An overview of Machine Learning

Pierre Gaillard – Inria, ENS Paris

September 28, 2017

Overview

Introduction

Supervised learning

- Empirical risk minimization: OLS, Logistic regression, Ridge, Lasso, Quantile regression
- Calibration of the parameters: cross-validation
- Local averages
- Deep learning

Unsupervised learning

- Clustering
- Dimensionality Reduction Algorithms

Tools for Machine Learning

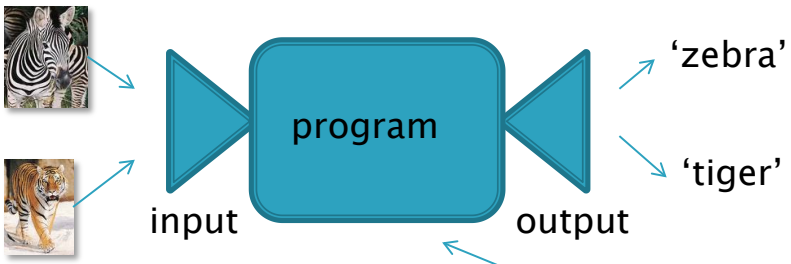
Introduction

What is it?

Machine Learning : artificial intelligence which can learn and model some phenomena without being explicitely programmed

Machine Learning $\subset$ Statistics + Computer Sciences

- Traditional programming:



- Machine learning:

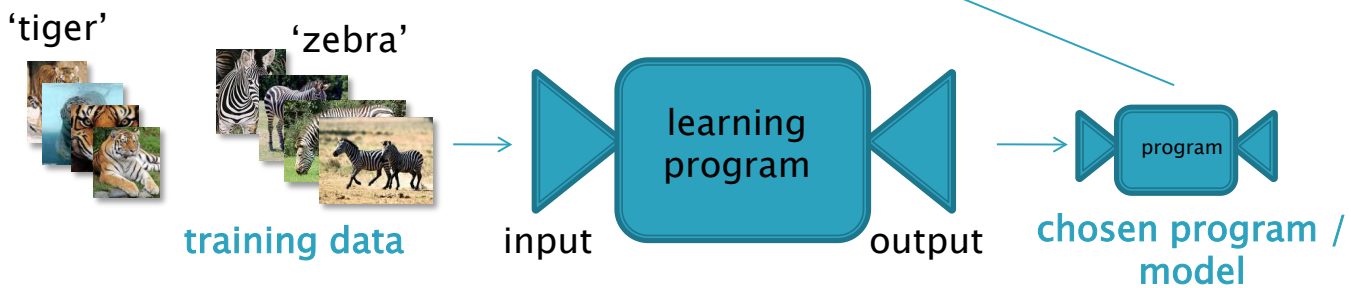


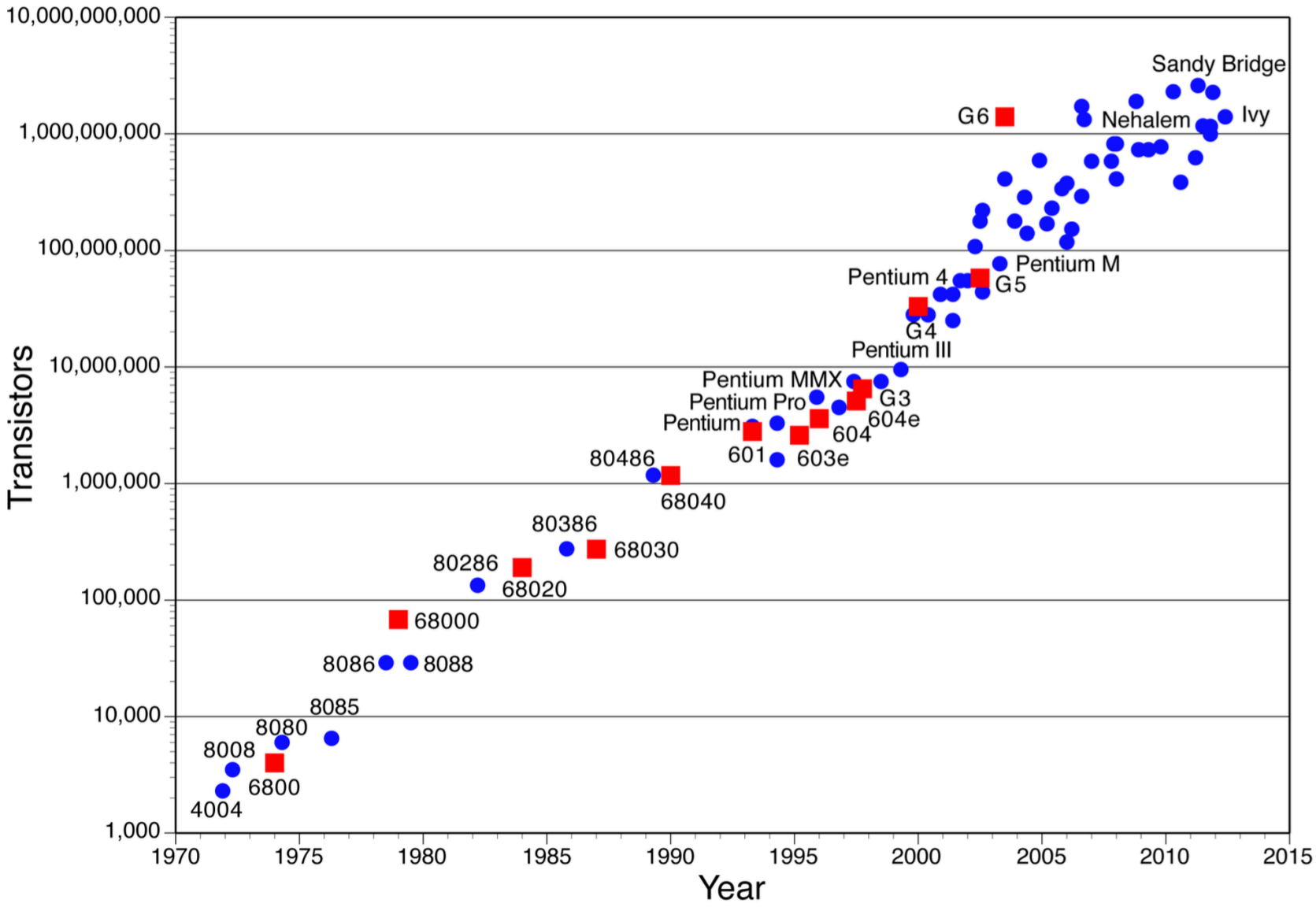
Image from Francis Bach lecture

The machine learning revolution

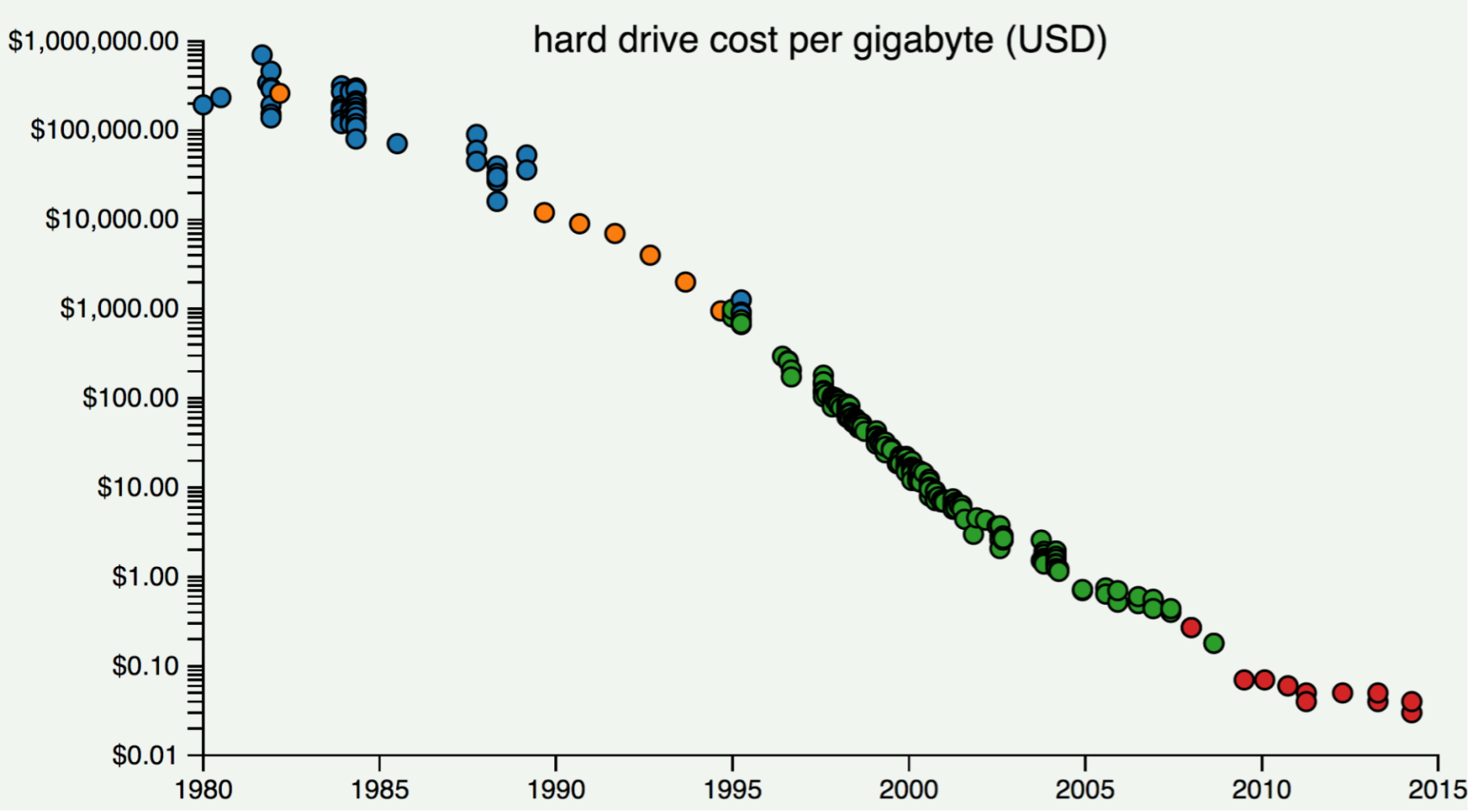
Big data / machine learning / data science / artificial intelligence / deep learning, a revolution?

- **Technical progress:** increase in computing power and storage capacity, lower costs

Moore's law: more computing power



Moore's Law: reduced costs



Limits : – debits do not follow

– miniaturization → reach the limits of classical physics → quantum mechanics

The machine learning revolution

Big data / machine learning / data science / artificial intelligence / deep learning, a revolution?

- **Technical progress:** increase in computing power and storage capacity, lower costs
- **Exponential increase in amount of data:** Volume, Variability, Velocity, Veracity
  - IBM: 10^{18} bytes created each day — 90% of the data $\leqslant$ 2 years
  - In all area: sciences, industries, personal life
  - In all forms: video, text, clicks, numbers

The machine learning revolution

Big data / machine learning / data science / artificial intelligence / deep learning, a revolution?

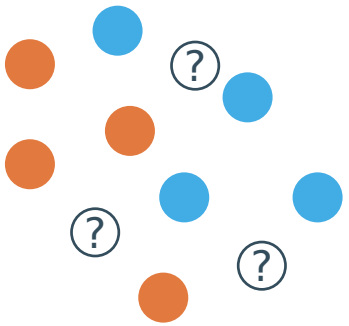
- **Technical progress:** increase in computing power and storage capacity, lower costs
- **Exponential increase in amount of data:** Volume, Variability, Velocity, Veracity
  - IBM: 10^{18} bytes created each day — 90% of the data $\leqslant$ 2 years
  - In all area: sciences, industries, personal life
  - In all forms: video, text, clicks, numbers
- **Methodological advancement** to analyze complex datasets: high dimensional statistics, deep learning, reinforcement learning,...

Overview of most popular machine learning methods

Two main categories of machine learning algorithms:

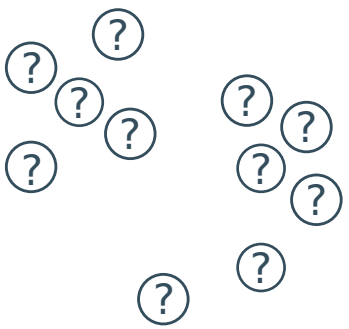
- **Supervised learning:** predict output Y from some input data X. The training data has a known label Y.

  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
  - X is a music track and Y are the audio signals of each instrument



- **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
  - cluster data in homogeneous groups
  - compress data without loosing much information
  - density estimation



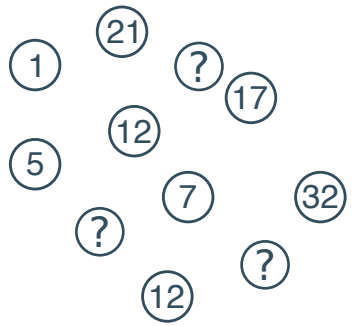
- **Others:** reinforcement learning, semi-supervised learning, online learning,...

Overview of most popular machine learning methods

Two main categories of machine learning algorithms:

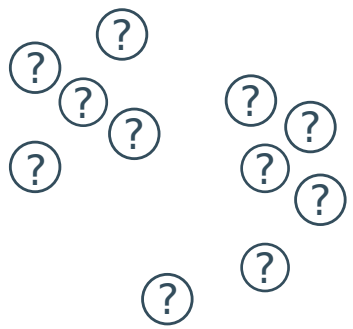
- **Supervised learning:** predict output Y from some input data X. The training data has a known label Y.

  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
  - X is a music track and Y are the audio signals of each instrument



- **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
  - cluster data in homogeneous groups
  - compress data without loosing much information
  - density estimation



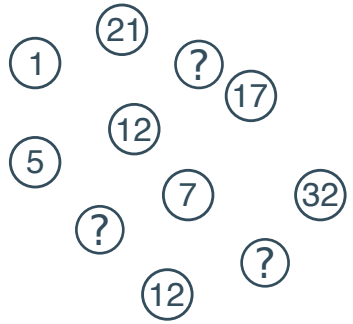
- **Others:** reinforcement learning, semi-supervised learning, online learning,...

Overview of most popular machine learning methods

Two main categories of machine learning algorithms:

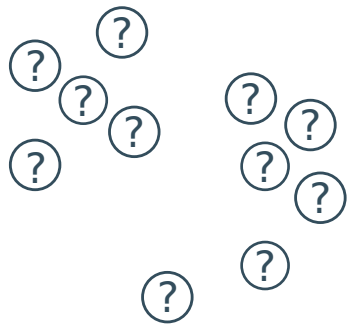
- **Supervised learning:** predict output *Y* from some input data *X*. The training data has a known label *Y*.

Classification	Regression
SVM Logistic regression Random Forest	Lasso, Ridge Nearest Neighbors Neural Networks



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

Clustering	Dimensionality reduction
K-means, the Apriori algorithm, Birch, Ward, Spectral Cluster	PCA, ICA word embedding



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

Supervised learning

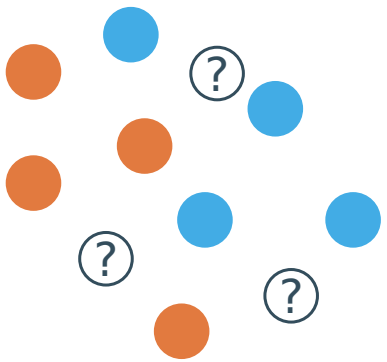
Supervised learning

Goal: from training data, we want to predict an output Y (or the best action) from the observation of some input X.

Difficulties: Y is not a deterministic function of X. There can be some noise:

$$Y = f(X) + \varepsilon$$

The function f is unknown and can be sophisticated.
$\rightarrow$ hard to perform well systematically



Possible theoretical approaches: perform well

- in the worst-case: minimax theory, game theory
- in average, or with high probability

Algorithmic approaches:

- local averages: K-nearest neighbors, decision trees
- empirical risk minimization: linear regression, lasso, spline regression, SVM, logistic regression
- online learning
- deep learning
- probabilist models: graphical models, Bayesian methods

Supervised learning: theory

Some data $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ is distributed according to a probability distribution P.

We observe training data $D_n := \{(X_1, Y_1), \dots, (X_n, Y_n)\}$.

We must form prediction into a decision set $\mathcal{A}$ by choosing a prediction function

$$f : \underbrace{\mathcal{X}}_{\text{observation}} \to \underbrace{\mathcal{A}}_{\text{decision}}$$

Our performance is measured by a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}$. We define the risk

$$R(f) := \mathbb{E}\big[\ell\big(f(X), Y\big)\big] \quad = \quad \text{expected loss of } f$$

Goal: minimize $R(f)$ by approaching the performance of the oracle $f^* = \arg\min_{f \in \mathcal{F}} R(f)$

	Least square regression	Classification
$\mathcal{A} = \mathcal{Y}$	$\mathbb{R}$	$\{0, 1, \dots, K-1\}$
$\ell(a, y)$	$(a-y)^2$	$\mathbb{1}_{a \neq y}$
$R(f)$	$\mathbb{E}\big[(f(X)-Y)^2\big]$	$\mathbb{P}(f(X) \neq Y)$
f^*	$\mathbb{E}[Y\|X]$	$\arg\max_k \mathbb{P}(Y = k\|X)$

Outline

Empirical risk minimization

Idea: estimate $R(f)$ thanks to the training data with the empirical risk

$$\underbrace{\widehat{R}_n(f) := \frac{1}{n}\sum_{i=1}^{n} \ell\big(f(X_i), Y_i\big)}_{\text{average error on training data}} \quad \approx \quad \underbrace{R(f) = \mathbb{E}\big[\ell\big(f(X), Y\big)\big]}_{\text{expected error}}$$

We estimate $\widehat{f}_n$ by minimizing the empirical risk

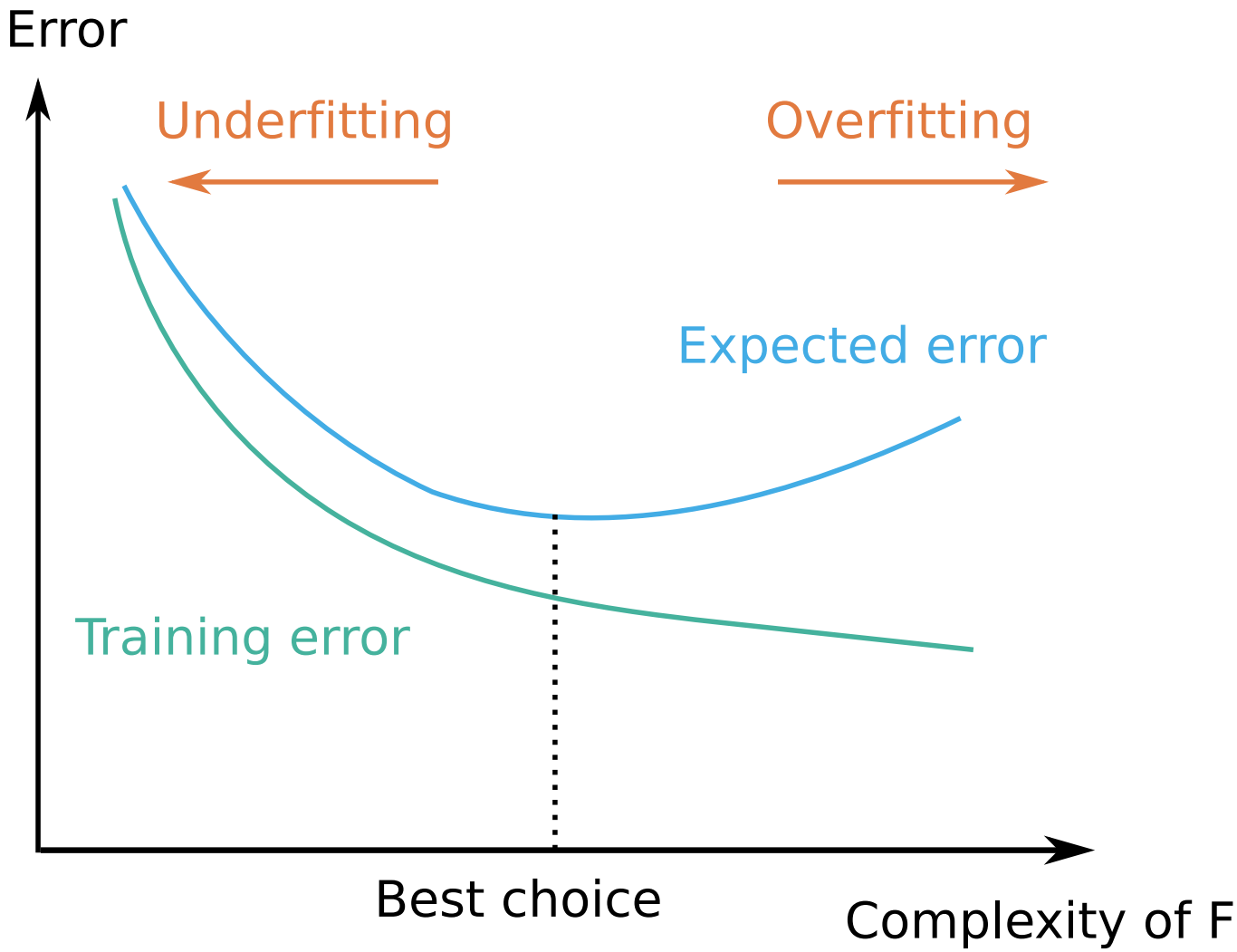
$$\widehat{f}_n \in \operatorname*{arg\,min}_{f \in \mathcal{F}} \widehat{R}_n(f)\,.$$

Many methods are based on empirical risk minimization: ordinary least square, logistic regression, Ridge, Lasso,...

