Introduction to machine learning and validation

Olivier Colliot

ARAMIS Lab - www.aramislab.fr
CNRS, Inria, Inserm, Sorbonne Université, AP-HP, Institut du Cerveau
Centre Inria de Paris

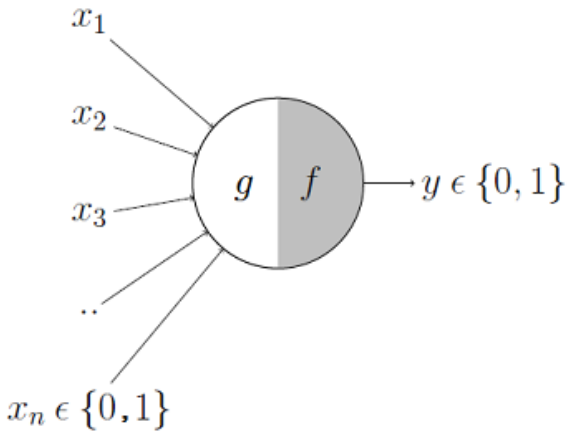
History and terminology

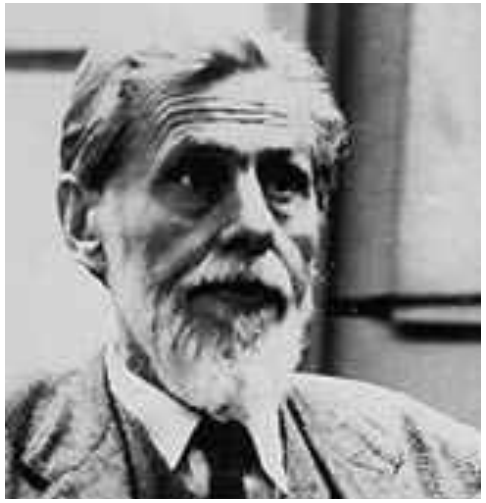
A bit of history

When did AI start?

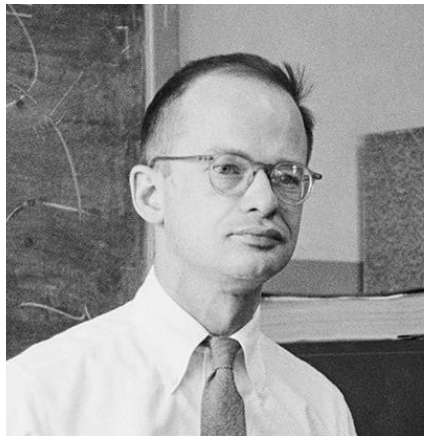
1943: artificial neuron model

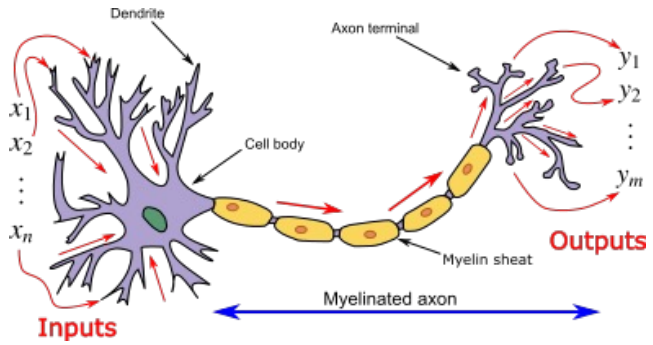
- **McCulloch-Pitt neuron**
  - Artificial neuron model





Warren McCulloch





Source: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

1950: the Turing test

- **A. Turing:** mathematician, founder of computer science
- A test for deciding if a machine can think

Alan Turing

MIND

A QUARTERLY REVIEW

OF

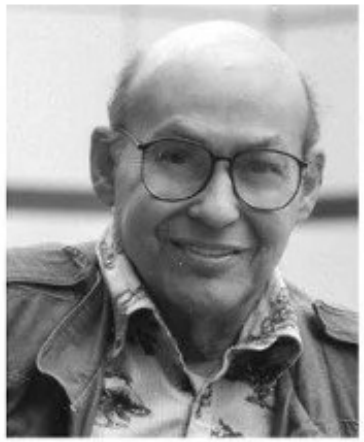
PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

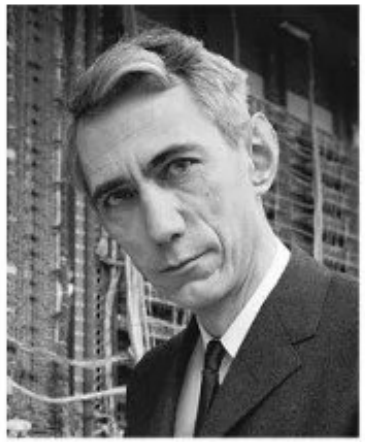
By A. M. Turing

1956: the Dartmouth workshop

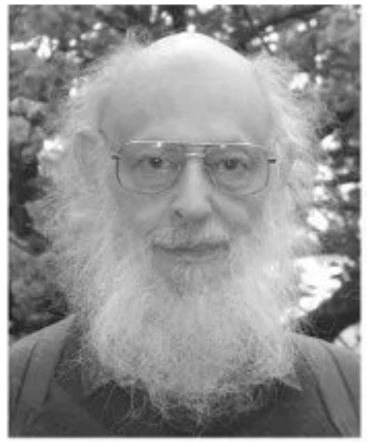
John MacCarthy



Marvin Minsky



Claude Shannon

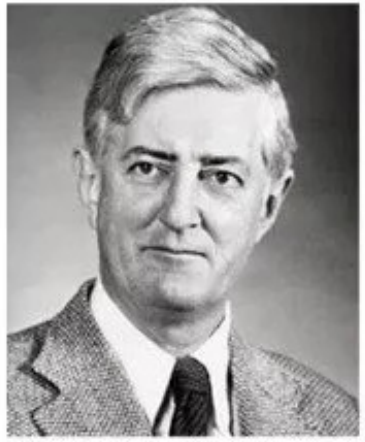


Ray Solomonoff

Alan Newell

Herbert Simon

Arthur Samuel



Oliver Selfridge

Nathaniel Rochester

Trenchard More

1956: the Dartmouth workshop

We propose that a **2 month, 10 man study of artificial intelligence be carried out during the summer of 1956** at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the **conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it**. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. **We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.**

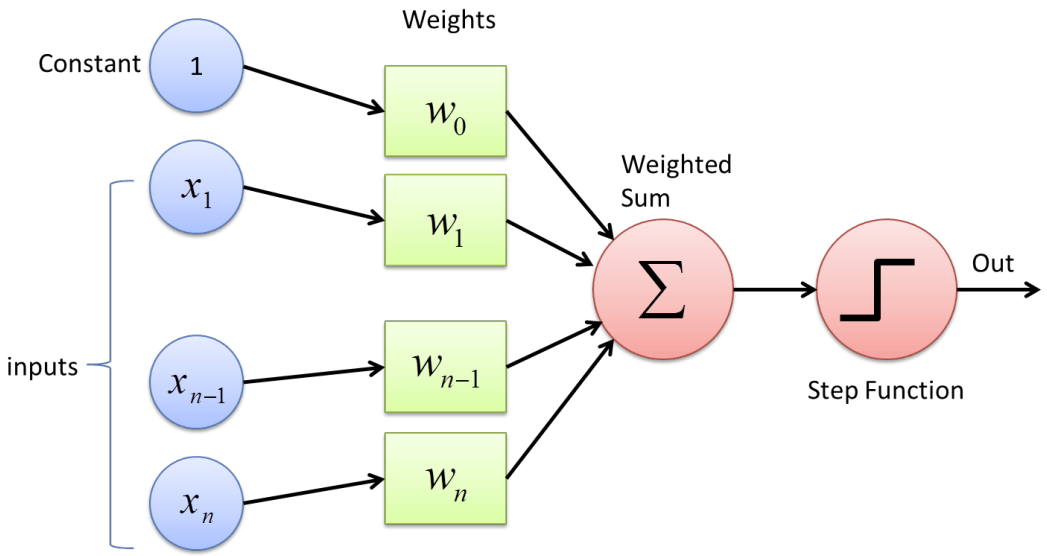
Source: Russell SJ, and Norvig P, Artificial intelligence: a modern approach, 2016 (Section 1.3).

1956: the Dartmouth workshop

- Coined the term **"artificial intelligence"**
- No major breakthrough
- Presentation of the **Logic Theorist**
  - First reasoning program
  - **Symbolic AI** is born

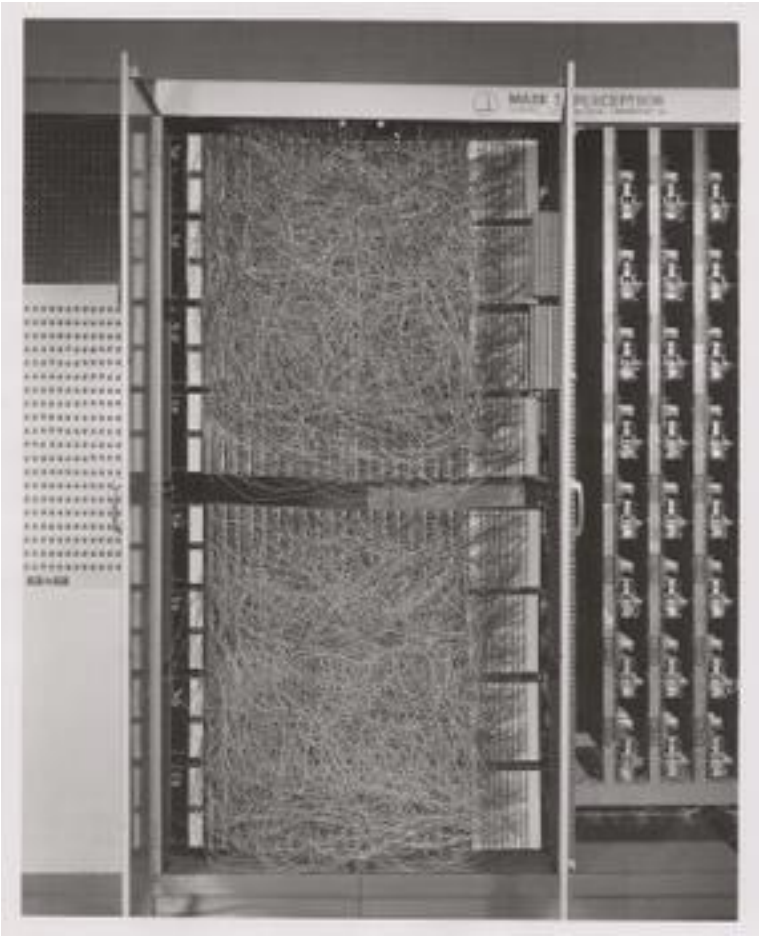
1958: the Perceptron

- **Artificial neural network**
- Could recognize letters and numbers



Frank Rosenblatt

1958: the Perceptron



Mark I Perceptron machine, hardware implementation of the perceptron algorithm. **It was connected to a camera** with 20×20 cadmium sulfide photocells to make a 400-pixel image. To the right, arrays of **potentiometers** that implemented the adaptive weights.

Funded by US Navy

https://en.wikipedia.org/wiki/Perceptron

1958: the Perceptron

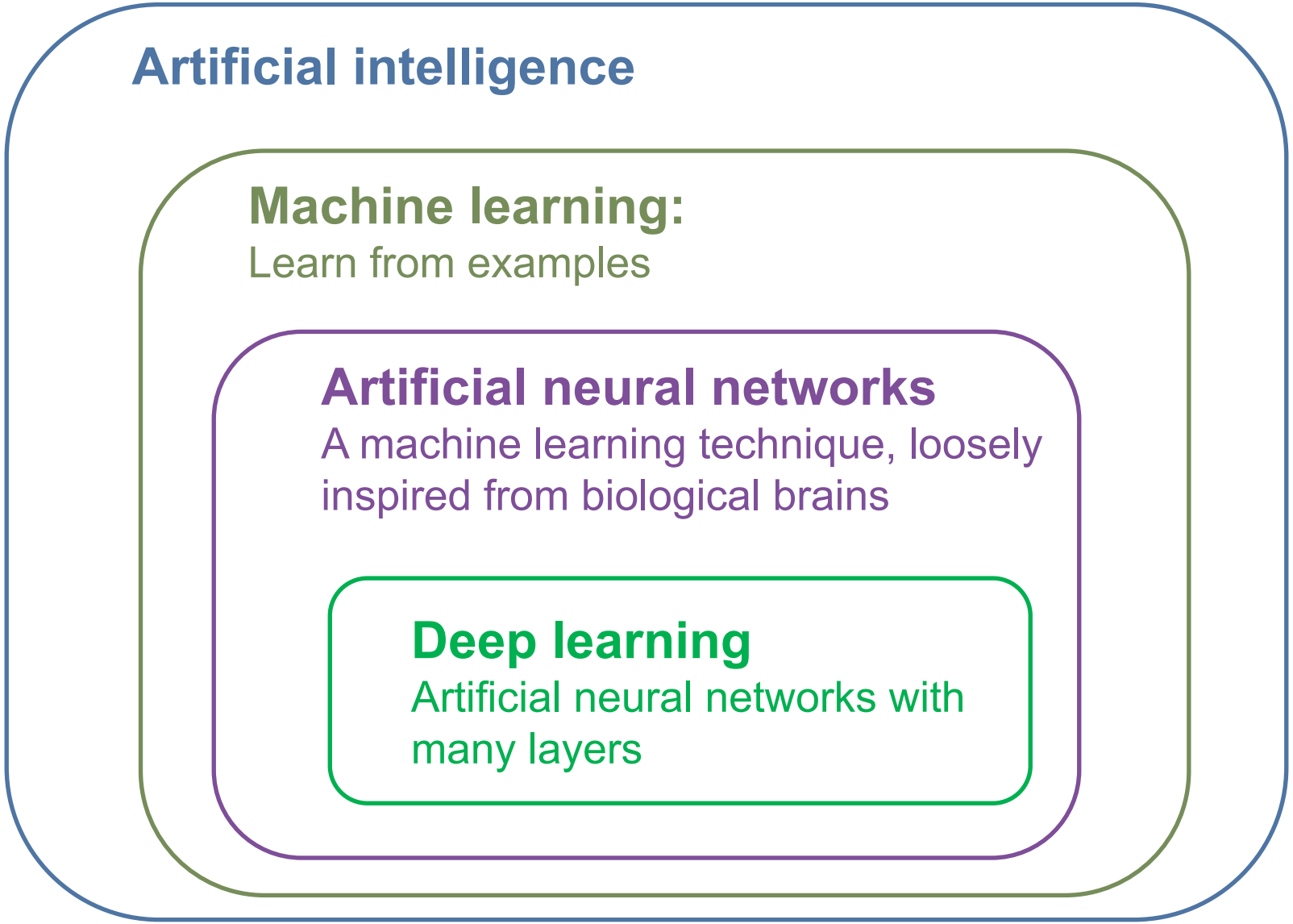
The New York Times

WASHINGTON, July 7 (UPI) -- The Navy revealed the embryo of an electronic computer today that it expects **will be able to walk, talk, see, write, reproduce itself and be conscious of its existence**.

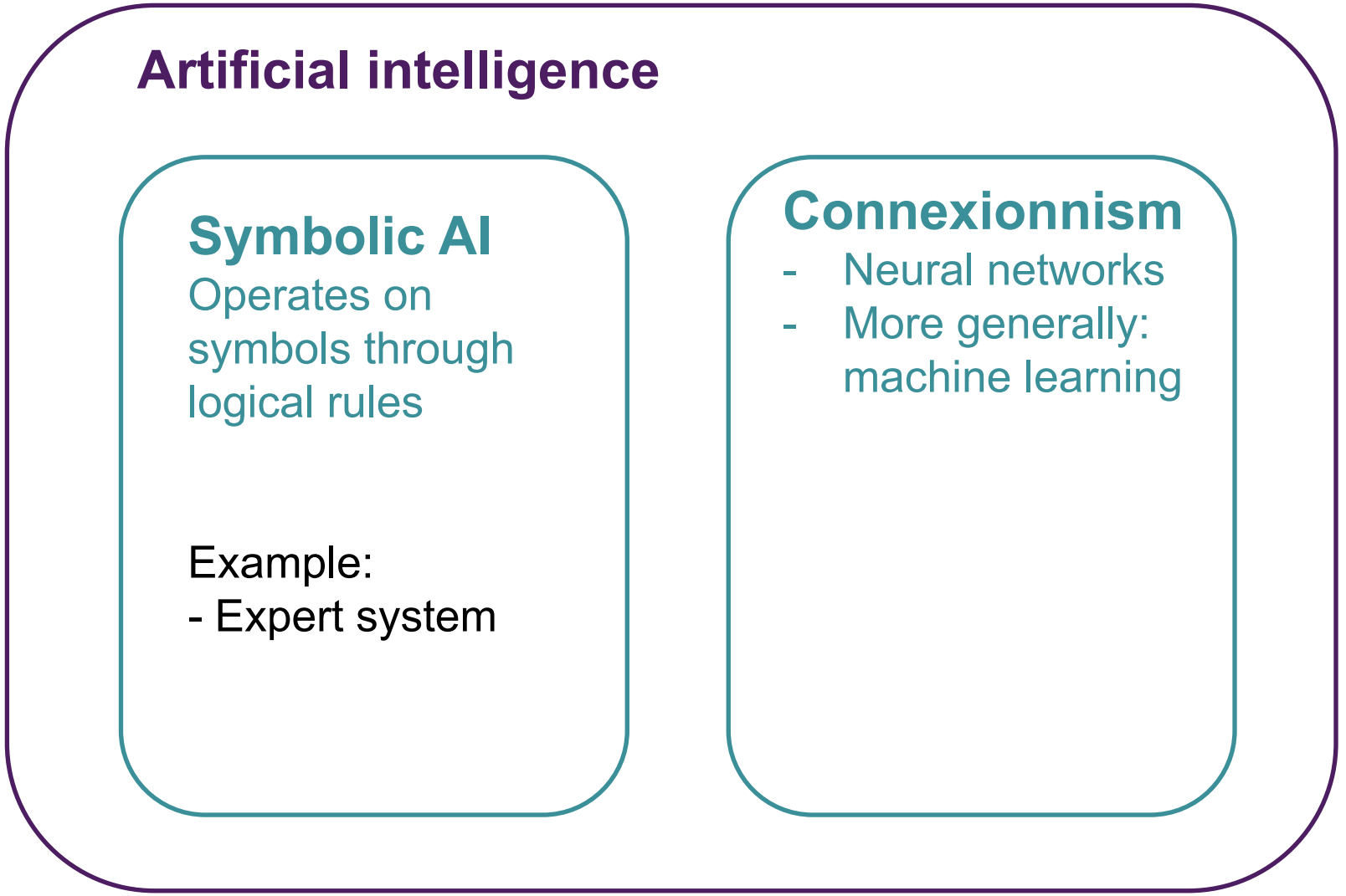
Terminology

AI, machine learning, deep learning, what is the difference?

Terminology



Terminology: the two main families of AI



Terminology: the two main families of AI

A fun read for your spare time

LA REVANCHE DES NEURONES

L'invention des machines inductives
et la controverse de l'intelligence artificielle

Dominique CARDON
Jean-Philippe COINTET
Antoine MAZIÈRES

La Découverte | « Réseaux »
2018/5 n° 211 | pages 173 à 220
ISSN 0751-7971
ISBN 9782348040689

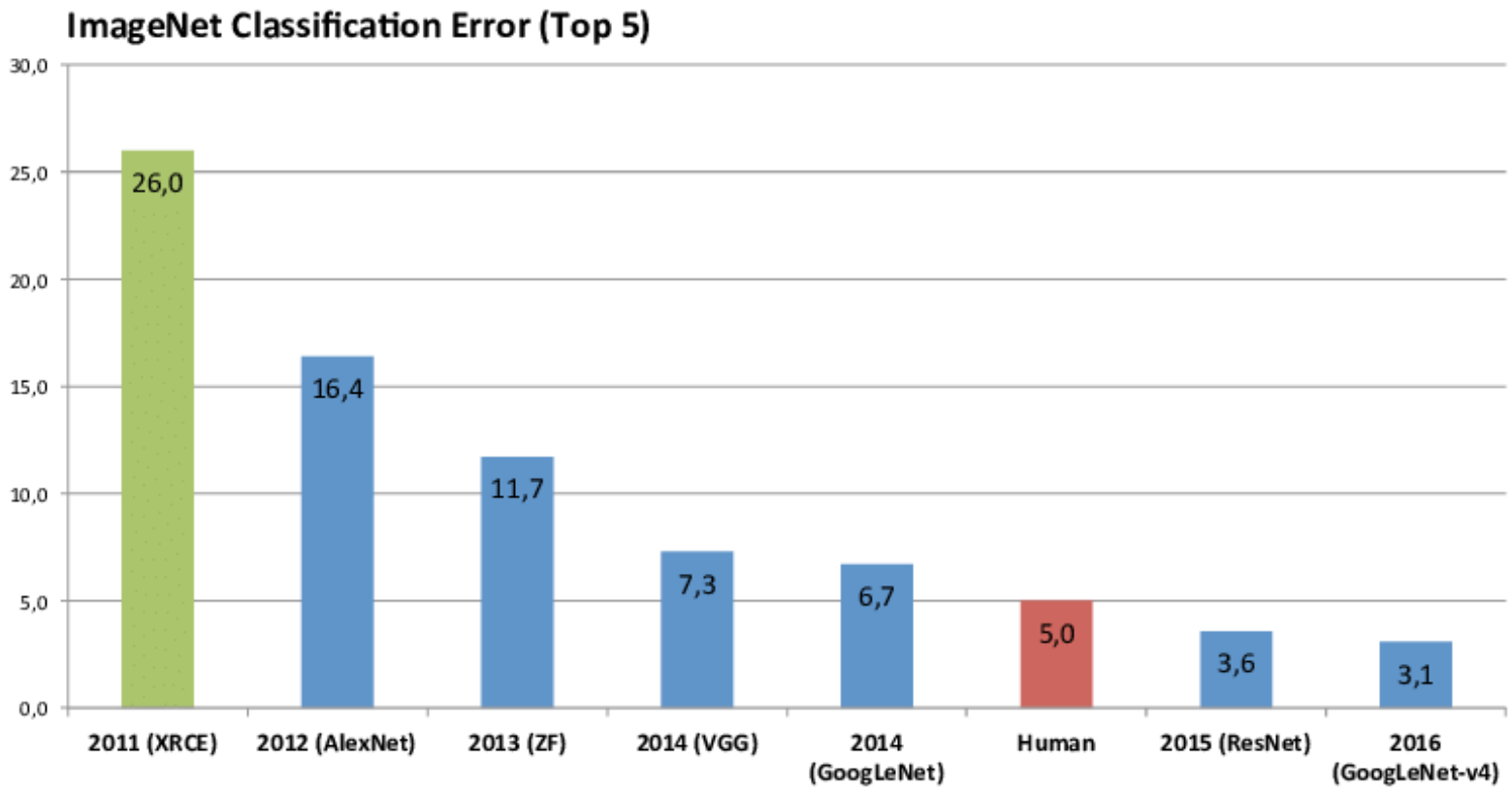
From (Cardon et al, Réseaux, 2018)

Since when everybody talks about deep learning?

ImageNet Challenge

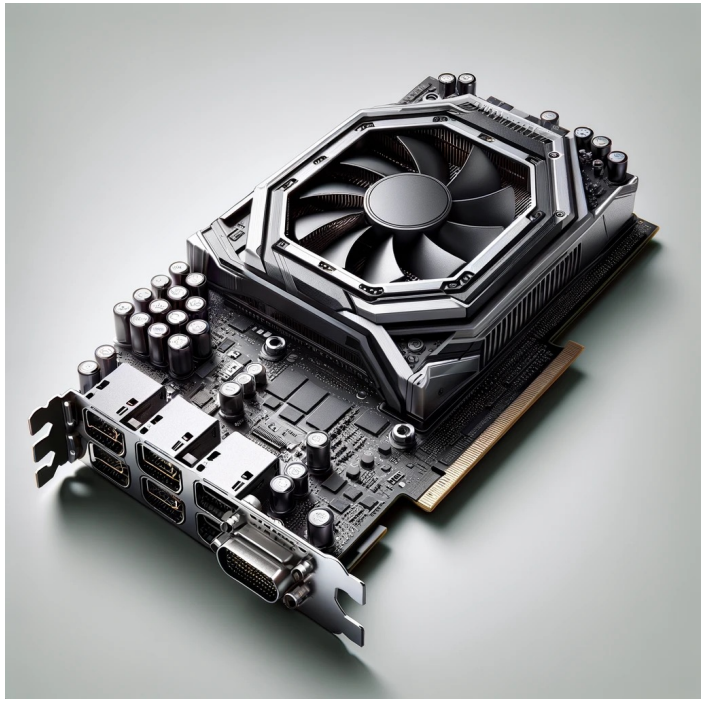
14 millions labelled images, 20000 categories

2012



Summary

- Machine learning is **one** approach to AI
- Deep learning is **<u>one</u>** machine learning technique
  - It has obtained impressive results on some types of data
- Basic principles are relatively old (1980s, 1990s)
- Obtained impressive results recently (after 2012):
  - Increase in the amount of data
  - Increase in computing power
    - Computations using GPU (graphical processing unit)



DALL-E is better at depicting GPUs than "big data"…

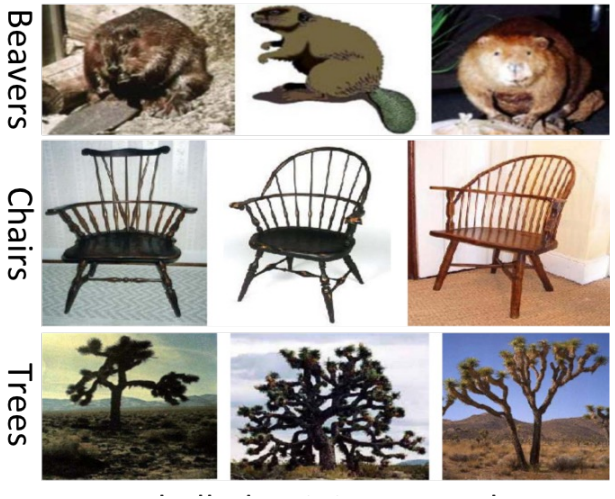
This session

- **We will focus on machine learning (ML)**
  - Including deep learning since it is a subcategory of ML
- **Basics of ML**
  - A "procedural" view: what does the machine compute?
  - More conceptual view: risk, overfitting, model selection
- **Special focus on validation**

Machine learning basics

ML Basics

Training samples: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$



Learning

Input

x

f

$y = f(x)$

Beaver

Output

y

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Loss function:
measures error in prediction

ML Basics

Training samples: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Learning

Input

x

f

$y = f(x)$

Beaver

Output

y

y is a class

Classification task

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

ML Basics

Training samples: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Learning

Input

x

f

$y = f(x)$

Beaver weight in kg

Output

y

y is a number

Regression task

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

ML Basics

Input

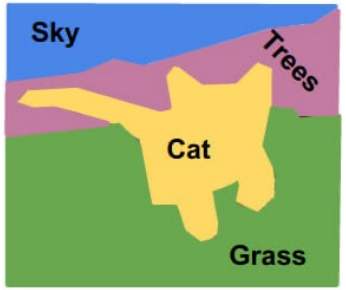
x

Training samples: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Learning

f

$y = f(x)$



Output

y

y is a labelled image

Segmentation task

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

ML Basics

Input

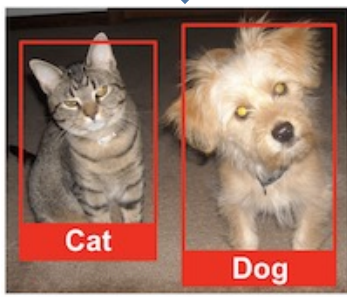
x

Training samples: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Learning

f

$y = f(x)$



Output

y

Detection task

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Training samples: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Loss function:
measures error in prediction

How far is my prediction from the truth

Example

If we are dealing with real numbers

Least squares loss

$$\ell(y, f(x)) = (y - f(x))^2$$

Remember? From linear regression
(Legendre 1805, Gauss 1809)

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Loss function:
measures error in prediction

How far is my prediction from the truth

Example

If we are dealing with real numbers

Least squares loss

$$\ell(y, f(x)) = (y - f(x))^2$$

Of course the output will not always be real number

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Averaging across training samples

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Trying to minimize the error

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

I want *f* such that the average loss is minimum

ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

The set of admissible functions f

sklearn.linear_model.LinearRegression

Example

The set of functions of the form

$$f : \left| \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & f(x) = ax + b \end{array} \right.$$

Univariate linear regression

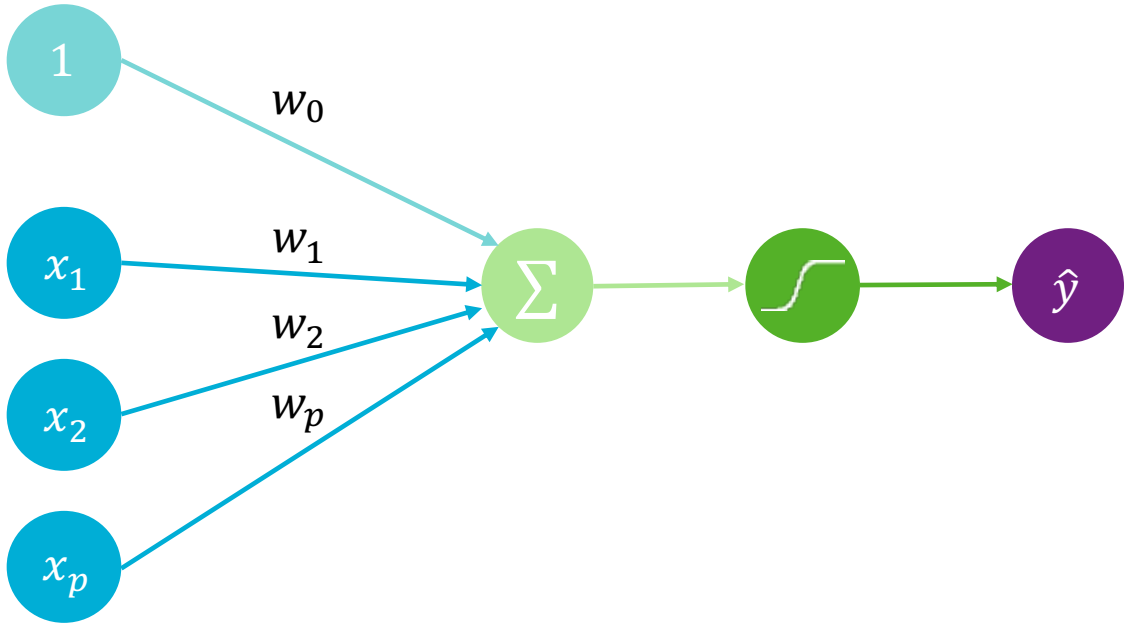
ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

The set of admissible functions *f*

Example

The set of neural networks with a given architecture



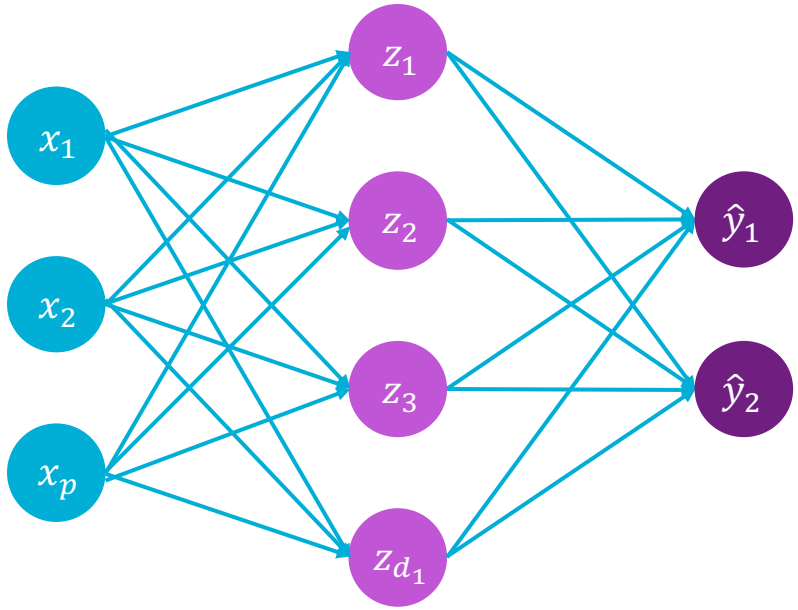
ML Basics

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

The set of admissible functions *f*

Example

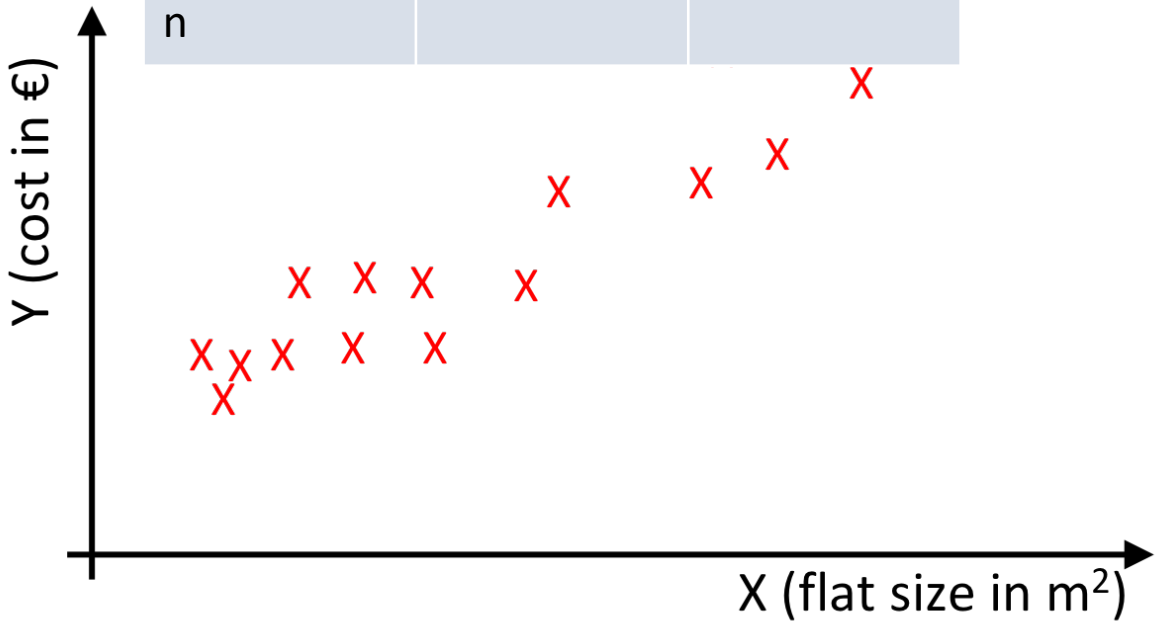
The set of neural networks with a given architecture



A very simple example: univariate linear regression

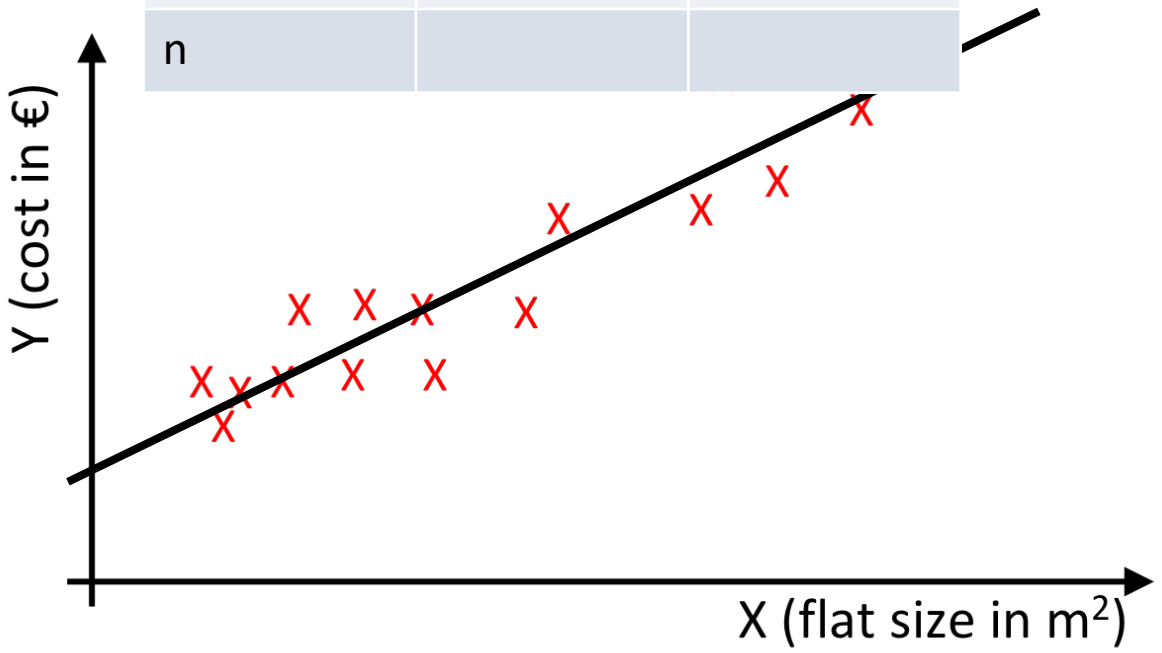
Training set

i	$x^{(i)}$ Size of flat (in m^2)	$y^{(i)}$ Cost (in €)
1	100	940,000
2	53	510,000
3	25	280,000
…		
n		