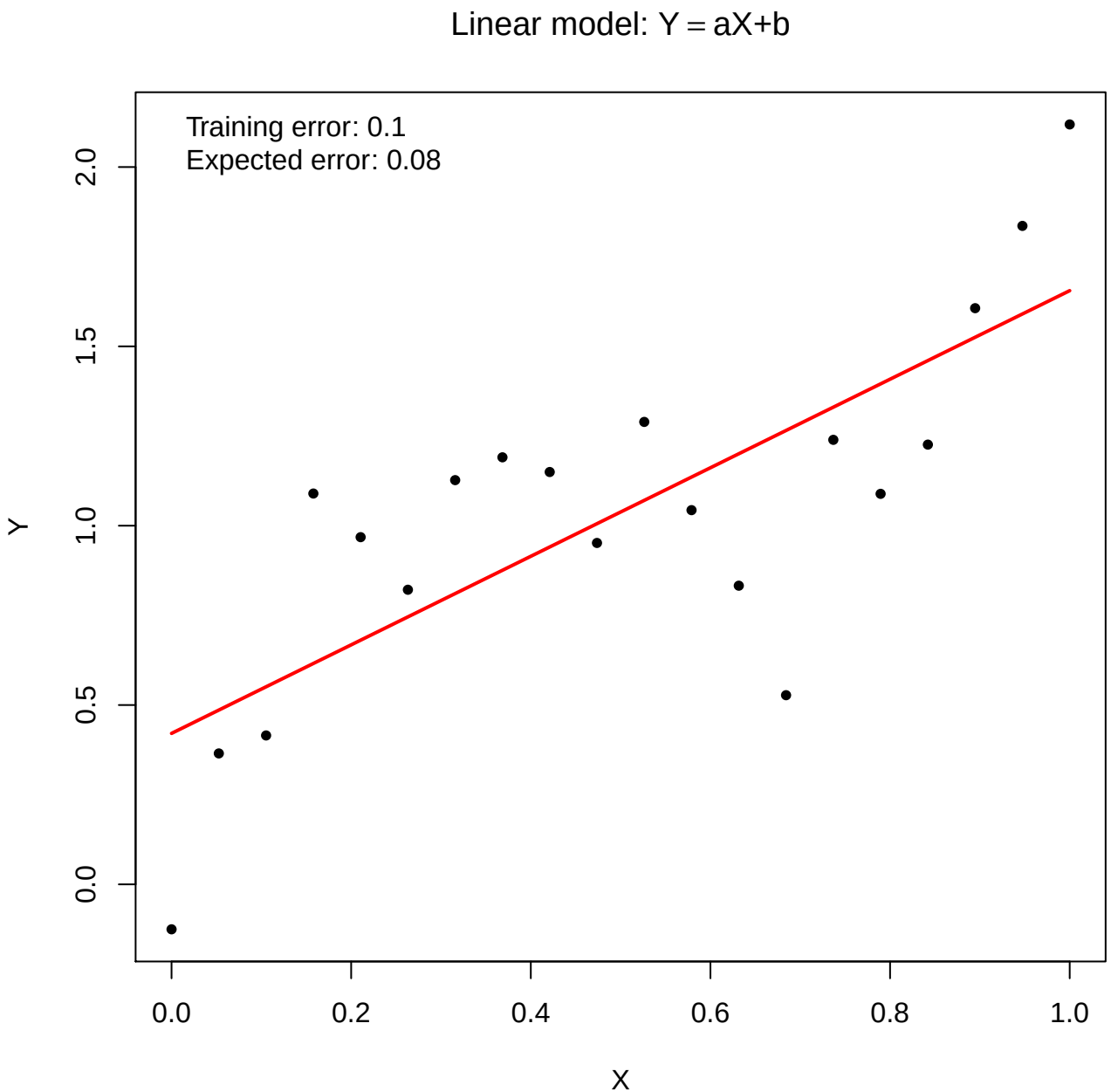
Choosing the right model: $\mathcal{F}$ is a set of models which needs to be properly chosen:

$$R(\widehat{f}_n) = \underbrace{\min_{f \in \mathcal{F}} R(f)}_{\text{Approximation error}} + \underbrace{R(\widehat{f}_n) - \min_{f \in \mathcal{F}} R(f)}_{\text{Estimation error}}$$

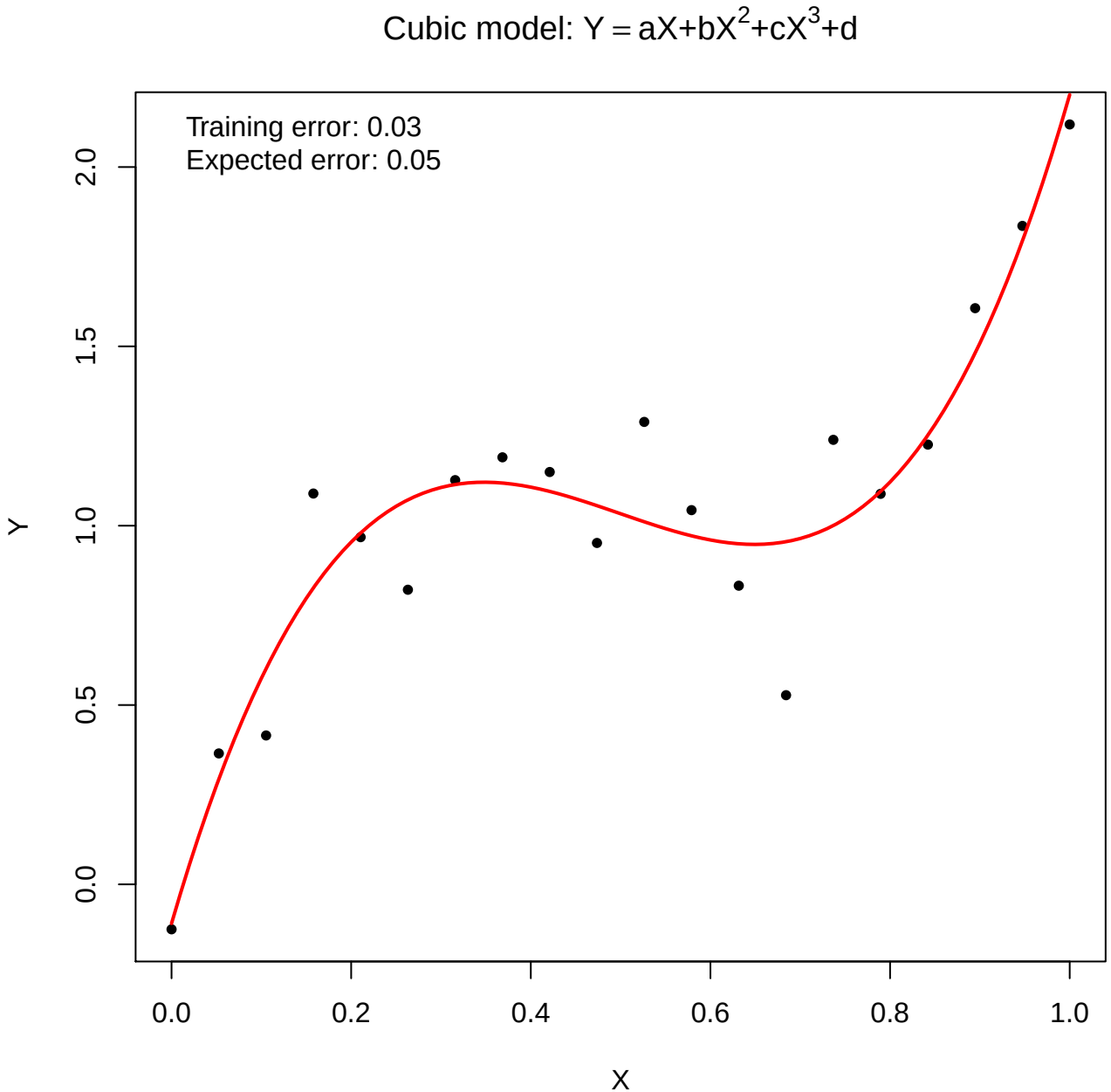
Overfitting



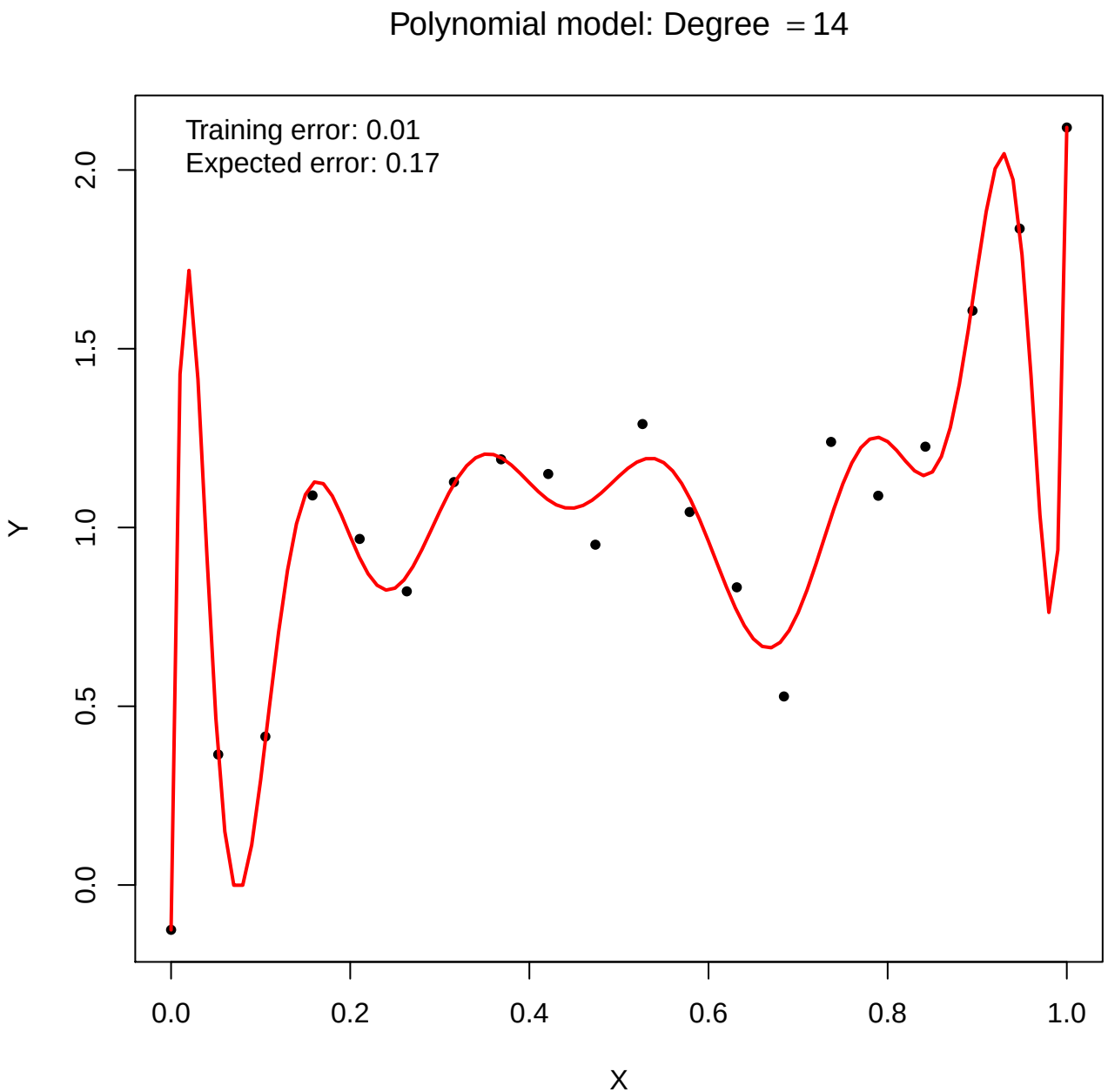
Overfitting: example in regression



Overfitting: example in regression



Overfitting: example in regression



Least square Linear regression

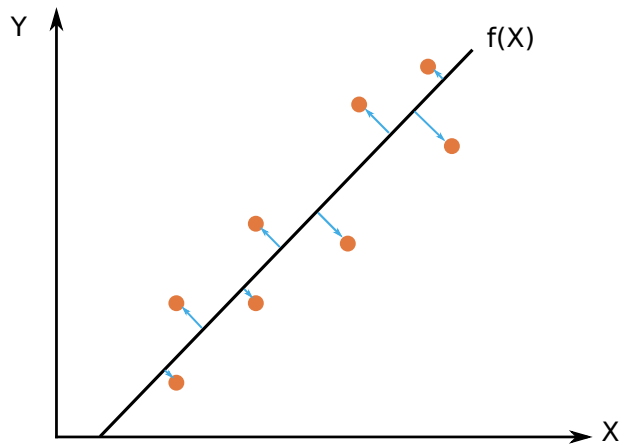
Given training data (X_i, Y_i) for $i = 1, \ldots, n$, with $X_i \in \mathbb{R}^d$ and $Y_i \in \{0, 1\}$ learn a predictor f such that our expected square loss

$$\mathbb{E}\left[(f(X) - Y)^2\right]$$

is small.

We assume here that f is a linear combination of the input $x = (x_1, \ldots, x_d)$

$$f_w(x) = \sum_{i=1}^{d} w_i x_i = w^\top x$$



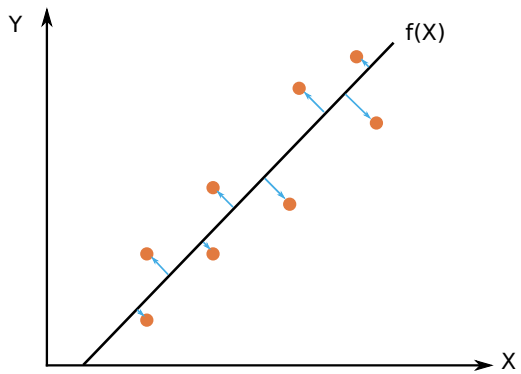
Ordinary Least Square

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

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

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

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

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

What happens if $d \gg n$?

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

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

$$\widehat{w}_n \in \arg\min_w \widehat{R}_n(w) = \left(X^\top X\right)^{-1} X^\top Y\,.$$

Question: how to compute it?

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

$$R\widehat{w} = Q^\top Y, \qquad \text{with} \qquad R = \begin{pmatrix} x & x & \cdots & x \\ & \ddots & & \\ & & \ddots & \\ & 0 & & x \end{pmatrix}$$

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

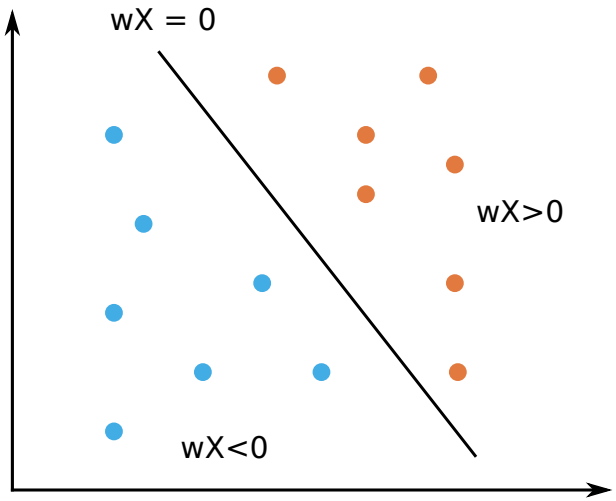
$$w_{i+1} = w_i - \eta \nabla \widehat{R}_n(w_i)$$

Classification

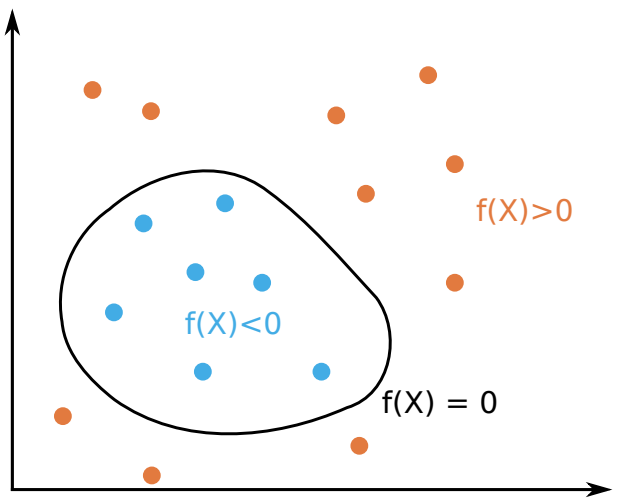
Given training data (X_i, Y_i) for $i = 1, \ldots, n$, with $X_i \in \mathbb{R}^d$ and $Y_i \in \{0, 1\}$ learn a classifier $f(x)$ such that

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

Linearly separable



Non Linearly separable



Linear classification

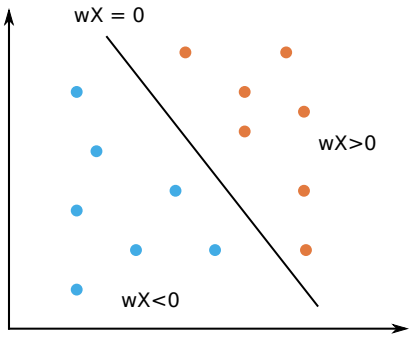
We would like to find the best linear classifier such that

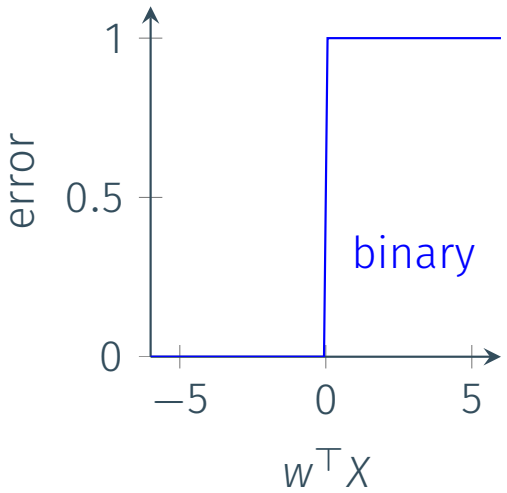
$$f_w(X) = w^\top X \begin{cases} \geqslant 0 & \Rightarrow \quad Y = +1 \\ < 0 & \Rightarrow \quad Y = 0 \end{cases}$$

Empirical risk minimization with the binary loss?

$$\widehat{w}_n = \operatorname*{arg\,min}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{Y_i \neq \mathbb{1}_{w^\top X_i \geqslant 0}} .$$

👎 This is not convex in w. Very hard to compute!

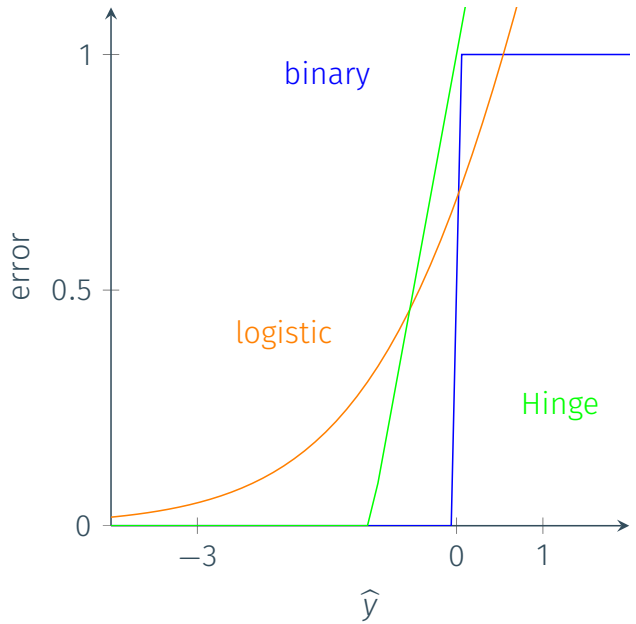


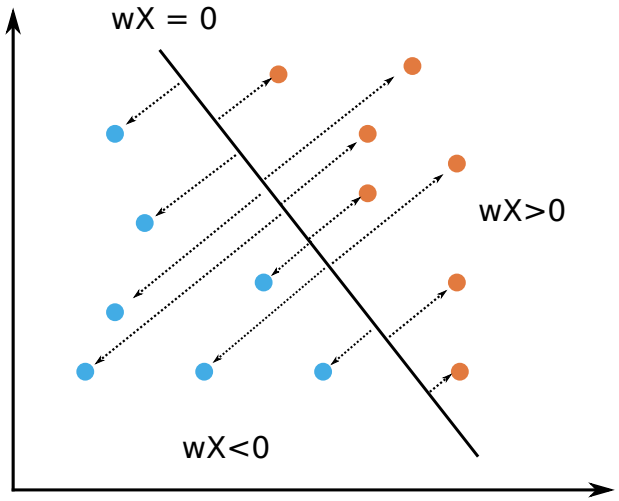


Logistic regression

Idea: replace the loss with a convex loss

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





Probabilistic interpretation: based on likelihood maximization of the model:

$$\mathbb{P}(Y = 1|X) = \frac{1}{1 + e^{-w^\top X}} \in [0, 1]$$

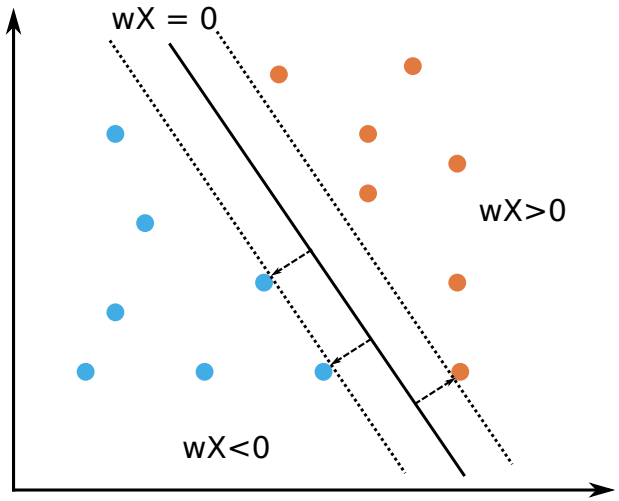
Satisfied for many distributions of $X|Y$: Bernoulli, Gaussian, Exponential, ...

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

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

Support Vector Machine (SVM)

In SVM, the linear separator (hyperplane) is chosen by maximizing the margin. Not by minimizing the empirical risk.



👍 Sparsity: it only depends on a few training points, called the support vectors

In practice, we use soft margins because no perfect linear separation is possible.

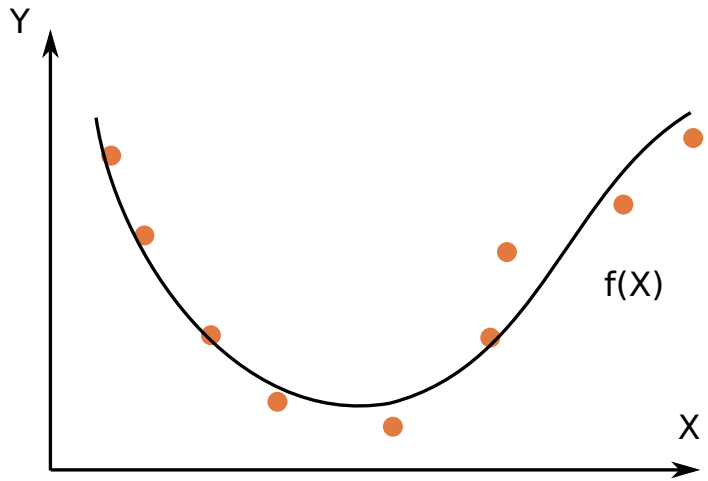
Non-linear regression/classification

Until now, we have only considered linear predictions of $x = (x_1, \ldots, x_d)$

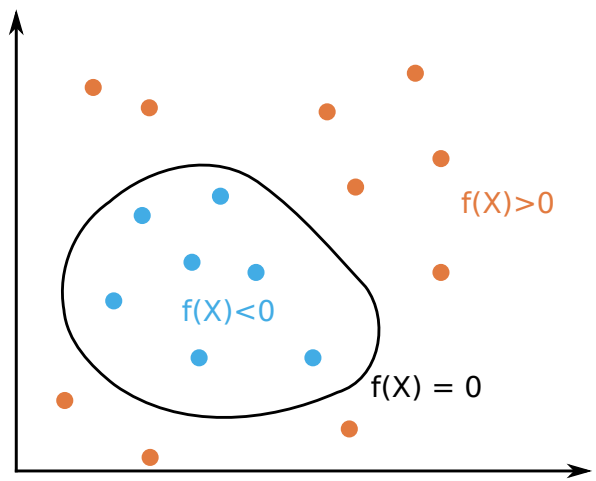
$$f_w(x) = \sum_{i=1}^{d} w_i x_i \, .$$

But this can perform pretty bad... How to perform non-linear regression?

Non linear regression



Non Linearly separable



Non-linear regression/classification

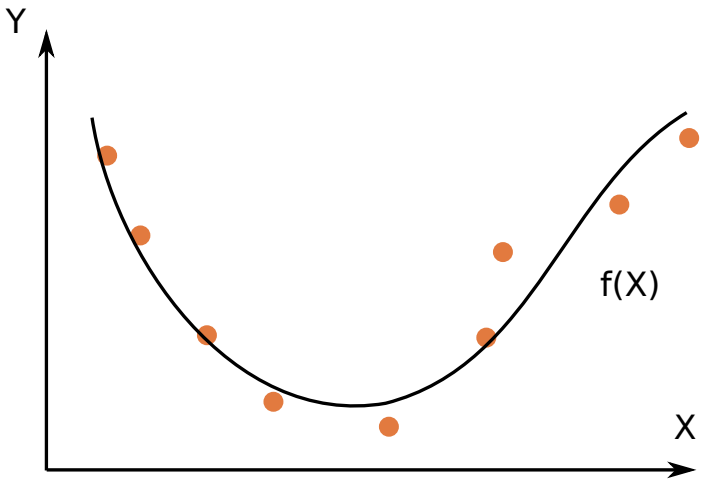
Idea: map the input X into a higher dimensional space where the problem is linear.

Example: given an input $x = (x_1, x_2, x_3)$ perform a linear method on a transformation of the input like

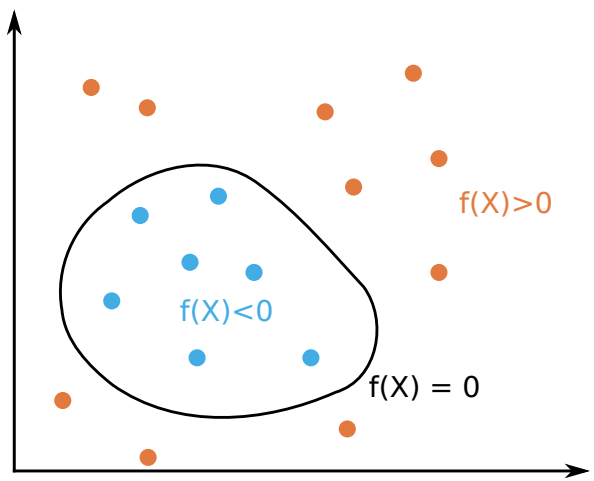
$$\Phi(x) = \left(x_1x_1, x_1x_2, \ldots, x_3x_2, x_3x_3\right) \in \mathbb{R}^9$$

Linear transformations of $\Phi(x)$ are polynomials of x! The previous methods works by replacing x with $\Phi(x)$.

Non linear regression



Non Linearly separable



Spline regression

A spline of degree p is a function formed by connecting polynomial segments of degree p so that:

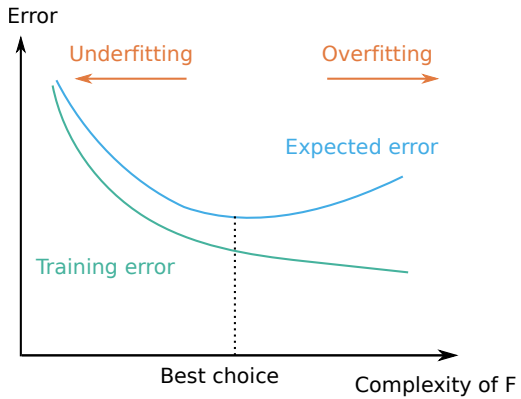
- the function is continuous
- the function has D – 1 continuous derivatives
- the pth-derivative is constant between knots

This can be done by choosing the good transformation $\Phi_p(x)$ and the right regularization $\|\Phi_p(x)\|$.

Difficulties: choose the number of knots and the degree

Regularization

How to avoid over-fitting if there is not enough data?



Control the complexity of the solution

- explicitly by choosing $\mathcal{F}$ small enough: choose the degree of the polynomials,...
- implicitly by adding a regularization term

$$\min_{f \in \mathcal{F}} \widehat{R}_n(f) + \lambda \|f\|^2$$

The higher the norm $\|f\|$ is, the more complex the function is.

👍 We do not need to know the best complexity $\mathcal{F}$ in advance

👎 Complexity controlled by λ, which need to be calibrated.

Ridge regression

The most classic regularization in statistics for linear regression:

$$\widehat{w}_n = \operatorname*{arg\,min}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2 \; + \; \lambda \sum_{i=1}^{d} w_i^2$$

The exact solution is unique because the problem is now strongly convex:

$$\widehat{w}_n = \left(X^\top X + n\lambda I\right)^{-1} X^\top Y$$

The regularization parameter λ controls the matrix conditioning:

- if $\lambda = 0$: ordinary linear regression
- if $\lambda \to \infty$: $\widehat{w}_n \to 0$

The Lasso: how to choose among a large set of variables with few observations

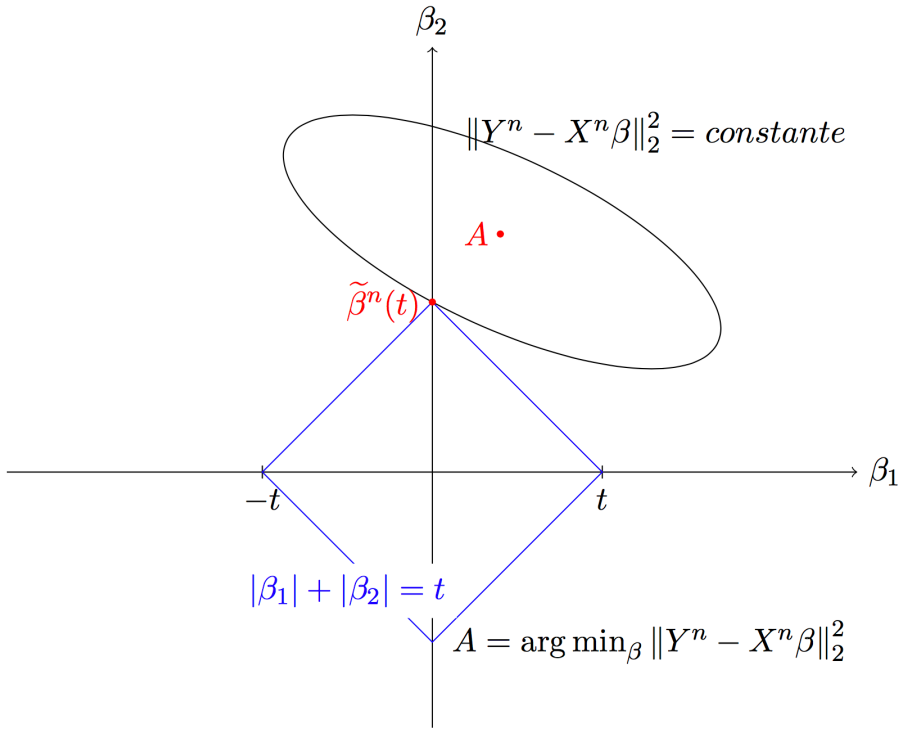
The Lasso corresponds to L_1 regularization:

$$\widehat{w}_n = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2 + \lambda \sum_{i=1}^{d} |w_i|$$

- Powerful if $d \gg n$: many potential variables, few observations
- $\widehat{w}_n$ is sparse: most of its values will be 0 $\rightarrow$ can be used to choose variables

Other formulation of the Lasso:
$\exists \beta > 0$ such that

$$\widehat{w}_n \in \arg\min_{\|w\|_1 \leqslant \beta} \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2$$



The Lasso: how to choose among a large set of variables with few observations

The Lasso corresponds to L_1 regularization:

$$\widehat{w}_n = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2 + \lambda \sum_{i=1}^{d} |w_i|$$

- 👍 Powerful if $d \gg n$: many potential variables, few observations
- 👍 $\widehat{w}_n$ is sparse: most of its values will be 0 $\rightarrow$ can be used to choose variables

- 👎 **The Lasso is biased**: $\widehat{w}_n^\top X \neq \mathbb{E}[Y|X]$. Hence, it is better to:

Perform Lasso
↓
Choose variables with $\widehat{w}_i > 0$
↓
Perform Ridge on this sub-model only

Another solution is Elastic Net:

$$\widehat{w}_n = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2 + \lambda_1 \sum_{i=1}^{d} |w_i| + \lambda_2 \sum_{i=1}^{d} w_i^2$$

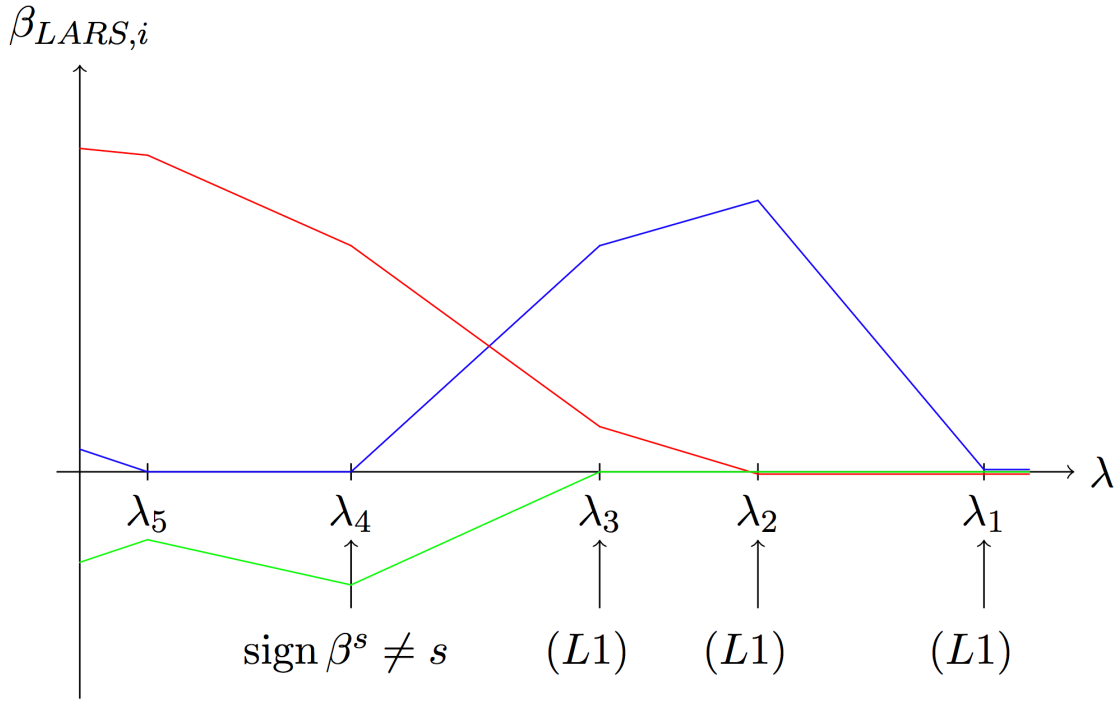
Many extensions of the Lasso exist: Group Lasso,...