Training set

i	$x^{(i)}$ Size of flat (in m^2)	$y^{(i)}$ Cost (in €)
1	100	940,000
2	53	510,000
3	25	280,000
…		
n		



Model

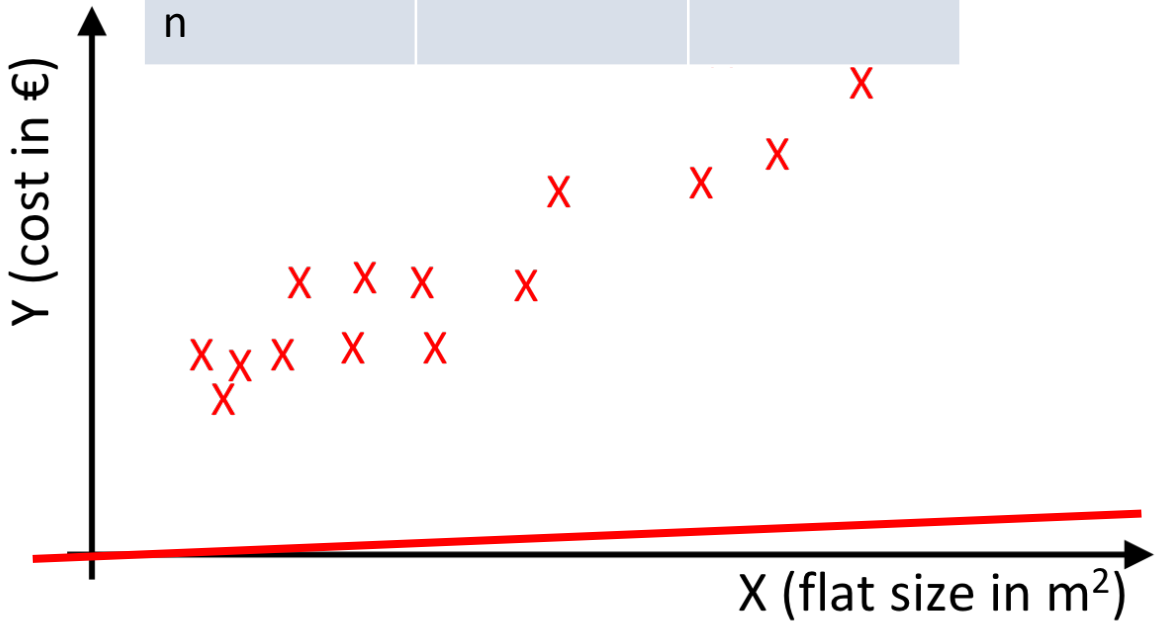
$$f(x) = w_1 x + w_0$$

Slope

Intercept

Training set

i	$x^{(i)}$ Size of flat (in m^2)	$y^{(i)}$ Cost (in €)
1	100	940,000
2	53	510,000
3	25	280,000
…		
n		



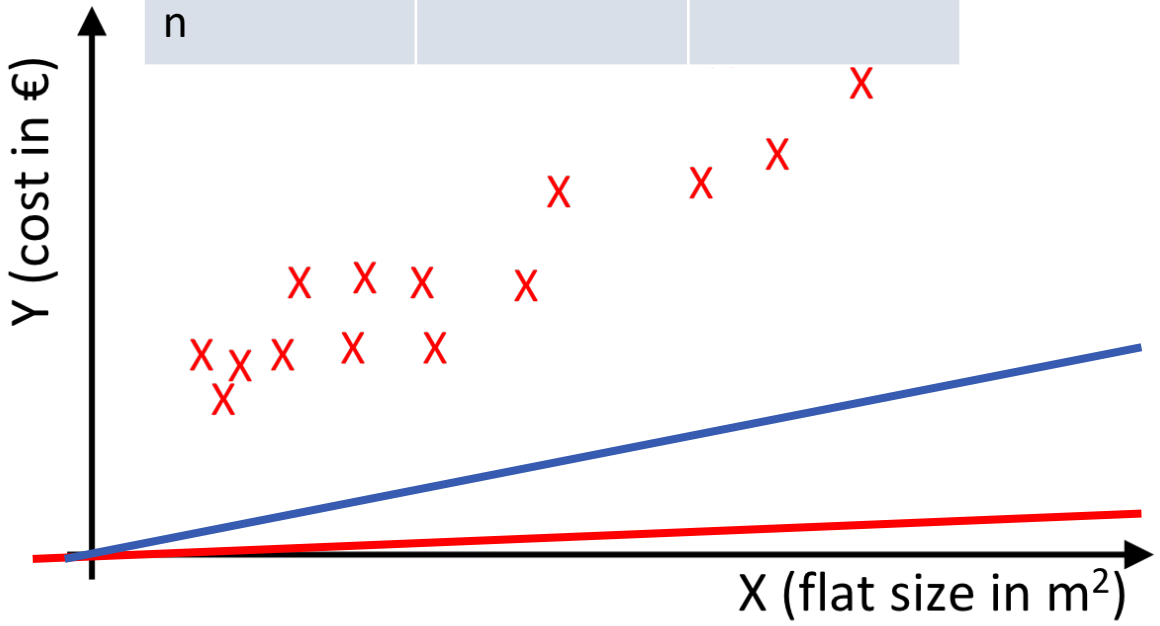
Model

$$f(x) = w_1 x + w_0$$

$$f(x) = 1000x + 0$$

Training set

i	$x^{(i)}$ Size of flat (in m^2)	$y^{(i)}$ Cost (in €)
1	100	940,000
2	53	510,000
3	25	280,000
…		
n		



Model

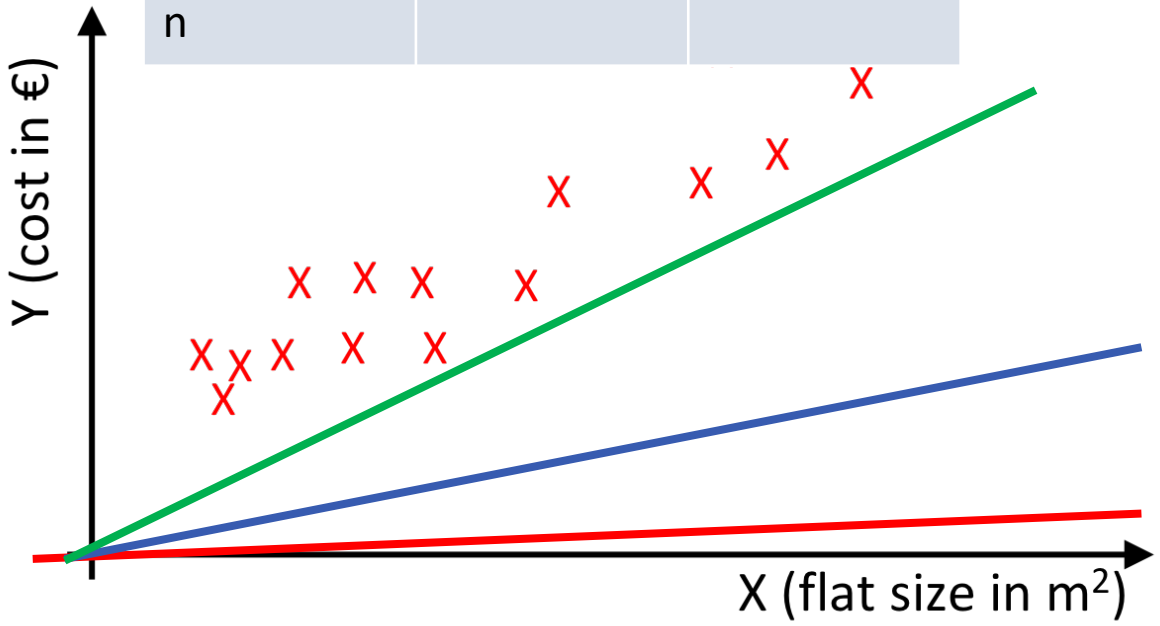
$$f(x) = w_1 x + w_0$$

$$f(x) = 1000x + 0$$

$$f(x) = 5000x + 0$$

Training set

i	$x^{(i)}$ Size of flat (in m^2)	$y^{(i)}$ Cost (in €)
1	100	940,000
2	53	510,000
3	25	280,000
…		
n		



Model

$$f(x) = w_1 x + w_0$$

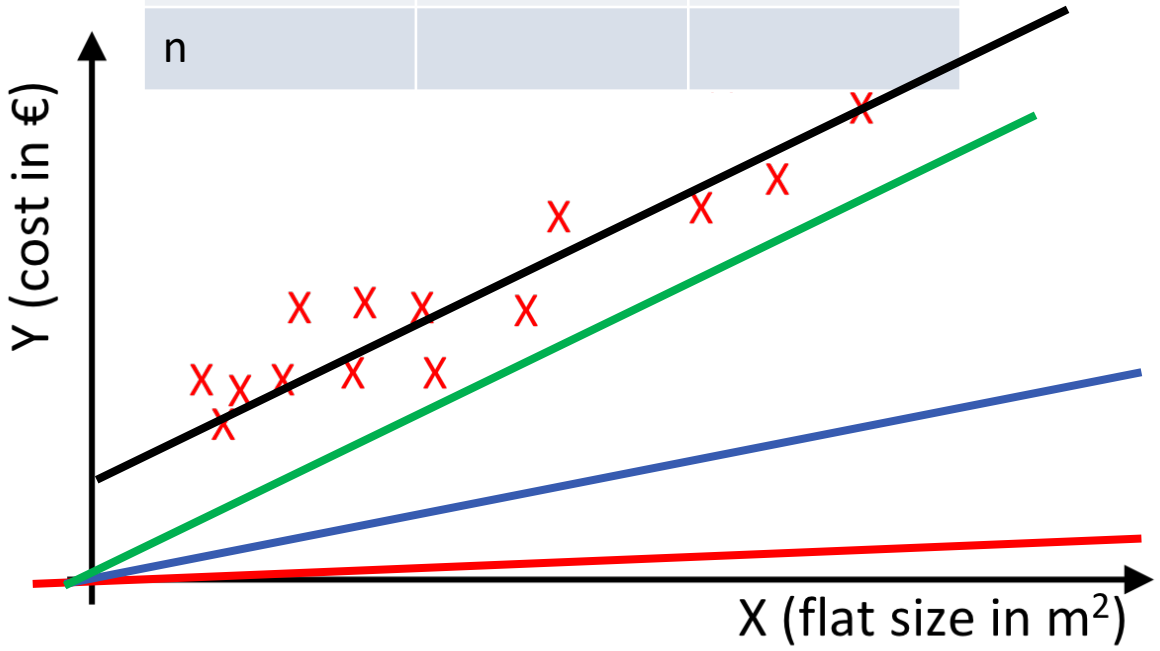
$$f(x) = 1000x + 0$$

$$f(x) = 5000x + 0$$

$$f(x) = 10000x + 0$$

Training set

i	$x^{(i)}$ Size of flat (in m^2)	$y^{(i)}$ Cost (in €)
1	100	940,000
2	53	510,000
3	25	280,000
…		
n		



Model

$$f(x) = w_1 x + w_0$$

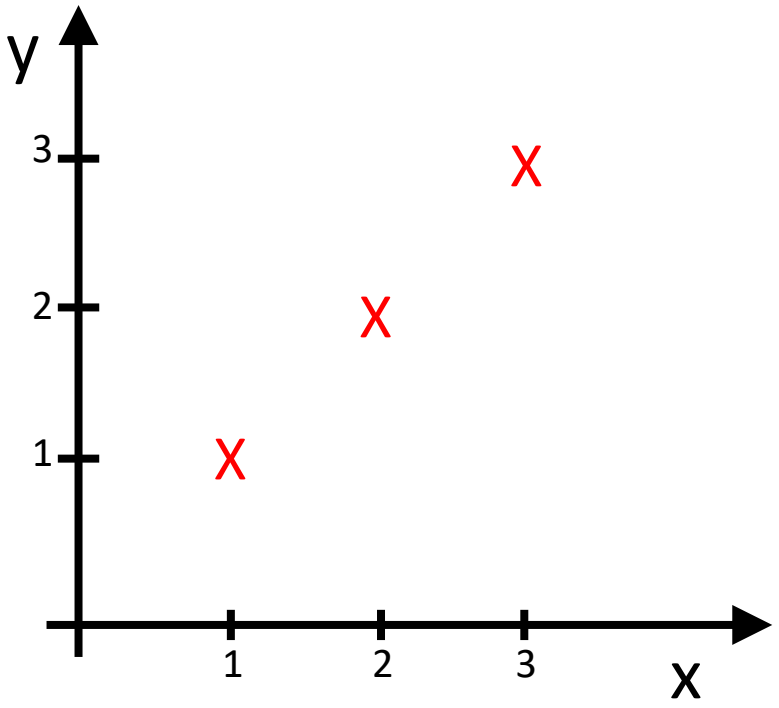
$$f(x) = 1000x + 0$$

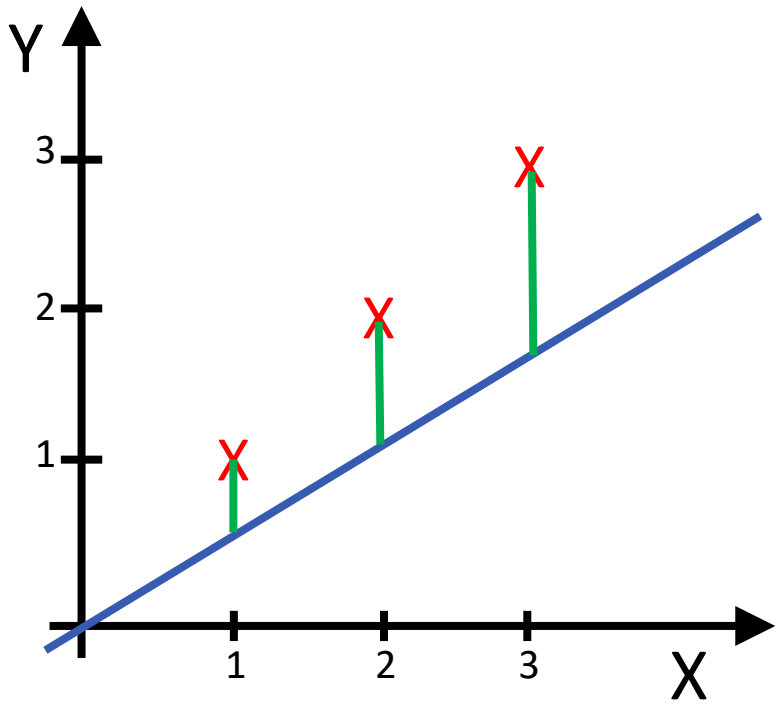
$$f(x) = 5000x + 0$$

$$f(x) = 10000x + 0$$

$$f(x) = 10000x + 30000$$

Model





Loss: $\ell(y, f(x)) = (y - f(x))^2$

Cost function:

$$J(f) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(y^{(i)}, f(x^{(i)}) \right)$$

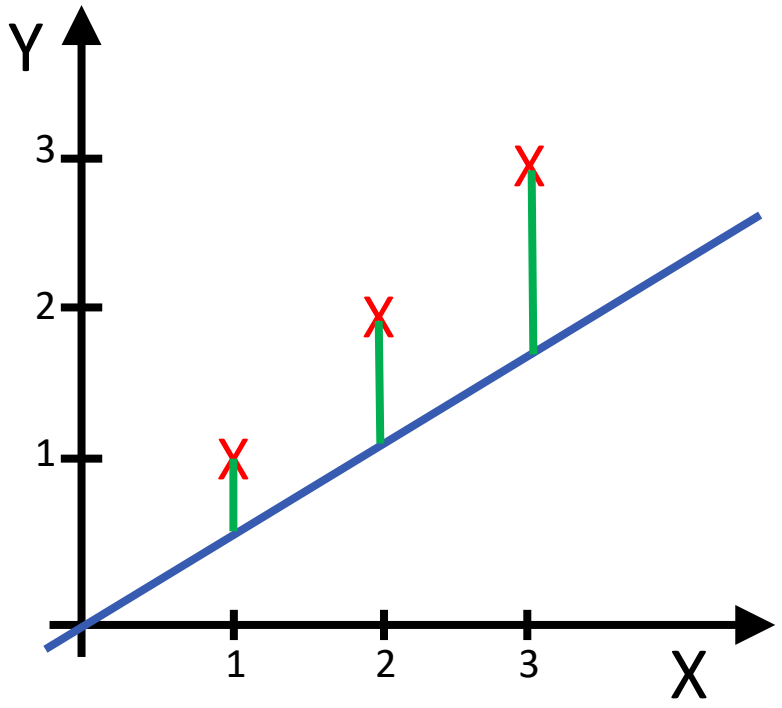
Cost function:

$$J(f) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - f(x^{(i)}) \right)^2$$

Learning:

$$\hat{f} = arg \min_{f \in \mathcal{F}} J(f)$$

No intercept



Univariate regression model: $y = f(x) = w_1 x + w_0$

Assume that $w_0 = 0$

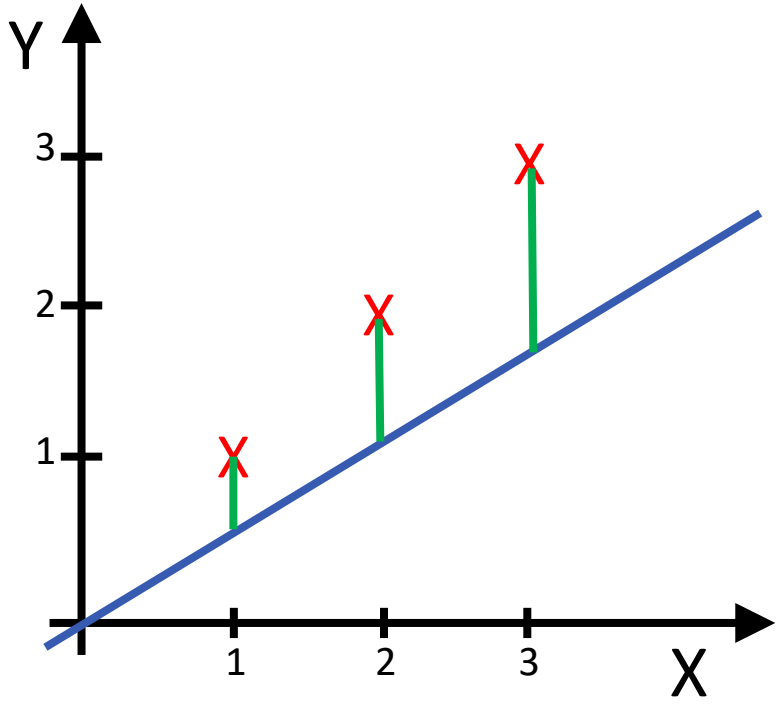
The model becomes: $y = f(x) = w_1 x$

Cost function:

$$J(w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - w_1 x^{(i)} \right)^2$$

Learning:

$$\hat{w_1} = arg \min_{w_1 \in \mathbb{R}} J(w_1)$$


Y
3
2
1
X
X
X
1
2
3
X

Model

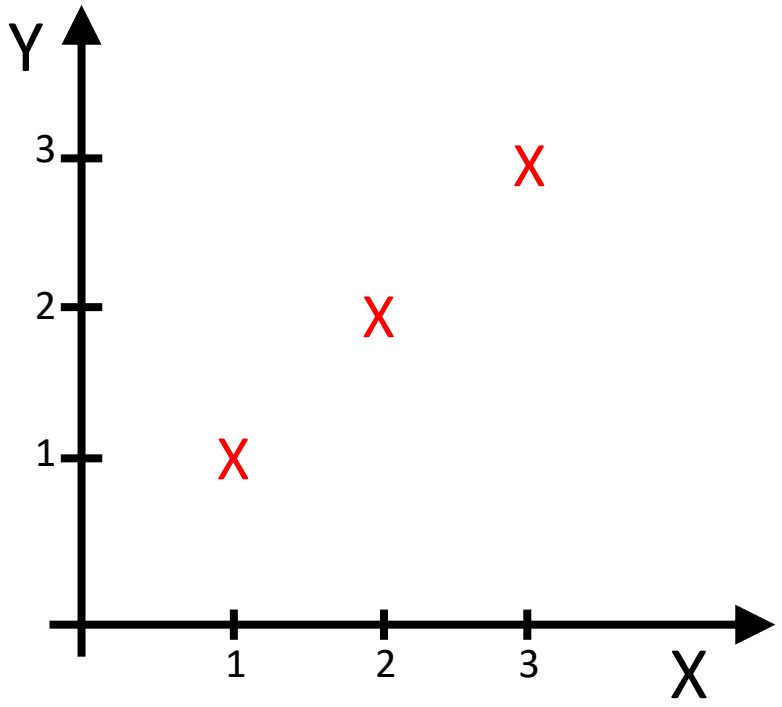
$$f(x) = w_1 x + w_0$$

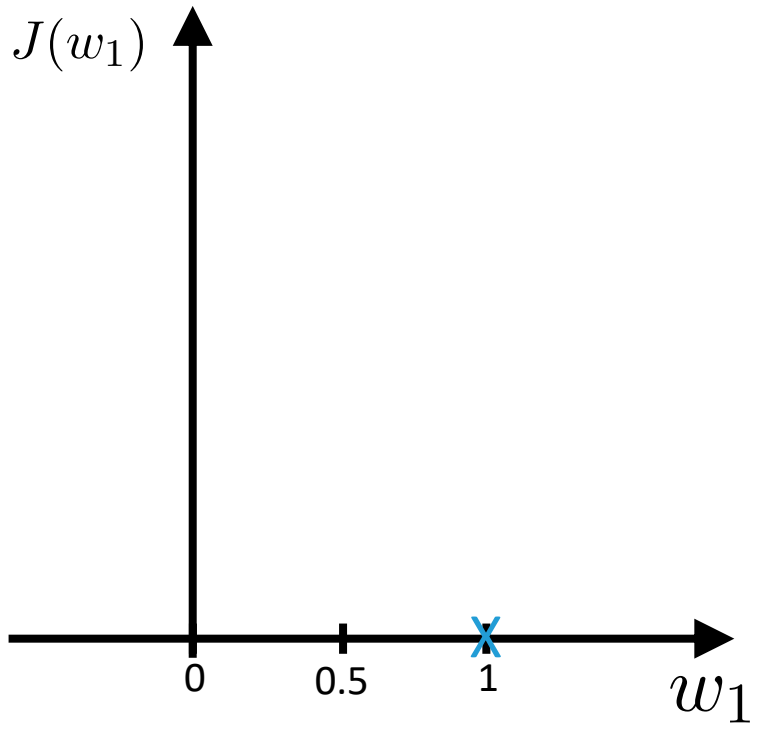
Cost function:

$$J(w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - w_1 x^{(i)} \right)^2$$

Learning:

$$\hat{w_1} = arg \min_{w_1 \in \mathbb{R}} J(w_1)$$





Model

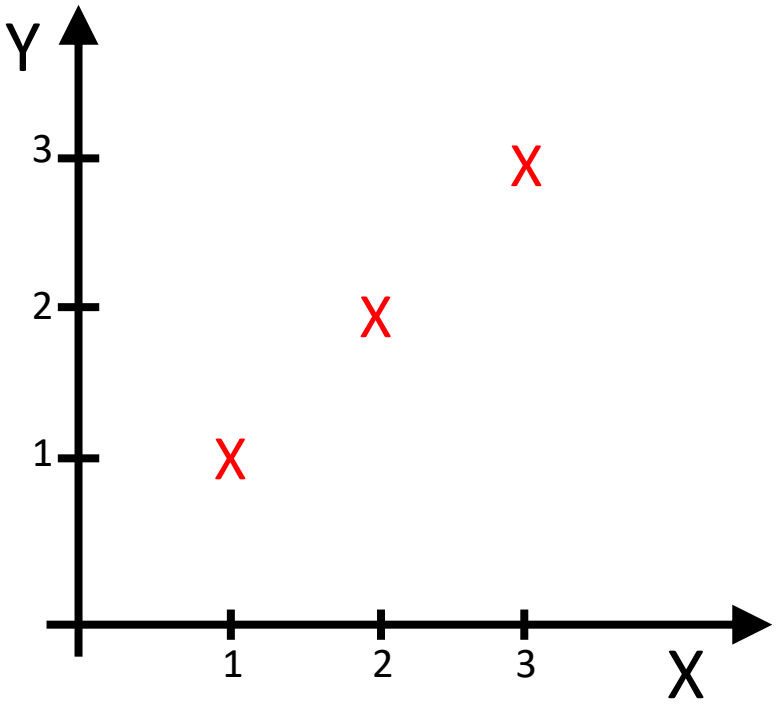
$$f(x) = w_1 x + w_0$$

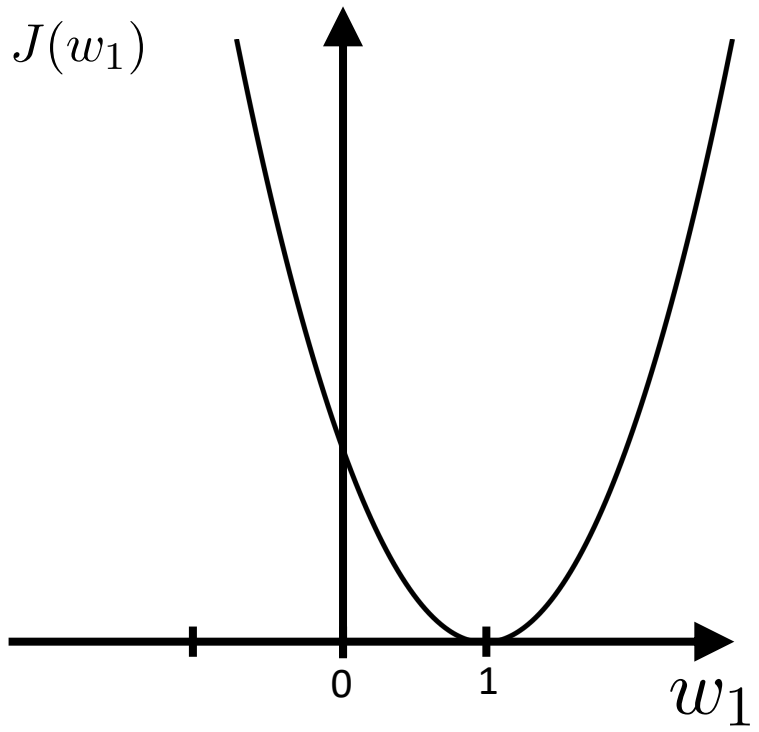
Cost function:

$$J(w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - w_1 x^{(i)} \right)^2$$

$J(w_1)$ is a quadratic function of w_1

Parabolic curve.





Model

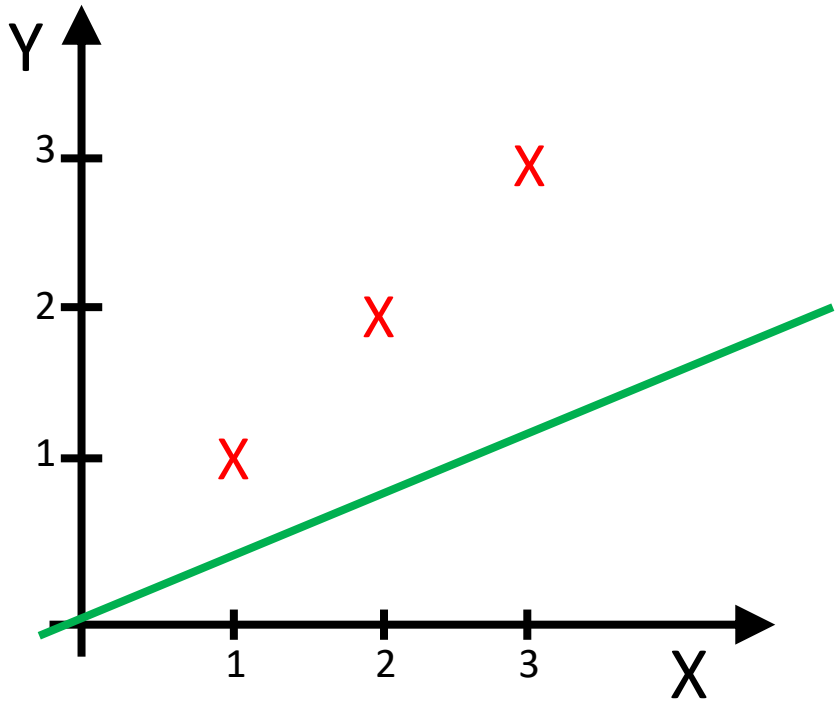
$$f(x) = w_1 x + w_0$$

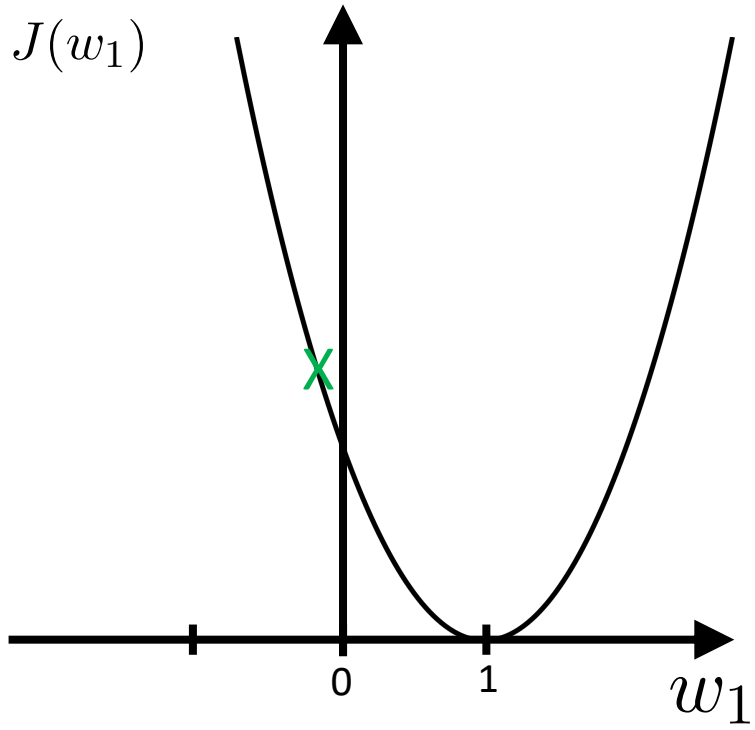
Cost function:

$$J(w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - w_1 x^{(i)} \right)^2$$

It has a unique global minimum which is, in our example, $w_1 = 1$

To each value w_1 corresponds a regression line with a corresponding value of $J(w_1)$





Model

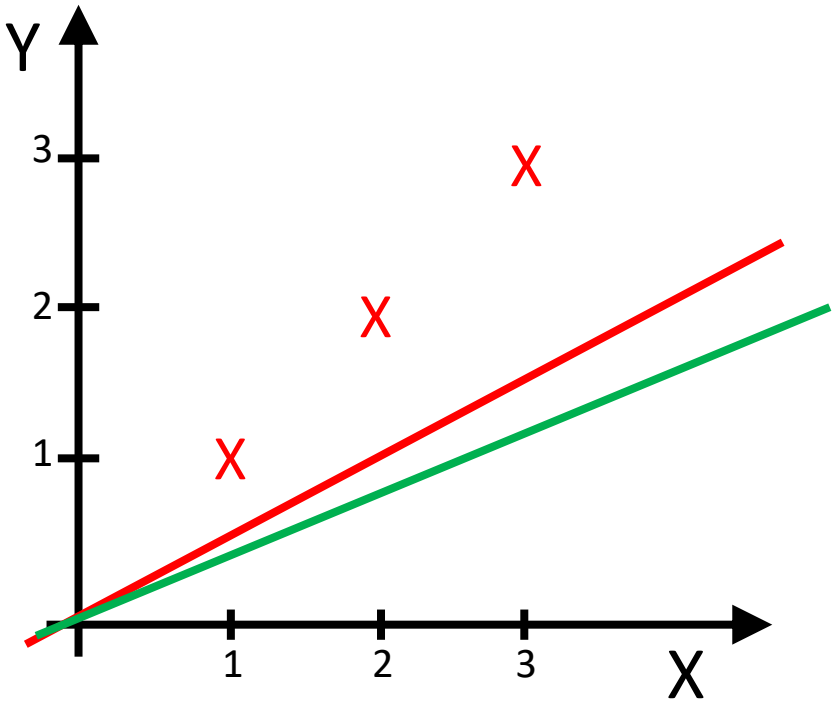
$$f(x) = w_1 x + w_0$$

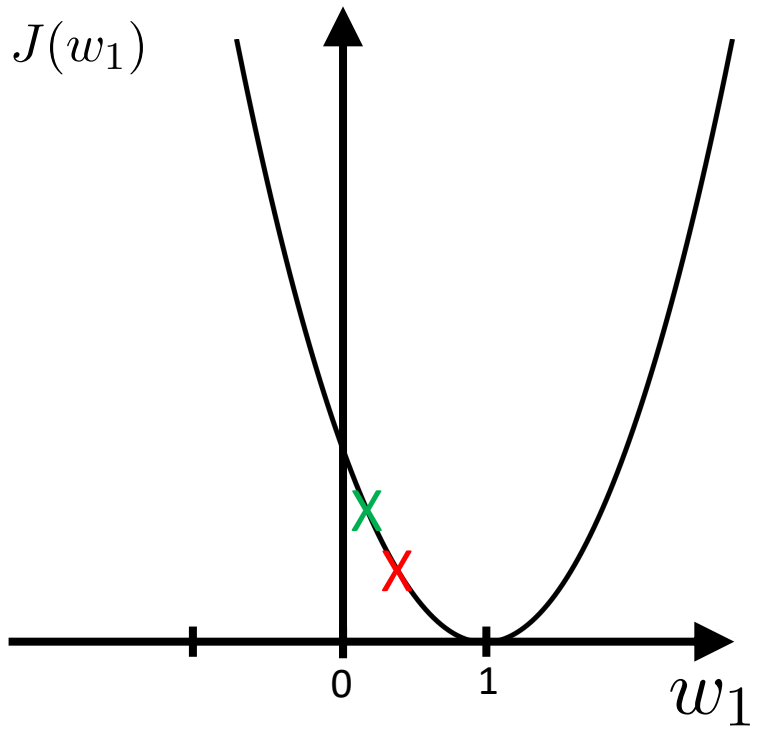
Cost function:

$$J(w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - w_1 x^{(i)} \right)^2$$

It has a unique global minimum which is, in our example, $w_1 = 1$

To each value w_1 corresponds a regression line with a corresponding value of $J(w_1)$