Lasso: the regularization path

The Lasso corresponds to L_1 regularization:

$$\widehat{w}_n = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2 + \lambda \sum_{i=1}^{d} |w_i|$$

Plot of the evolution of the coefficients of $\widehat{w}_n$ as a function of λ:

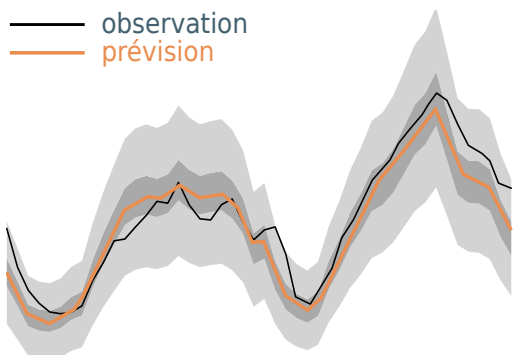


Probabilistic prediction

In some situation, we are not interested by prediction the average case $\mathbb{E}[Y|X]$ only, but by the distribution of $Y|X$. $\rightarrow$ give a measure of uncertainty of our prediction

Solution: modify the loss function:

- square loss $\ell(a, y) = (a - y)^2$: prediction of the expected value
- absolute loss $\ell(a, y) = |a - y|$: prediction of the median
  (50% to be above Y, and 50% chance to be below)
- pinball loss $\ell(a, y) = (a - y)(\tau - \mathbb{1}_{a<y})$: prediction of the τ-quantile
  ($(1 - \tau)$ chance to be above Y and τ chance to be below)

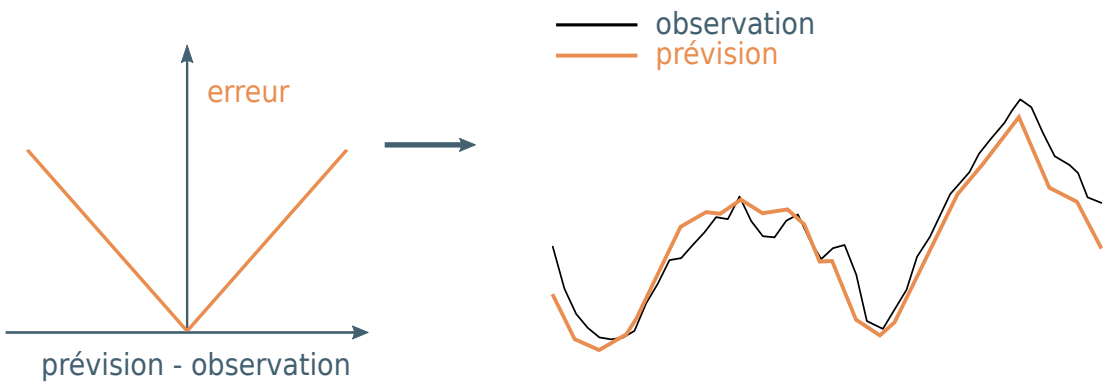


Probabilistic prediction

In some situation, we are not interested by prediction the average case $\mathbb{E}[Y|X]$ only, but by the distribution of $Y|X$. $\rightarrow$ give a measure of uncertainty of our prediction

Solution: modify the loss function:

- square loss $\ell(a, y) = (a - y)^2$: prediction of the expected value
- absolute loss $\ell(a, y) = |a - y|$: prediction of the median
  (50% to be above Y, and 50% chance to be below)
- pinball loss $\ell(a, y) = (a - y)(\tau - \mathbb{1}_{a<y})$: prediction of the τ-quantile
  ($(1 - \tau)$ chance to be above Y and τ chance to be below)

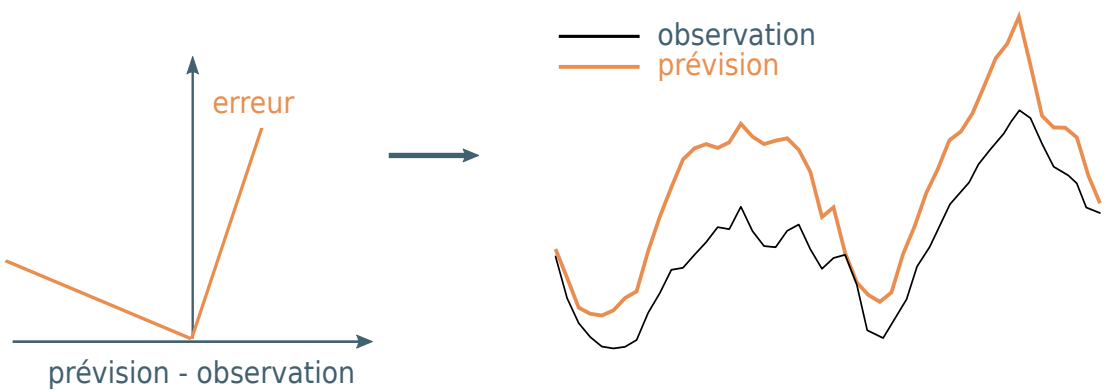


Probabilistic prediction

In some situation, we are not interested by prediction the average case $\mathbb{E}[Y|X]$ only, but by the distribution of $Y|X$. $\rightarrow$ give a measure of uncertainty of our prediction

Solution: modify the loss function:

- square loss $\ell(a, y) = (a - y)^2$: prediction of the expected value
- absolute loss $\ell(a, y) = |a - y|$: prediction of the median
  (50% to be above Y, and 50% chance to be below)
- pinball loss $\ell(a, y) = (a - y)(\tau - \mathbb{1}_{a<y})$: prediction of the τ-quantile
  ($(1 - \tau)$ chance to be above Y and τ chance to be below)

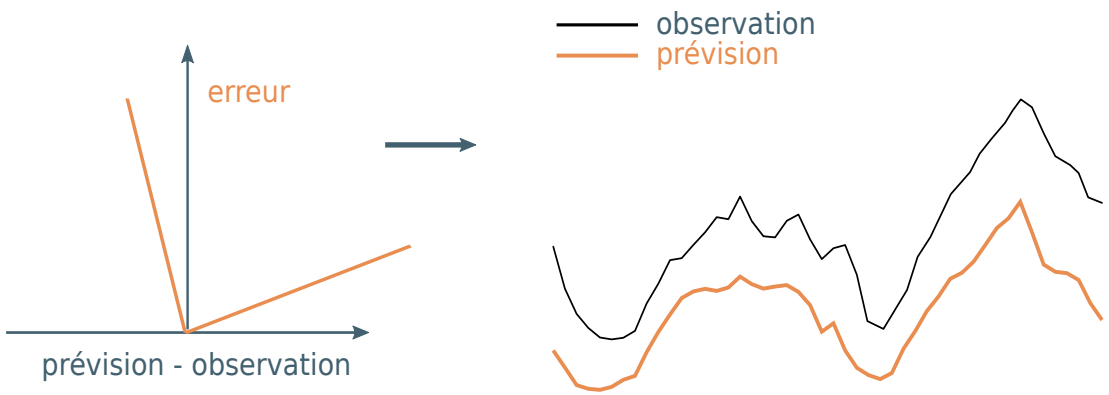


Probabilistic prediction

In some situation, we are not interested by prediction the average case $\mathbb{E}[Y|X]$ only, but by the distribution of $Y|X$. $\rightarrow$ give a measure of uncertainty of our prediction

Solution: modify the loss function:

- square loss $\ell(a, y) = (a - y)^2$: prediction of the expected value
- absolute loss $\ell(a, y) = |a - y|$: prediction of the median
  (50% to be above Y, and 50% chance to be below)
- pinball loss $\ell(a, y) = (a - y)(\tau - \mathbb{1}_{a<y})$: prediction of the τ-quantile
  ($(1 - \tau)$ chance to be above Y and τ chance to be below)

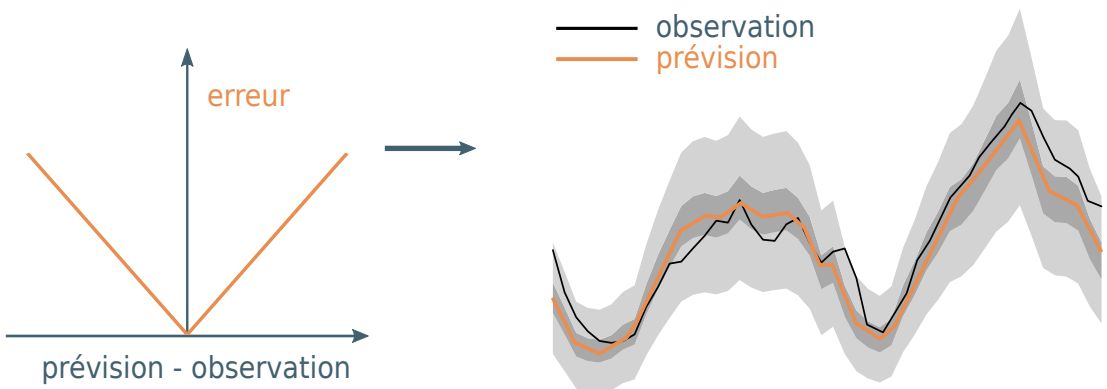


Probabilistic prediction

In some situation, we are not interested by prediction the average case $\mathbb{E}[Y|X]$ only, but by the distribution of $Y|X$. $\rightarrow$ give a measure of uncertainty of our prediction

Solution: modify the loss function:

- square loss $\ell(a, y) = (a - y)^2$: prediction of the expected value
- absolute loss $\ell(a, y) = |a - y|$: prediction of the median
  (50% to be above Y, and 50% chance to be below)
- pinball loss $\ell(a, y) = (a - y)(\tau - \mathbb{1}_{a<y})$: prediction of the τ-quantile
  ($(1 - \tau)$ chance to be above Y and τ chance to be below)



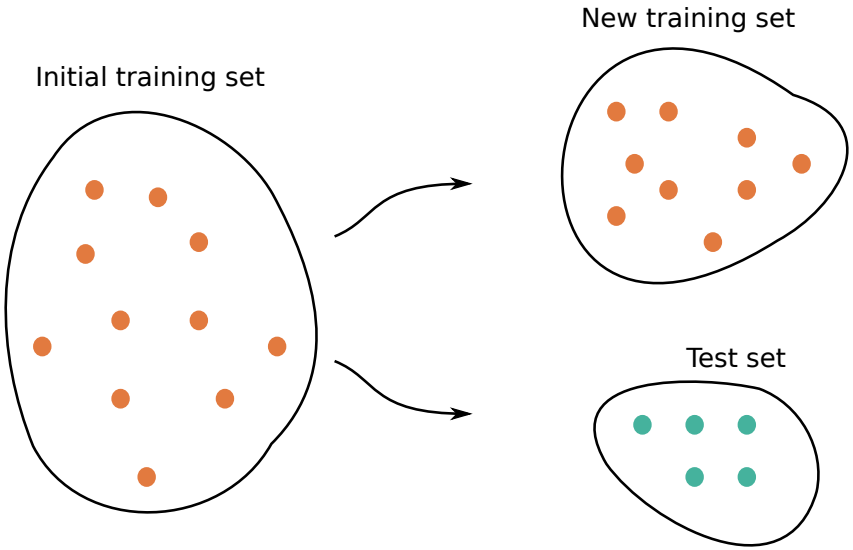
Outline

How to choose the parameters? Test set

All the methods in machine learning depend on learning parameters.

How to choose them? First solution: use a test set.

- randomly choose 70% of the data to be in the training set
- the remainder is a test set



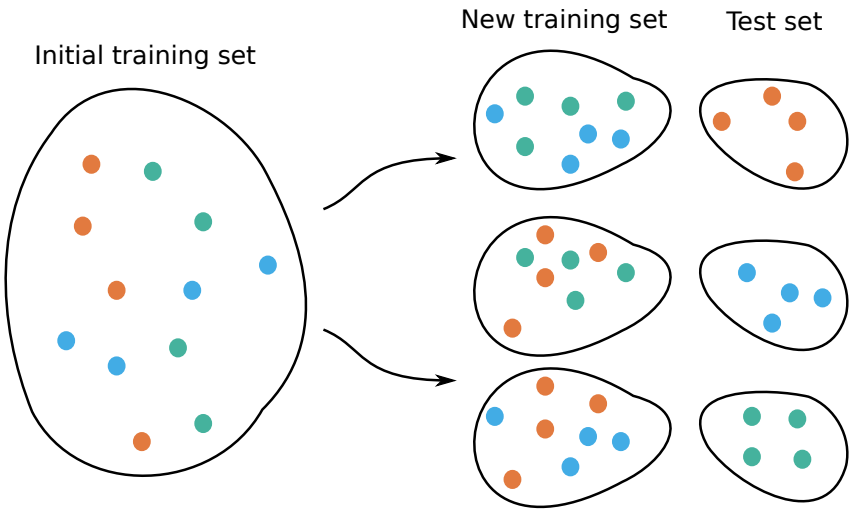
We choose the parameter with the smallest error on the test set.

- very simple
- waste data: the best method is fitted only with 70% of the data
- with bad luck the test set might be lucky or unlucky

How to choose the parameters? Cross-validation

Cross-validation:

- randomly break data into K groups
- for each group, use it as a test set and train the data on the $(K-1)$ other groups



We choose the parameter with the smallest average error on the test sets.

👍 only $1/K$ of the data lost for training
👎 K times more expensive

In practice: choose $K \approx 10$.

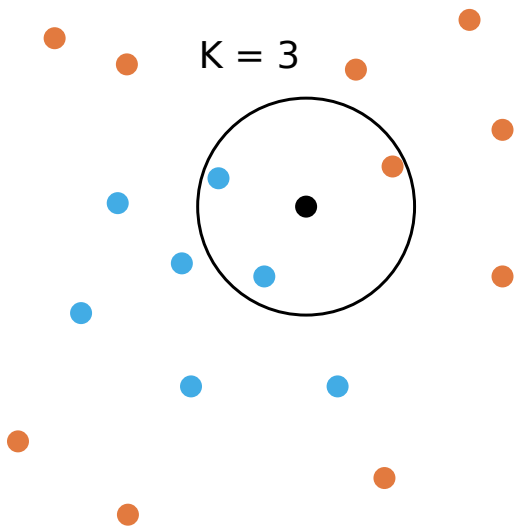
Outline

K-Nearest Neighbors

Classify data based on similarity with neighbors.

When observing a new input x, find the k-closest training data points to x and for

- classification: predict the most frequently occuring class
- regression: predict the average value

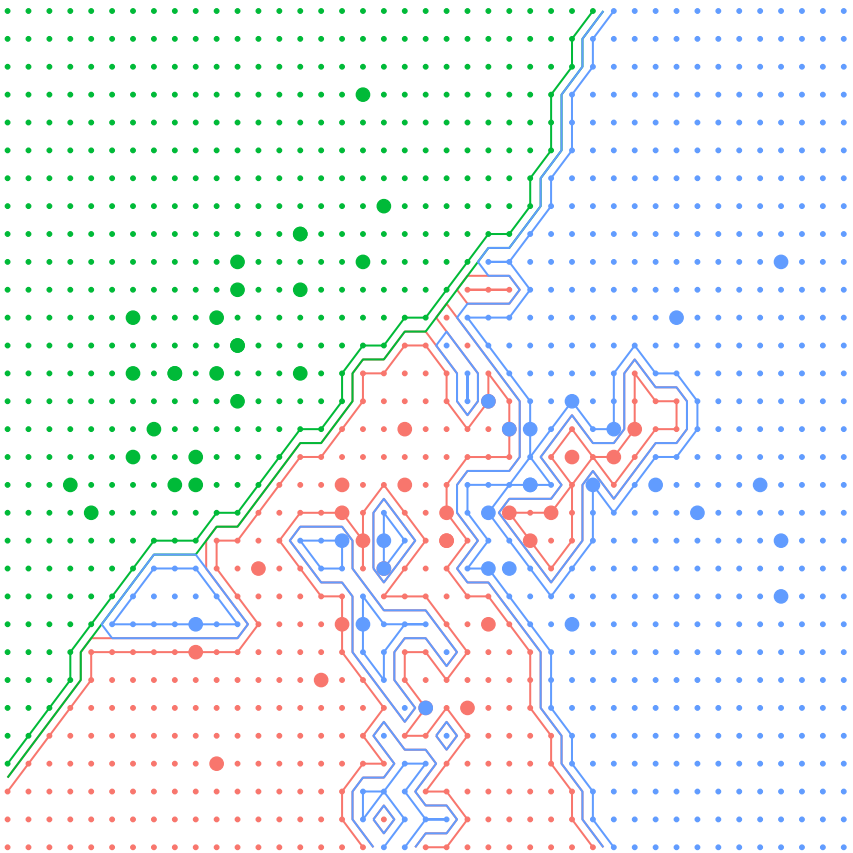


K-Nearest Neighbors

Classify data based on similarity with neighbors.

When observing a new input x, find the k-closest training data points to x and for

- classification: predict the most frequently occuring class
- regression: predict the average value



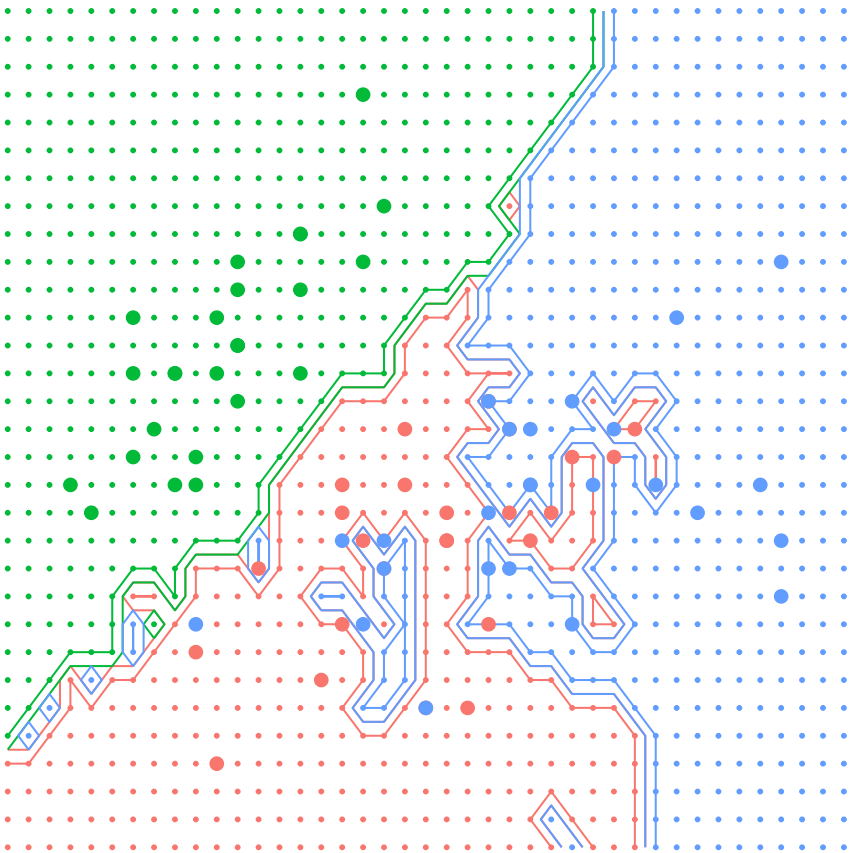
$K = 1$

K-Nearest Neighbors

Classify data based on similarity with neighbors.