Model

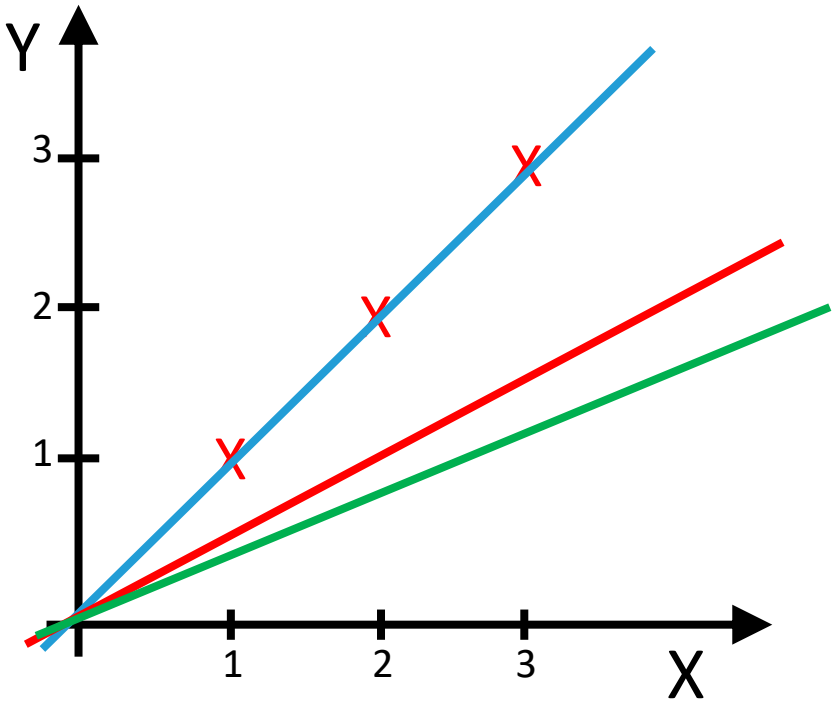
$$f(x) = w_1 x + w_0$$

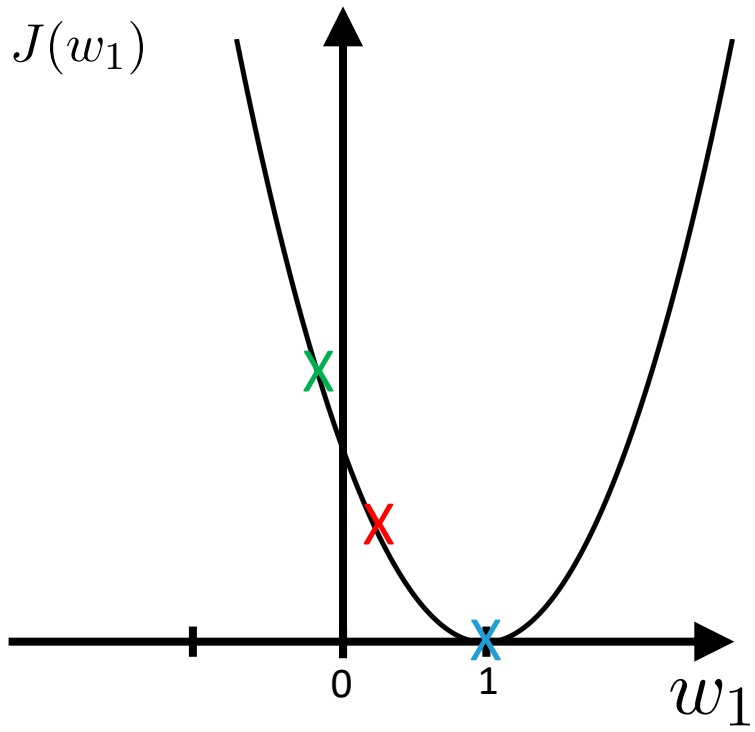
Cost function:

$$J(w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - w_1 x^{(i)} \right)^2$$

It has a unique global minimum which is, in our example, $w_1 = 1$

To each value w_1 corresponds a regression line with a corresponding value of $J(w_1)$





Let's add an intercept

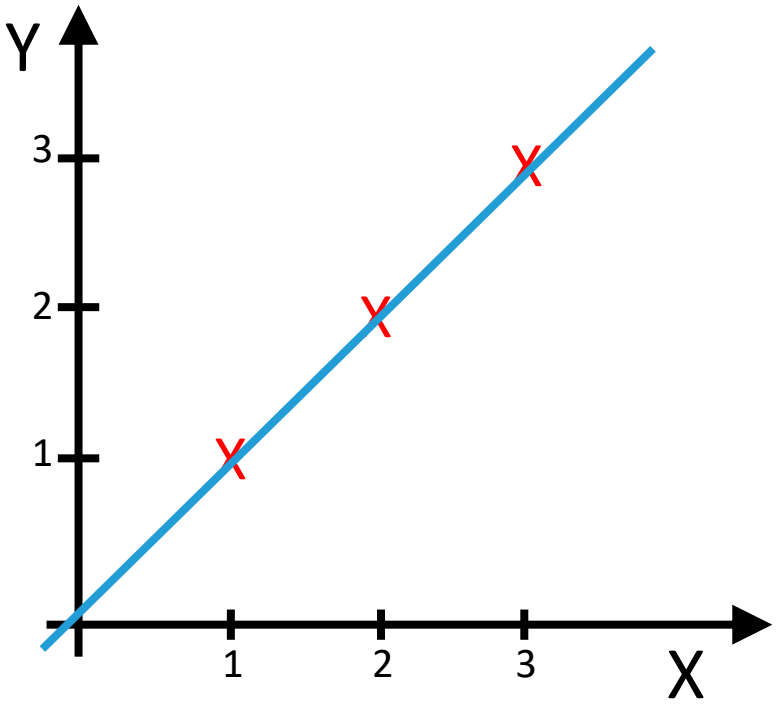
Univariate regression model: $y = f(x) = w_1 x + w_0$

Cost function:

$$J(w_1, w_0) = \frac{1}{N} \sum_{i=1}^{n} (y^{(i)} - (w_1 x^{(i)} + w_0))^2$$

Learning:

$$(\hat{w}_0, \hat{w}_1) = \underset{(w_0, w_1) \in \mathbb{R}^2}{\arg\min} \; J(w_0, w_1)$$



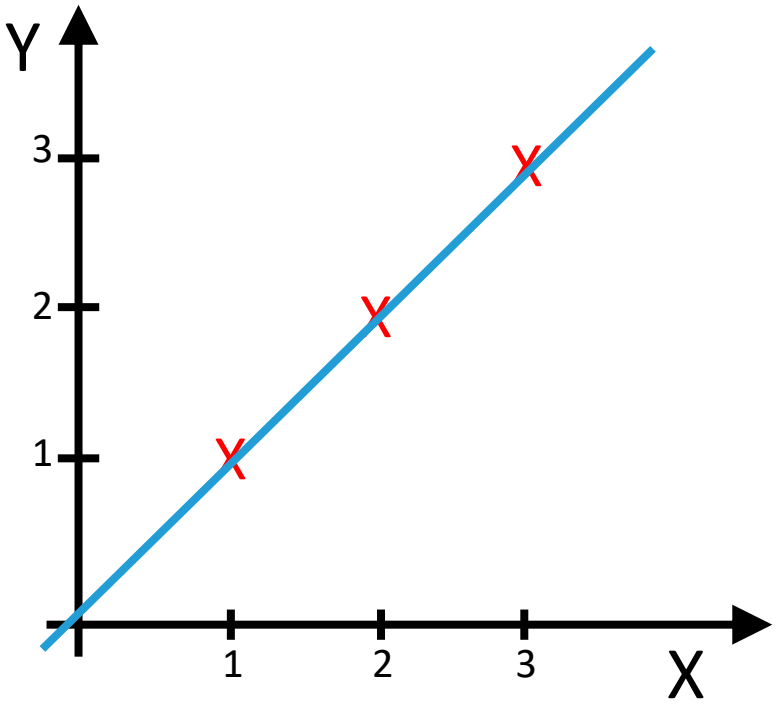
Univariate regression model: $y = f(x) = w_1 x + w_0$

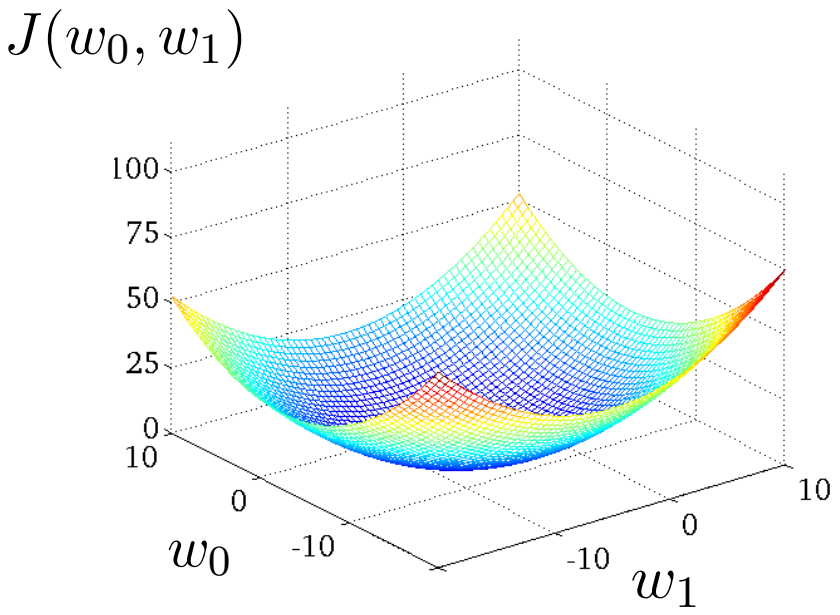
Cost function:

$$J(w_1, w_0) = \frac{1}{N} \sum_{i=1}^{n} (y^{(i)} - (w_1 x^{(i)} + w_0))^2$$

Learning:

$$(\hat{w}_0, \hat{w}_1) = \underset{(w_0, w_1) \in \mathbb{R}^2}{\arg\min} \; J(w_0, w_1)$$





The solution can be computed analytically

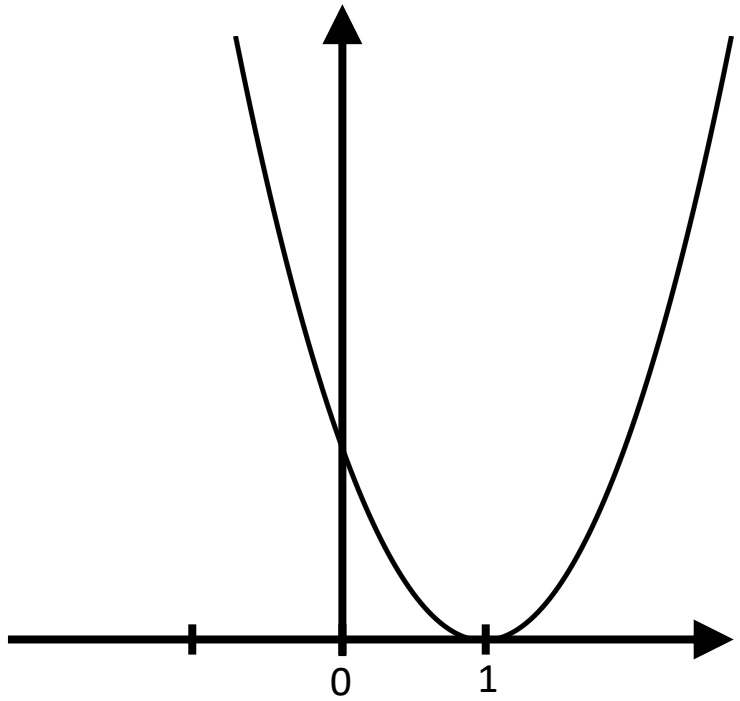
But let’s assume it cannot

Gradient descent

Idea:

- Start with an initial value of w_1
- Keep changing w_1 so that it reduces $J(w_1)$

Case with one parameter

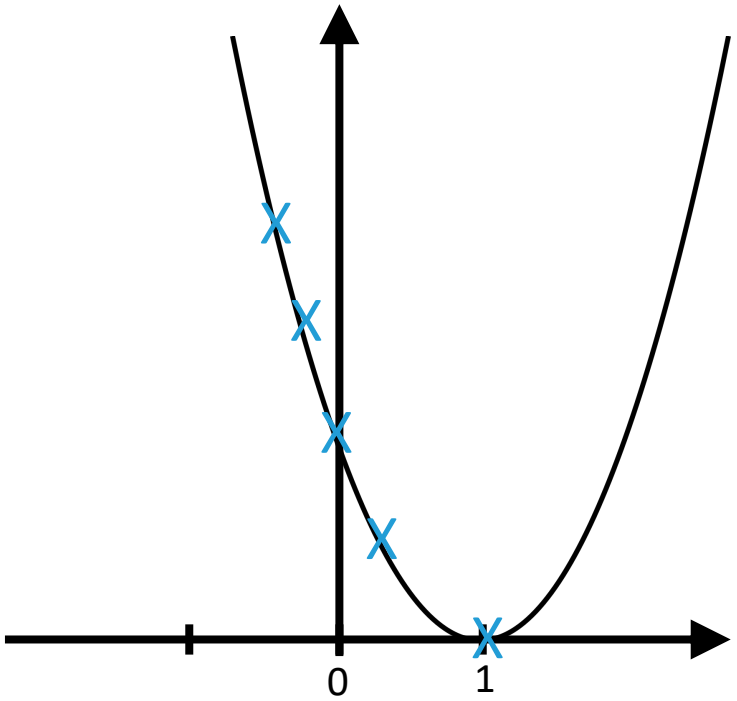


Gradient descent

Idea:

- Start with an initial value of w_1
- Keep changing w_1 so that it reduces $J(w_1)$

Case with one parameter

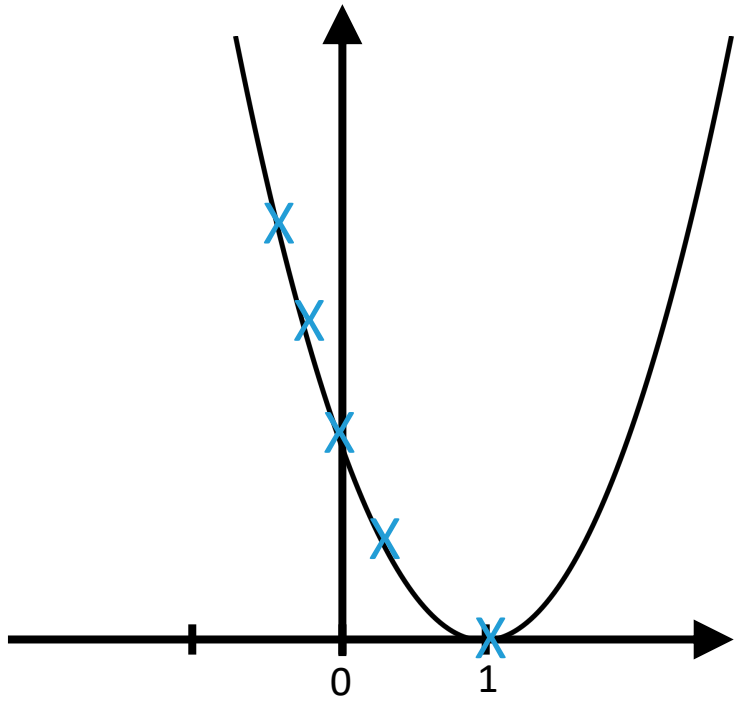


Gradient descent

Idea:

- Start with an initial value of w_1
- Keep changing w_1 so that it reduces $J(w_1)$

Case with one parameter

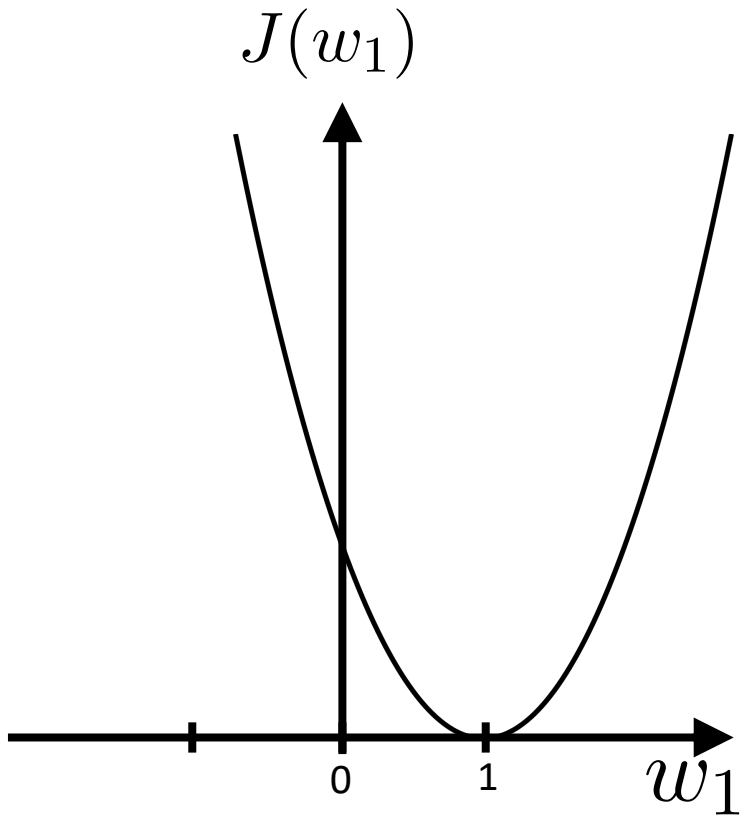


Gradient descent

How do we update w_1 ?

- Use the derivative of $J(w_1)$

Case with one parameter

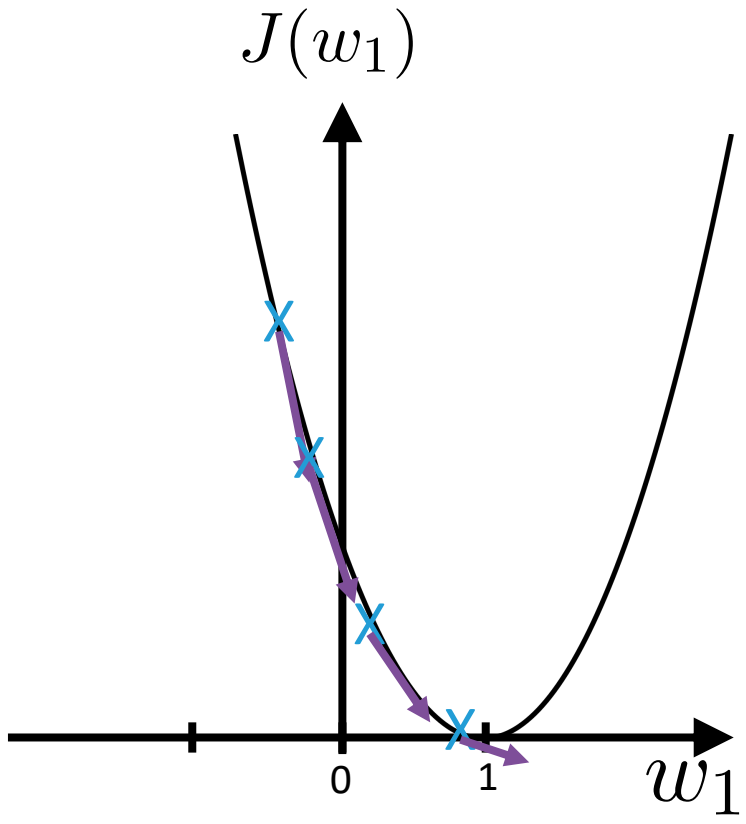


Gradient descent

How do we update w_1 ?

- Use the derivative of $J(w_1)$ $\frac{dJ}{dw_1}$

Case with one parameter



Gradient descent

Gradient descent algorithm

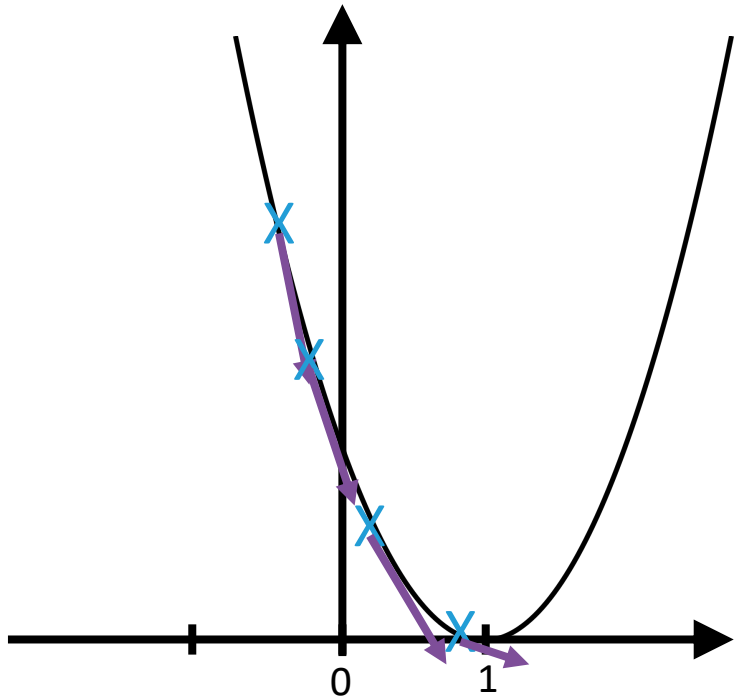
Repeat until convergence {

$$w_1 \leftarrow w_1 - \eta \frac{dJ}{dw_1}$$

}

derivative

Learning rate

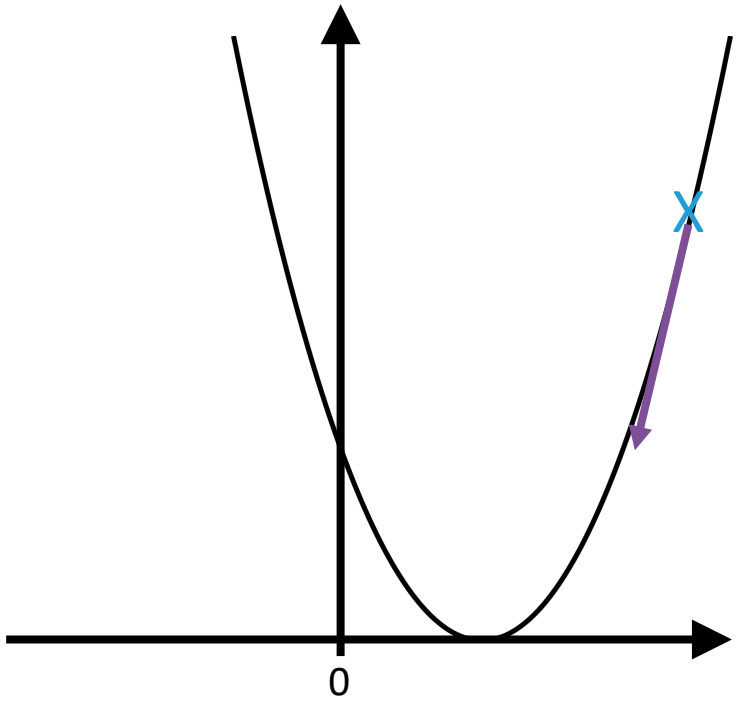


Case with one parameter

Gradient descent

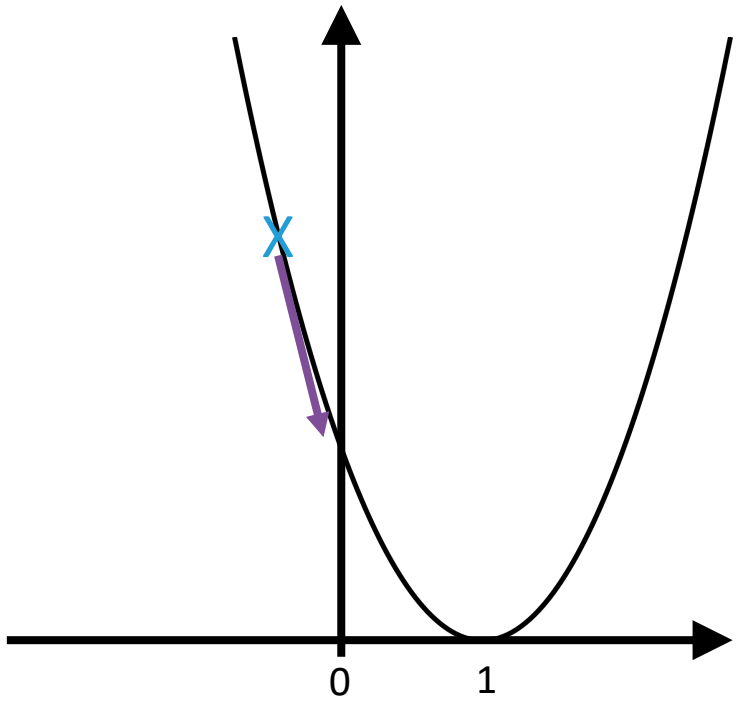
$$w_1 \leftarrow w_1 - \eta \frac{dJ}{dw_1}$$

positive



$$w_1 \leftarrow w_1 - \eta \frac{dJ}{dw_1}$$

negative

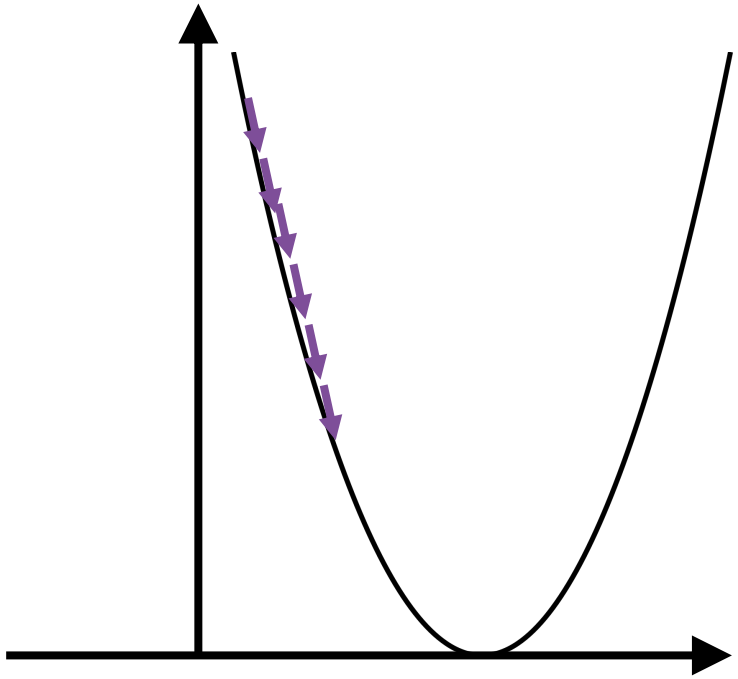


Gradient descent

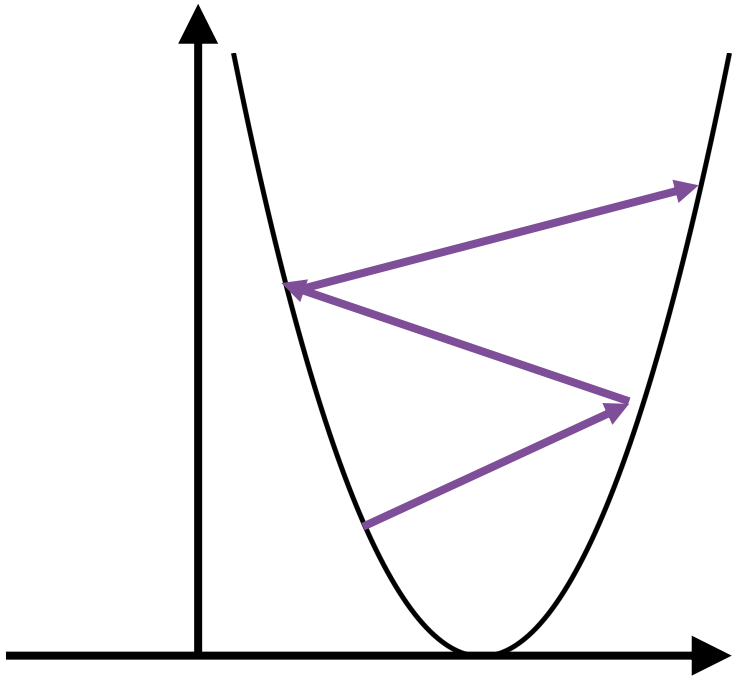
How to choose the learning rate.

$$w_1 \leftarrow w_1 - \eta \frac{dJ}{dw_1}$$

- If η is too small: slow to converge



- If η is too large: may diverge



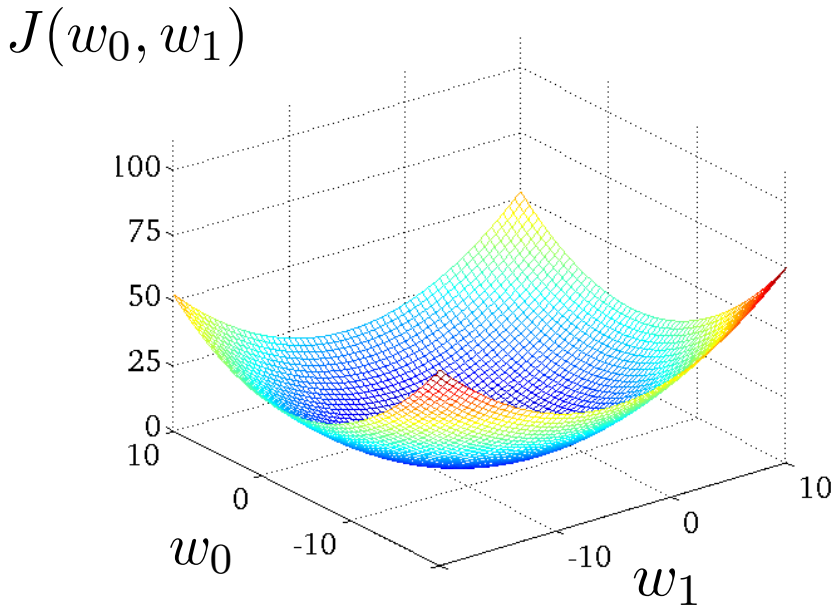
Gradient descent

Cost function $J(w_0, w_1)$

Partial derivatives $\frac{\partial J}{\partial w_0}$ $\frac{\partial J}{\partial w_1}$

Gradient $\nabla J = \left(\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}\right)$ $\boldsymbol{\nabla} J$ or $\frac{\partial J}{\partial \boldsymbol{w}}$

Case with two parameters



Gradient descent

Gradient descent algorithm

Repeat until convergence {

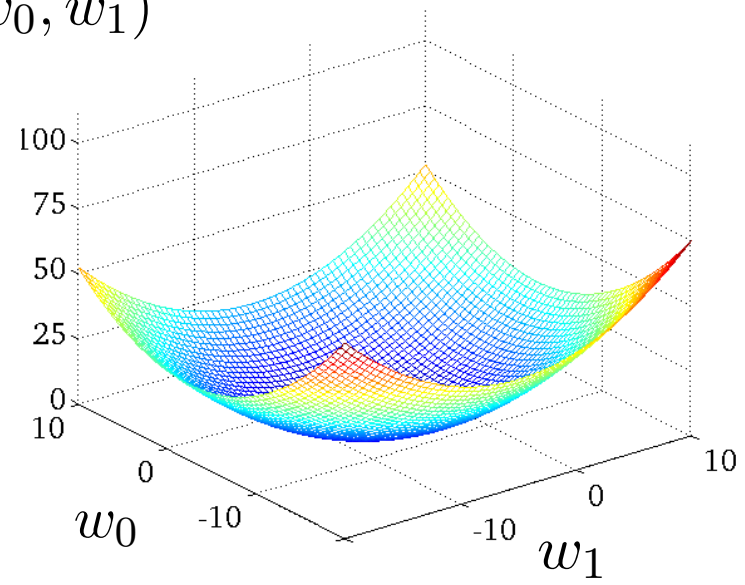
$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla J$$

gradient

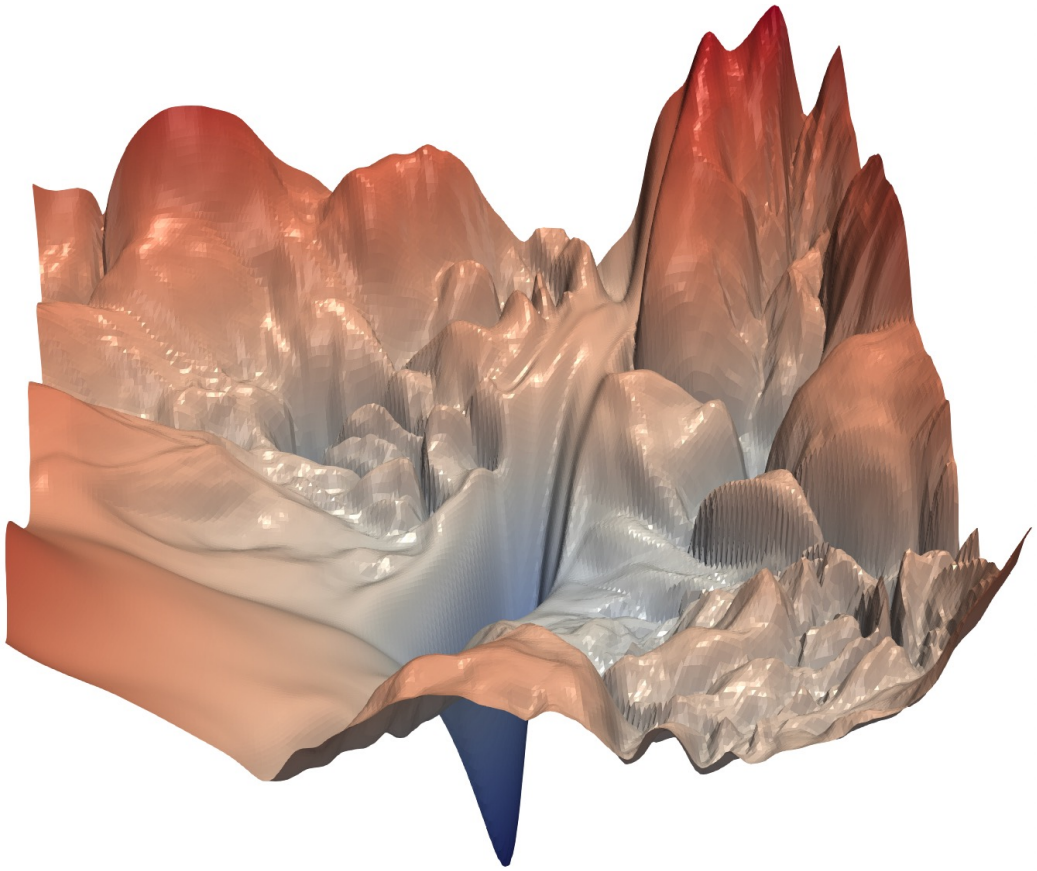
Learning rate

Case with two parameters

$J(w_0, w_1)$



Gradient descent



Li et al., Visualizing the Loss Landscape of Neural Nets, NeurIPS, 2018

Features

- In the previous examples, inputs were series of numbers

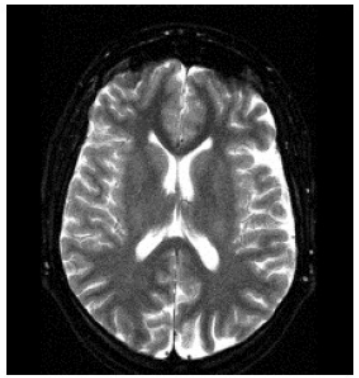
i	$x_{i,1}$ Size of flat (in m^2)	$x_{i,2}$ Number of rooms	$x_{i,3}$ Floor	y_i Cost (in €)
1	100	5	1	940,000
2	53	3	5	510,000
3	25	1	6	280,000
…				
N				

- In ML, an input variable is often called a **feature**

Features

- There are many cases where the inputs are not numbers

Natural images



Magnetic resonance images

1. 'To be, or not to be: that is the question'

(Hamlet Act 3, Scene 1)

Text

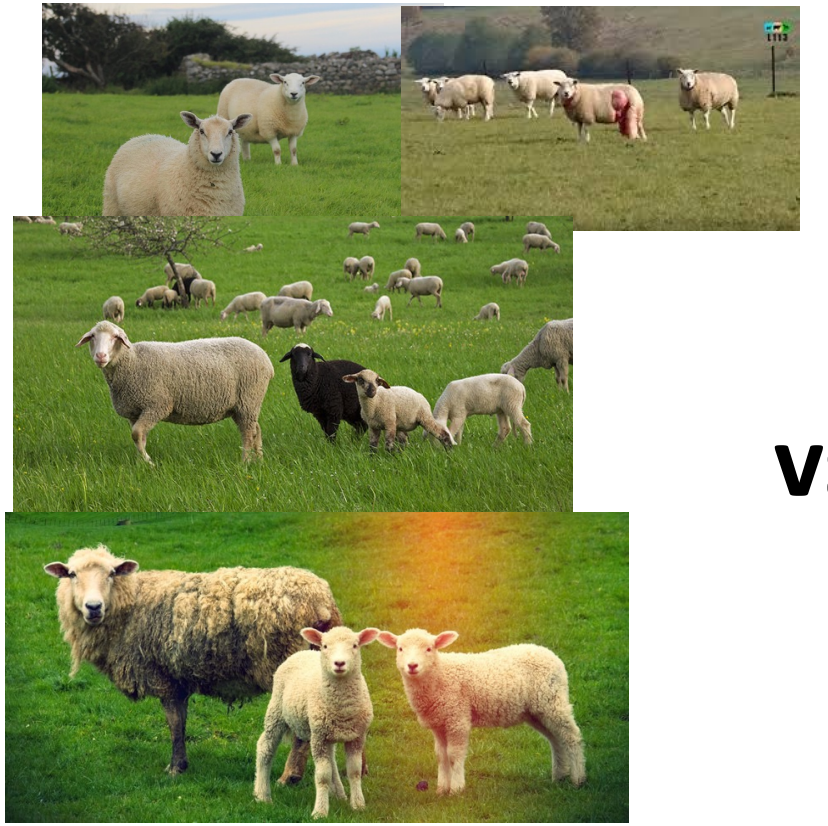
DNA sequence

- Of course, in a computer, everything is stored as numbers
  - But using their direct encoding may not be meaningful
    - ASCII characters for text
    - An image has a structure richer than a simple vector of pixel values

Features

- One solution is to extract variables from the input so that we can apply a ML algorithm that works on numbers
- These variables, which are going to be used as input of the algorithm, are called **features**.
- They represent the characteristics of the data that are relevant for the machine learning task that is considered

Features



vs

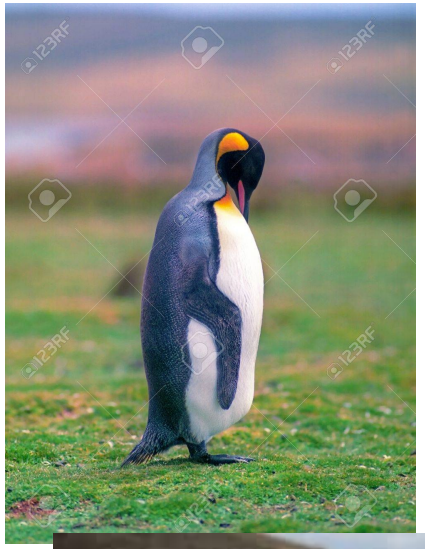
Classify images of sheeps vs images of penguins

Features:

- x1: amount of green in the image
- x2: amount of white in the image

Features

How course, this will not always work



Features

Labelled training examples

$(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Learning

Input

x

f

$y = f(x)$

Beaver

Output

y

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Features

Labelled training examples

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$$

Learning

Input

x

f $y = f(\phi(x))$

Beaver

Output

y

ϕ is the feature extractor

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \ell\left(y^{(i)}, f\left(\phi(x^{(i)})\right)\right)$$

Features

Labelled training examples

Learning

Input

x

f $y = f(\phi(x))$

Beaver

Output

y

$(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

$\left(\phi(x^{(i)})\right)\Big)$

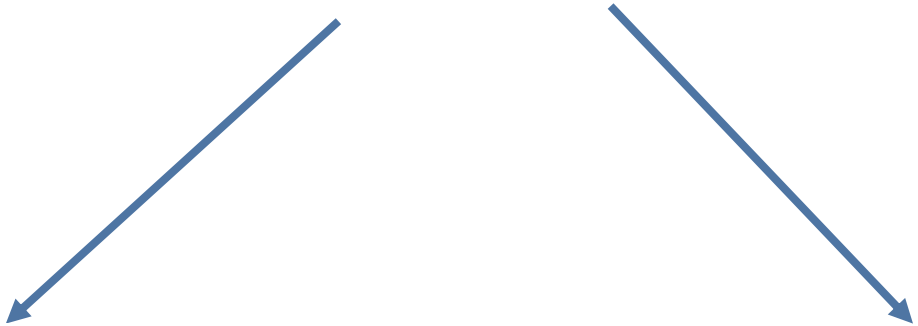
$i=1$

ϕ is the feature extractor

ϕ maps the set of inputs to the set of features $\mathbb{R}^p$

Features

How to define ϕ?