When observing a new input x, find the k-closest training data points to x and for

- classification: predict the most frequently occuring class
- regression: predict the average value



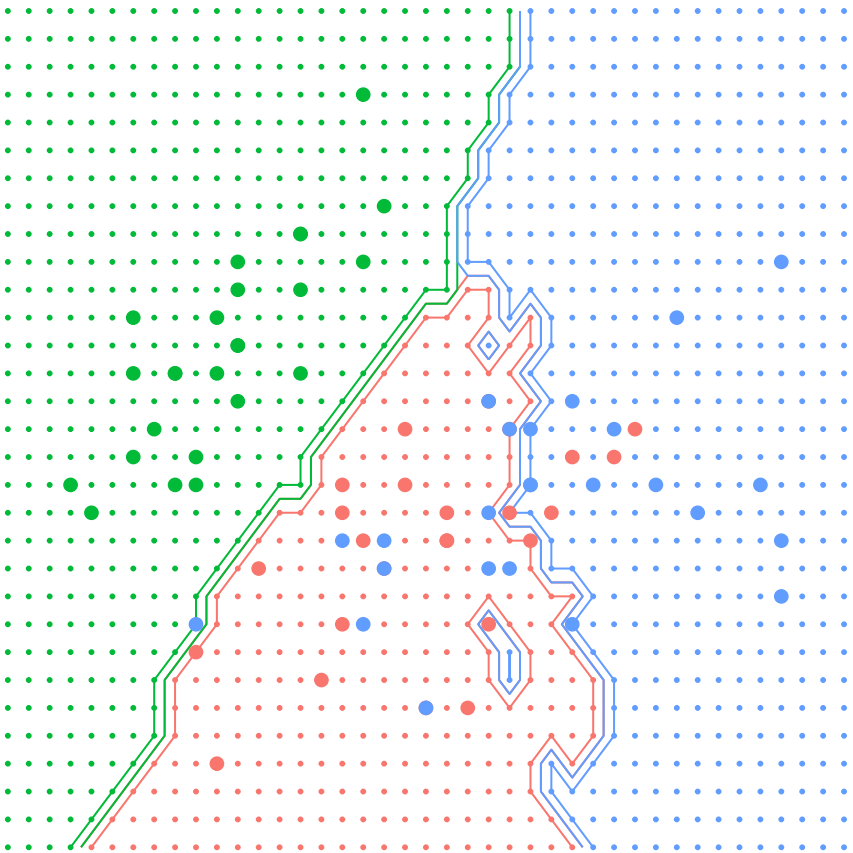
$K = 3$

K-Nearest Neighbors

Classify data based on similarity with neighbors.

When observing a new input x, find the k-closest training data points to x and for

- classification: predict the most frequently occuring class
- regression: predict the average value



$K = 20$

K-Nearest Neighbors

Advantages:

- No optimization or training
- Easy to implement
- Can get very good performance

Drawbacks:

- Slow at query time: must pass through all training data at each
- Easily fooled by irrelevant inputs
- Bad for high-dimensional data ($d > 20$)

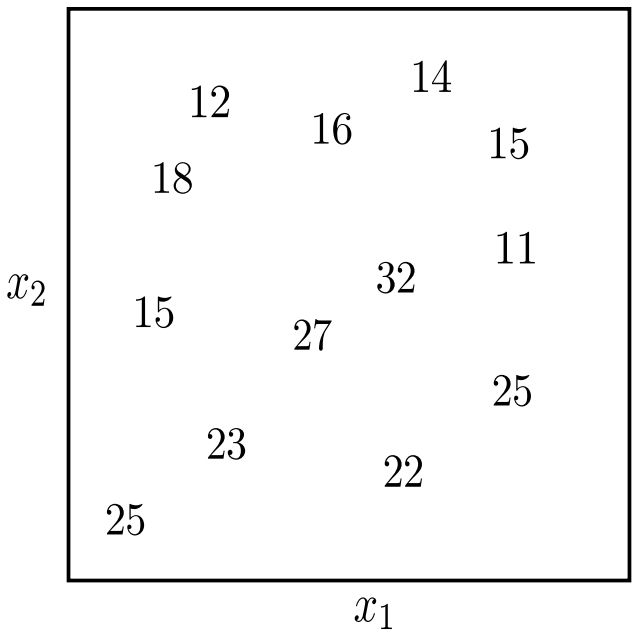
Difficulties:

- choice of K
- what distance for complex data?

Decision Tree

Introduced by Breiman et al. 1984

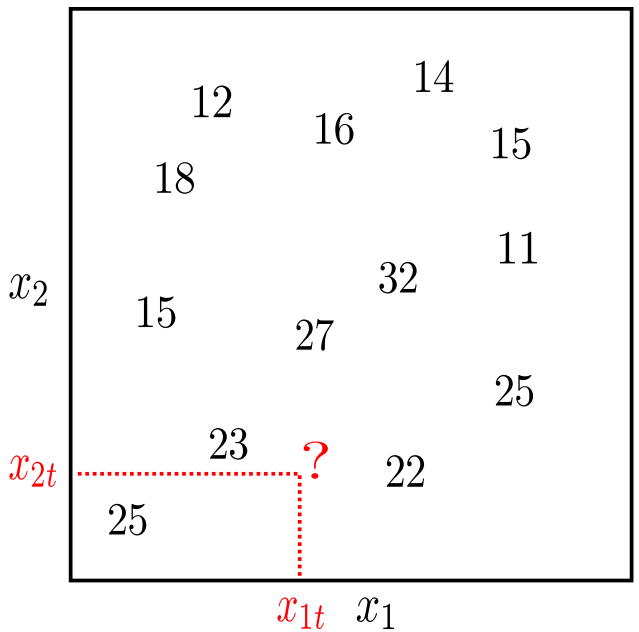
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

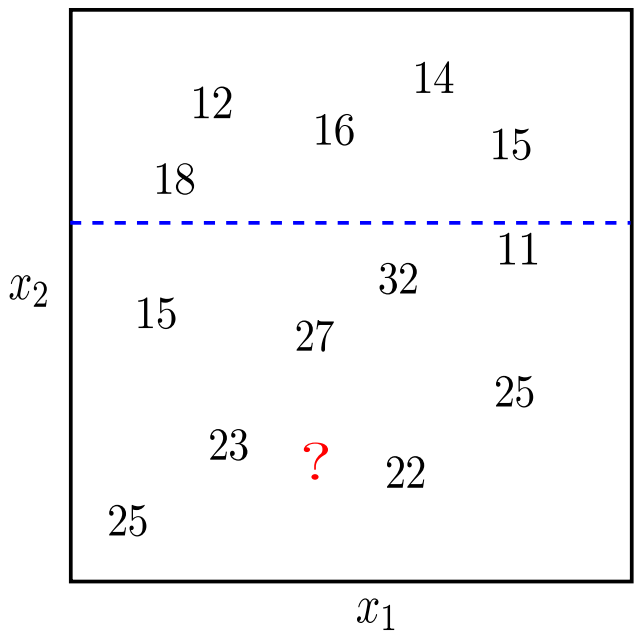
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

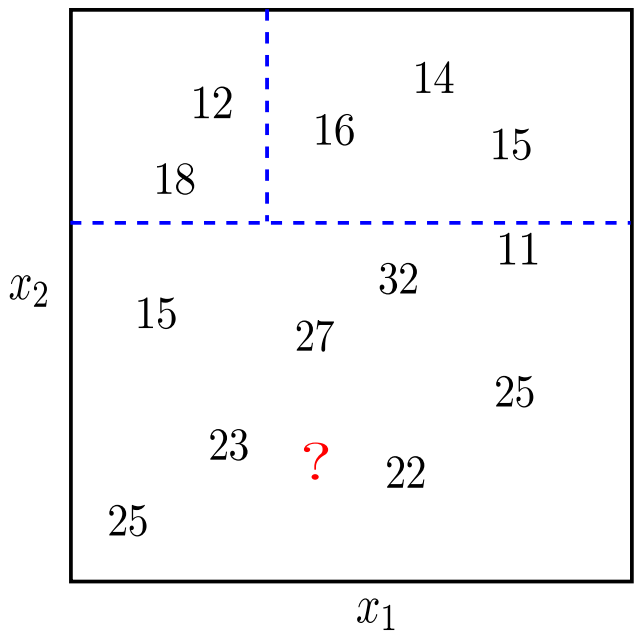
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

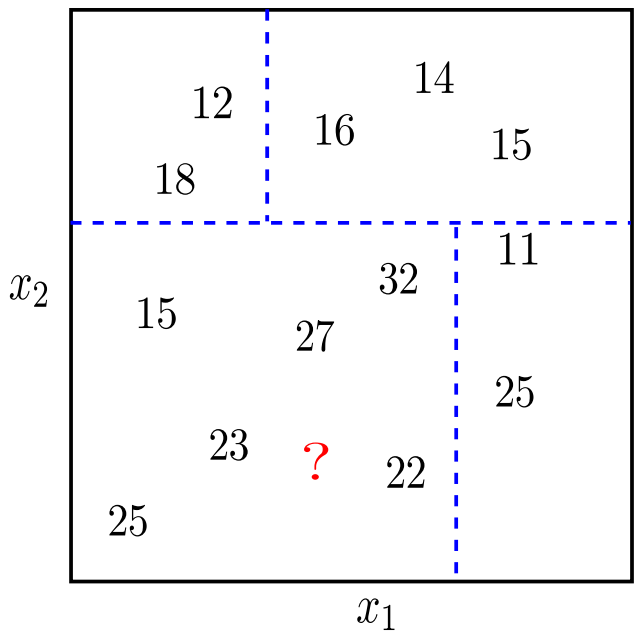
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

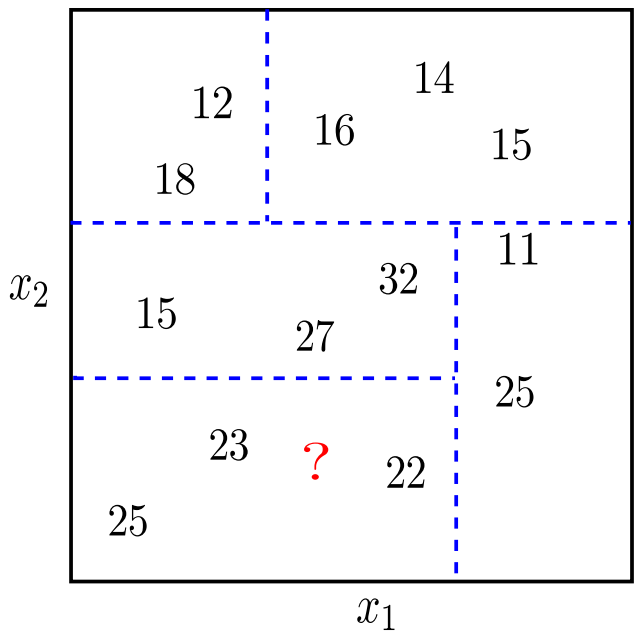
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

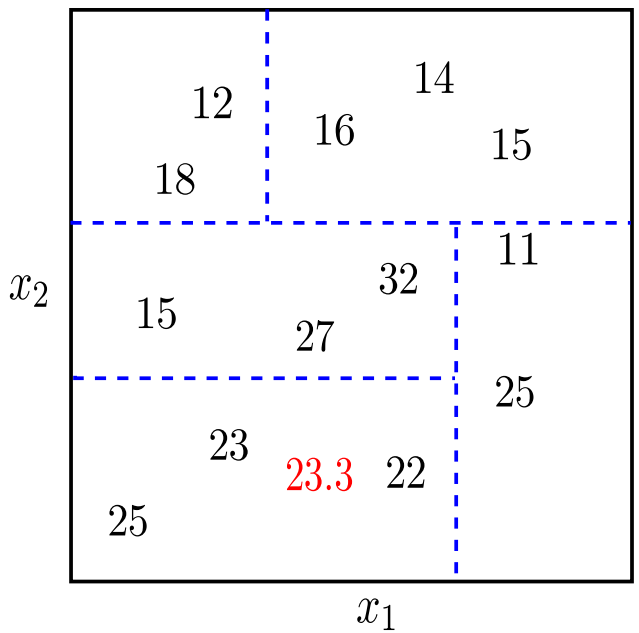
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

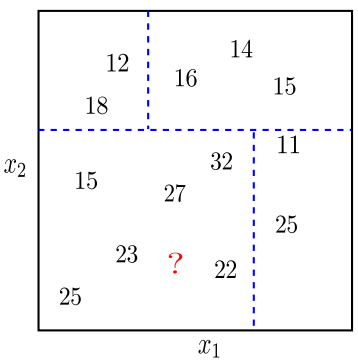
Idea: partitioned the input space in an inductive and diadic fashion.



Decision Tree

Introduced by Breiman et al. 1984

Idea: partitioned the input space in an inductive and diadic fashion.



To construct the tree, we need to answer two questions:

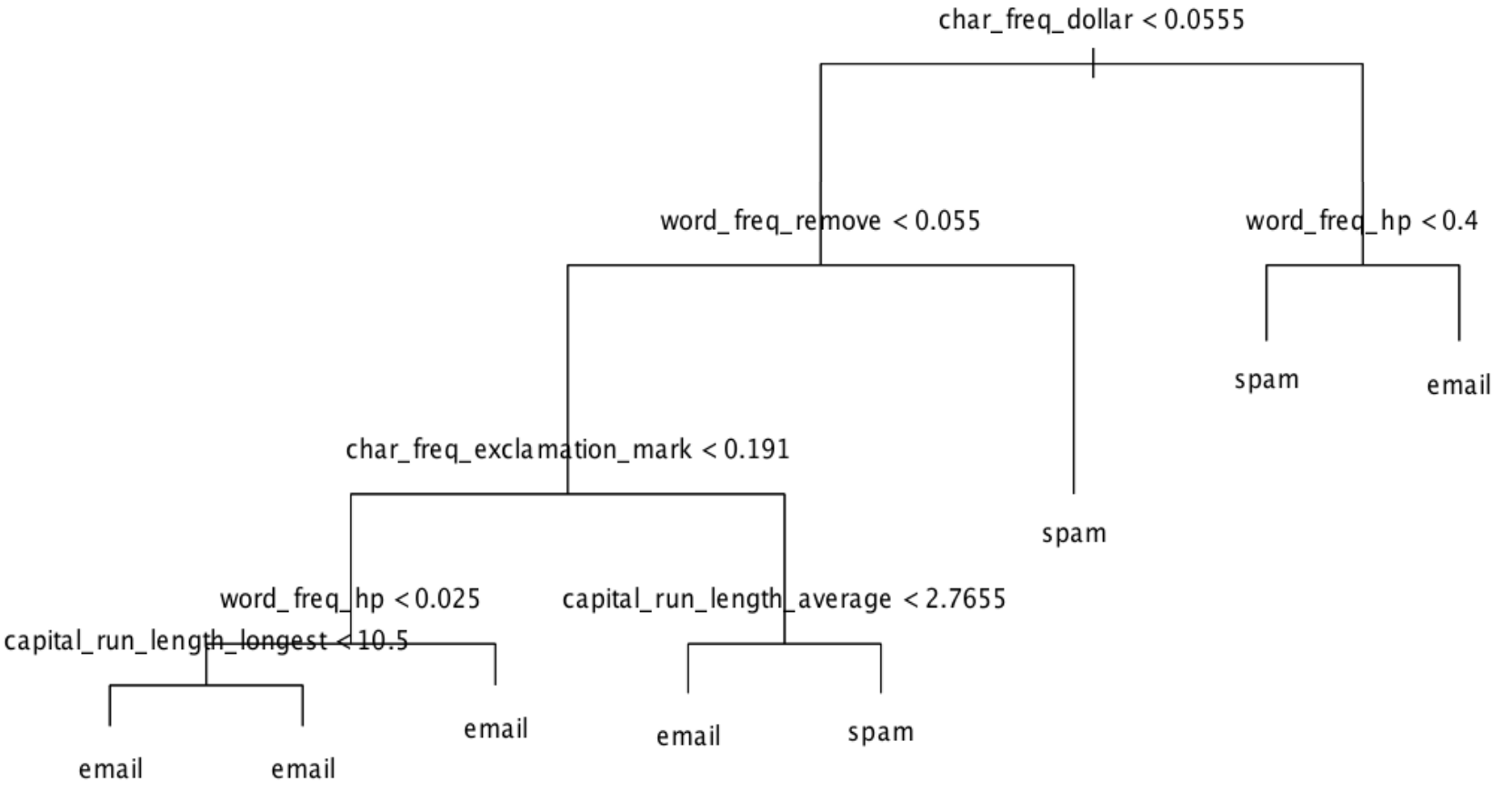
- **Location of the cuts:** which variable, what threshold?
  $\rightarrow$ minimize the inter-groups variance
- **Depth of the tree:** when do we stop? Over-fitting risk!
  - continue while variance decreases enough
  - pruning: build a large tree and prune it by minimizing a penalized error:

$$\text{Test error}(T) + \lambda \text{size}(T)$$

👍 **Advantage:** interpretable computational cost

👎 **Drawbacks:** instable (butterfly effect),

Decision trees: example for spam detection



Ensemble algorithms

Ensemble algorithms are based on the following idea: averaging adds stability.

Example: Assume that $Y \in \{0, 1\}$ and that you have K independent classification methods $f_k, k = 1, \dots, K$ such that $\mathbb{P}(f_k(X) \neq Y) \leqslant \varepsilon$. Then from Hoeffding's inequality:

$$\mathbb{P}\big(\text{majority voting of } f_k(X) \neq Y\big) \lesssim e^{-K\varepsilon^2}$$

$\rightarrow$ exponential decrease to 0!

Idea: build base methods as independent as possible and average them.

1. split the training set into K subsets of size n/K
2. train a different "base learner" on each subset

Issue: n may be too small $\rightarrow$ not enough data per "base learner" $\rightarrow$ Bagging

Bagging (Boostrap) AGGregatING

Introduced by Breiman 1996

To fit a new "base learner"

1. sample n data with replacement from the training set
2. train the "base learner" on this subset of observations

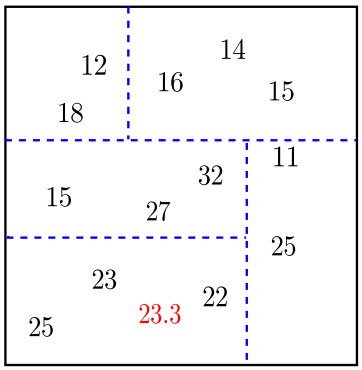
Each base learner gets $\approx 36.8\%$ of the data. Remaining points are called "out-of-bag".

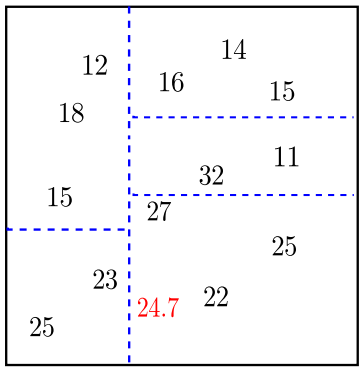
We can estimate the performance of each base learner with the out-of-bag error

Random Forests

Introduced by Breiman 2001

Idea: build many (≈ 400) random decisions trees and average their predictions.





predict $\frac{24.7+23.3}{2} = 24$

How to build uncorrelated trees?

- bagging: each tree is built over sample of training points
- random choice of the covariate to cut

👍 **Advantages**:

- No over-fitting (the more trees we build, the better)
- Easy computation of an error estimate: "out-of-bag": no-need of cross validation
- efficient for small data sets n

👎 **Drawbacks**: computational cost, black box

Variable selection with random forests

Random forests is a powerful tool to order explanatory variables by predictive importance.

First, we build the forest and compute E its "out-of-bag" error.

For each variable X_i, we compute its importance as follows

- randomly permute the values of X_i among training data
- update the "out-of-bag" error E_i
- get the importance of X_i given by $E_i - E$

Outline

Deep learning

Successful application domains: Image (object recognition), Audio (speech recognition), Text (parsing)

What is it used for?

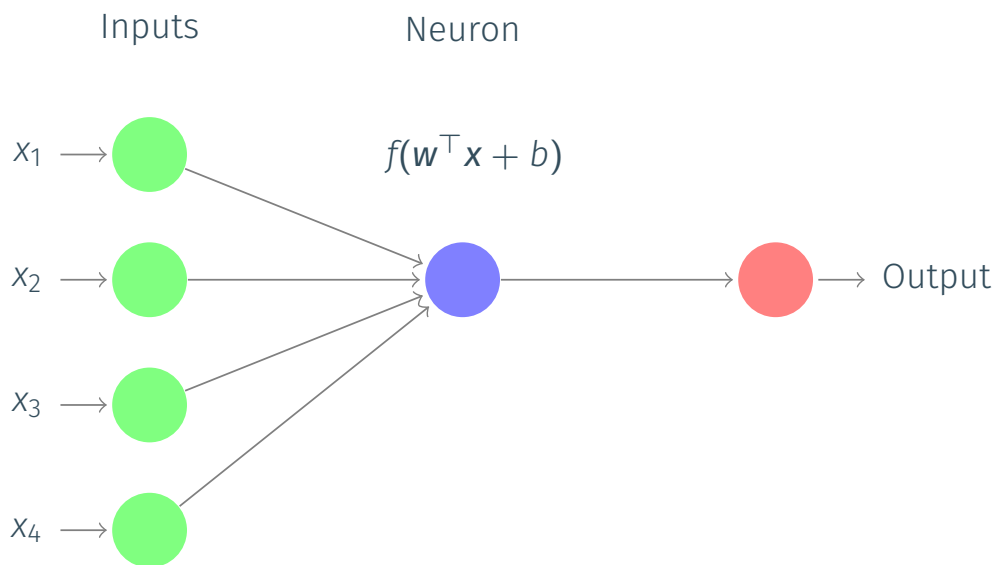
- Prediction: regression, classification,
- Generation: denoising, reconstruction of partial/missing data, generation of new data

What is it?

- Models with graphs structure (networks) with multiple layers (deep)
- Typically non-linear models

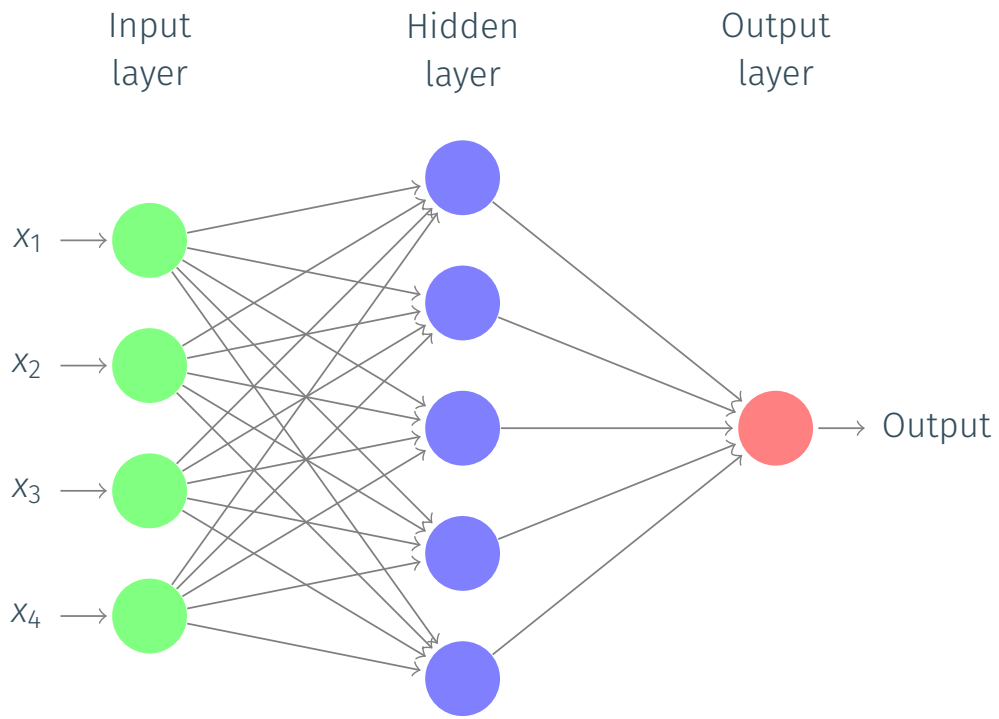
Deep neural network

- A **neuron** is a non-linear transformation of a linear combination of inputs.
- A **column of neurons** taking the same input $\boldsymbol{x}$ forms a new layer



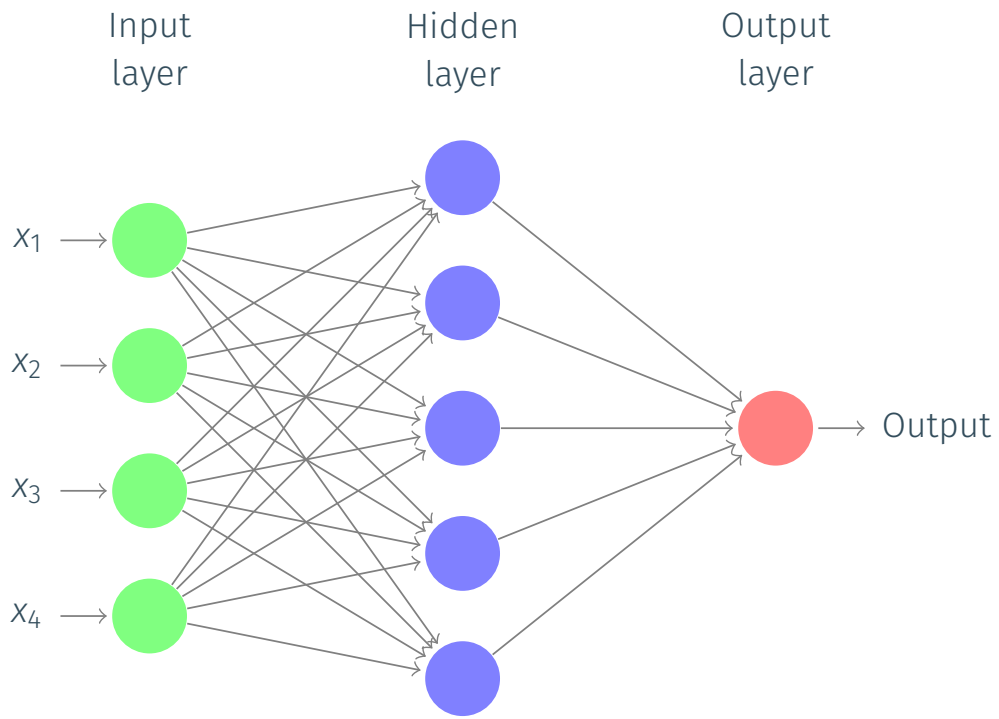
Deep neural network

- A **neuron** is a non-linear transformation of a linear combination of inputs.
- A **column of neurons** taking the same input $\boldsymbol{x}$ forms a new layer



Deep neural network

- A **neuron** is a non-linear transformation of a linear combination of inputs.
- A **column of neurons** taking the same input $\boldsymbol{x}$ forms a new layer



Training a neural networks: backpropagation (gradient descent using $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$).

Avoid over-fitting: dropout [Hinton et al. 2012]

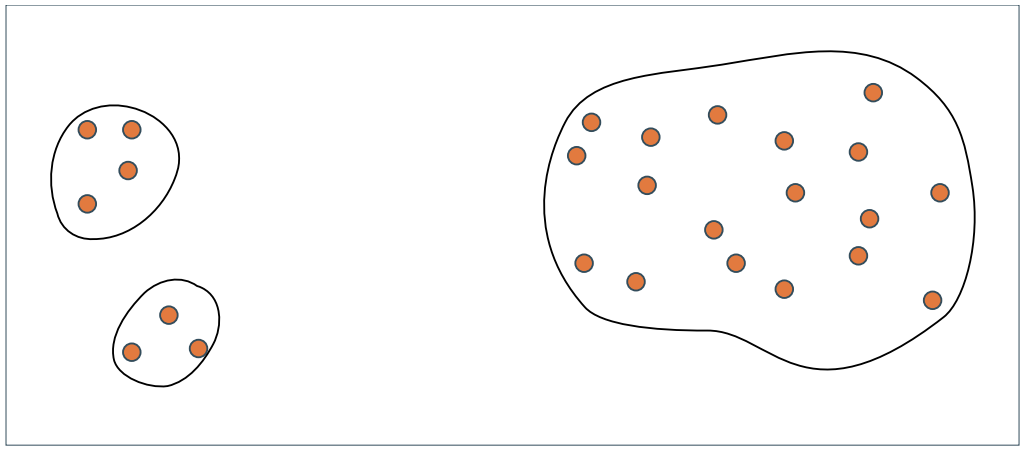
Build data-specific models: convolutional neural networks [LeCun et al. 1998]

Unsupervised learning

Outline

Clustering

- **Idea:** group together similar instances
- Requires data but no labels
- Useful when you don't know what you are looking for



The similarity is measured by a metric (ex: $\|x - y\|_2^2$).

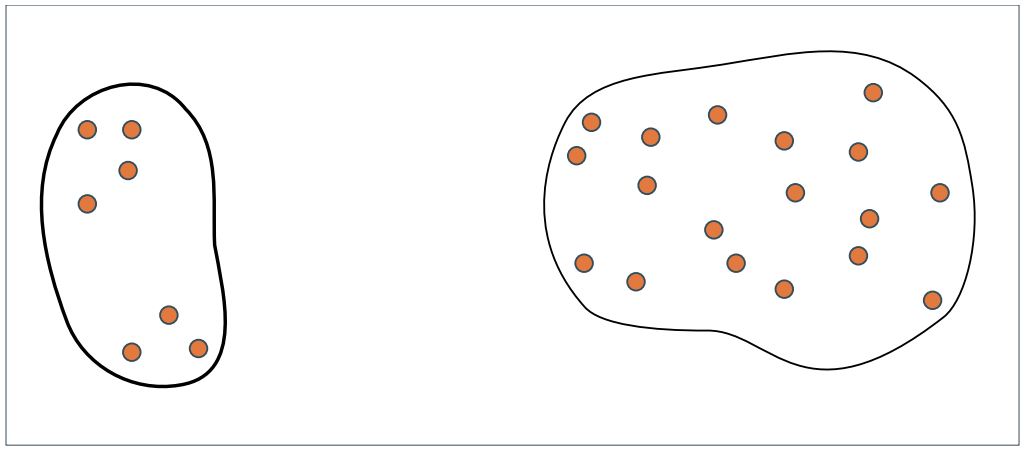
The results crucially depends on the metric choice: depends on data.

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)

Clustering

- **Idea:** group together similar instances
- Requires data but no labels
- Useful when you don't know what you are looking for



The similarity is measured by a metric (ex: $\|x - y\|_2^2$).

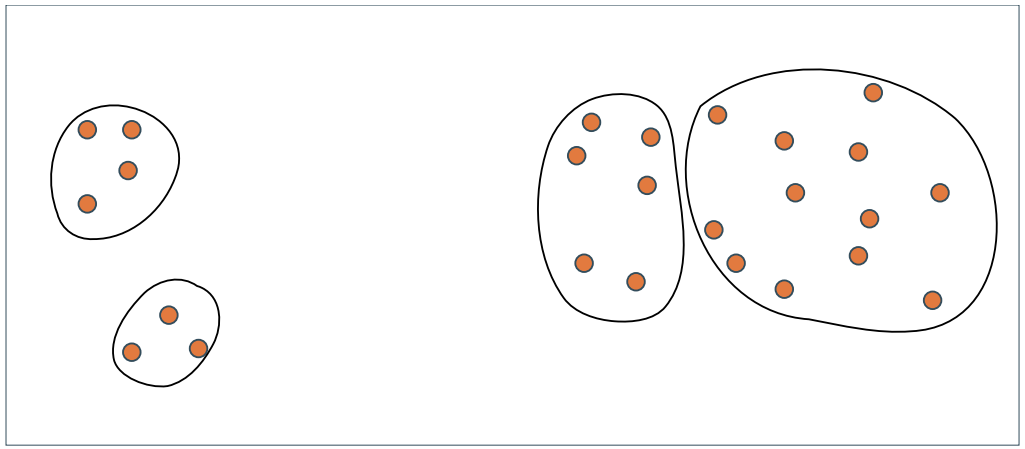
The results crucially depends on the metric choice: depends on data.

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)

Clustering

- **Idea:** group together similar instances
- Requires data but no labels
- Useful when you don't know what you are looking for



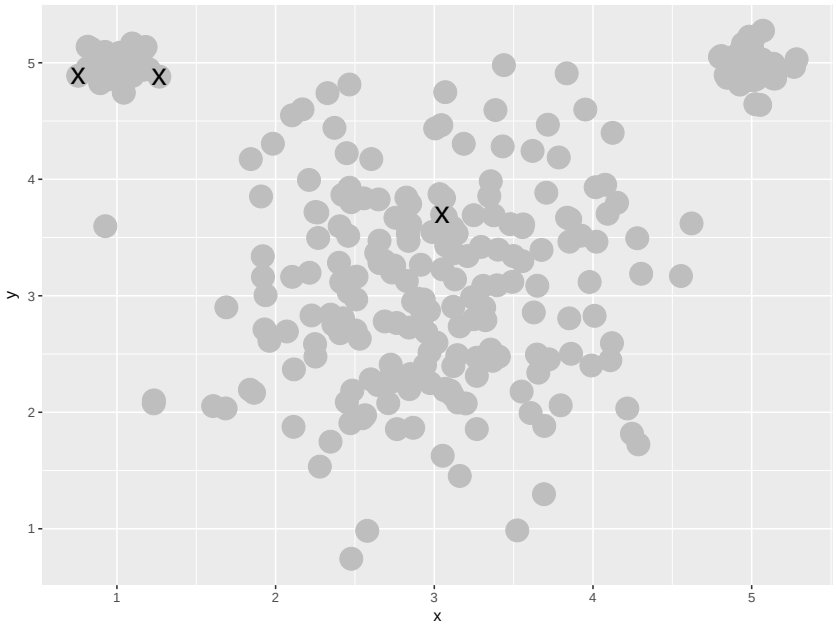
The similarity is measured by a metric (ex: $\|x - y\|_2^2$).

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

Types of clustering algorithms:

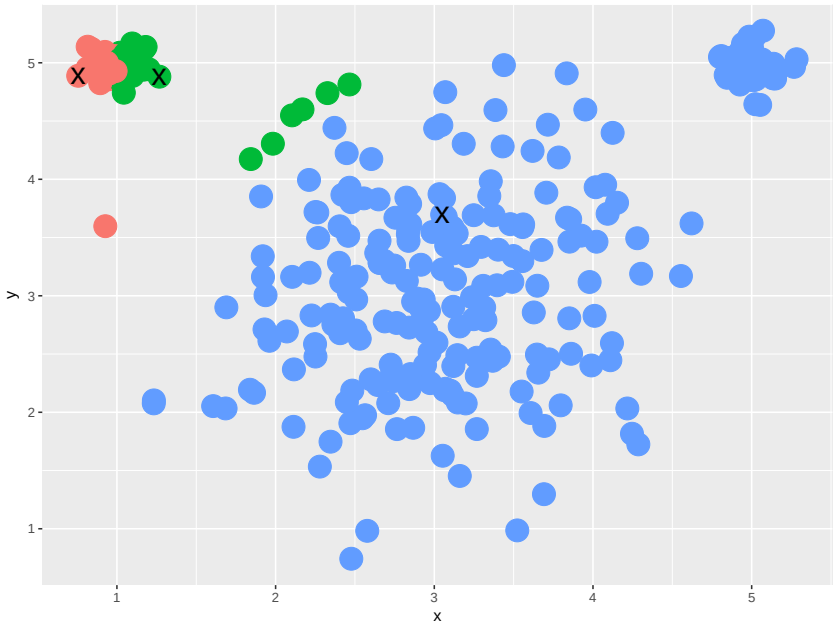
- model based clustering (mixture of Gaussian)
- hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Flat clustering: no hierarchy (k-means, spectral clustering)

K-means



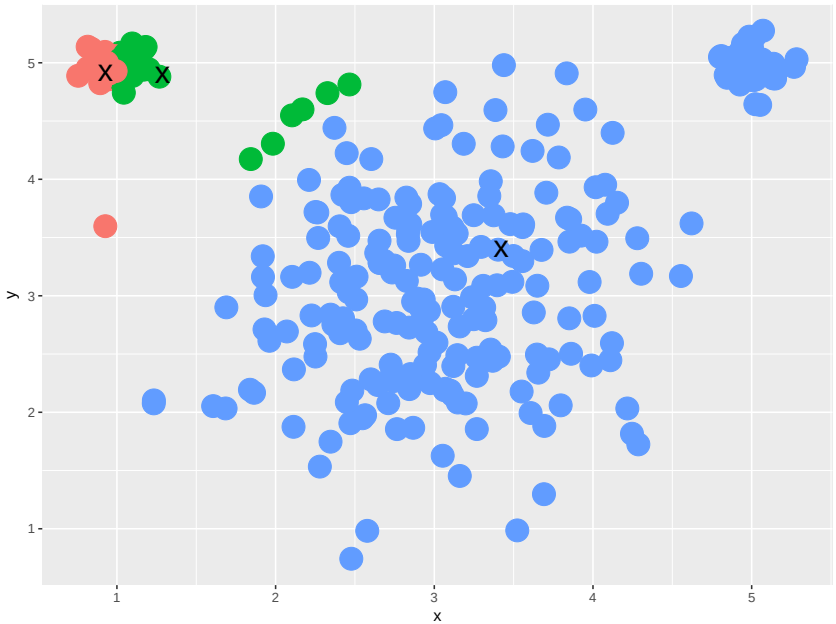
- Initialization: sample K points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- Stop when no point's assignment change.