Define it "manually"

Feature engineering
"Hand-crafted" features

Learn it

Deep learning

Summary

Input variable (multivariate): $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$

Output: y

Model: f, $y = f(x)$ The "artificial intelligence"

Model parameters: $\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$, indexed by j

Model, depending on parameters: $f(\boldsymbol{x}; \boldsymbol{w})$

Summary

Input variable (multivariate): $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$

Input features

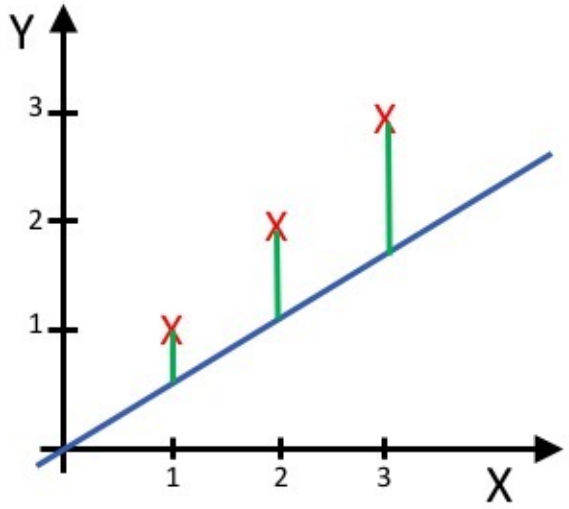
Output: y

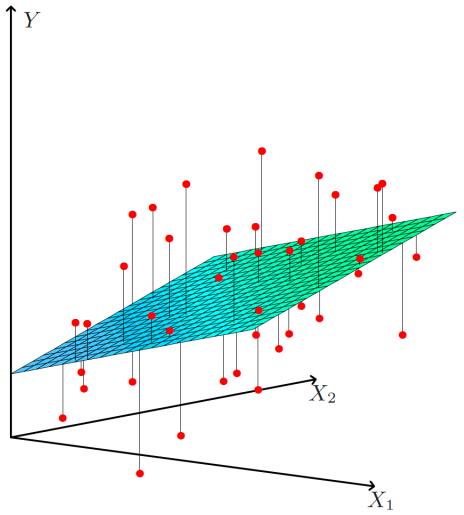
Model: f, $y = f(x)$

The "artificial intelligence"

Loss: $\ell(y, x)$

Quantifies how much the prediction is far from the true output





Summary

Input variable (multivariate): $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$

Input features

Output: y

Model: $f,\ y = f(x)$

The "artificial intelligence"

Loss: $\ell(y, x)$

Quantifies how much the prediction is far from the true output

Cost function:

$$J(f) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(y^{(i)}, f(x^{(i)}) \right)$$

How far are we from the true output across all training examples ?

Learning:

$$\hat{f} = arg \min_{f \in \mathcal{F}} J(f)$$

Learning: find the model with the minimal error

Optimization algorithm: method to find the minimum

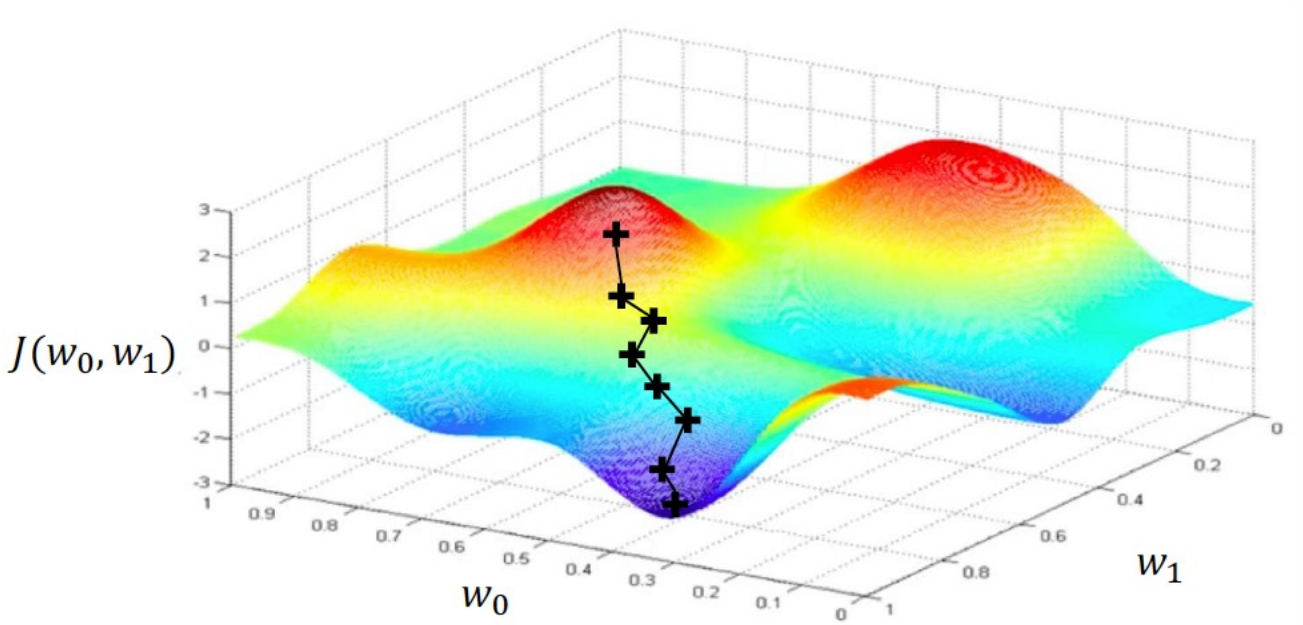
Summary

Learning:

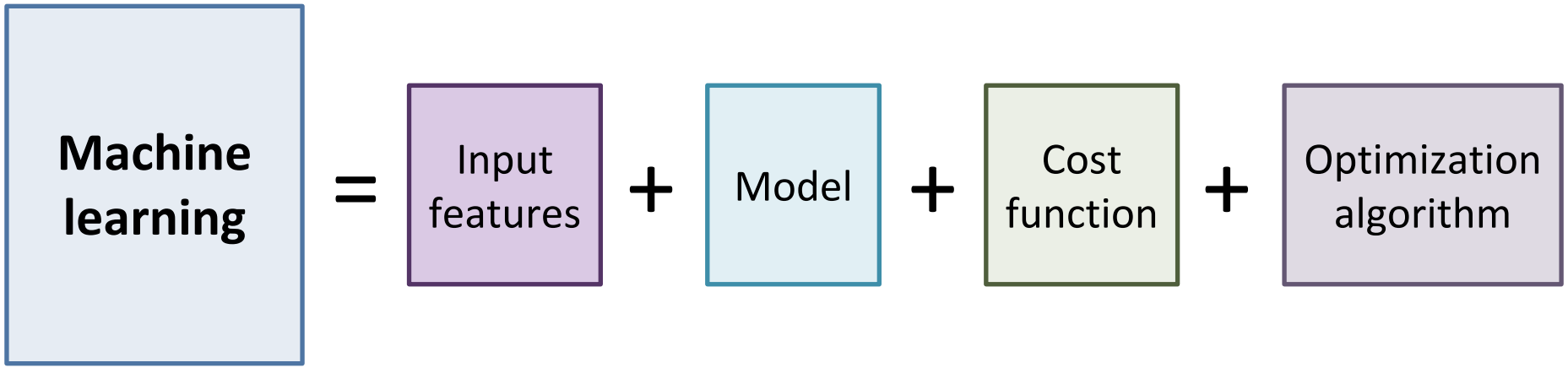
$$\hat{f} = arg\min_{f \in \mathcal{F}} J(f)$$

Learning: find the model with the minimal error

Optimization algorithm: method to find the minimum



Summary


Machine learning
=
Input features
+
Model
+
Cost function
+
Optimization algorithm

ML basics – overfitting, underfitting and model selection

Empirical risk minimization

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Empirical risk minimization

Empirical risk minimization

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Computed from observations → **Empirical** risk minimization

Empirical risk minimization

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

A stupid solution which minimizes the empirical risk

Let f such that $f(x^{(i)}) = y^{(i)}$ for all training examples and $f(x)$ takes a random value otherwise

Empirical risk minimization

$$\hat{f} = arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f(x^{(i)})\right)$$

Ideally, I would want $\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbf{E}[l(f(x), y)] = \arg\min_{f \in \mathcal{F}} \int l(f(x), y) dP(x, y)$

But of course, I don't know $P(x, y)$

Empirical risk minimization

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

$$R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f)$$

Empirical risk minimization

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

$$R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f)$$

Was called J(f) in the previous section

Risk vs empirical risk

$$R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(y^{(i)}, f \left(x^{(i)} \right) \right)$$

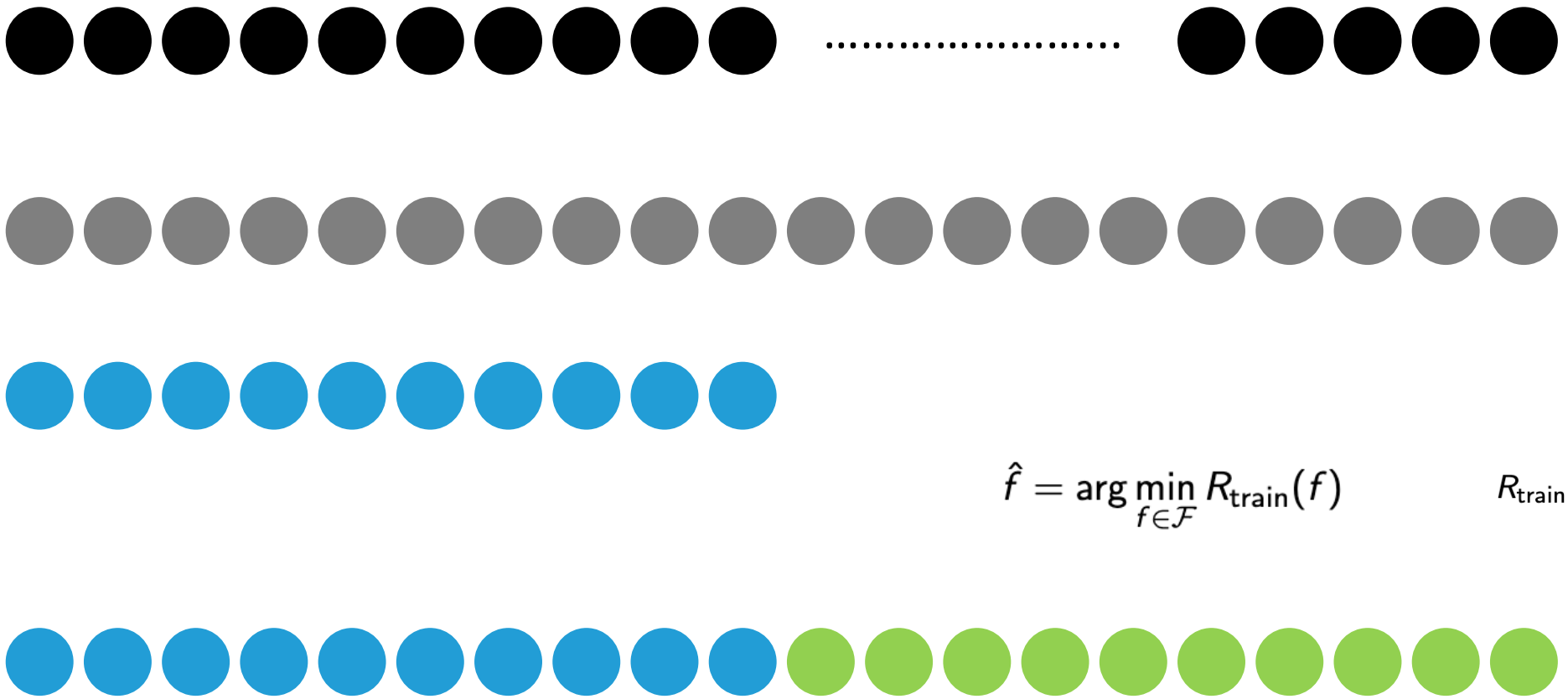
vs

$$R(f) = \mathbf{E}[l(f(x), y)]$$

Is $R_{\text{train}}(\hat{f})$ a good approximation of $R(\hat{f})$?

No, it is overoptimistic because $\hat{f}$ has been trained using R_{train}

Training and validation sets



Population

$$R(f) = \mathbf{E}[l(f(x), y)]$$

Sample

Training set

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f)$$

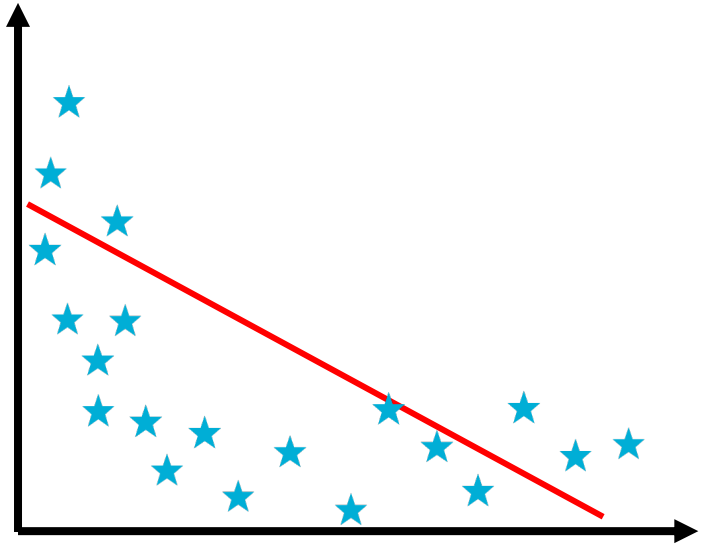
$$R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

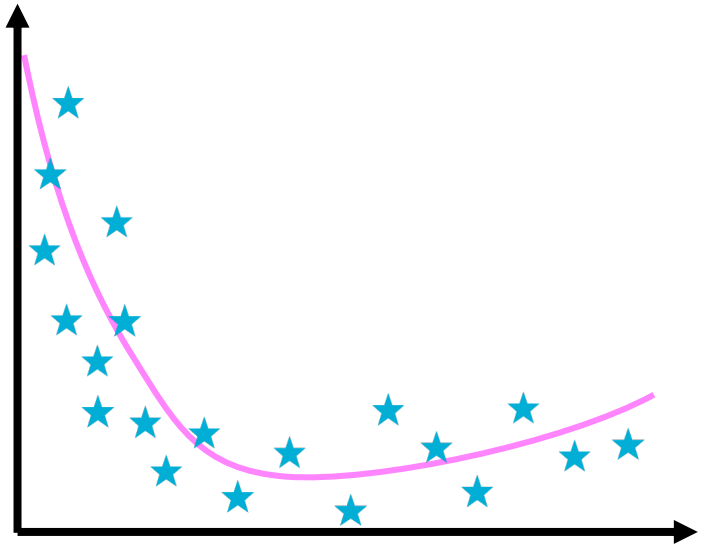
Validation set

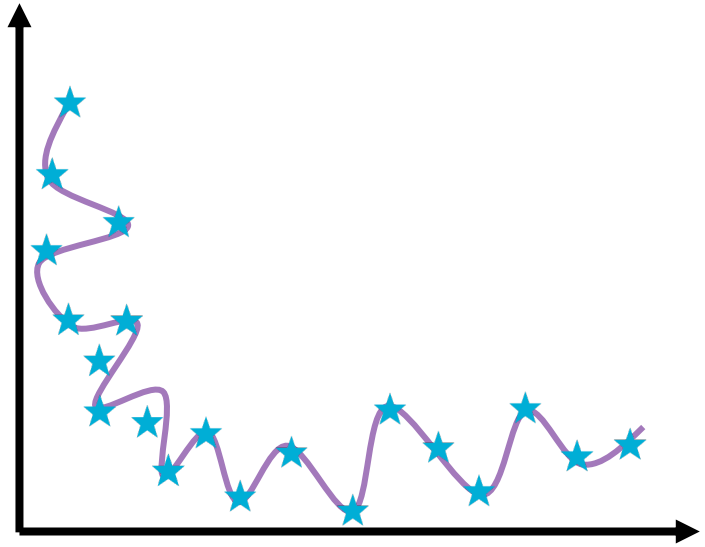
$$R_{\text{validation}}(\hat{f}) = \frac{1}{m} \sum_{i=1}^{m} \ell\left(y'^{(i)}, \hat{f}\left(x'^{(i)}\right)\right)$$

Overfitting (and underfitting)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f) \qquad R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

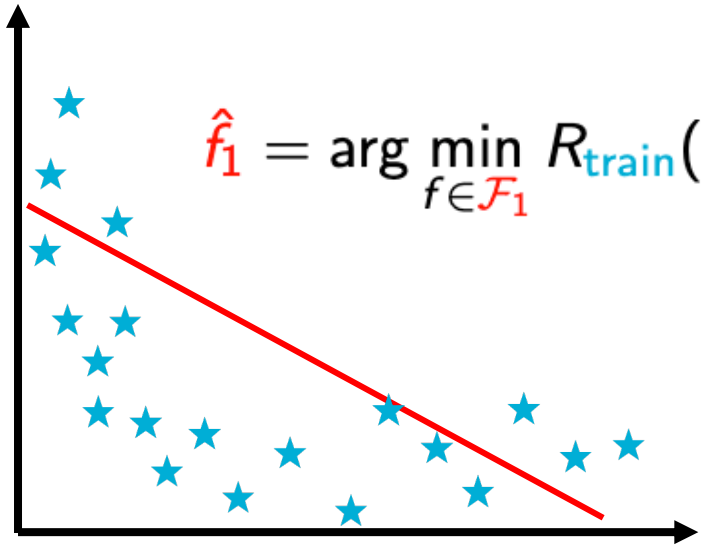


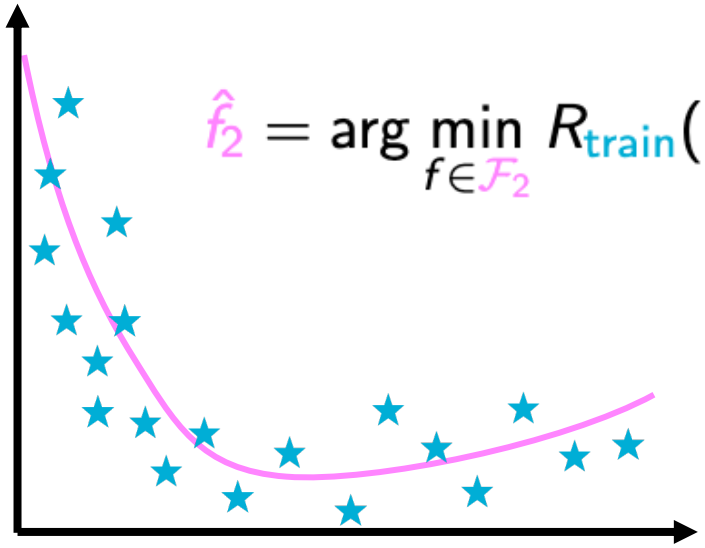


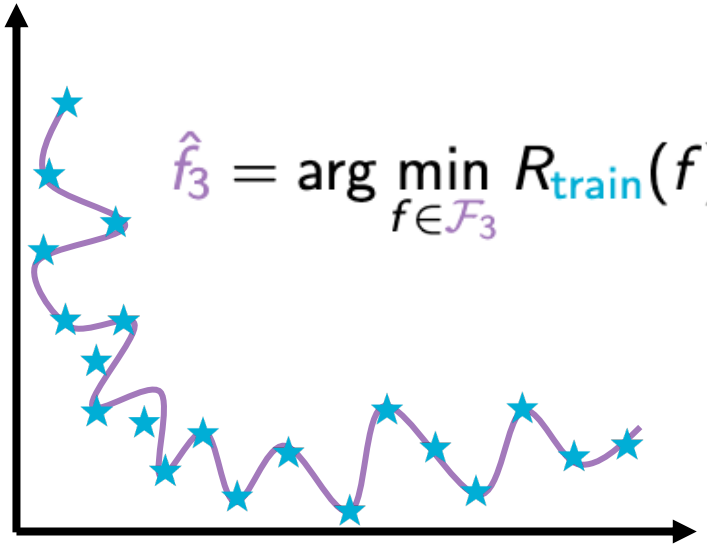


Overfitting (and underfitting)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f) \qquad R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

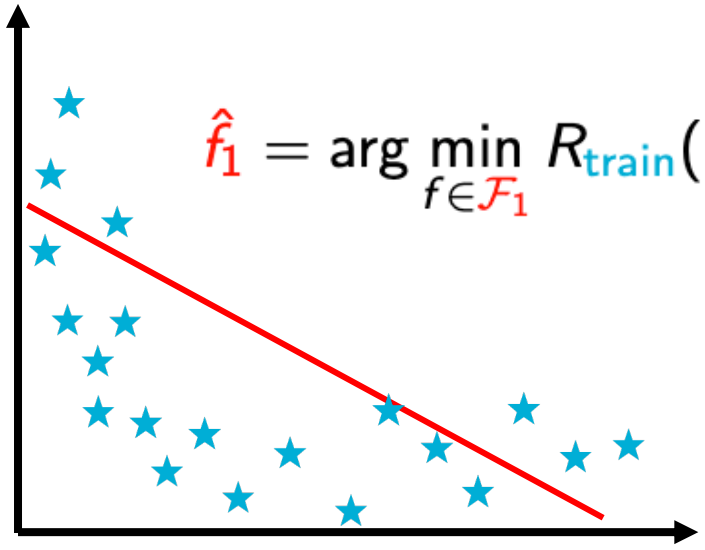


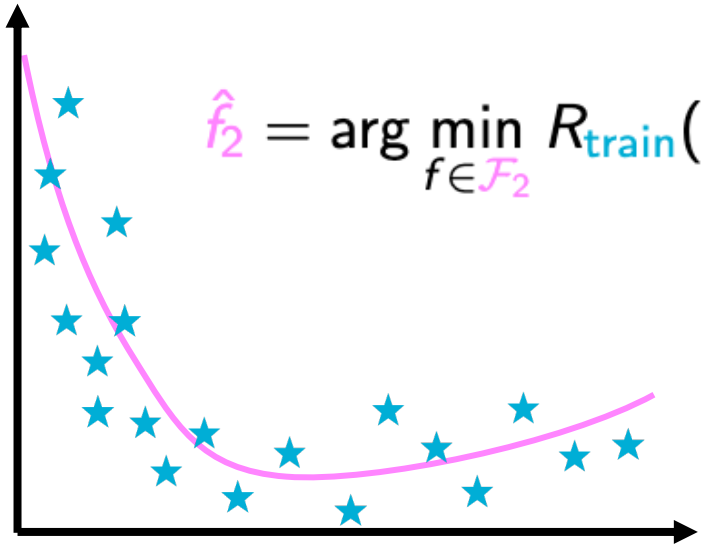


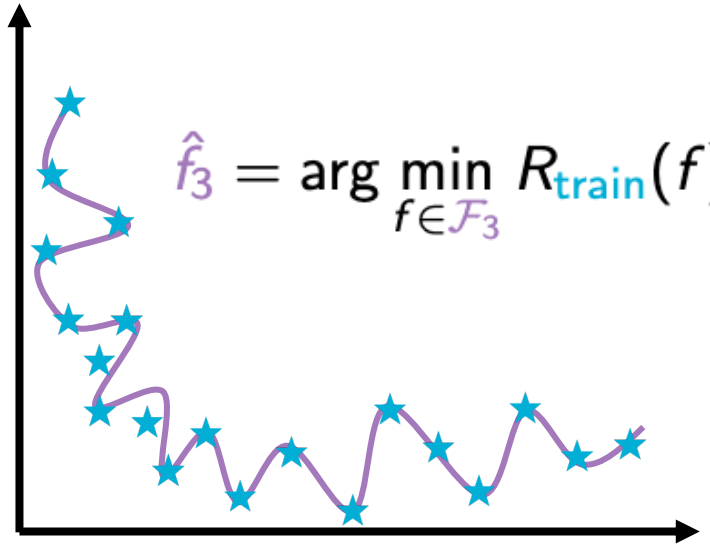


Overfitting (and underfitting)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f) \qquad R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$



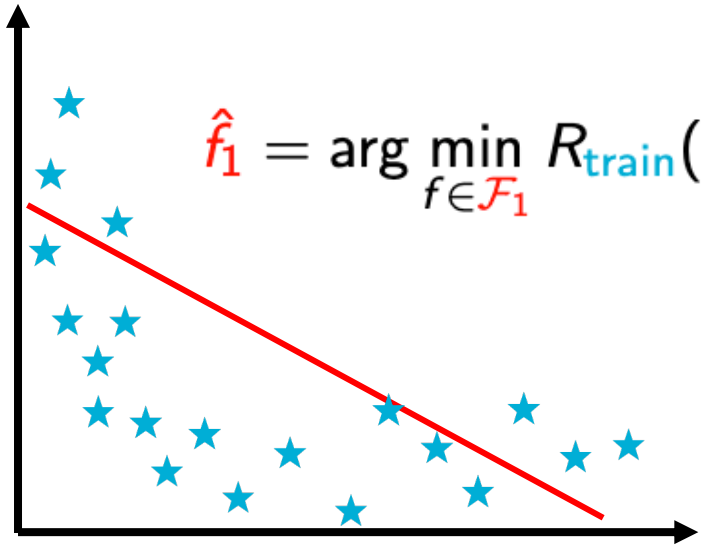


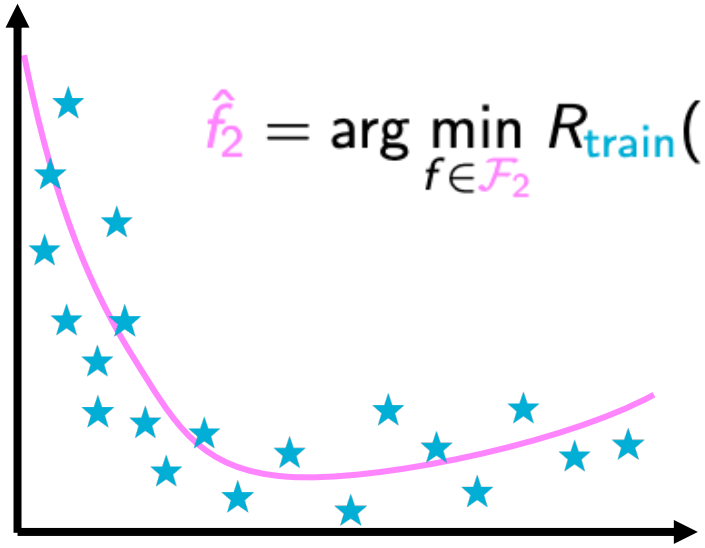


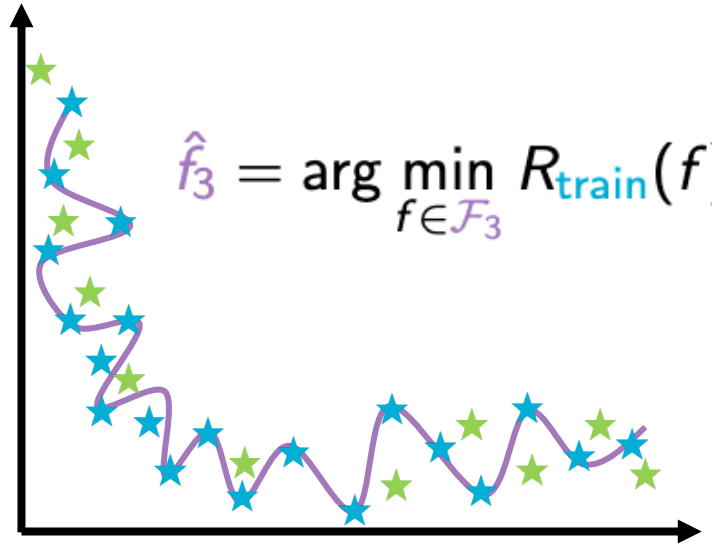
$R_{\text{train}}(\hat{f}_1) > R_{\text{train}}(\hat{f}_2) > R_{\text{train}}(\hat{f}_3) \longrightarrow$ Is $\hat{f}_3$ the best?

Overfitting (and underfitting)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f) \qquad R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$





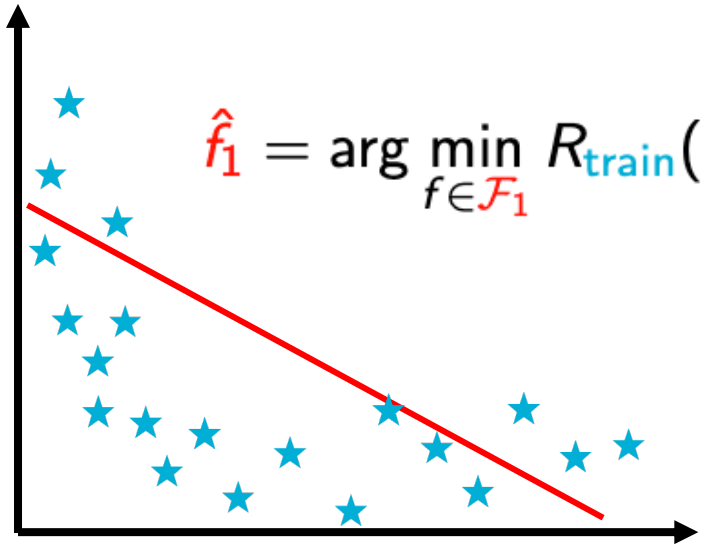


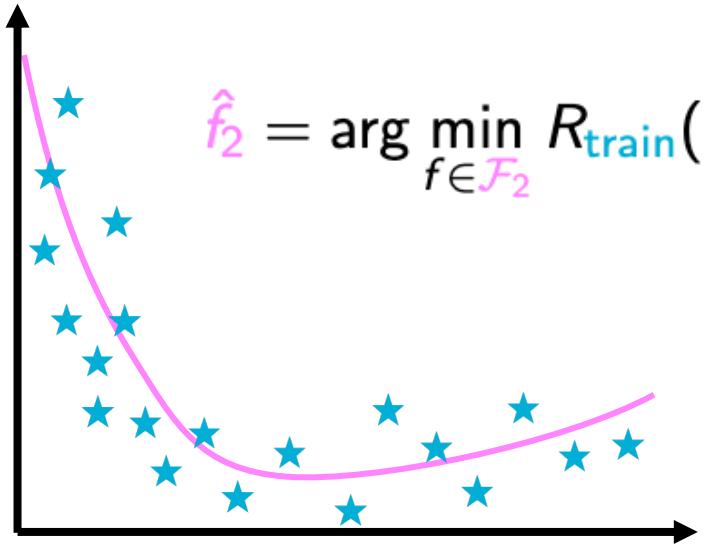
$R_{\text{train}}(\hat{f}_1) > R_{\text{train}}(\hat{f}_2) > R_{\text{train}}(\hat{f}_3) \longrightarrow$ Is $\hat{f}_3$ the best?

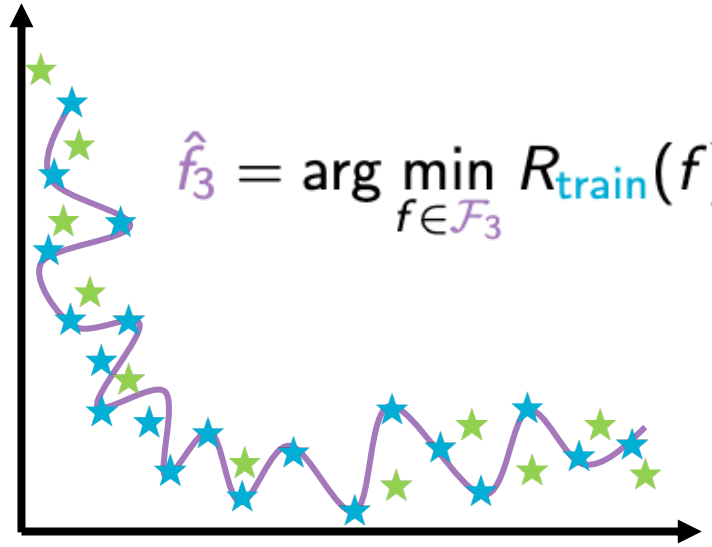
No, because $R_{\text{validation}}(\hat{f}_3) \gg R_{\text{train}}(\hat{f}_3)$

Overfitting (and underfitting)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f) \qquad R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$







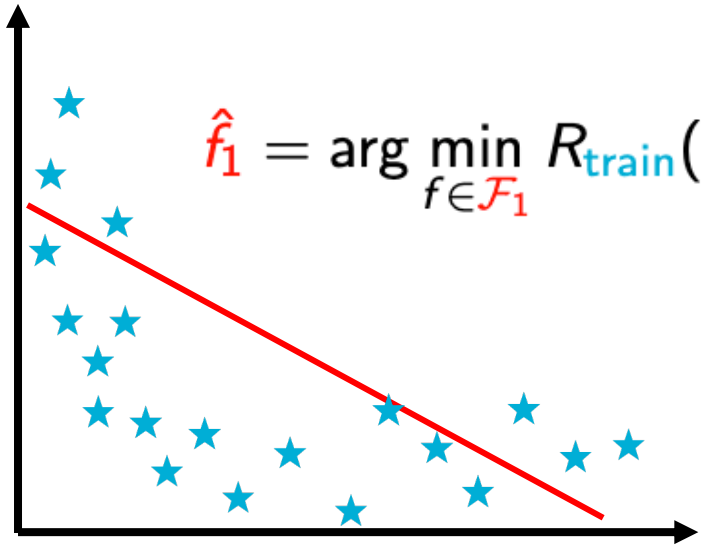
$R_{\text{train}}(\hat{f}_1) > R_{\text{train}}(\hat{f}_2) > R_{\text{train}}(\hat{f}_3) \longrightarrow$ Is $\hat{f}_3$ the best?