K-means



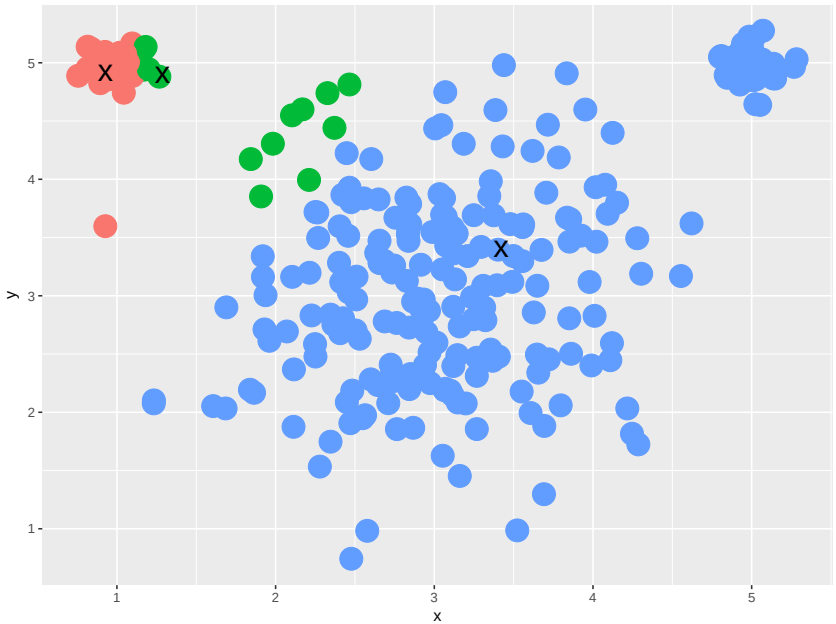
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



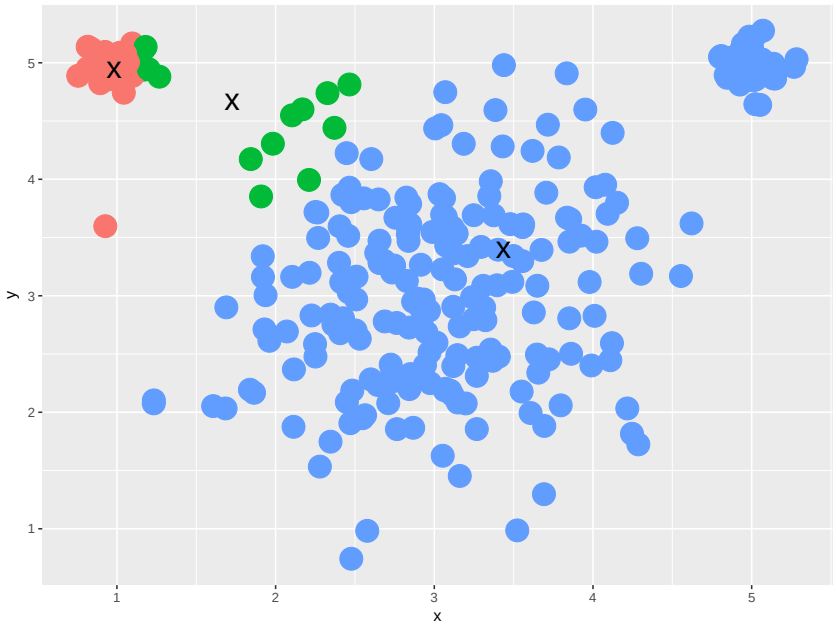
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



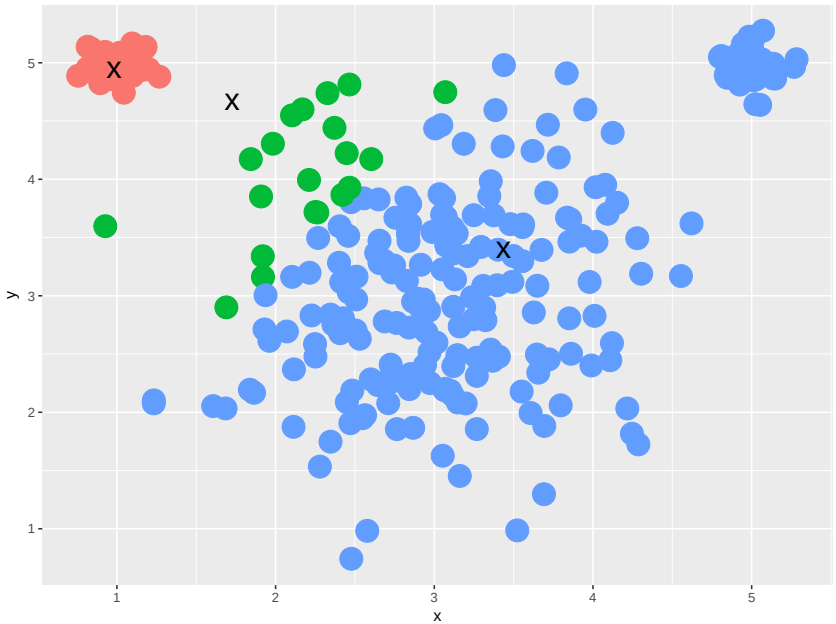
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



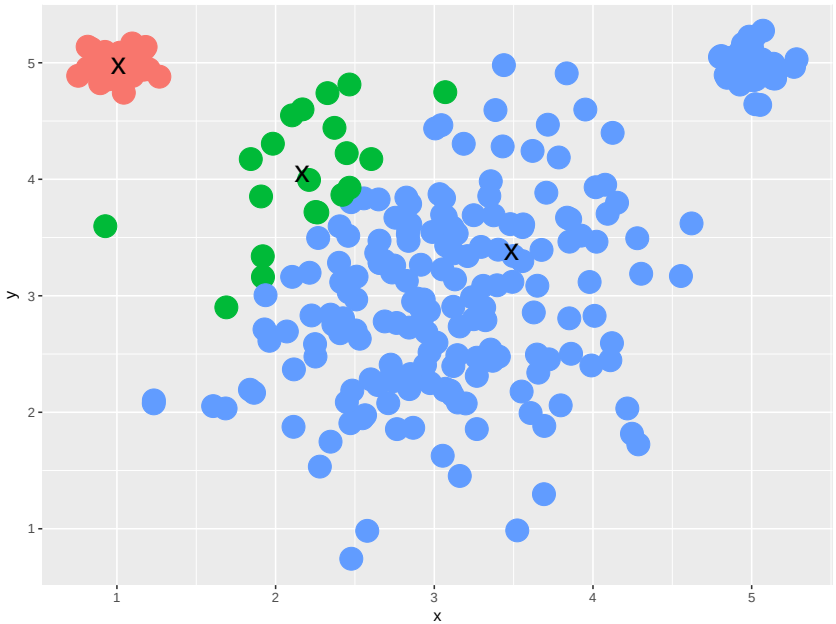
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



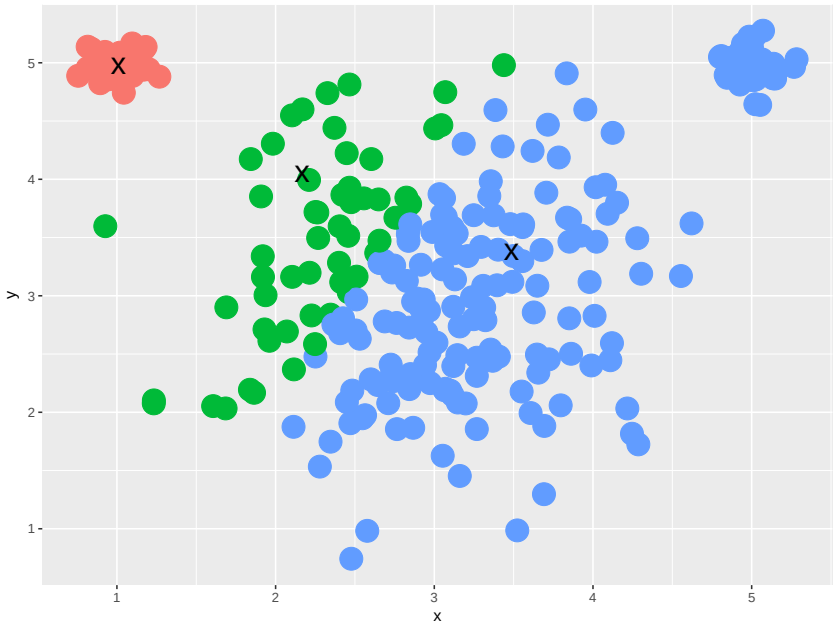
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



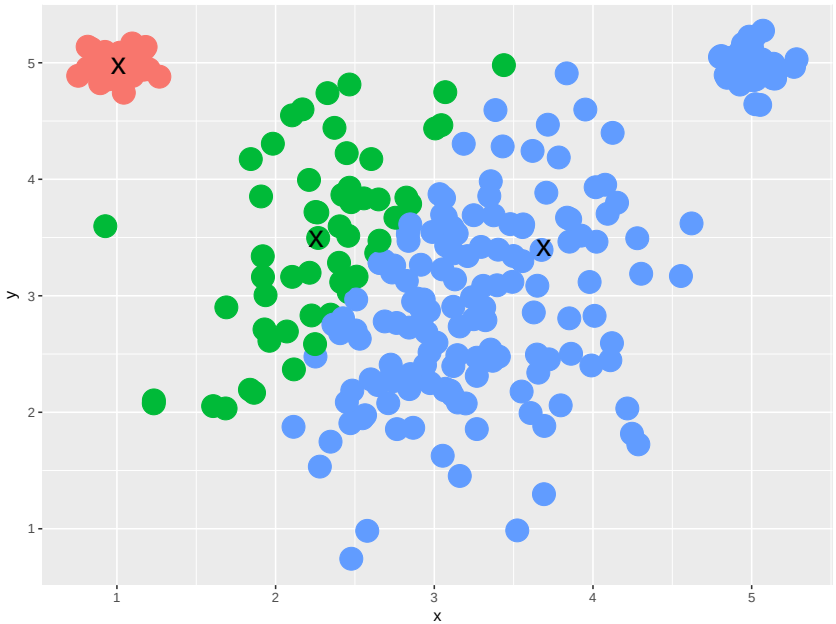
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



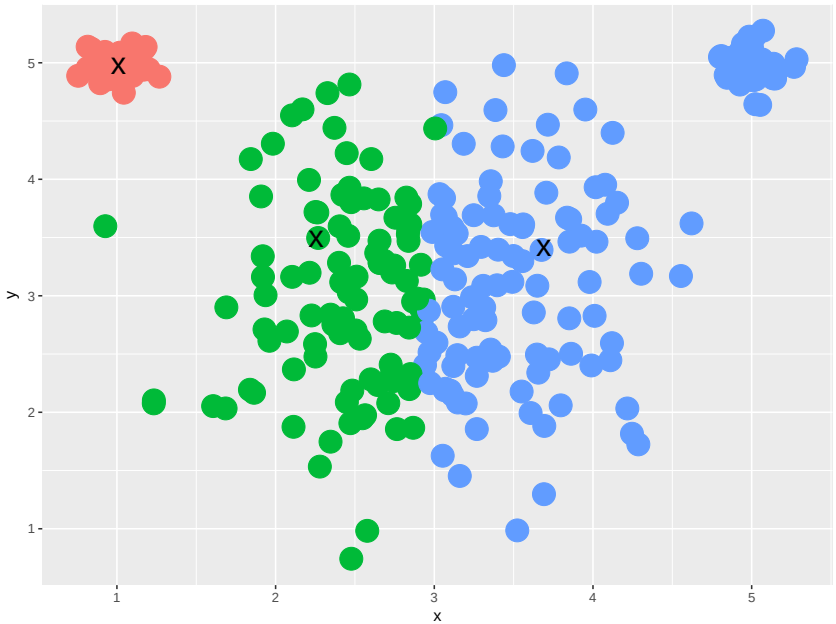
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



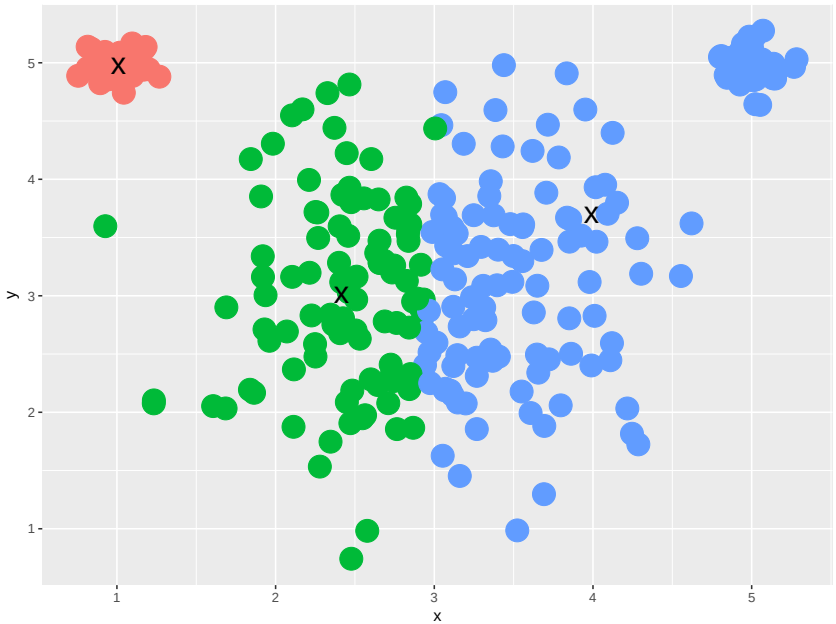
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



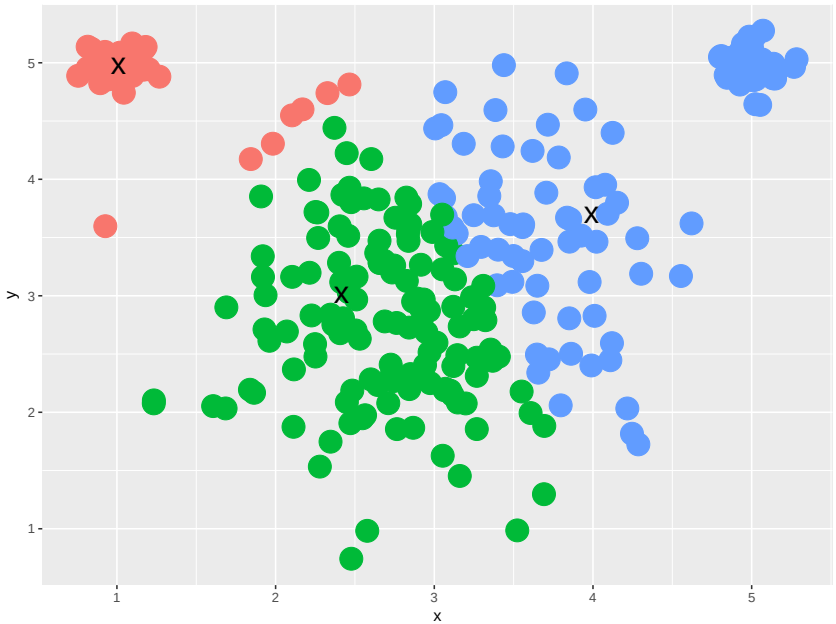
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



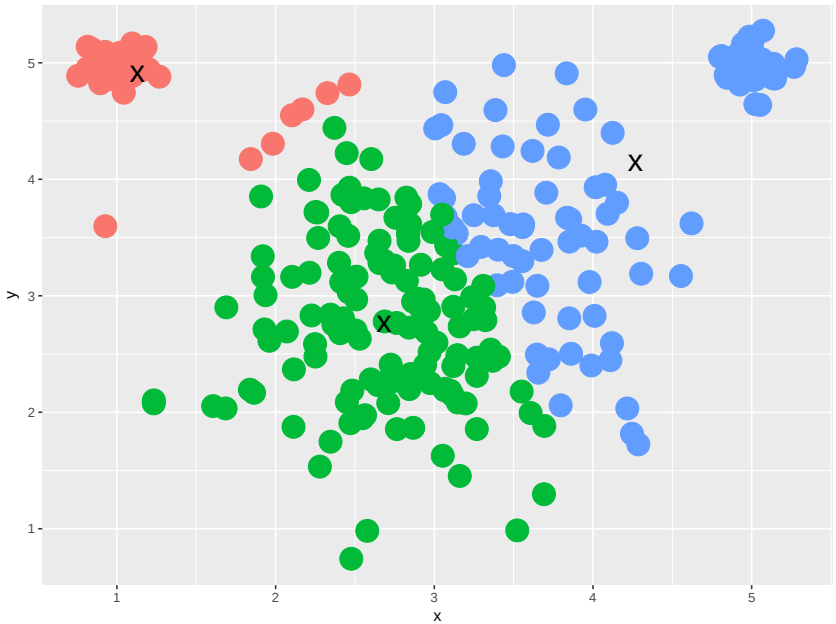
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



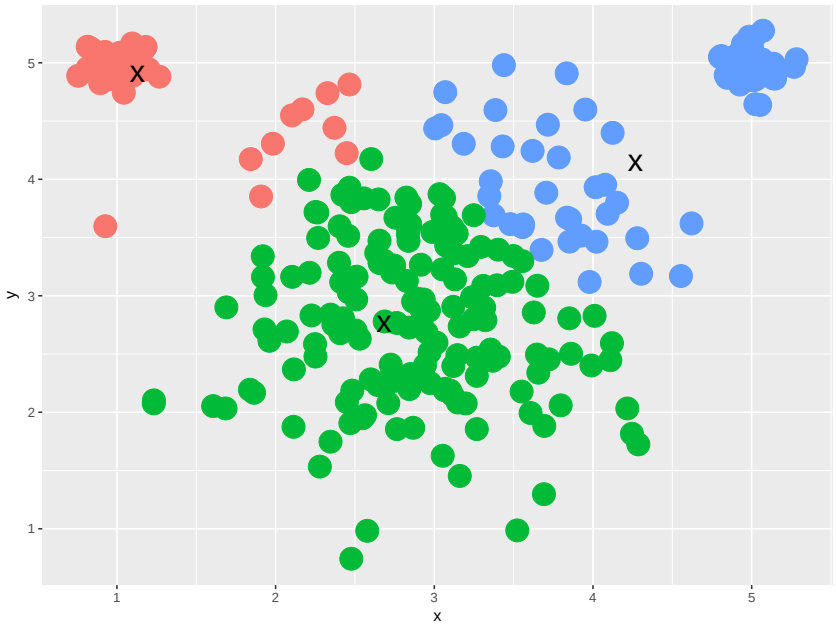
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



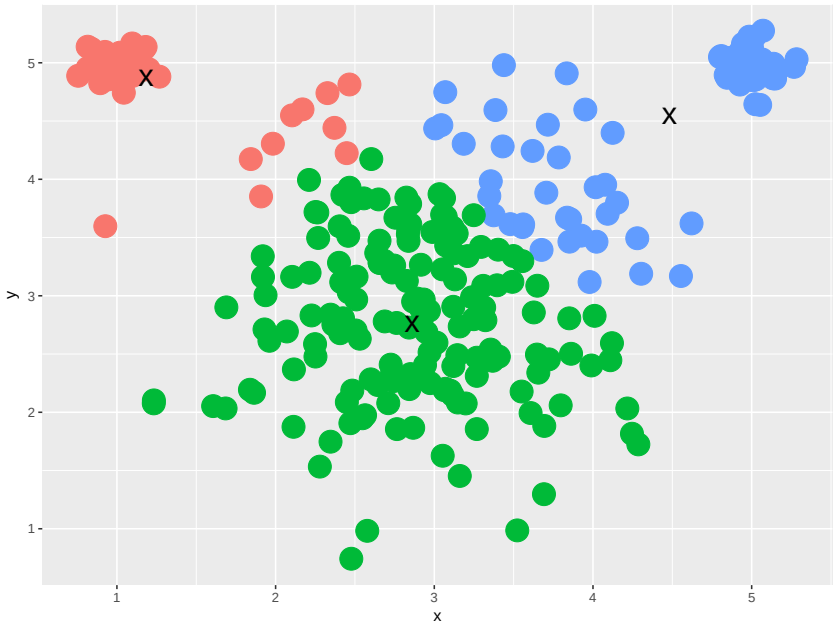
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