No, because $R_{\text{validation}}(\hat{f}_3) \gg R_{\text{train}}(\hat{f}_3)$

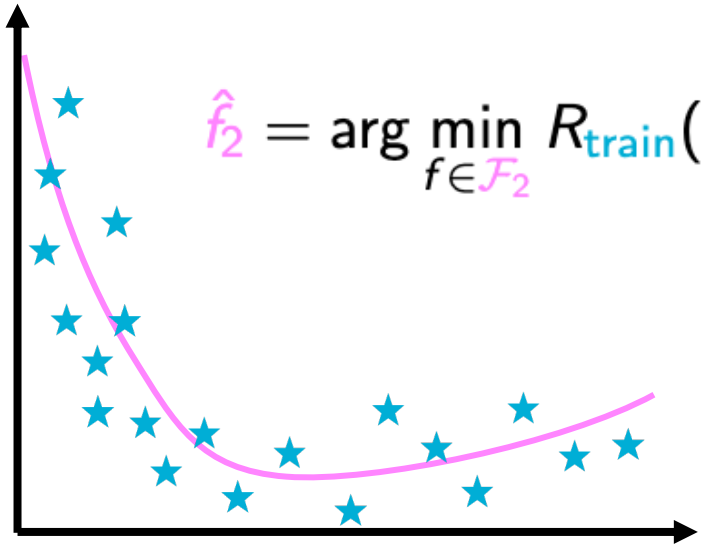
And likely $R_{\text{validation}}(\hat{f}_3) \gg R_{\text{validation}}(\hat{f}_2)$

Overfitting (and underfitting)

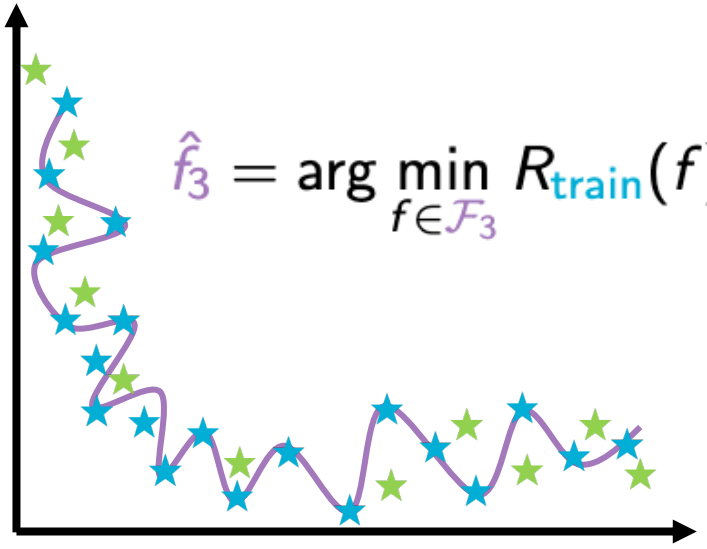
$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f) \qquad R_{\text{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$



Underfitting



Ideal fit



Overfitting

The bias-variance decomposition

Remember from statistical courses?

Bias: difference between the expected value of the estimator and the true value of the parameter being estimated

$$\text{Bias}(\hat{\theta}) = \mathbf{E}[\hat{\theta}] - \theta$$

Variance: expected squared deviation of the estimator from its expected value

$$\text{Var}(\hat{\theta}) = \mathbf{E}[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2]$$

Mean Squared Error (MSE): expected value of the squared deviation of the estimator from the true parameter value

$$\text{MSE}(\hat{\theta}) = \mathbf{E}[(\hat{\theta} - \theta)^2]$$

Bias-variance decomposition of MSE:

$$\text{MSE}(\hat{\theta}) = \text{Var}(\hat{\theta}) + [\text{Bias}(\hat{\theta})]^2$$

The bias-variance decomposition

Remem

Bias: difference betwe
the true value

$$\hat{f} = \arg\min_{f \in \mathcal{F}} R_{\text{train}}(f)$$

train is a random sample $\longrightarrow$ $\hat{f}$ is a random variable

$$\text{Bias}(\hat{\theta}) = \mathbf{E}[\hat{\theta}] - \theta$$

Variance: expected squared deviation of the estimator from its expected value

$$\text{Var}(\hat{\theta}) = \mathbf{E}[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2]$$

Mean Squared Error (MSE): expected value of the squared deviation of the estimator from the true parameter value

$$\text{MSE}(\hat{\theta}) = \mathbf{E}[(\hat{\theta} - \theta)^2]$$

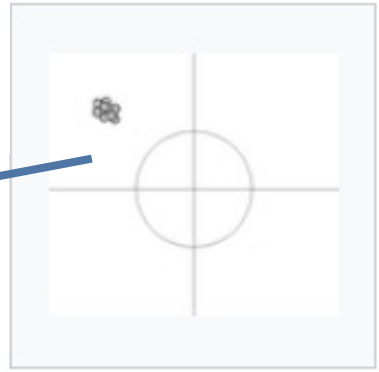
Bias-variance decomposition of MSE:

$$\text{MSE}(\hat{\theta}) = \text{Var}(\hat{\theta}) + [\text{Bias}(\hat{\theta})]^2$$

The bias-variance decomposition

$$\hat{f}_1 = \arg\min_{f \in \mathcal{F}_1} R_{\text{train}}(f)$$

When you learn, you always end up in the same place (i.e. same $\hat{f}$) but this place is wrong

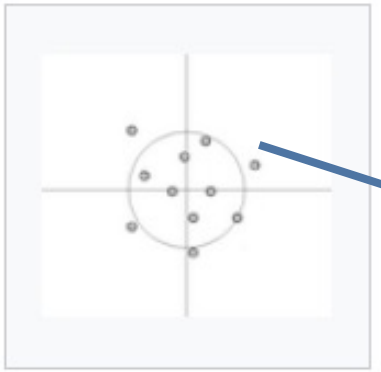


High bias, low variance

High bias, high variance

$$\hat{f}_2 = \arg\min_{f \in \mathcal{F}_2} R_{\text{train}}(f)$$

Low bias, low variance



Low bias, high variance

$$\hat{f}_3 = \arg\min_{f \in \mathcal{F}_3} R_{\text{train}}(f)$$

On average, you end up in the right place but there is a high variability (i.e. the $\hat{f}$ are very different from each other)

And since you will train only a few times at best, this is not great

Overfitting (and underfitting)

- **If the model is too "simple" for the data**
  - Underfitting
  - High bias

- **There is a relationship between model complexity and overfitting but it is not trivial**
  - High variance ⇒ complex model
  - The reverse may not be true (some complex models have low variance)

- **It is not that easy to define what is a complex model**
  - E.g. $f(x) = a\sin(bx)$ has only two parameters but can fit almost anything

Preventing overfitting

Empirical risk minimization

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

Preventing overfitting

Penalties (structural risk minimization)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) + \lambda H(f)$$

Preventing overfitting

Penalties

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) + \lambda H(f)$$

Regularization term:
High if the model is too “complex”

Preventing overfitting

Penalties

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) + \lambda H(f)$$

Trade-off parameter

- Small λ
  - Encourages fitting the data
  - At the risk of overfitting
- Large λ
  - Encourages simpler models
  - At the risk of underfitting

Regularization term:
High if the model is too “complex”

Preventing overfitting

Penalties

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) + \lambda H(f)$$

Trade-off parameter

- Small λ
  - Encourages fitting the data
  - At the risk of overfitting
- Large λ
  - Encourages simpler models
  - At the risk of underfitting

Needs to be tuned

Regularization term:
High if the model is too “complex”

Preventing overfitting

Penalties / contraints

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) + \lambda H(f)$$

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) \quad \text{subject to} \quad H(f) < t$$

The constraint reduces the set $\mathcal{F}$ of possible functions

Preventing overfitting

Example: ridge regression

(linear regression with l2 penalization term)

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right) + \lambda H(f)$$

where

$$f(x) = w_0 + \sum_{i=1}^{p} w_i x_i$$

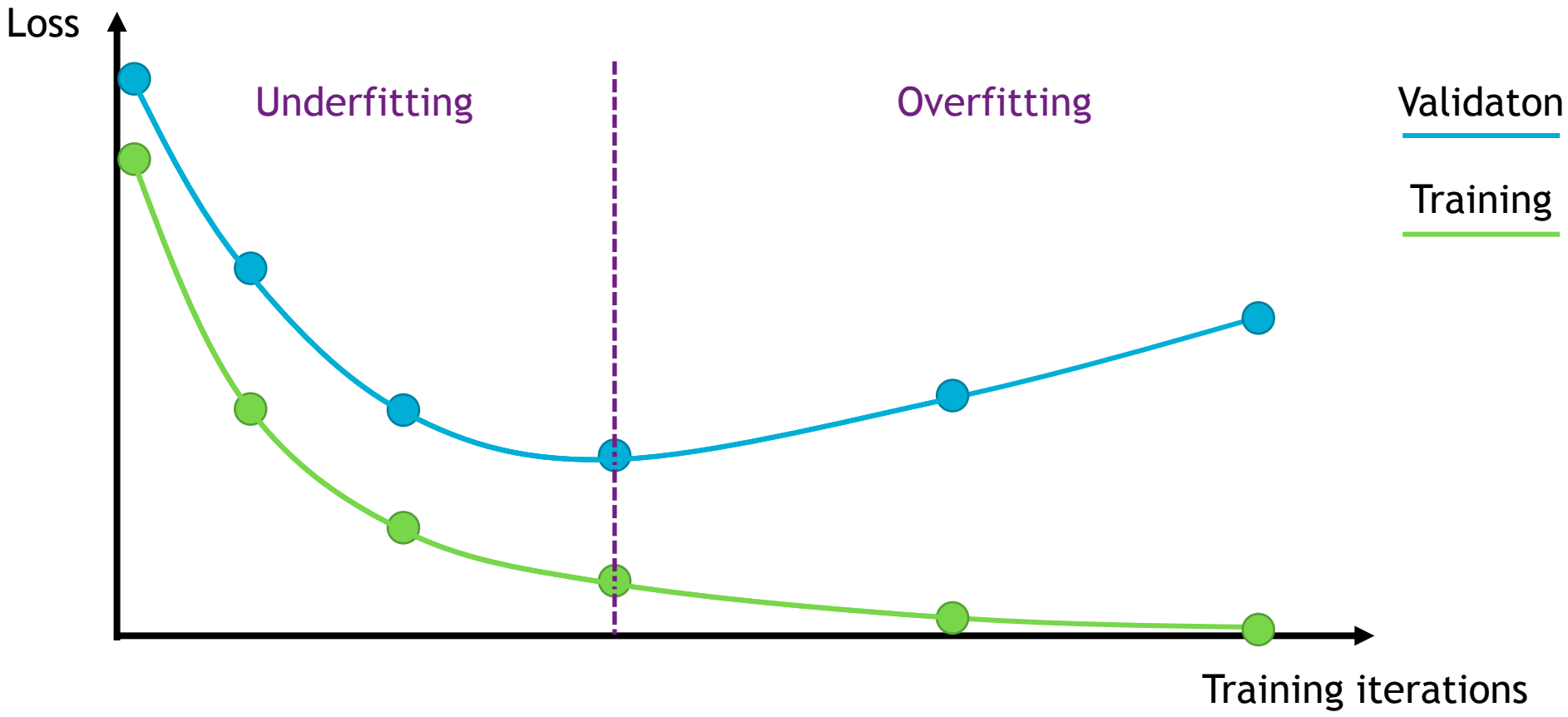
sklearn.linear_model.Ridge

and

$$H(f) = \|\mathbf{w}\|_2^2 = \sum_{i=1}^{p} w_i^2$$

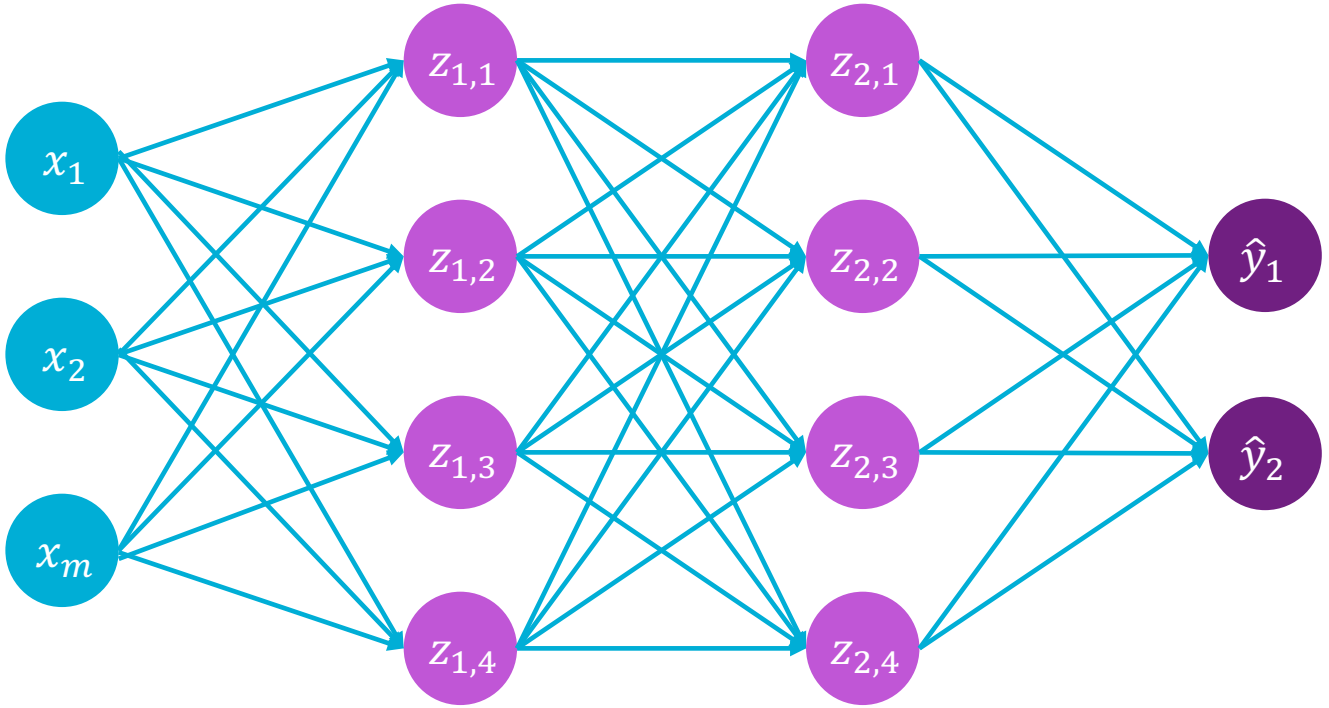
Preventing overfitting in deep learning

Tackling overfitting in DL: early stopping



Preventing overfitting in deep learning

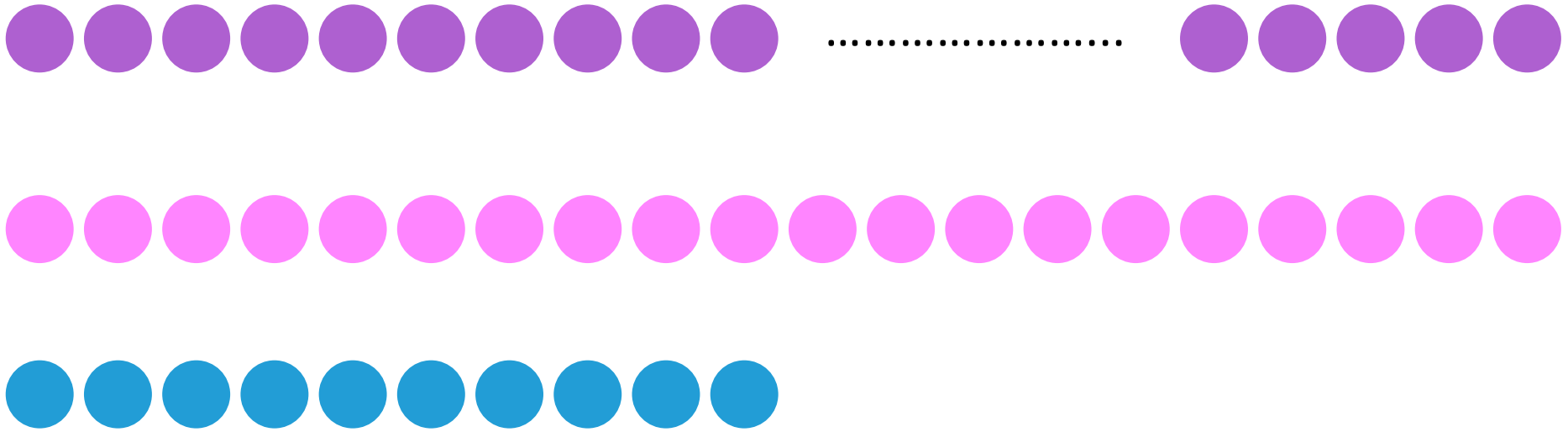
Tackling overfitting in DL: dropout



Model selection

- **Model selection**
  - We need to choose the family of functions
    - In practice we are going to test a few: linear models, penalized linear models, random forests, various deep learning architectures
  - For a given family of functions, there may be some hyperparameters to set
  - We may need to choose when we stop training

Model selection



Population

$$R(f) = \mathbf{E}[l(f(x), y)]$$

Sample

Training set

$$R_{\text{train}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

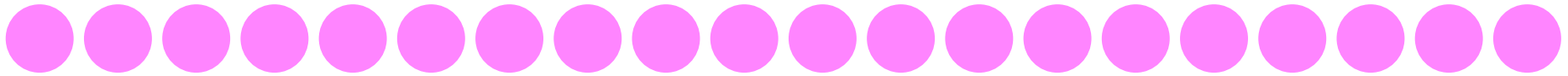
Of course, we cannot use the training set for model selection

Otherwise, we will simply choose the model that overfits the most

Model selection

Population

$$R(f) = \mathbf{E}[l(f(x), y)]$$



Sample

Training set

$$R_{\text{train}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

Validation set

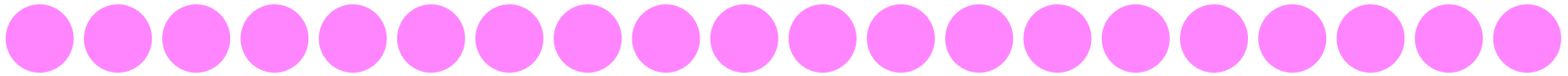
$$R_{\text{validation}}(\hat{f}) = \frac{1}{m}\sum_{i=1}^{m} \ell\left(y'^{(i)}, \hat{f}\left(x'^{(i)}\right)\right)$$

We will use the validation set for model selection

Model selection

Population

$$R(f) = \mathbf{E}[l(f(x), y)]$$



Sample

Training set

$$R_{\text{train}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

Validation set

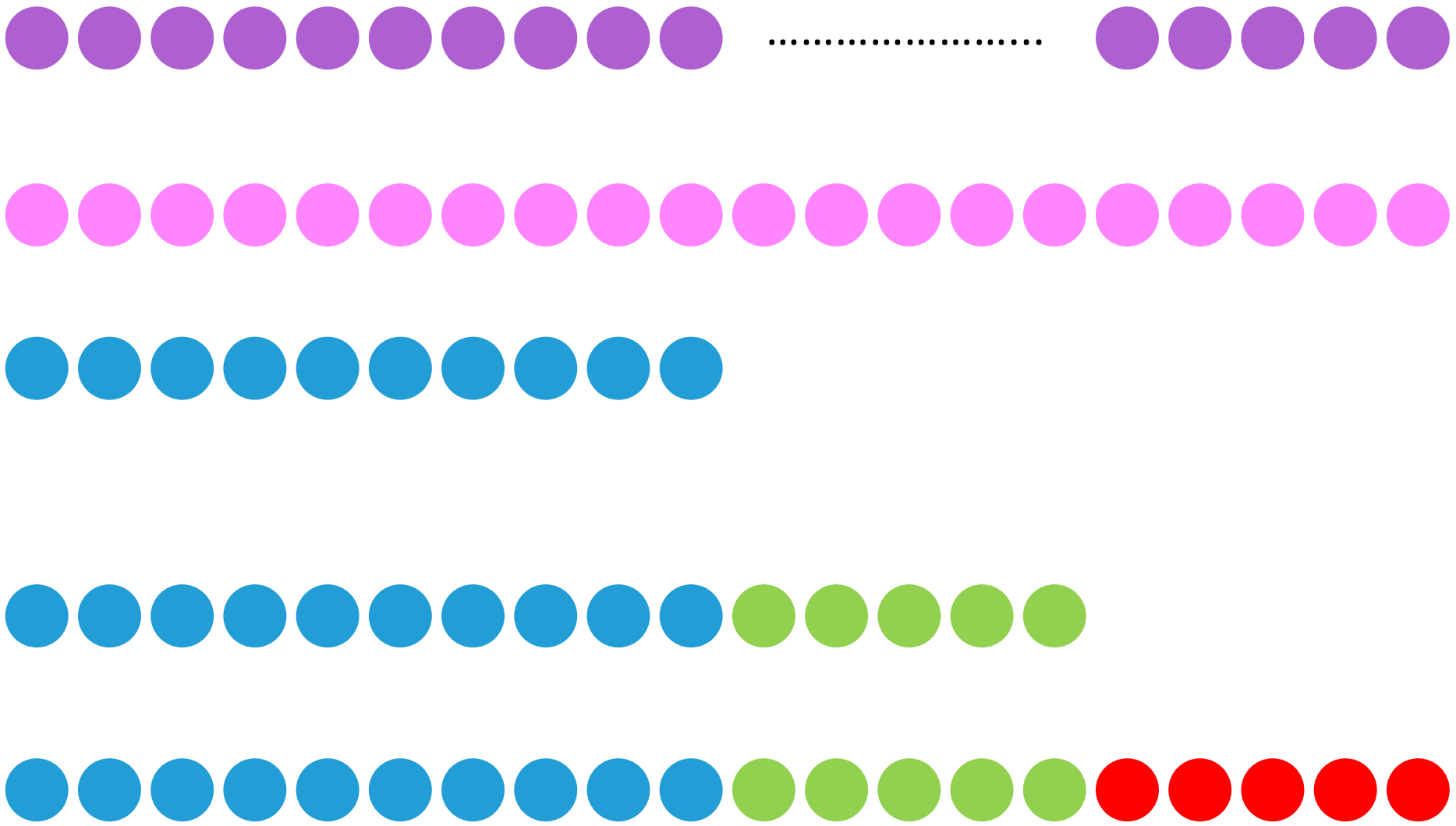
$$R_{\text{validation}}(\hat{f}) = \frac{1}{m}\sum_{i=1}^{m} \ell\left(y'^{(i)}, \hat{f}\left(x'^{(i)}\right)\right)$$

We will use the validation set for model selection

But then, how can we have an unbiased estimate of the performance?

The validation estimate is biased by model selection

Model selection



Population

$$R(f) = \mathbf{E}[l(f(x), y)]$$

Sample

Training set

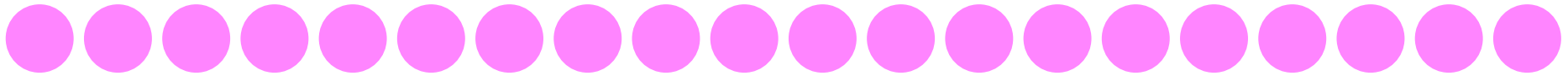
Validation set

Test set

Model selection

Population

$R(f) = \mathbf{E}[l(f(x), y)]$



Sample

Training set

Train several models $\hat{f}_1, \hat{f}_2, \hat{f}_3$….

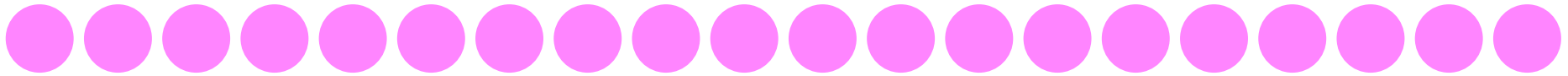
Validation set

Test set

Model selection

Population

$R(f) = \mathbf{E}[l(f(x), y)]$



Sample

Training set

Train several models $\widehat{f_1}, \widehat{f_2}, \widehat{f_3}$….

Validation set

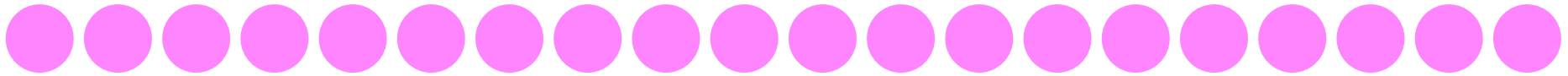
Select a model $\widehat{f_i}$ among $\widehat{f_1}, \widehat{f_2}, \widehat{f_3}$….

Test set

Model selection

Population

$R(f) = \mathbf{E}[l(f(x), y)]$



Sample

Training set

Train several models $\widehat{f_1}, \widehat{f_2}, \widehat{f_3}$….

Validation set

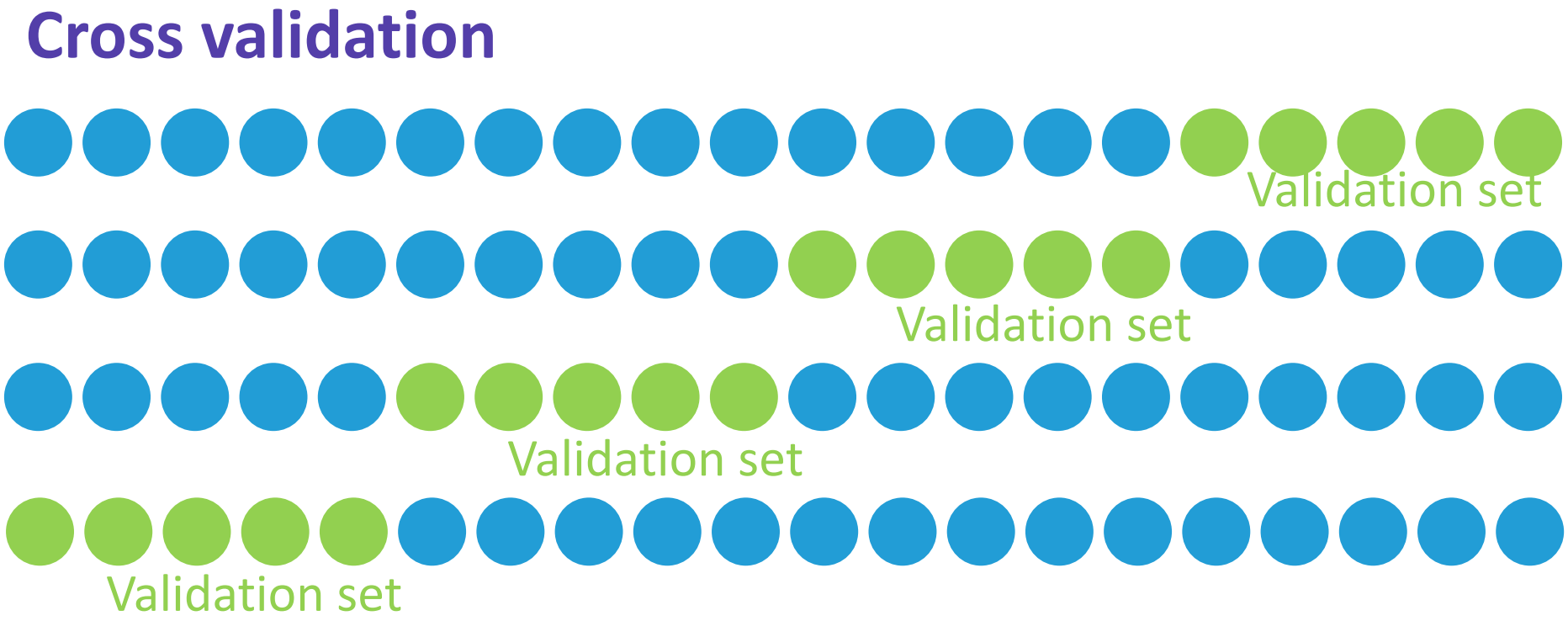
Select a model $\widehat{f_i}$ among $\widehat{f_1}, \widehat{f_2}, \widehat{f_3}$….

Evaluate the performances of $\widehat{f_i}$

Test set

Model selection

OK but if I have only one training set (and one validation set), I may get a lucky $\hat{f}$



Validation

Introduction

- Validation aims at **evaluating the performance of an ML model**
- **Ideally**, it should be representative of **how the model would perform in real life**
  - Difficult to achieve in practice, at least at the stage of research
- **At the very least**, it should provide an **unbiased estimate of how the model would perform on new data** that is similar to that used for training (but not the same data of course!!)
- Provide information about the **variability of the performance** and the **precision of its estimation**

Introduction

- We want a model that performs well on **new, never-before seen, data**.
- That is equivalent to saying we want our model **to generalise well**.
  - We want it to recognise only those characteristics of the data that are general enough to also apply to some unseen data
  - … while ignoring the characteristics of the training data that are overly specific to the training data
- Because of this, **we never test on training data, but use separate test data**

Introduction

- In this part, we address
  - **How to quantify the performance of the model?**
    - Performance metrics
  - **How to estimate the performance metrics?**
    - Validation strategies
  - **What kind of statistical analysis should be performed?**

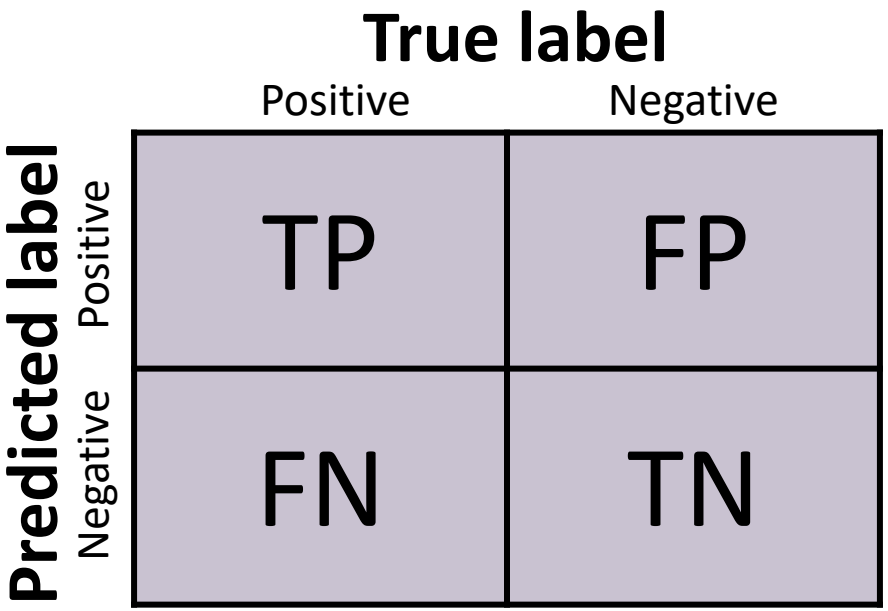
Some of this information is based upon (Varoquaux and Colliot, 2023 - https://hal.science/hal-03682454/)

Performance metrics

Metrics for classification

Metrics for classification

Confusion matrix



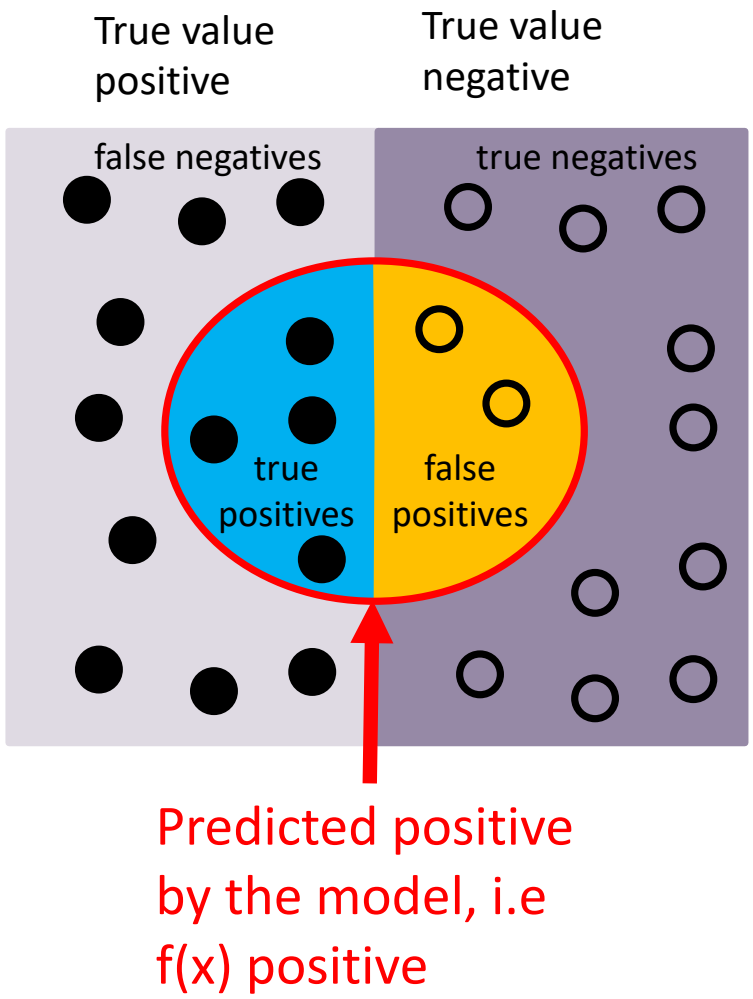
Metrics for classification

True Positives (TP): cases when the actual class of the data point was 1 and the predicted is also 1
Ex. The patient has cancer (1) and the model classifies his case as cancer(1)

True Negatives (TN): cases when the actual class of the data point was 0 and the predicted is also 0
Ex. The patient does not have cancer (0) and the model classifies his case as non-cancer (0)

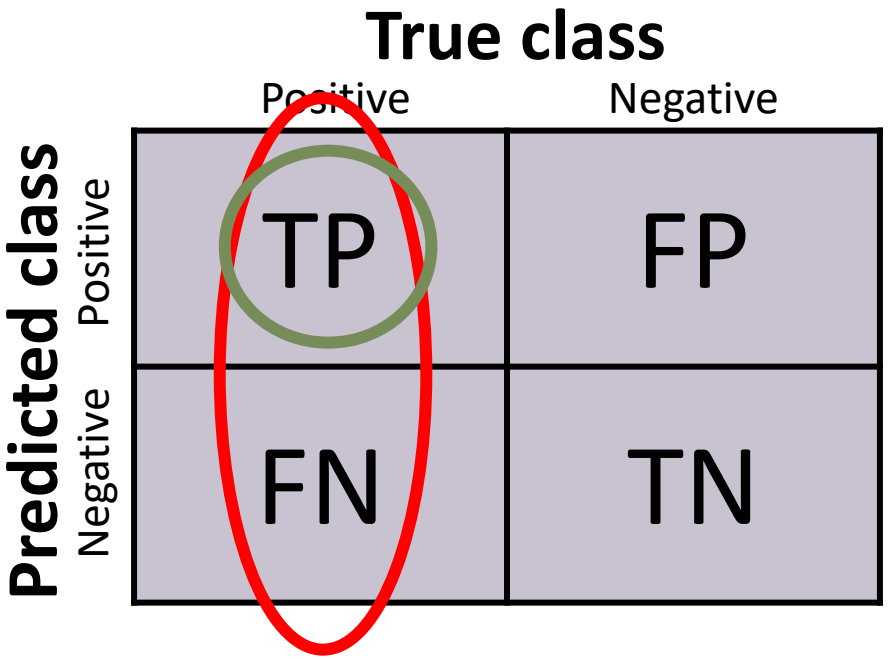
False Negatives (FN): cases when the actual class of the data point was 1 and the predicted is 0
Ex. The patient has cancer (1) and the model classifies his case as non-cancer(0)

False Positives (FP): cases when the actual class of the data point was 0 and the predicted is also 0
Ex. The patient does not have cancer (0) and the model classifies his case as cancer (1)



Metrics for classification

Sensitivity (also called recall)

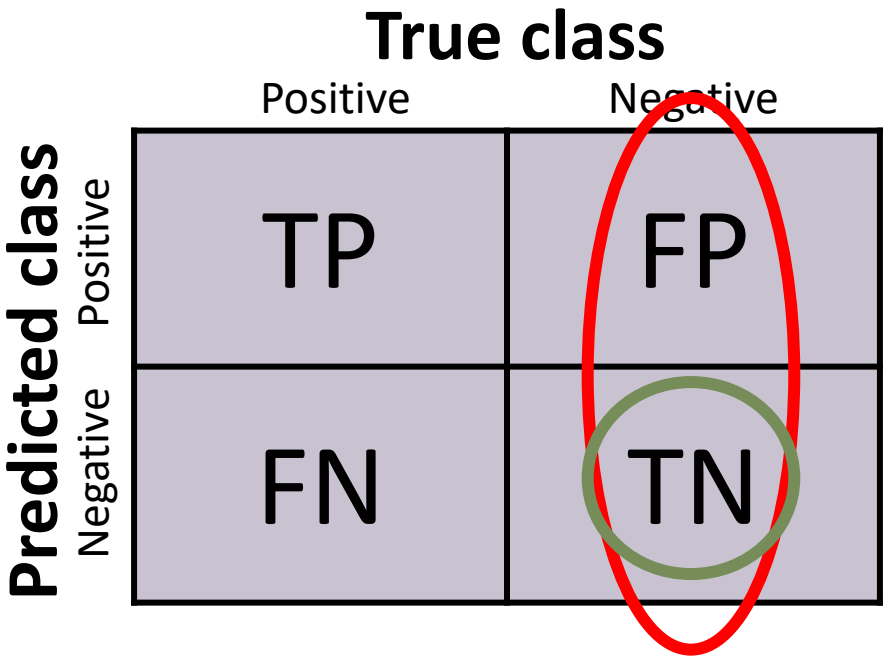


How much of the positives do we retrieve?

$$Sensitivity = Recall = \frac{TP}{TP + FN}$$

Metrics for classification

Specificity

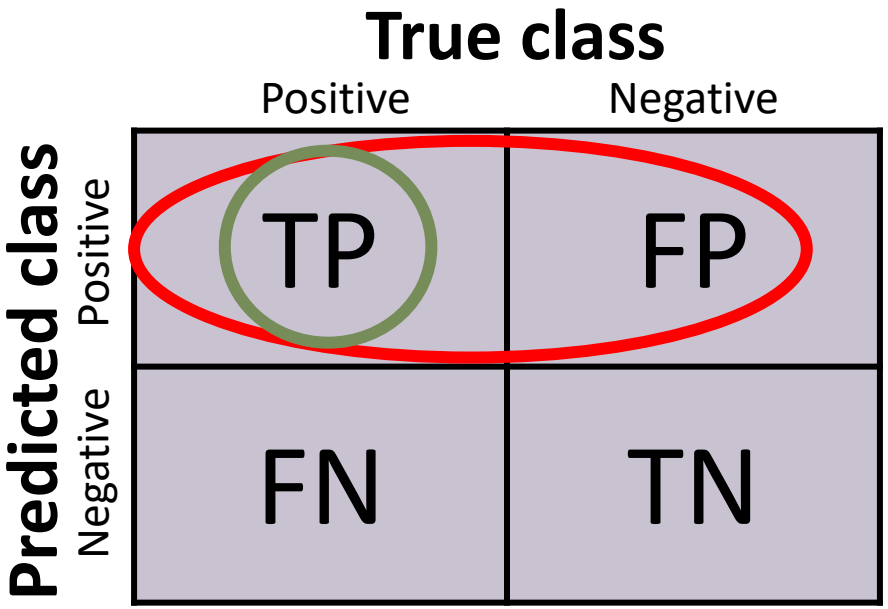


How much of the negatives do we retrieve?

$$Specificity = \frac{TN}{TN + FP}$$

Metrics for classification

Precision (also called positive predictive value - PPV)

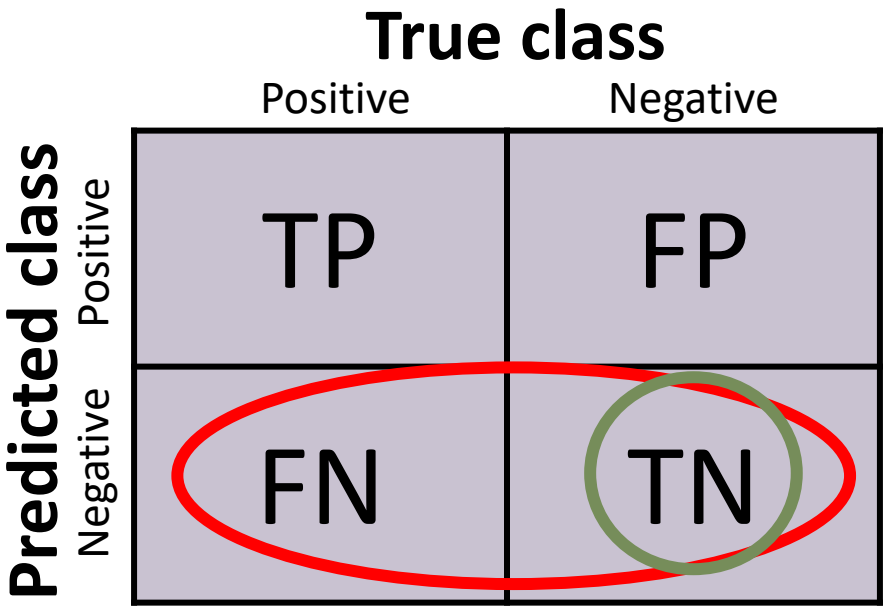


How much of those classified as positives are indeed positives?

$$PPV = Precision = \frac{TP}{TP + FP}$$

Metrics for classification

Negative predictive value - NPV



How much of those classified as negatives are indeed negatives?

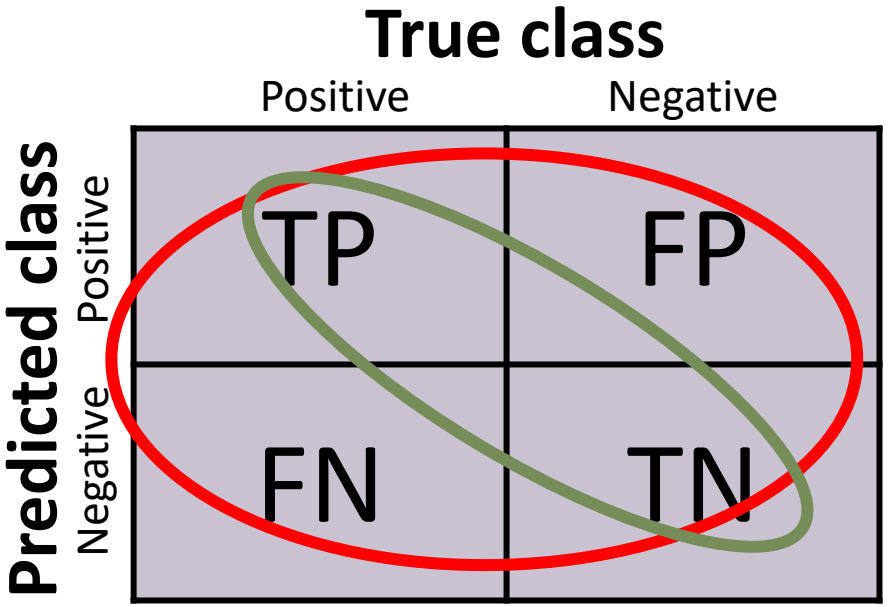
$$NPV = \frac{TN}{TN + FN}$$

Metrics for classification

The four previous metrics each describe only part of the confusion matrix

Often, one wants to have a summary in a single metric

Accuracy



Among all samples, how much are correctly classified?

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Metrics for classification

Problem with accuracy

- **Do not use when the data is imbalanced** (the number of cases in each class is not the same)
  - Example :
    - 990 non-cancer and 10 cancer
    - Trivial majority classifier: nobody has cancer
    - Accuracy: 99%

- Possible solution: **balanced accuracy (BA)**

$$BA = \frac{Sensitivity + Specificity}{2}$$

Metrics for classification

Problem with balanced accuracy

Suppose that you take a diagnostic test for a given disease

- The test turns out positive
- The **sensitivity of the test is 99%**, i.e. 99% of sick people are detected
- The **specificity is 90%**, i.e. 10% of healthy people are diagnosed as positive
- So **BA=95%** which is excellent
- What is the probability that you have the disease?

We don't have enough information.
Sensitivity=P(test positive|sick)
Specificity=P(test negative|healthy)
We are interested in P(sick|test positive)

Metrics for classification

We are interested in P(sick|test positive)

P(sick|test positive)=P(test positive|sick)*P(sick)/P(test positive)

P(sick|test positive)=**Sensitivity***P(sick)/P(test positive)

P(test positive)=P(test positive|healthy)*P(healthy)+ P(test positive|sick)*P(sick)

P(test positive)=(1-specificity)*(1-P(sick)) + sensitivity*P(sick)

Thus, we are missing P(sick) which is the **prevalence of the disease**.

Let the prevalence be 1/1000.

P(test positive)=0.10*0.999+0.99*0.001=0.0999+0.00099=10.089%

P(sick|test positive)=0.99*0.001/0.10089=0.01

So you have only 1% chance to be sick!

Metrics for classification

N= 10000 samples, prevalence= 0.001
Sensitivity: 0.99
Specificity: 0.99

PPV: 0.09
NPV: 0.9999
Accuracy: 0.99
Balanced accuracy: 0.99

is very bad: 91% of positively predicted samples are wrong

Looks good: 99% of samples are correctly classified

Balanced accuracy also looks good

Metrics for classification

Remember

- **For a diagnostic test, sensitivity and specificity are not enough**

- You need to also know the **prevalence**
  - Or the positive and negative predictive values

- Be careful at the prevalence in your sample. If you have a case-control study (for instance with equal numbers of cases and controls) the prevalence is likely wrong

- Ideally, you would need the prevalence in the situation in which the test is meant to be used (general population for a screening test)

Metrics for classification

NPV and PPV as a function of prevalence

- Sensitivity and specificity are fixed

$$\text{PPV} = \frac{\text{sensitivity} \times \text{prevalence}}{\text{sensitivity} \times \text{prevalence} + (1 - \text{specificity}) \times (1 - \text{prevalence})}$$

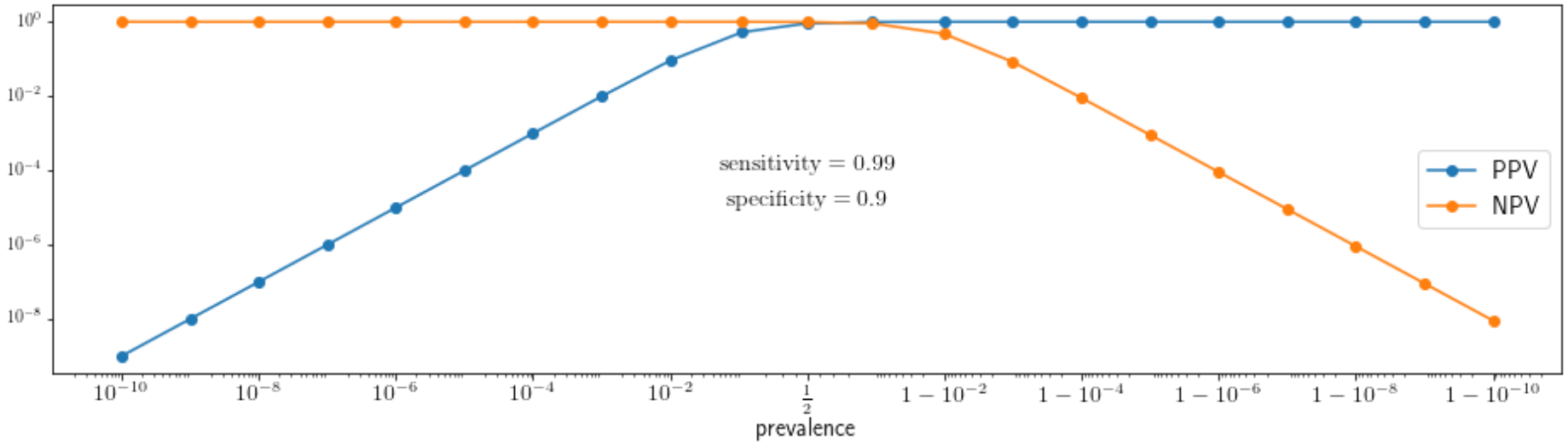
$$\text{PPV} \sim \frac{\text{sensitivity}}{1 - \text{specificity}} \times \text{prevalence} \quad \texttt{when} \quad \text{prevalence} \longrightarrow 0$$

$$\text{PPV} \longrightarrow 1 \quad \texttt{when} \quad \text{prevalence} \longrightarrow 1$$

$$\text{NPV} = \frac{\text{specificity} \times (1 - \text{prevalence})}{\text{specificity} \times (1 - \text{prevalence}) + (1 - \text{sensitivity}) \times \text{prevalence}}$$

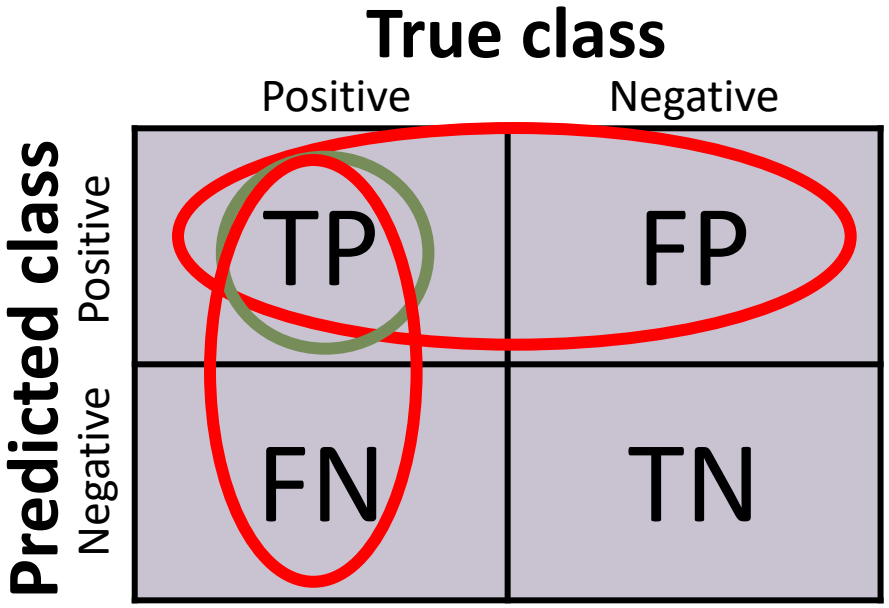
$$\text{NPV} \longrightarrow 1 \quad \texttt{when} \quad \text{prevalence} \longrightarrow 0$$

$$\text{NPV} \sim \frac{\text{specificity}}{1 - \text{sensitivity}} \times (1 - \text{prevalence}) \quad \texttt{when} \quad \text{prevalence} \longrightarrow 1$$



Metrics for classification

F1 score



Harmonic mean of precision and recall

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2\frac{Precision\ x\ Recall}{Precision + Recall}$$

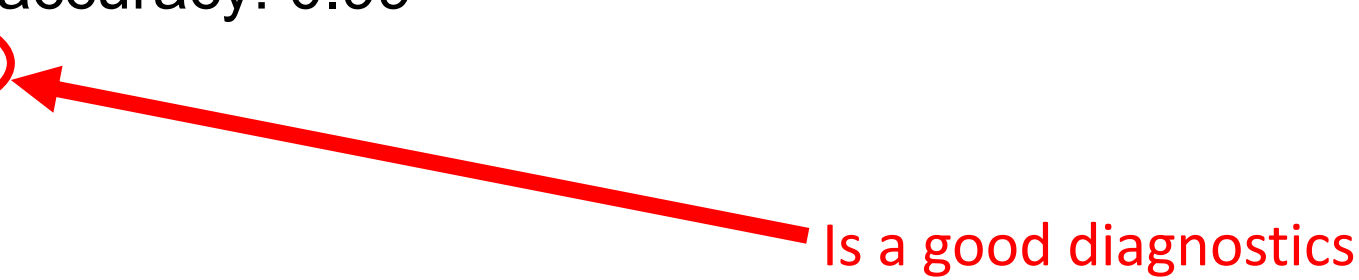
$$F1 = \frac{2TP}{2TP + FP + FN}$$

Metrics for classification

Coming back to the previous example

N= 10000 samples, prevalence= 0.001
Sensitivity: 0.99
Specificity: 0.99

PPV: 0.09
NPV: 0.9999
Accuracy: 0.99
Balanced accuracy: 0.99
F1: 0.16



Metrics for classification

N= 10000 samples, prevalence= 0.9999
Sensitivity: 0.99
Specificity: 0.99

PPV: 0.9999
NPV: 0.0098
Accuracy: 0.99
Balanced accuracy: 0.98
F1: 0.994

Is a poor diagnostics

Metrics for classification

N= 10000 samples, prevalence= 0.9999
Sensitivity: 0.99
Specificity: 0.99

PPV: 0.9999
NPV: 0.0098
Accuracy: 0.99
Balanced accuracy: 0.98
F1: 0.994

Is a poor diagnostics

Solution: switch classes
F1: 0.019

Is a good diagnostics

F1 should focus on the minority class to be informative

Metrics for classification

Matthews Correlation Coefficient (MCC)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

Makes use of all the information in the confusion matrix

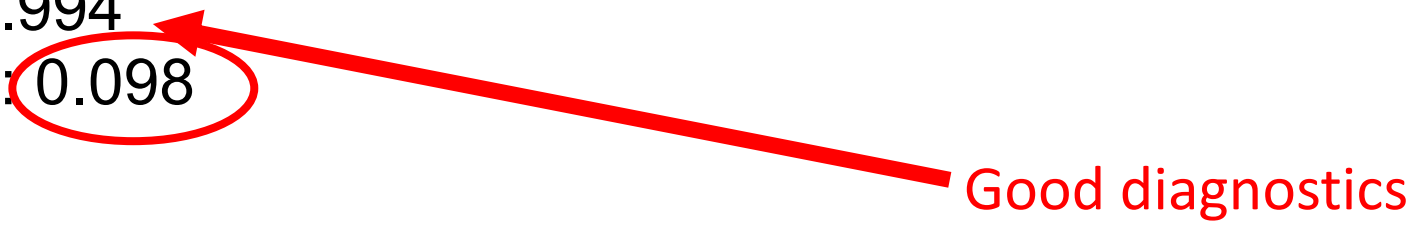
Ranges between +1 and -1
+1 is perfect prediction
0 is random prediction
-1 is perfectly wrong prediction

Metrics for classification

Coming back to the previous example

N= 10000 samples, prevalence= 0.9999
Sensitivity: 0.99
Specificity: 0.99

PPV: 0.9999
NPV: 0.0098
Accuracy: 0.99
Balanced accuracy: 0.98
F1: 0.994
MCC: 0.098



Metrics for classification

Conclusion

- Accuracy and BA **are useful because they are easy to interpret**. However, taken alone, **they are not sufficient and can be misleading**

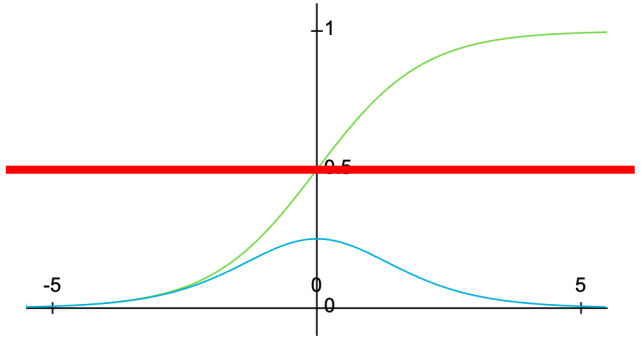
- Same thing for F1

- MCC is a good summary metric but probably less intuitive

Metrics for classification

Continuous outputs

- Many ML methods output continuous values
- This is in particular the case of neural networks
- Often one simply takes the class with highest probability
- However, there are applications where one is interested to study the performance for varying thresholds on the output

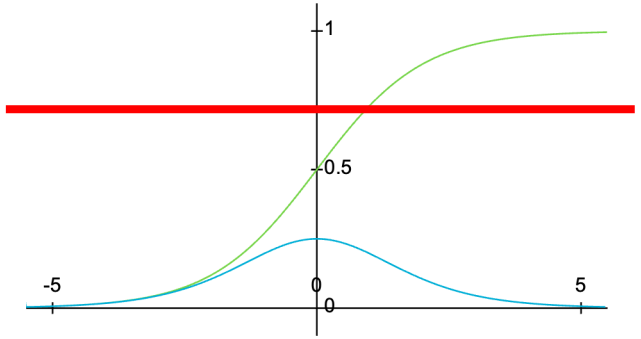


$$g(z) = \frac{1}{1 + e^{-z}}$$

Metrics for classification

Continuous outputs

- Many ML methods output continuous values
- This is in particular the case of neural networks
- Often one simply takes the class with highest probability
- However, there are applications where one is interested to study the performance for varying thresholds on the output

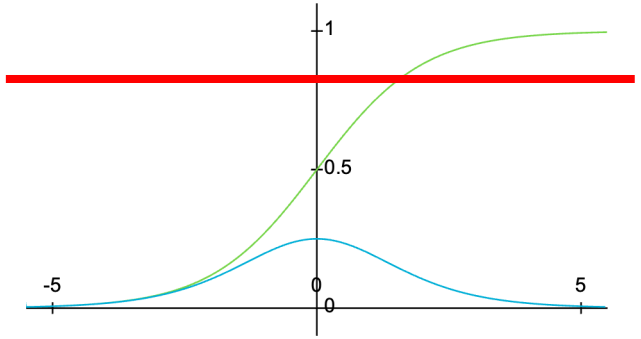


$$g(z) = \frac{1}{1 + e^{-z}}$$

Metrics for classification

Continuous outputs

- Many ML methods output continuous values
- This is in particular the case of neural networks
- Often one simply takes the class with highest probability
- However, there are applications where one is interested to study the performance for varying thresholds on the output

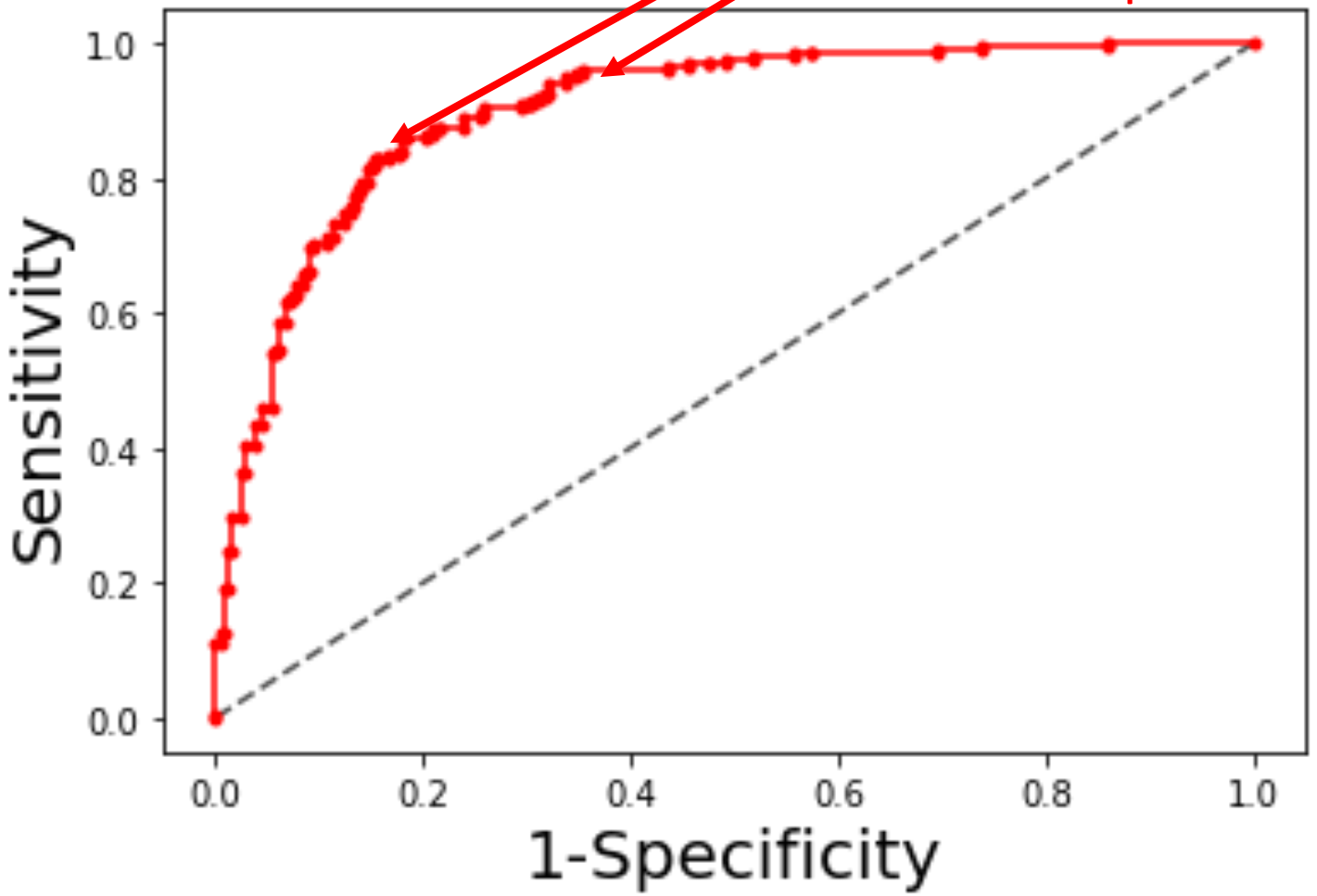


$$g(z) = \frac{1}{1 + e^{-z}}$$

Metrics for classification

ROC curve

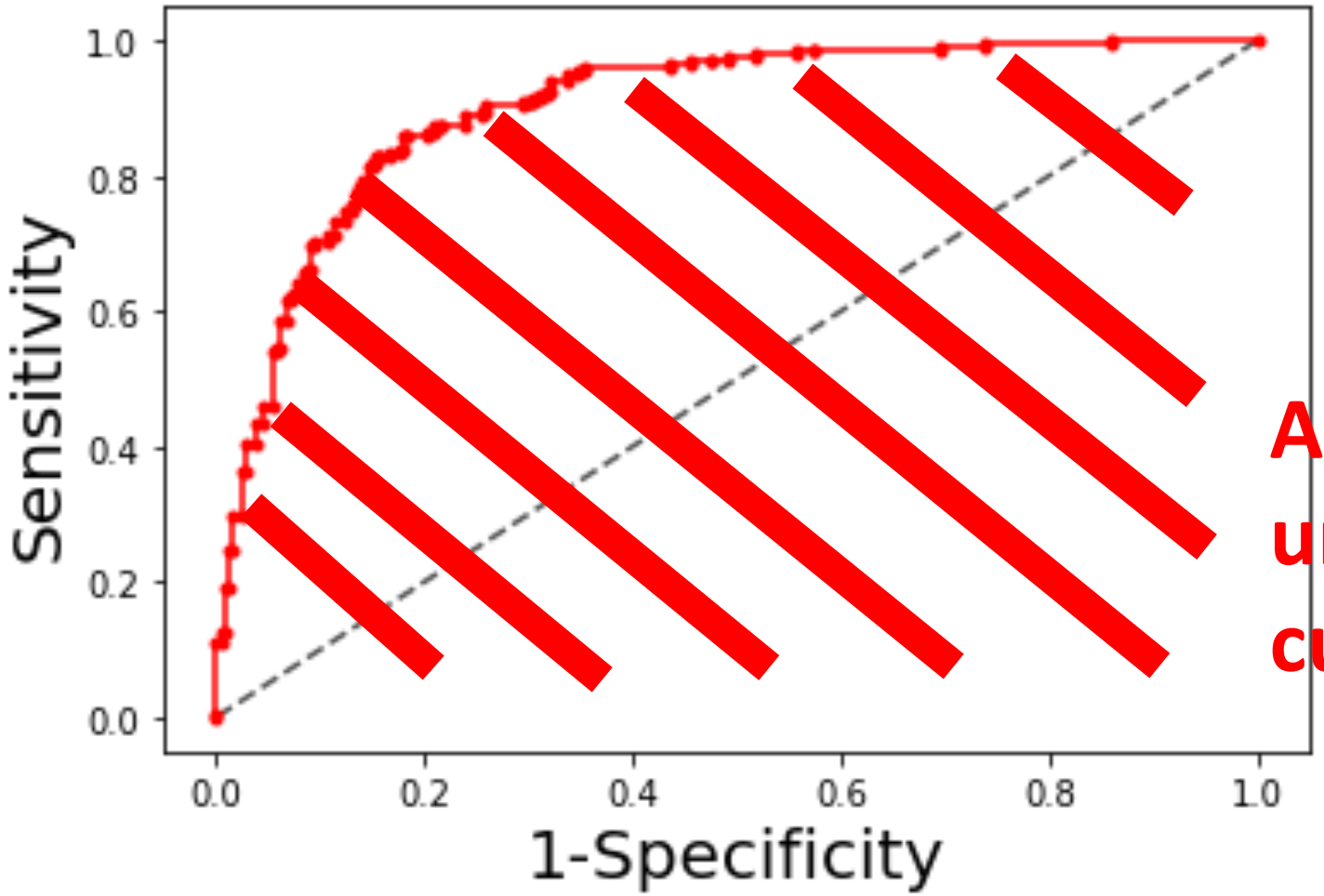
The curve corresponds to different thresholds on the output



Metrics for classification

ROC curve

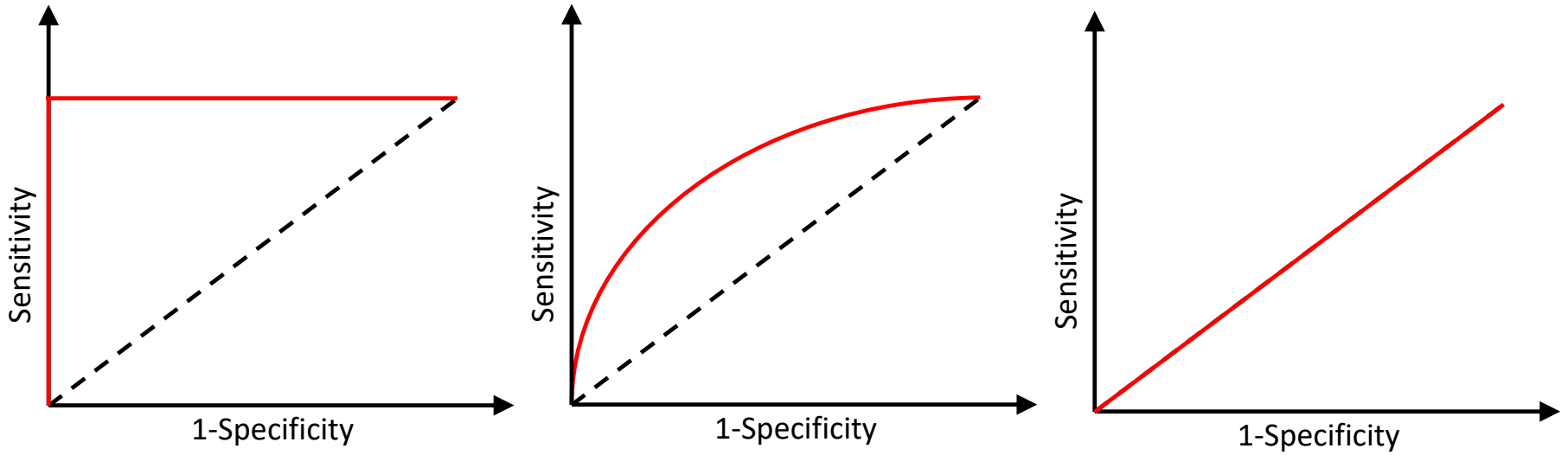
Interpretation of ROC AUC: probability that a positive sample has a higher classification score (as positive) than a negative sample



AUC (Area under curve)