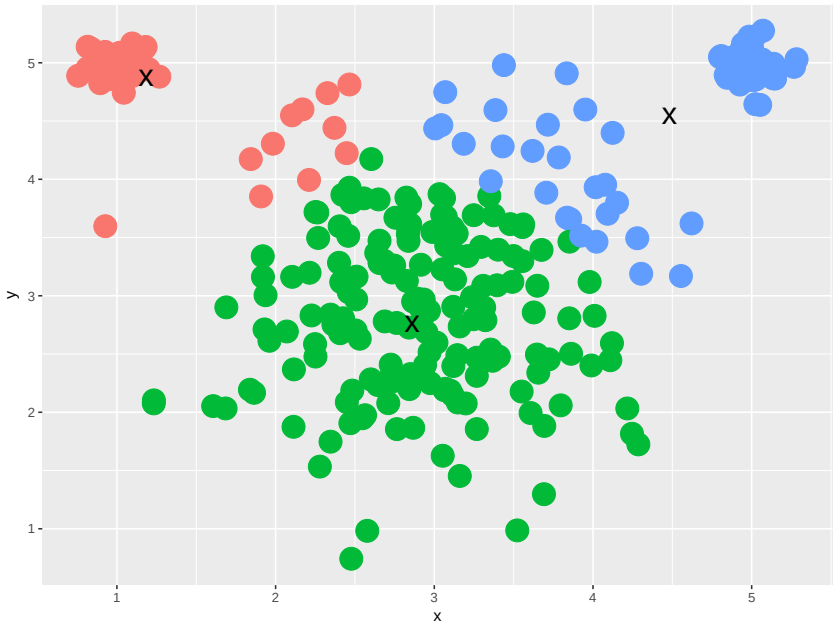
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



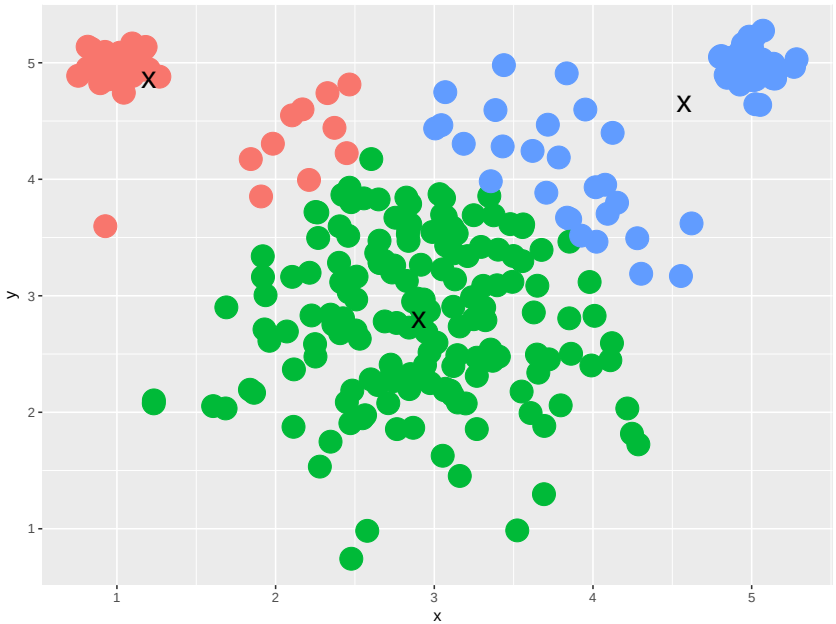
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



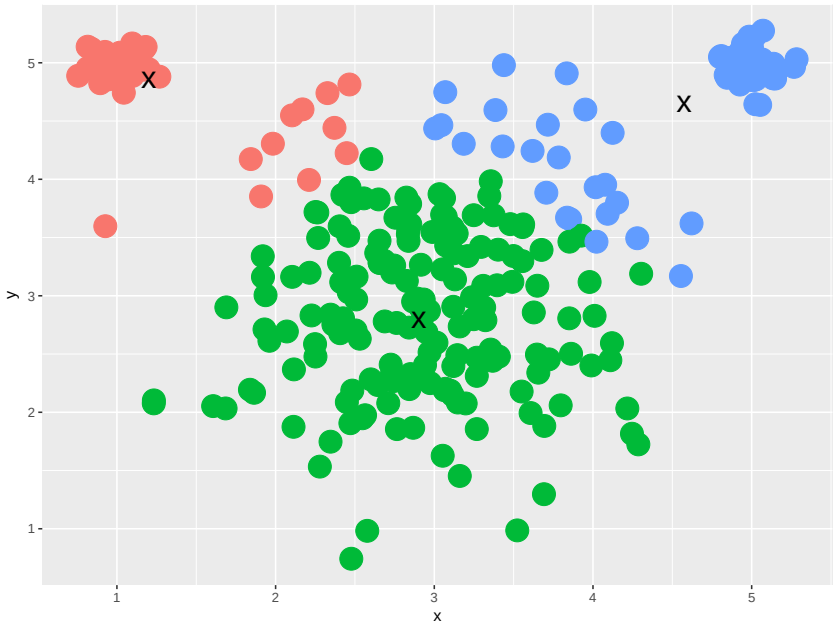
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



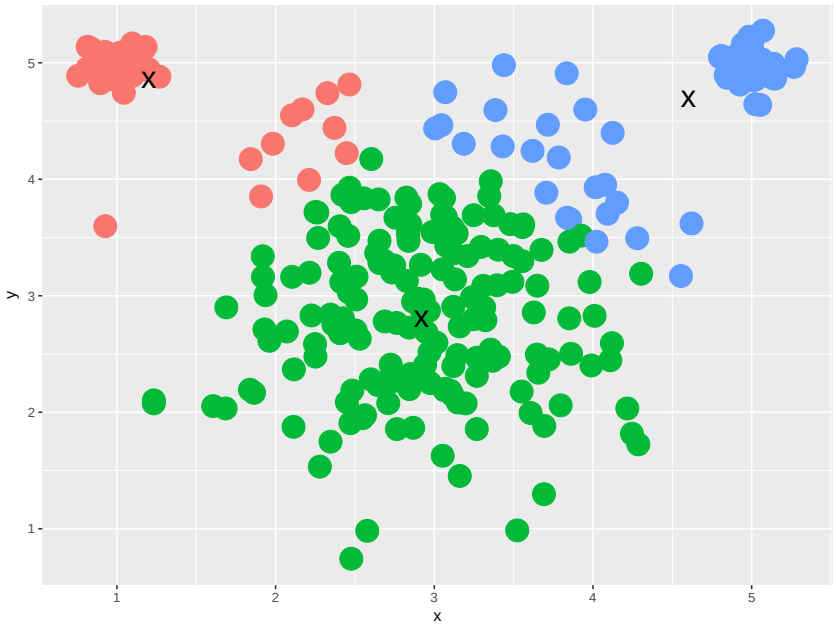
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



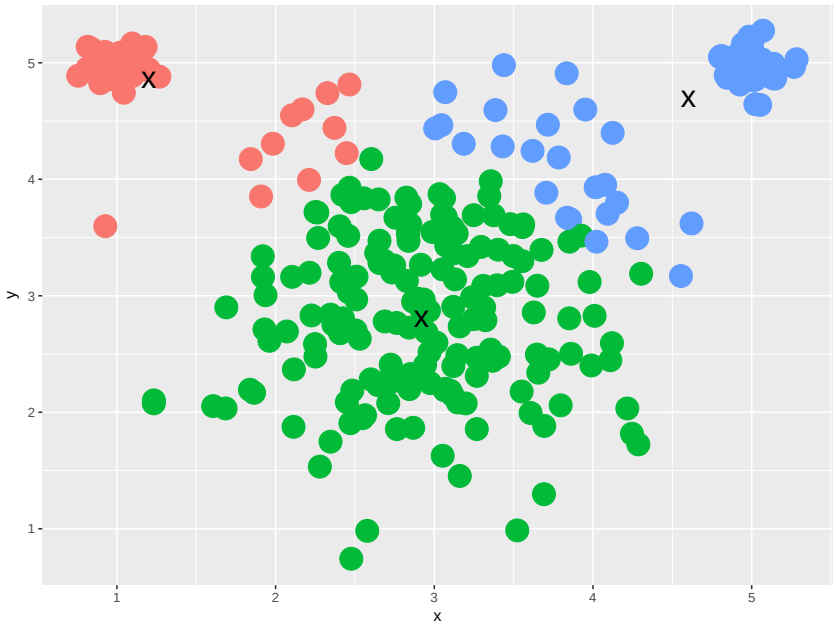
- **Initialization**: sample *K* points as cluster centers
- Alternate:
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

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.

Outline

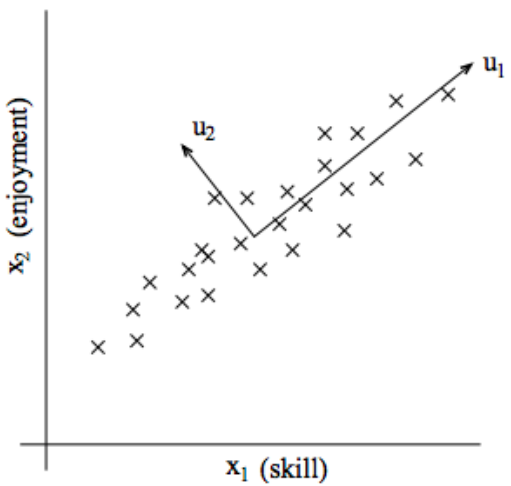
Principal component analysis

Assume that you have a data matrix (with column-wise zero empirical mean)

$$X := \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ \vdots & \vdots & \dots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}$$

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



PCA reduces the p dimensions of the data set X down to k principal components.

Principal component analysis

Assume that you have a data matrix (with column-wise zero empirical mean)

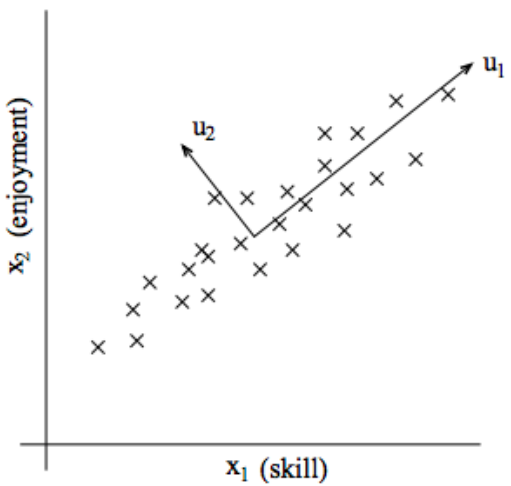
$$X := \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ \vdots & \vdots & \dots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}$$

How does it work?

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

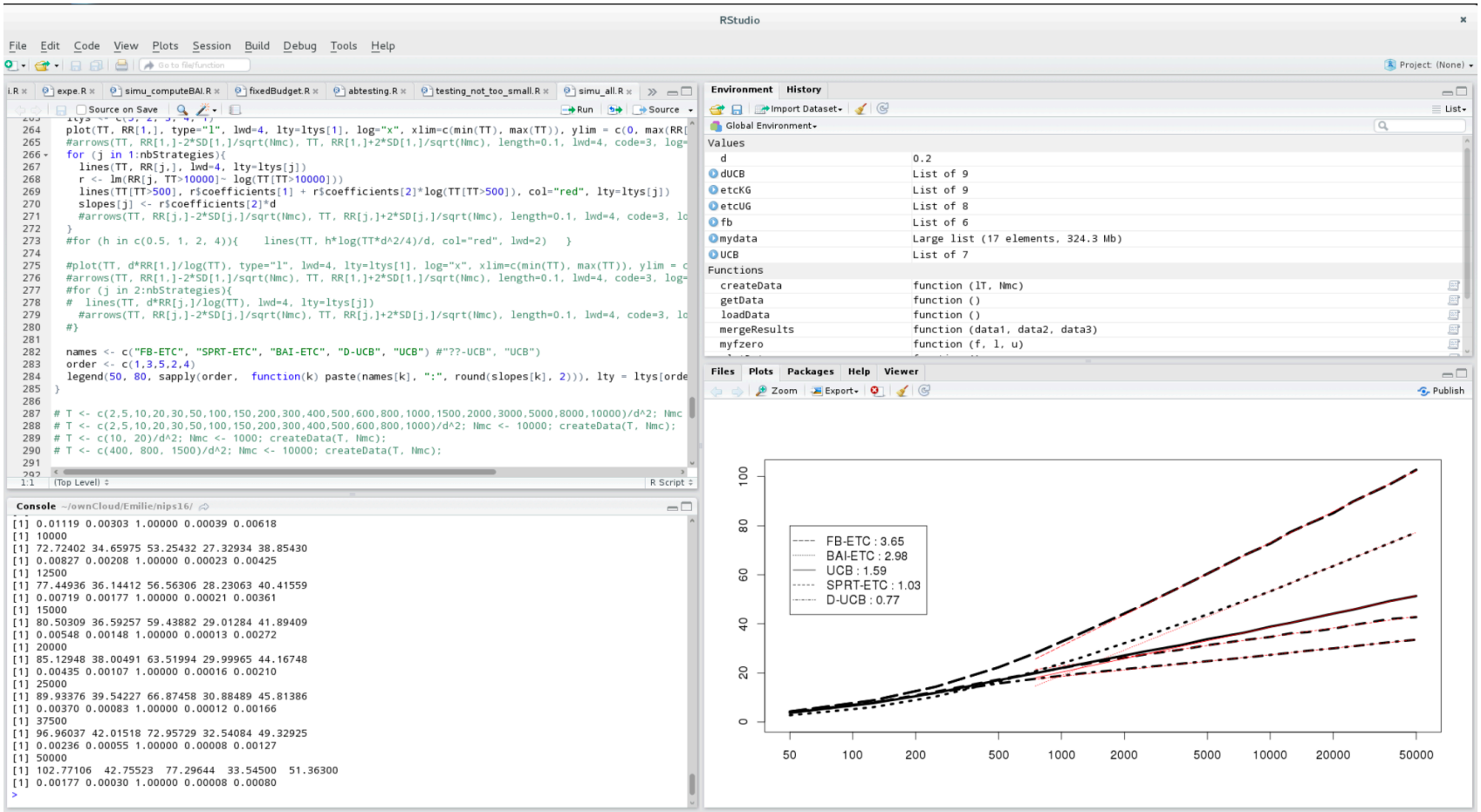
$$u_1 := \underset{\|u\|=1}{\arg\max} \|X^\top \boldsymbol{u}\|^2 = u^\top X^\top X u$$

   $\Rightarrow$ this is the principal eigenvector of $X^\top X$.
2. More generally, if we wish a k-dimensional subspace we choose $u_1, \dots, u_k$ the top k eigenvectors of $X^\top X$.
3. The u_i form a new orthogonal basis of the data



Tools for Machine Learning

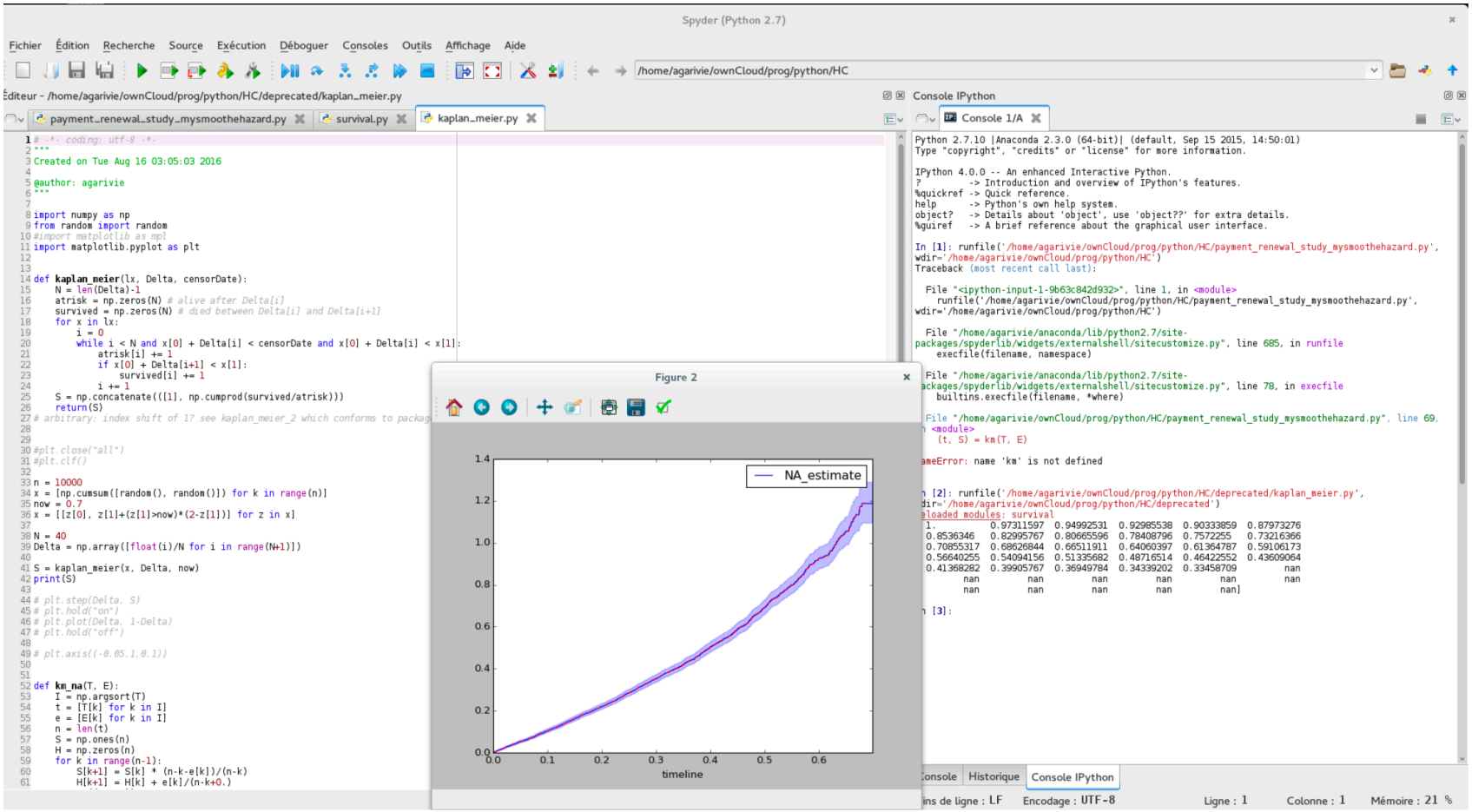
R



broadly used by the statistic community, huge library, well-known

slow, less used by computer scientists, old language

python



general-purpose language, growing fast

not (yet?) so good for statistical analysis (smaller library)

References

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

L. Breiman. “Bagging predictor”. In: *Machine Learning* 24.2 (1996), pp. 123–140.

L. Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.

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.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

Wikipedia. The free encyclopedia.