Metrics for classification

ROC curve

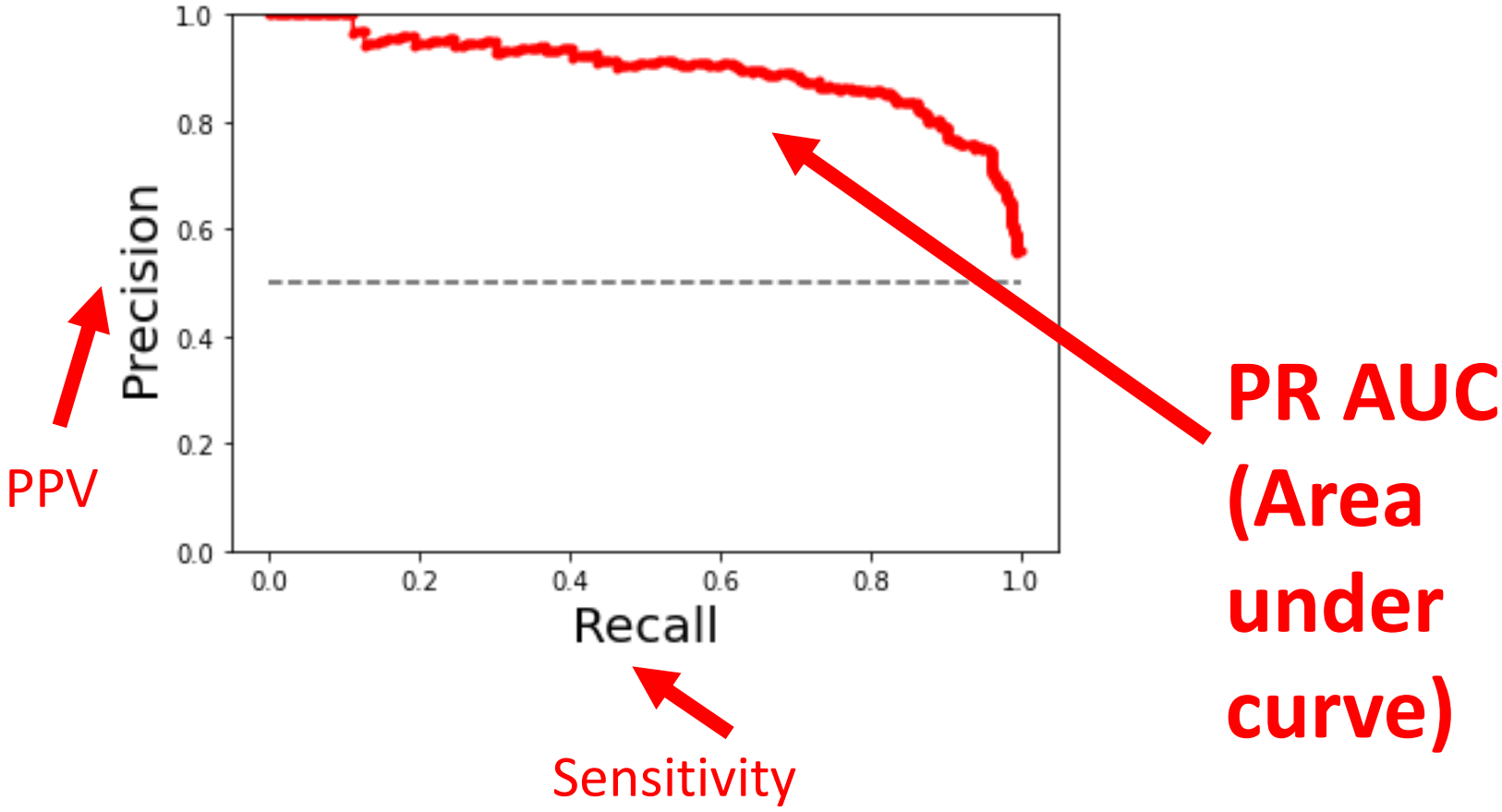


Perfect
AUC=1

Random
AUC=0.5

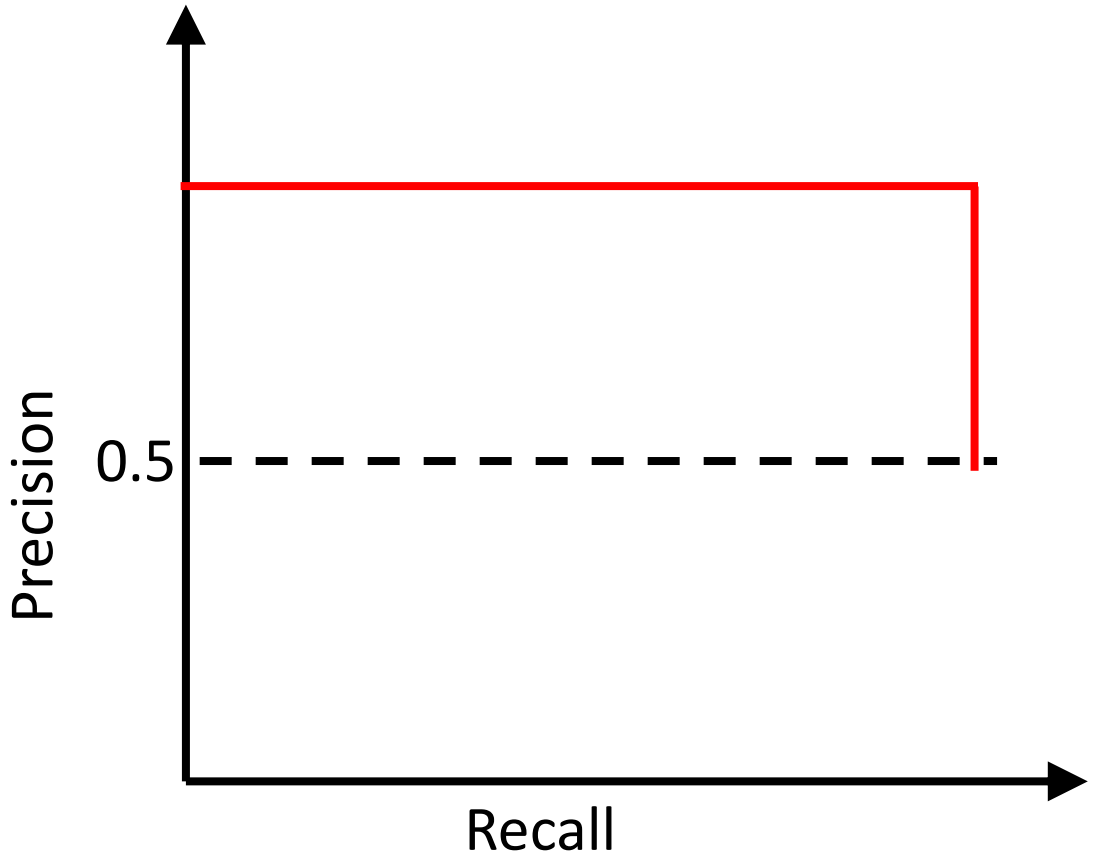
Metrics for classification

Precision-Recall (PR) curve



Metrics for classification

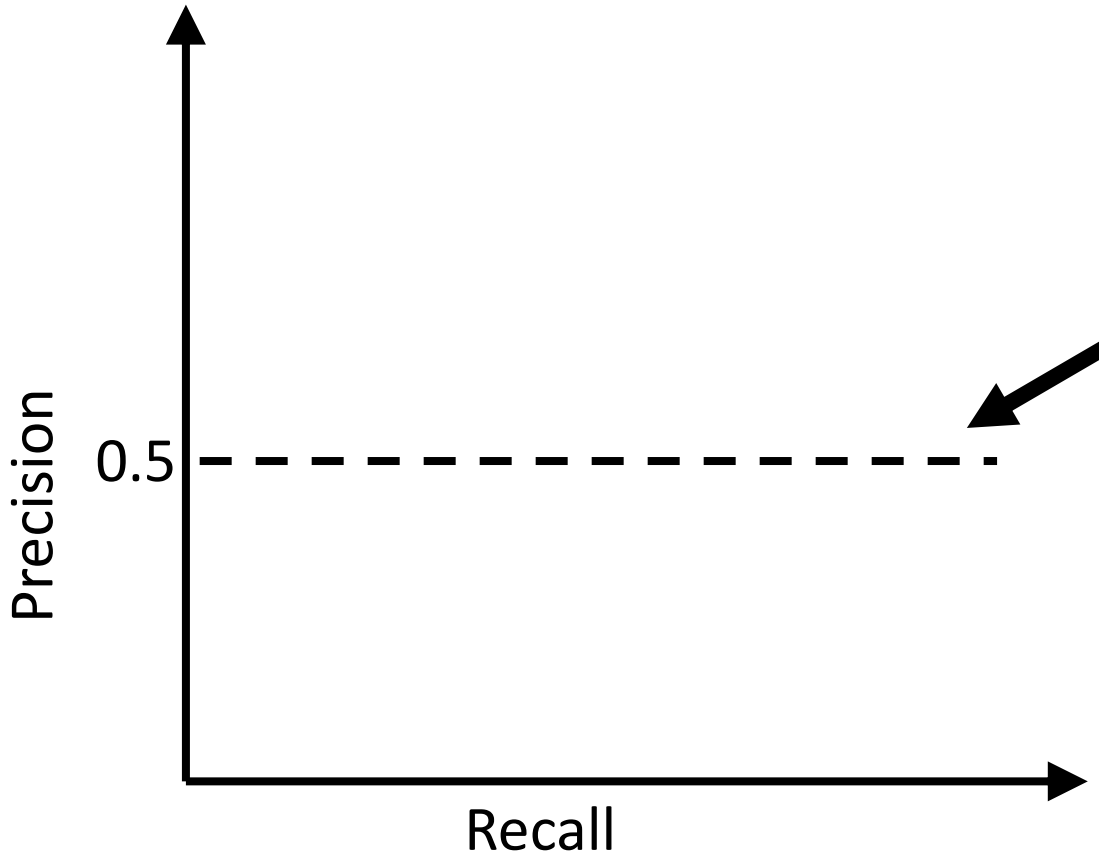
Precision-Recall (PR) curve



Perfect
PR AUC=1

Metrics for classification

Precision-Recall (PR) curve

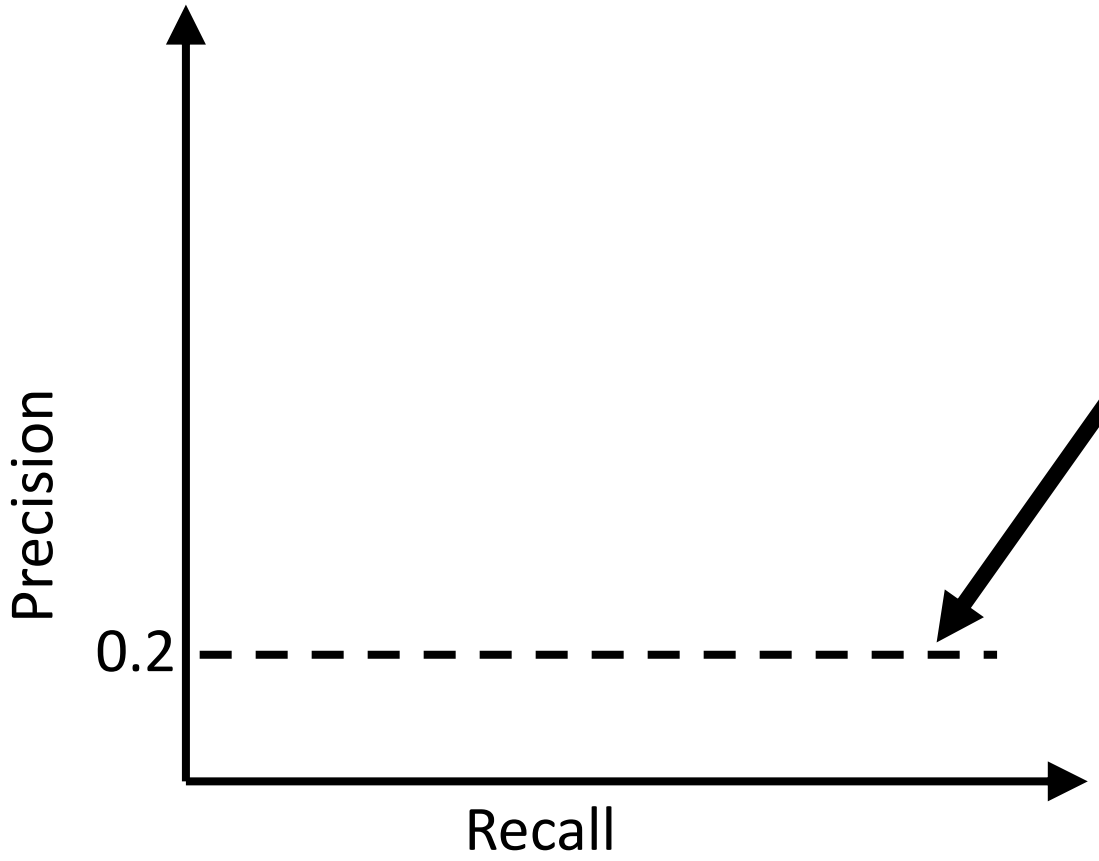


Dummy classifier performance depends on prevalence

Prevalence=0.5
PR AUC=0.5

Metrics for classification

Precision-Recall (PR) curve



Dummy classifier performance depends on prevalence

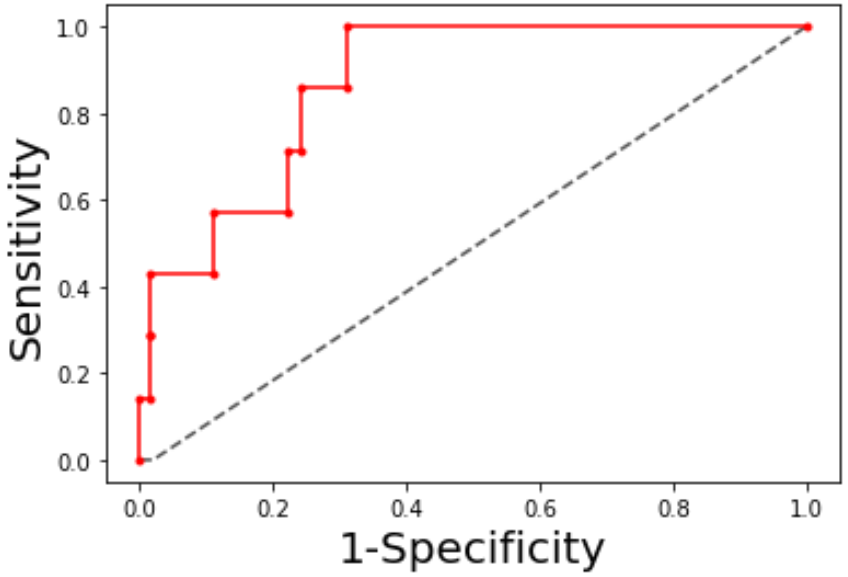
Prevalence=0.2
PR AUC=0.2

Interpretation of the PR AUC depends on the prevalence

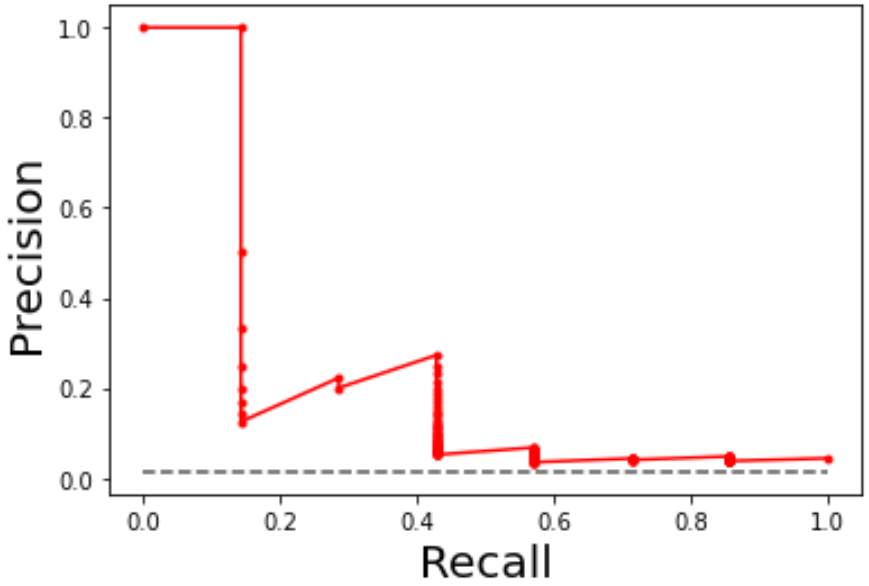
Metrics for classification

Imbalanced datasets

Example for prevalence=0.01



ROC AUC=0.87



PR AUC=0.23

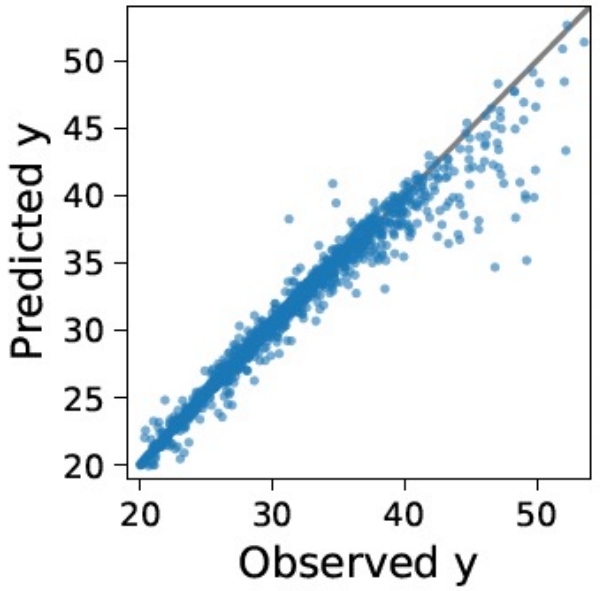
As accuracy, ROC AUC can be misleading when the dataset is imbalanced

Metrics for regression

Metrics for regression

Visualize prediction errors

Figure 6: ***Visualizing prediction errors*** *– plotting the predicted outcome as a function of the observed one enables to detect structure in the error beyond summary metric. Here the error increases for large values of y, for which there is also a systematic undershoot.*



Metrics for regression

R2 - coefficient of determination

$$R2 = 1 - \frac{\mathrm{SS}_{res}}{\mathrm{SS}_{tot}}$$

$$\mathrm{SS}_{res} = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

$$\mathrm{SS}_{tot} = \sum_{i=1}^{n} (y^{(i)} - \overline{y^{(i)}})^2$$

- This is computed on the test set (*out-of-sample*)
  - Can be negative (hence call it R2 and not R^2)
  - Is not the square of the correlation coefficient
- Sort of ”explained variance” (but for many authors, explained variance ignores bias)
- Do not use the correlation coefficient (between y and $\hat{y}$) because it discards errors on the mean and the scale -> **important in practice**
- Do not use to compare models because it depends on variance of y

Metrics for regression

Absolute error measures: RMSE and MAE

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}{n}}$$

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|}{n}$$

- Good to compare models
- Give an error in the scale of the outcome (e.g. outcome in years, error in years)
- MAE is easier to interpret
- RMSE will put more weight on rare large errors

$$error = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 100]$$

$$\text{MAE} = 10 \qquad \text{RMSE} \approx 30.17$$

Note that if the error was uniformly equal to the same value (10, for instance), both measures would give the same result.

More about metrics

There are dozens of metrics depending on

- Data types (images, text….)
- Task (classification, segmentation…)
- Problem at hand

Choose the right ones

- And never rely on a single metric

A great resource if you work with images

nature methods

Perspective

https://doi.org/10.1038/s41592-023-02150-0

Understanding metric-related pitfalls in image analysis validation

Received: 9 February 2023

Accepted: 12 December 2023

Published online: 12 February 2024

Check for updates

A list of authors and their affiliations appears at the end of the paper

Validation metrics are key for tracking scientific progress and bridging the current chasm between artificial intelligence research and its translation into practice. However, increasing evidence shows that, particularly in image analysis, metrics are often chosen inadequately. Although taking into account the individual strengths, weaknesses and limitations of validation metrics is a critical prerequisite to making educated choices, the relevant knowledge is currently scattered and poorly accessible to individual researchers. Based on a multistage Delphi process conducted by a multidisciplinary expert consortium as well as extensive community feedback, the present work provides a reliable and comprehensive common point of access to information on pitfalls related to validation metrics in image analysis. Although focused on biomedical image analysis, the addressed pitfalls generalize across application domains and are categorized according to a newly created, domain-agnostic taxonomy. The work serves to enhance global comprehension of a key topic in image analysis validation.

Reinke et al, Nature Methods, 2024

https://www.nature.com/articles/s41592-023-02150-0

nature methods

Perspective

https://doi.org/10.1038/s41592-023-02151-z

Metrics reloaded: recommendations for image analysis validation

Received: 9 February 2023

Accepted: 12 December 2023

Published online: 12 February 2024

Check for updates

A list of authors and their affiliations appears at the end of the paper

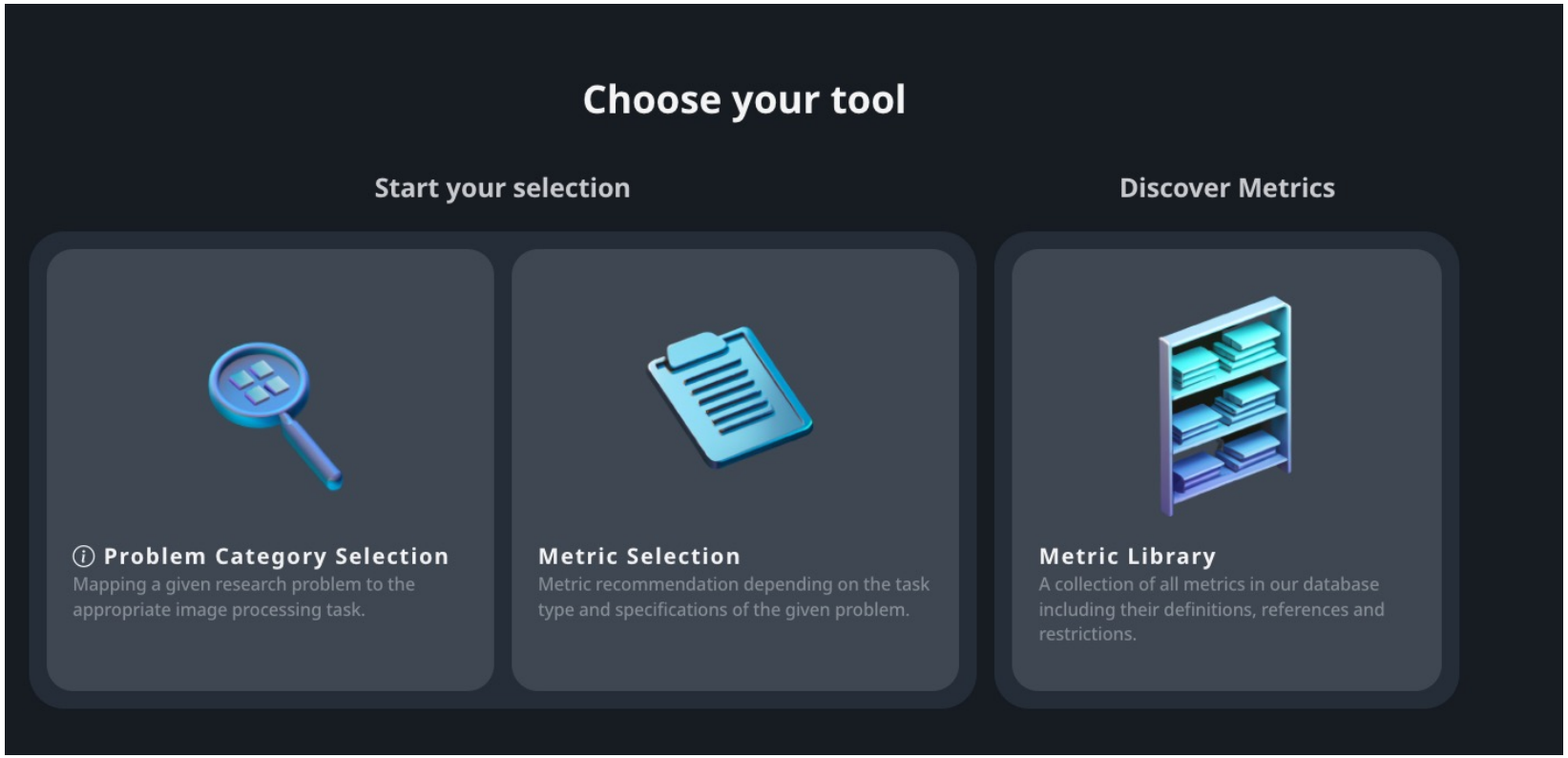
Increasing evidence shows that flaws in machine learning (ML) algorithm validation are an underestimated global problem. In biomedical image analysis, chosen performance metrics often do not reflect the domain interest, and thus fail to adequately measure scientific progress and hinder translation of ML techniques into practice. To overcome this, we created Metrics Reloaded, a comprehensive framework guiding researchers in the problem-aware selection of metrics. Developed by a large international consortium in a multistage Delphi process, it is based on the novel concept of a problem fingerprint—a structured representation of the given problem that captures all aspects that are relevant for metric selection, from the domain interest to the properties of the target structure(s), dataset and algorithm output. On the basis of the problem fingerprint, users are guided through the process of choosing and applying appropriate validation metrics while being made aware of potential pitfalls. Metrics Reloaded targets image analysis problems that can be interpreted as classification tasks at image, object or pixel level, namely image-level classification, object detection, semantic segmentation and instance segmentation tasks. To

Maier-Hein, Reinke et al, Nature Methods, 2024

https://www.nature.com/articles/s41592-023-02151-z

A great resource if you work with images

Metrics Reloaded



https://metrics-reloaded.dkfz.de/

Validation strategy


There is a a special place in hell for people
who validate using the training set

Validation strategies

Invidiuals

Validation strategies

Hold out

`sklearn.model_selection.ShuffleSplit(n_splits=1)`

Training set

Validation set

Larger training set:
better learning

Larger validation set:
better estimation of performance

But data is not infinite → **cross validation**
Idea: repeatedly exchange training and testing data

Validation strategies

Stratification

sklearn.model_selection.StratifiedSuffleSplit(n_splits=1)

Training set

Validation set

Keep the same proportion of each class in the training and validation sets

In the above example 1/3 of samples are diseased and 2/3 are healthy

Validation strategies

Stratification in a broader sense

In many cases, you want the **distribution of several variables** to be the same in the training and validation set (and not only the proportions of the different classes)

For example: age, sex…

This is **very important for medical data** (this issue may be less relevant in other areas such as computer vision)

Validation strategies

Stratification in a broader sense

Example

Table 2. Summary of participant demographics, mini-mental state examination (MMSE) and global clinical dementia rating (CDR) scores at baseline for ADNI.

	Subjects	Sessions	Age	Gender	MMSE	CDR
CN	330	1 830	74. 4 ± 5.8 [59.8, 89.6]	160 M / 170 F	29.1 ± 1.1 [24, 30]	0: 330
AD	336	1 106	75.0 ± 7.8 [55.1, 90.9]	185 M / 151 F	23.2 ± 2.1 [18, 27]	0.5: 160; 1: 175; 2: 1

Values are presented as mean ± SD [range]. M: male, F: female

Split into validation and test set while preserving **the most important variables**

Validation strategies

Stratification in a broader sense

It is often very difficult to acheive identical (or almost identical) distributions, in particular when controlling for many variables

In practice, one would often be happy if the mean and SD (for continuous variables) and the proportion (for categorical variables) are approximately preserved

Validation strategies

Stratification in a broader sense

Training set

	n_subjects	mean_age	std_age	min_age	max_age	sexF	sexM	mean_MMSE	std_MMSE	min_MMSE	max_MMSE
AD	236	74.995763	7.982102799	55.1	90.9	106	130	23.16949153	2.088325437	18	27
CN	230	74.42087	5.704597622	59.8	88.6	118	112	29.12173913	1.120153919	24	30

Validation set

	n_subjects	mean_age	std_age	min_age	max_age	sexF	sexM	mean_MMSE	std_MMSE	min_MMSE	max_MMSE
AD	100	74.993	7.330733319	55.9	90.3	45	55	23.25	1.986831649	19	27
CN	100	74.415	5.90662975	59.9	89.6	52	48	29.01	1.135737646	26	30

Validation strategies

Stratification in a broader sense

There is no scikit-learn function to perform this

One will often do this using ad-hoc procedures

This is usually done for a separated test set but not for a cross-validation

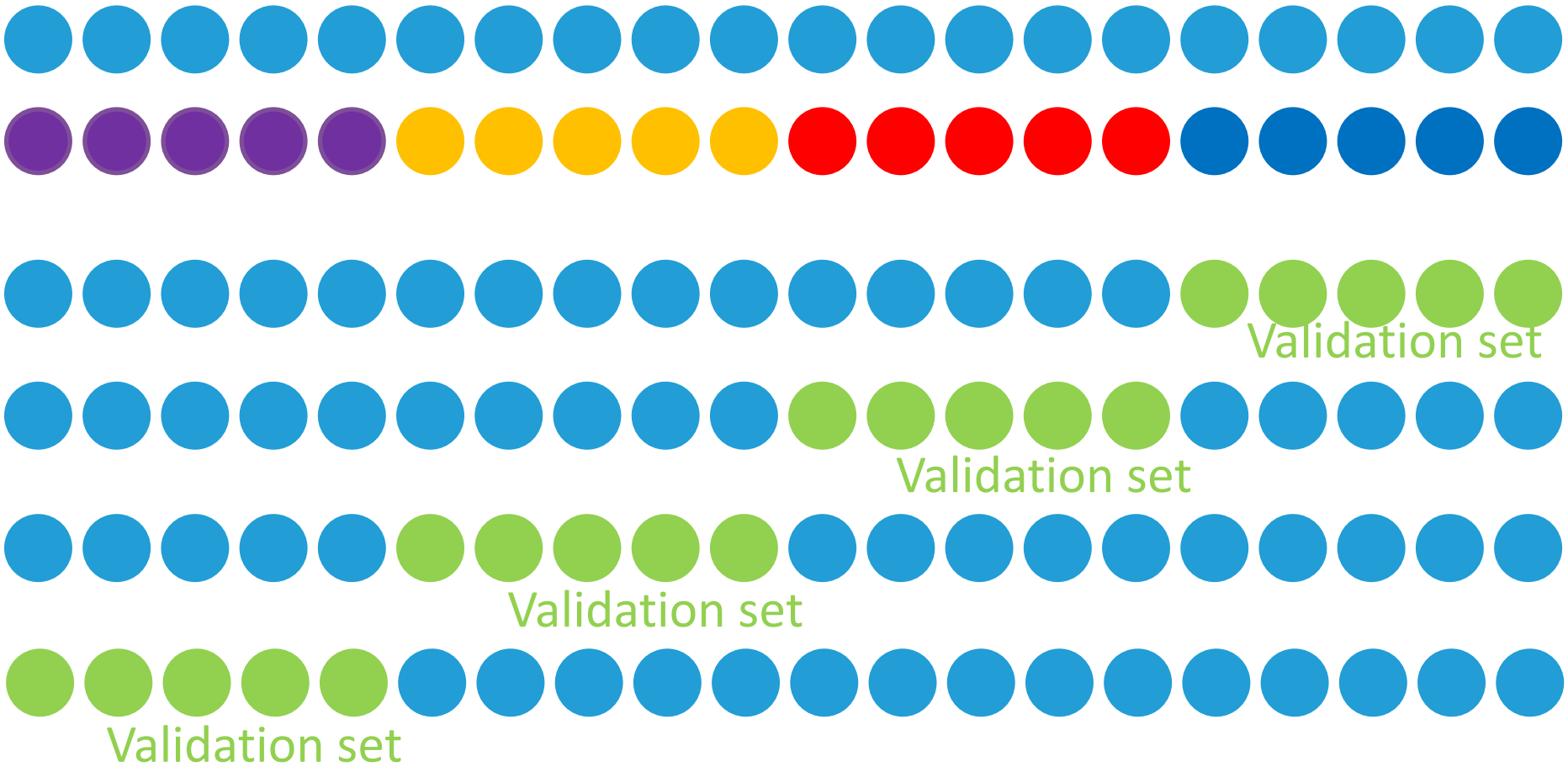
Validation strategies

k-fold cross validation

Here k=4

sklearn.model_selection.KFold

sklearn.model_selection.StratifiedKFold



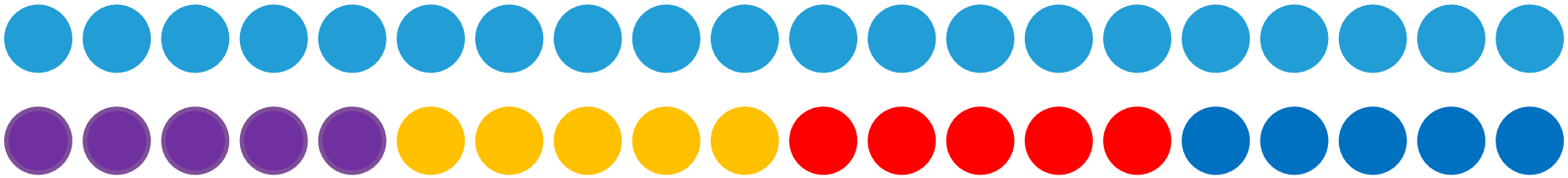
Validation strategies

k-fold cross validation

sklearn.model_selection.KFold

sklearn.model_selection.StratifiedKFold

Samples



Advantage: most efficient (efficient = less computation time) way to use all the samples for training and testing

Drawback: less comprehensive evaluation of the variability of the performance

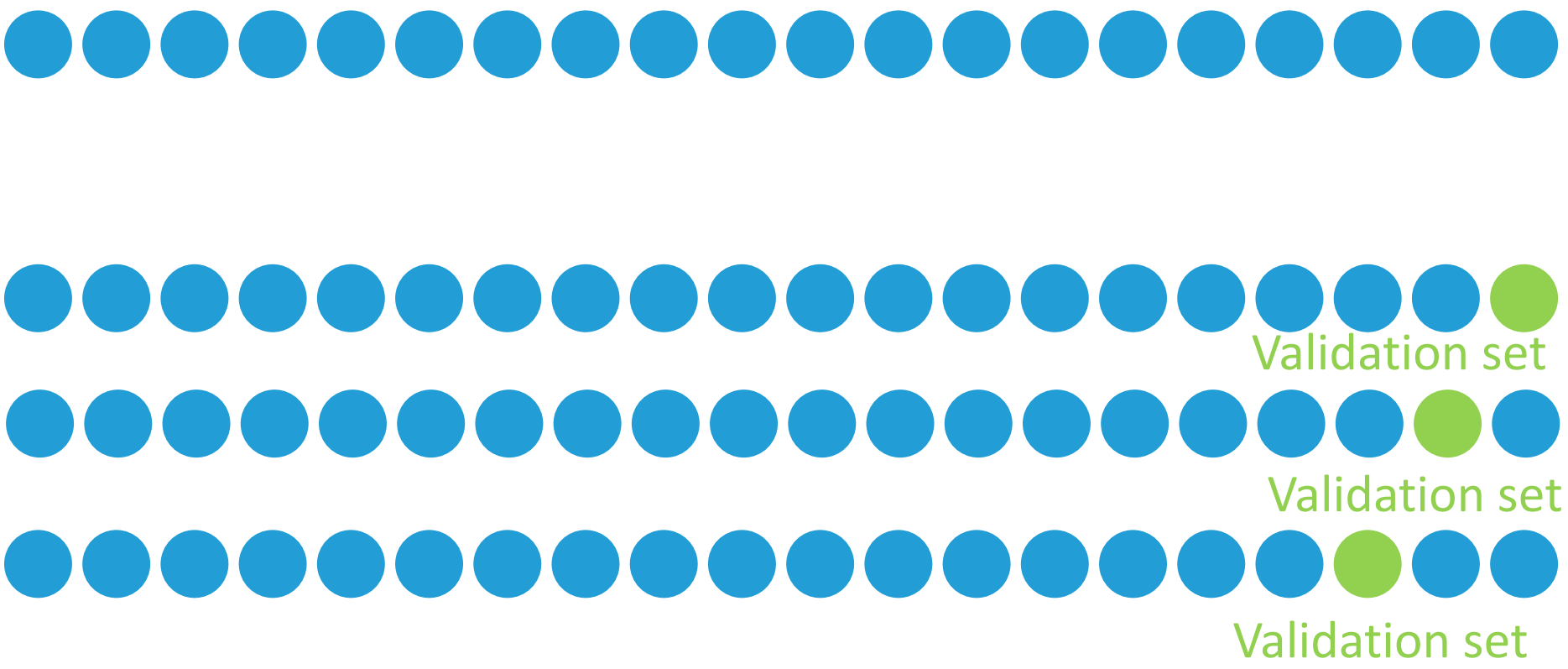
Typical values of k: 5, 10

Validation strategies

sklearn.model_selection.LeaveOneOut

Leave-one-out cross validation

Special case of k-fold with k=n



… In general, one should prefer smaller values of k unless n is really small

Validation strategies

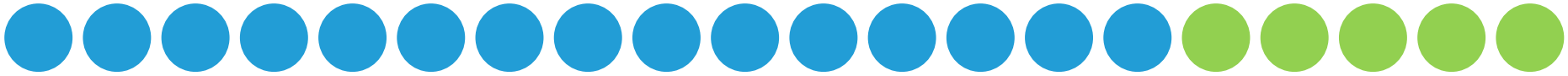
Repeated hold out

sklearn.model_selection.ShuffleSplit(n_splits)

sklearn.model_selection.StratifiedSuffleSplit(n_splits)

Repeat k times (with large k, for instance 100)

Validation set



Training set

Advantage: comprehensive evaluation of the variability of the performance

Drawback: computationnally expensive

Validation strategies

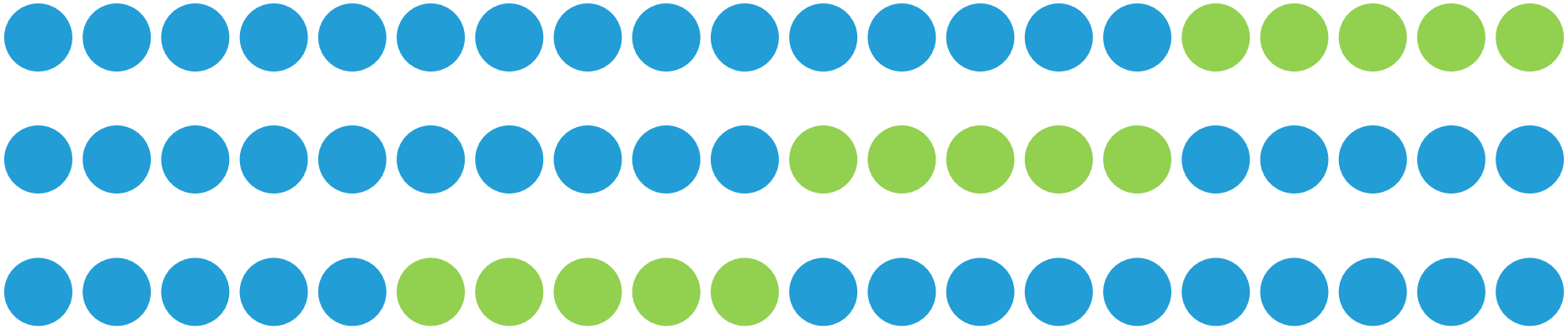
Is this enough?

- If there is no feature selection and a single model without any hyperparameter, yes
- But this is rarely the case

Bad practices

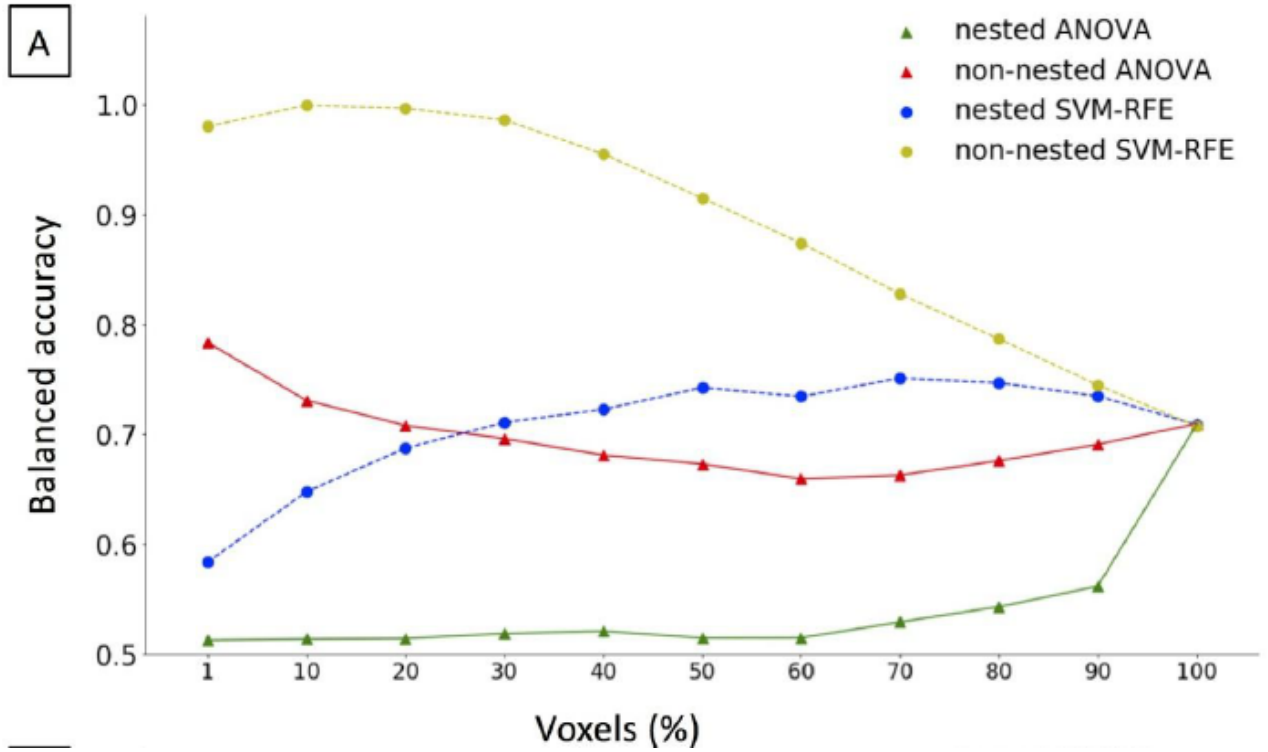
Use all samples for **feature selection**

Then cross-validate the model using the selected features as input



Bad practices

Use all samples for feature selection

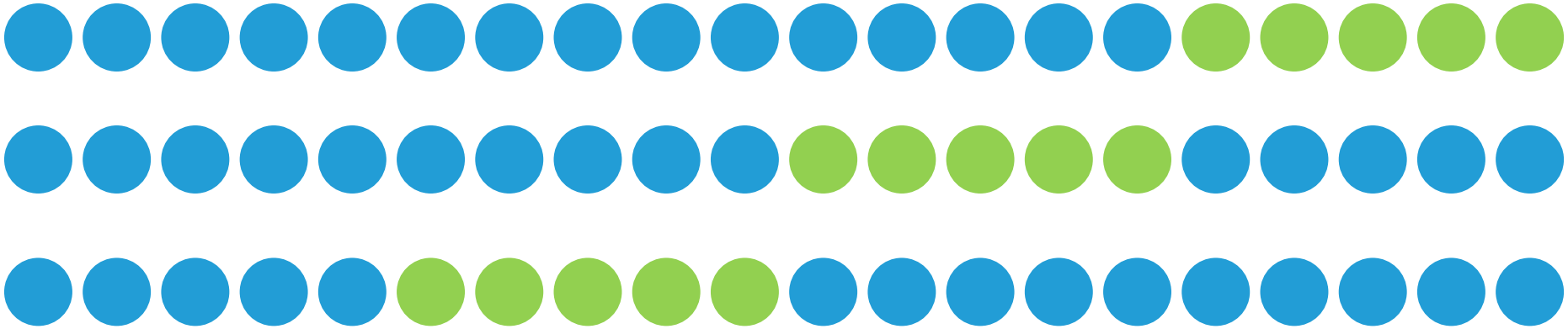


Wen et al, 2018

Bad practices

Use all samples for **dimensionality reduction (e.g PCA)**

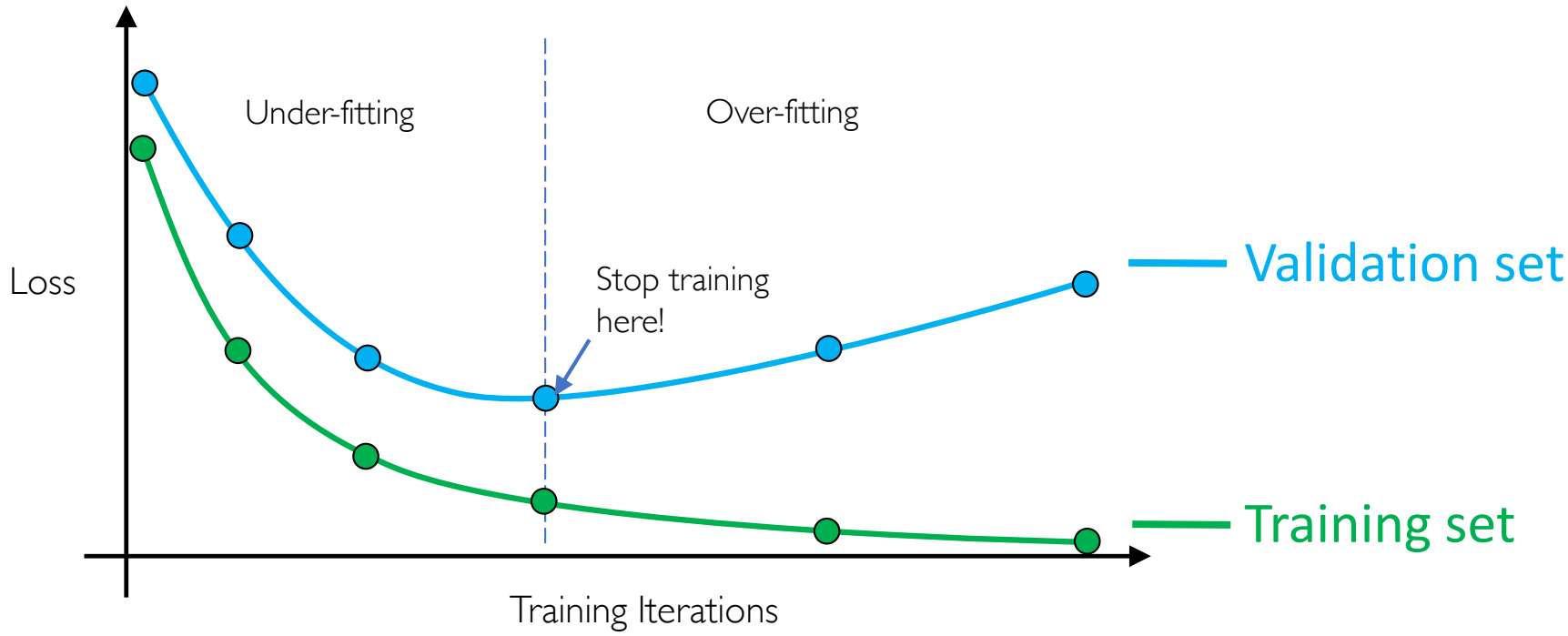
Then cross-validate the model using the reduced features as input



This should not be done but it is probably much less serious than in the case of feature selection

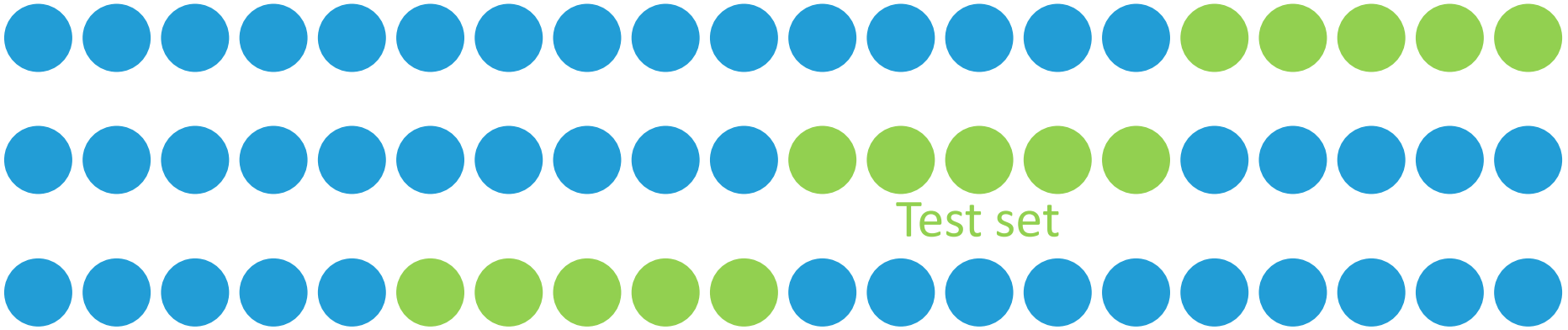
Bad practices

Report the performance obtained on the validation set that was used to decide when to stop training (in deep learning)



Bad practices

Test many possible models and architectures using multiple runs of CV and report the performance of the best performing model



Data leakage

These bad practices are called **data leakage**: some information from the validation set has leaked into the building of the model

Is data leakage prevalent?

Litterature survey of studies using CNNs for Alzheimer's classification from anatomical MRI

(A) Studies without data leakage

Study	DOI	Accuracy	Data leakage
		AD vs CN	
Aderghal et al, 2017	10.1007/978-3-319-51811-4_56	83,70%	None detected
Aderghal et al, 2018	10.1109/CBMS.2018.00067	90%	None detected
Backstrom et al, 2018 *	10.1109/ISBI.2018.8363543	90,11%	None detected
Cheng et al, 2017	10.1117/12.2281808	87,15%	None detected
Cheng and Liu, 2017	10.1109/CISP-BMEI.2017.8302281	85,47%	None detected
Islam and Zhang, 2018 **	10.1186/s40708-018-0080-3	(CN/mild/moderate/ severe: 93,18%)	None detected
Korolev et al, 2017	10.1109/ISBI.2017.7950647	80,00%	None detected
Li et al, 2018	10.1109/IST.2017.8261566	88,31%	None detected
Li et al, 2018	10.1016/j.compmedimag.2018.09.009	89,50%	None detected
Liu et al, 2018	10.1007/s12021-018-9370-4	84,97%	None detected
Liu. et al, 2018	10.1016/j.media.2017.10.005	91,09%	None detected
Liu. et al, 2018	10.1109/JBHI.2018.2791863	90,56%	None detected
Senanayake et al, 2018	10.1109/ISBI.2018.8363832	76%	None detected
Shmulev et al, 2018	10.1007/978-3-030-00689-1_9	(sMCI/pMCI: 62%)	None detected
Valliani and Soni, 2017	10.1145/3107411.3108224	81,30%	None detected

(B) Studies with potential data leakage

Study	DOI	Accuracy	Data leakage	Categories		
		AD vs CN		1	2	3
Aderghal et al, 2017	10.1145/3095713.3095749	91,41%	Unclear	X		
Hon and Khan, 2017	10.1109/BIBM.2017.8217822	96,25%	Unclear	X		X
Hosseini-Asl et al, 2018	10.2741/4606	99,30%	Unclear	X	X	
Islam and Zhang, 2017	10.1007/978-3-319-70772-3_20	(CN/mild/moderate/ severe: 73,75%)	Unclear		X	
Taqi et al, 2018	10.1109/MIPR.2018.00032	100%	Unclear		X	
Vu et al, 2017	10.1109/BIGCOMP.2017.7881683	85,24%	Unclear	X		
Wang et al, 2018	10.1007/s10916-018-0932-7	97,65%	Unclear		X	
Backstrom et al, 2018 *	10.1109/ISBI.2018.8363543	98,74%	Clear	X		
Farooq et al, 2017	10.1109/IST.2017.8261460	(AD/LMCI/EMCI/ CN: 98,88%)	Clear	X		
Gunawardena et al, 2017	10.1109/M2VIP.2017.8211486	(AD/MCI/CN: 96%)	Clear	X	X	
Vu et al, 2018	10.1007/s00500-018-3421-5	86,25%	Clear	X		X
Wang S. et al, 2017	10.1007/978-3-319-68600-4_43	(MCI/CN: 90,60%)	Clear	X		

Table 1. Summary of the studies performing classification of AD using CNNs on anatomical MRI. When d in brackets. (A) Studies without data leakage; (B) Studies with potential data leakage.
Data leakage categories: 1: Biased split; 2: No independent test set ; 3: Late split.
* (Backstrom et al., 2018) experimented two data-partitioning strategies to study the consequences of a l labels.
** Use of imbalanced accuracy on an imbalanced dataset, leading to an over-optimistic estimation of performance.

Over 40% of studies are suspect of data leakage!

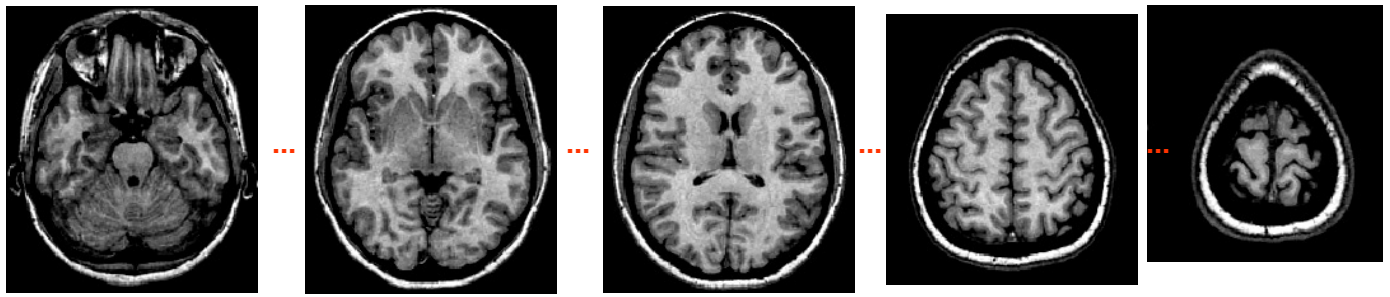
(Wen*, Thibeau—Sutre* et al, 2020)

Fifty shades of data leakage

Bad split of samples between the training, validation and test sets

3D MRI

Splitted at the slice level and not the patient level



5–fold accuracy (with data leakage) 1.00 ± 0 [1.00, 1.00, 1.00, 1.00, 1.00]

True 5 –fold accuracy 0.79 ± 0.04 [0.83, 0.83, 0.72, 0.82, 0.73]

Fifty shades of data leakage

Bad split of samples between the training, validation and test sets

Several visits per patient
Split at the visit level and not the patient level

One can use the following functions to do the split at the patient level (slices or visits will be grouped into patients)

sklearn.model_selection.LeaveOneGroupOut

sklearn.model_selection.LeavePGroupsOut

sklearn.model_selection.GroupKFold

What should we do?

Training, validation and test sets

Use to test different models, to choose when to stop training…

Often through cross-validation

Use only to report performance

What should we do?

Training, validation and test sets

Cross-validation

Standard (minimal) good practice for deep learning

Training, validation and test sets

Use cross-validation, often with k=3 to 5, to train the model, experiment with different architectures…

Standard (minimal) good practice for deep learning

Training, validation and test sets

Use cross-validation, often with k=5, to train the model, experiment with different architectures…

CV is useful to study the variability with respect to the training set

The test set

Where should I keep the test set?

In a safe!

The test set

When should I separate the test set?

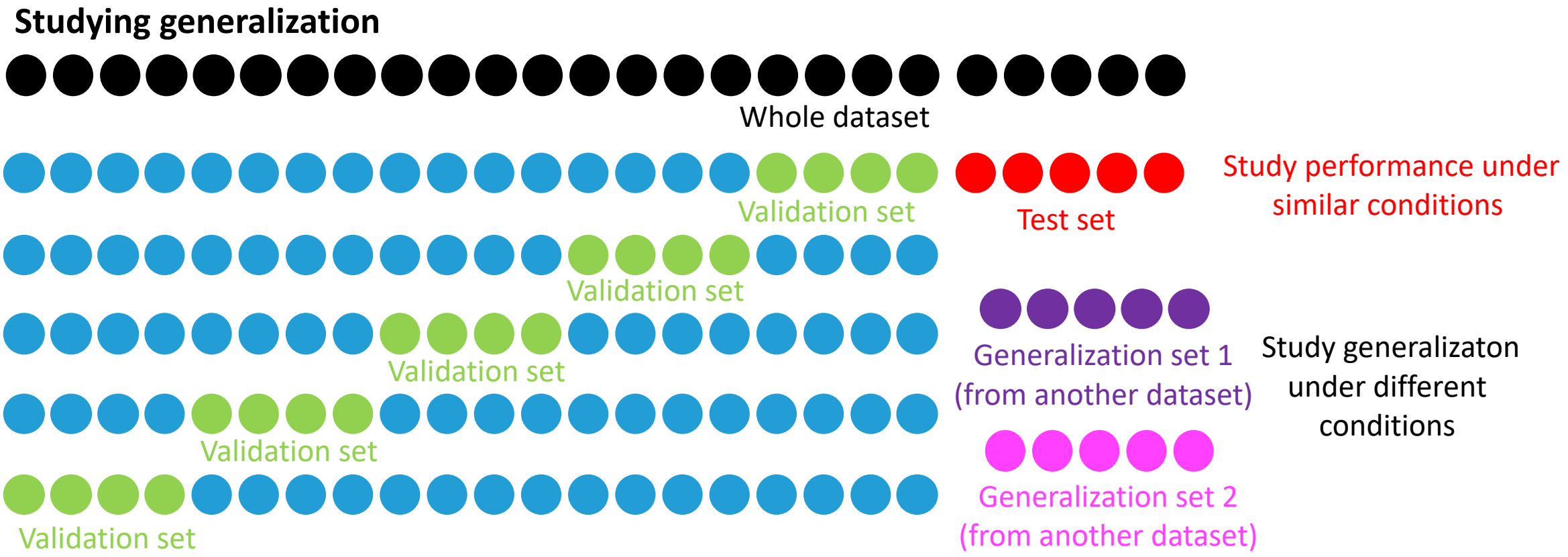
Before starting the work

Standard (minimal) good practice for deep learning

Training, validation and test sets

Use cross-validation, often with k=3 to 5, to train the model, experiment with different architectures…

Even better



Use to train the model, experiment with different architectures…

Statistical analysis

	Balanced accuracy
3D ResNet	72.1
TrickOfTheTradeCNN [11]	72.3
ZorglubFormer (proposed)	**72.7**

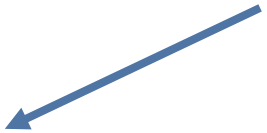
Should we be satisfied with this?

Statistical analysis

- **We are going to consider two cases**
  - **Trained models**
    - The model has been trained, the only source of variation is the test set
  - **Learning procedure**
    - You want to know how variable is the performance with respect to different sources
      - Test set
      - Training set
      - Hyperparameters
      - Initialization
      - ….

Statistical analysis

- **We are going to consider two cases**
  - **Trained models**
    - The model has been trained, the only source of variation is the test set
  - **Learning procedure**
    - You want to know how variable is the performance with respect to different sources
      - Test set
      - Training set
      - Hyperparameters
      - Initialization
      - ….



Main focus for today's course

Statistical analysis

- **We are going to look at two types of statistics**
  - **Descriptive statistics**
    - How variable is your performance?
  - **Inferential statistics**
    - How precise is the estimate of your performance?
    - Can you claim that one model is better than another?

Descriptive statistics

- **Can you give some examples of ways to assess of variability the performance?**

Descriptive statistics

- **Can you give some examples of ways to assess of variability the performance?**
  - **Measures**
    - Standard-deviation
    - IQR (inter-quartile range)
    - Min, max
    - Deciles, centiles
  - **Graphs**
    - Bar plots
    - Box plots
    - Violin plots
    - Jittered points

Descriptive statistics

- **Can you give some examples of ways to assess of precision of an estimate?**

Descriptive statistics

- **Can you give some examples of ways to assess of precision of an estimate?**
  - **Measures**
    - Confidence interval
    - Standard error
  - **Graphs**
    - Bar plots
    - Box plots
    - …

Statistical analysis: variability (descriptive statistics)

Trained models

Statistical analysis

- **We are going to consider two cases**
  - **Trained models**
    - The model has been trained, the only source of variation is the test set
  - **Learning procedure**
    - You want to know how variable is the performance with respect to different sources
      - Test set
      - Training set
      - Hyperparameters
      - Initialization
      - ….

Statistical analysis

- **We are going to consider two cases**
  - **Trained models**
    - The model has been trained, the only source of variation is the test set
  - **Learning procedure**
    - You want to know how variable is the performance with respect to different sources
      - Test set
      - Training set
      - Hyperparameters
      - Initialization
      - ….

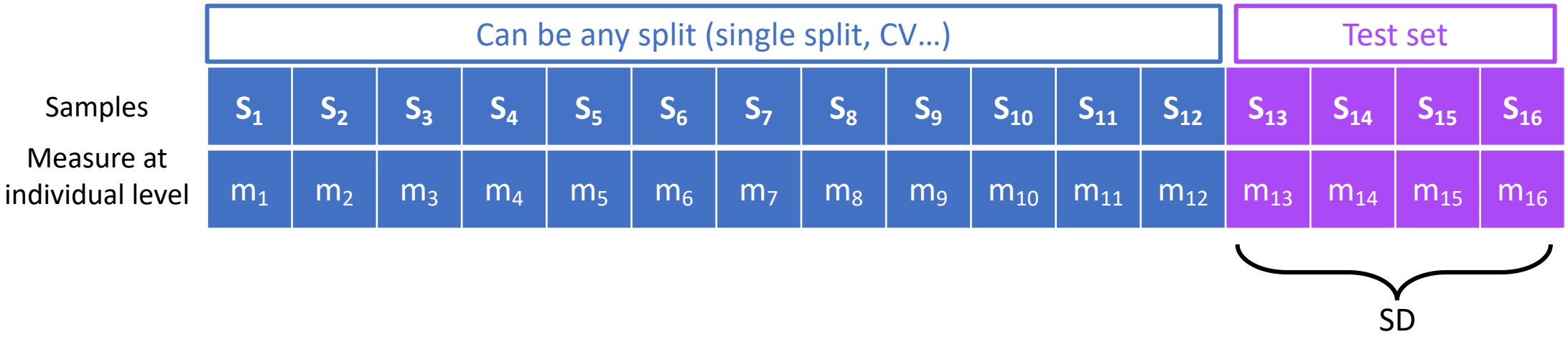
Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases?
    (complete failures)

- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph

Descriptive statistics: trained models

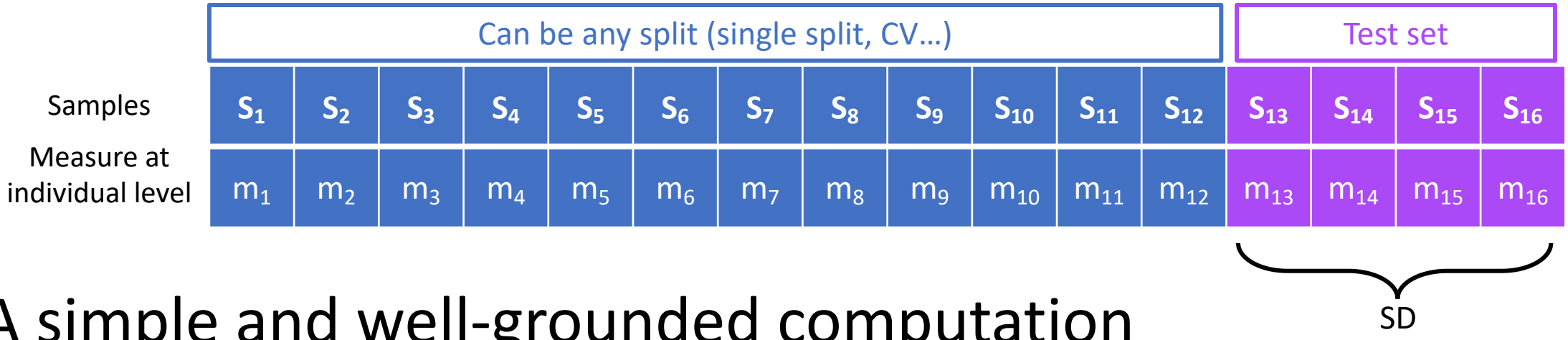
SD for a trained model: what does it mean?



Descriptive statistics: trained models

SD for a trained model: what does it mean?

It means SD on an independent test set
The model is already trained, it does not change
Only source of variance: test samples

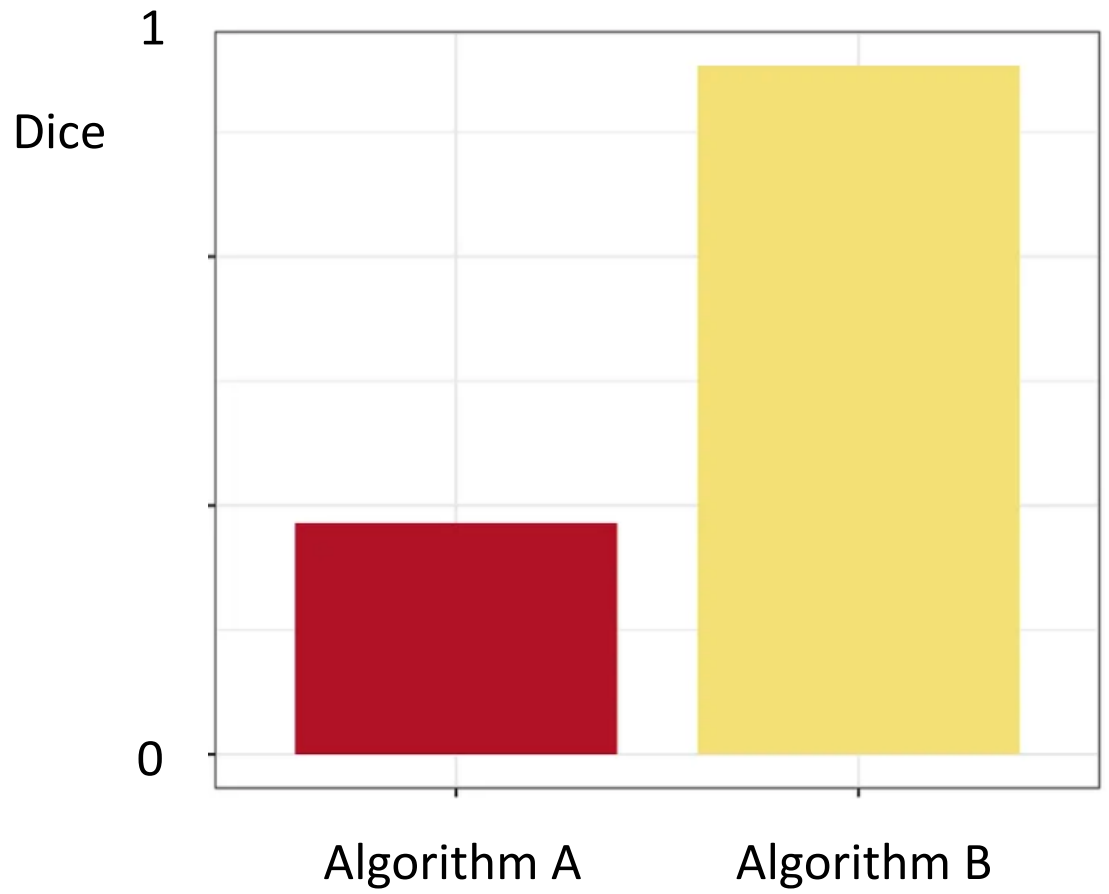


A simple and well-grounded computation

Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

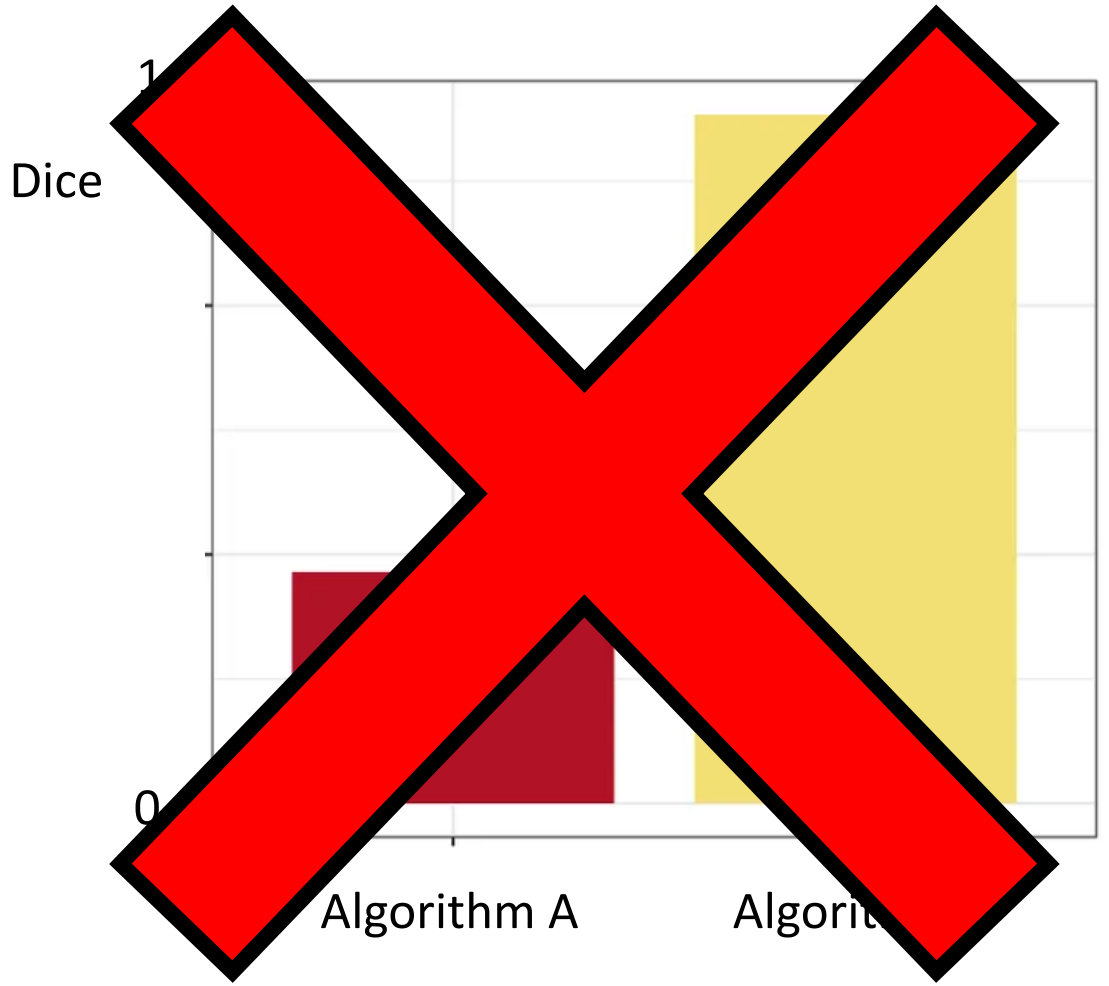
- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph



Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

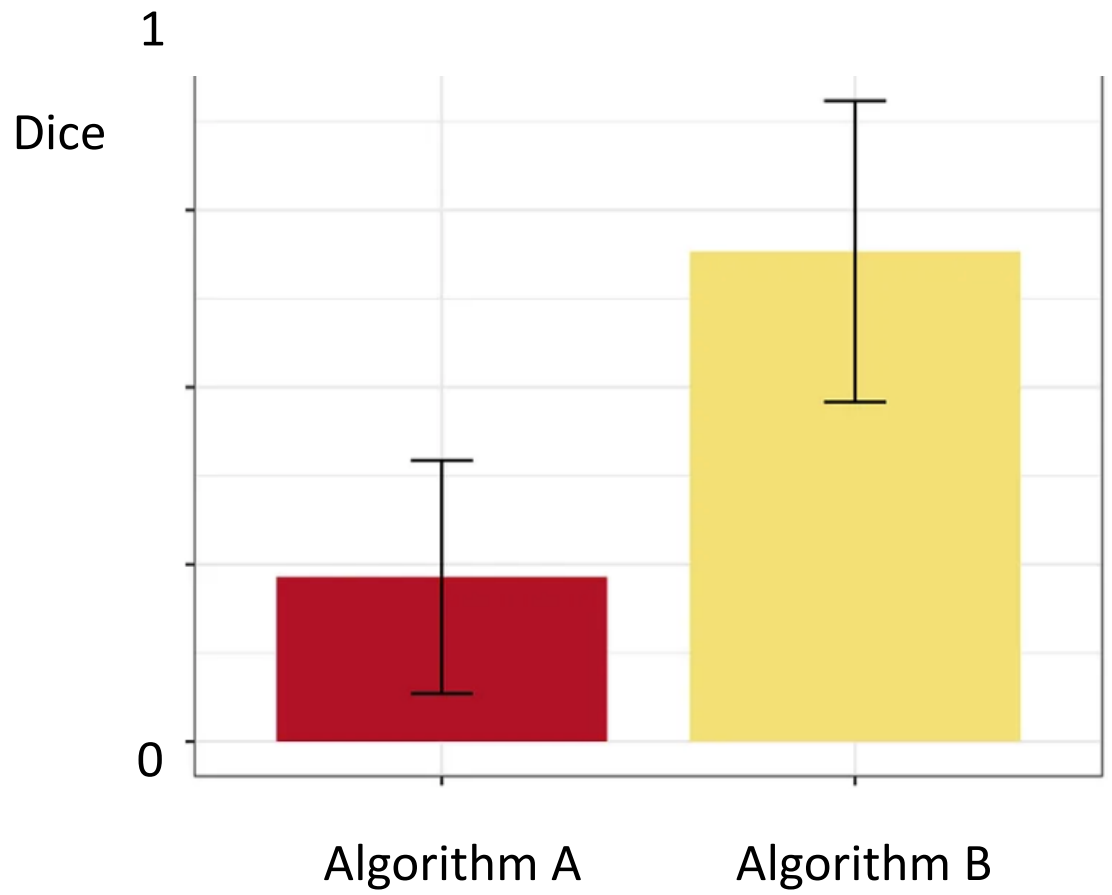
- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph



Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph

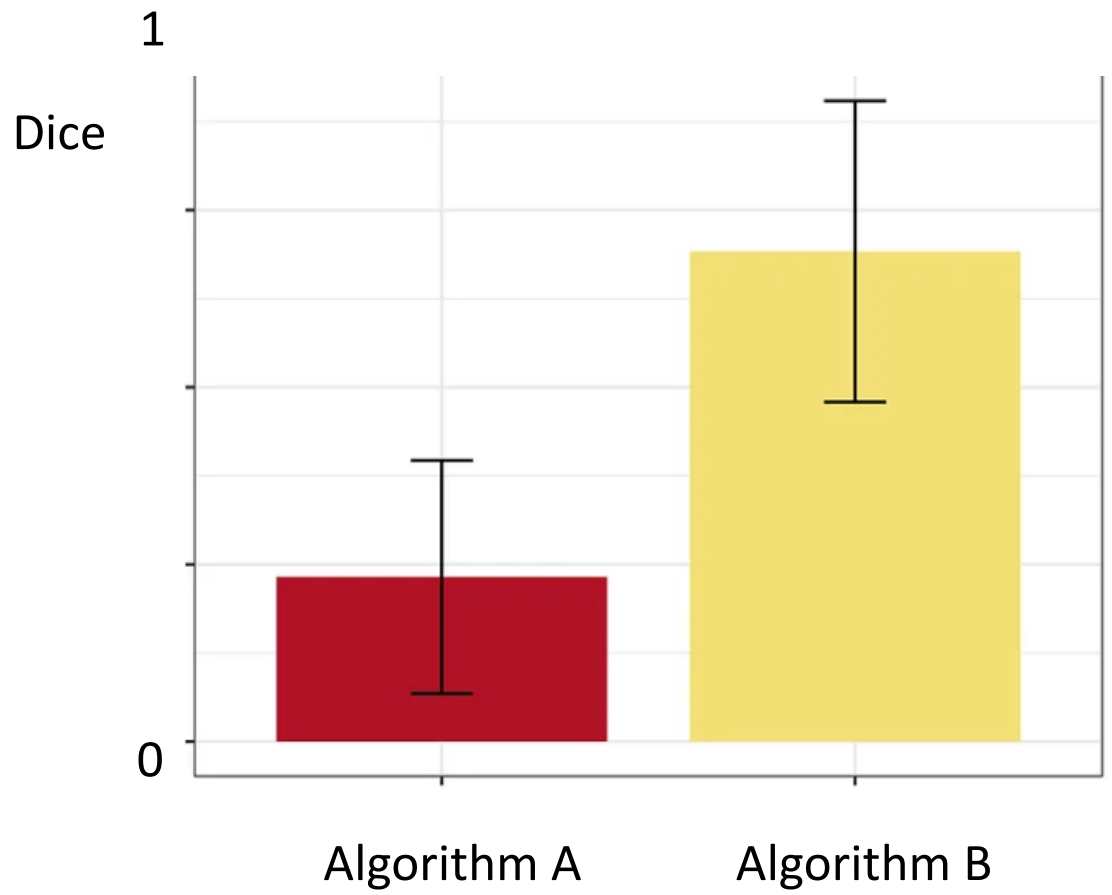


Is it enough?

Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph

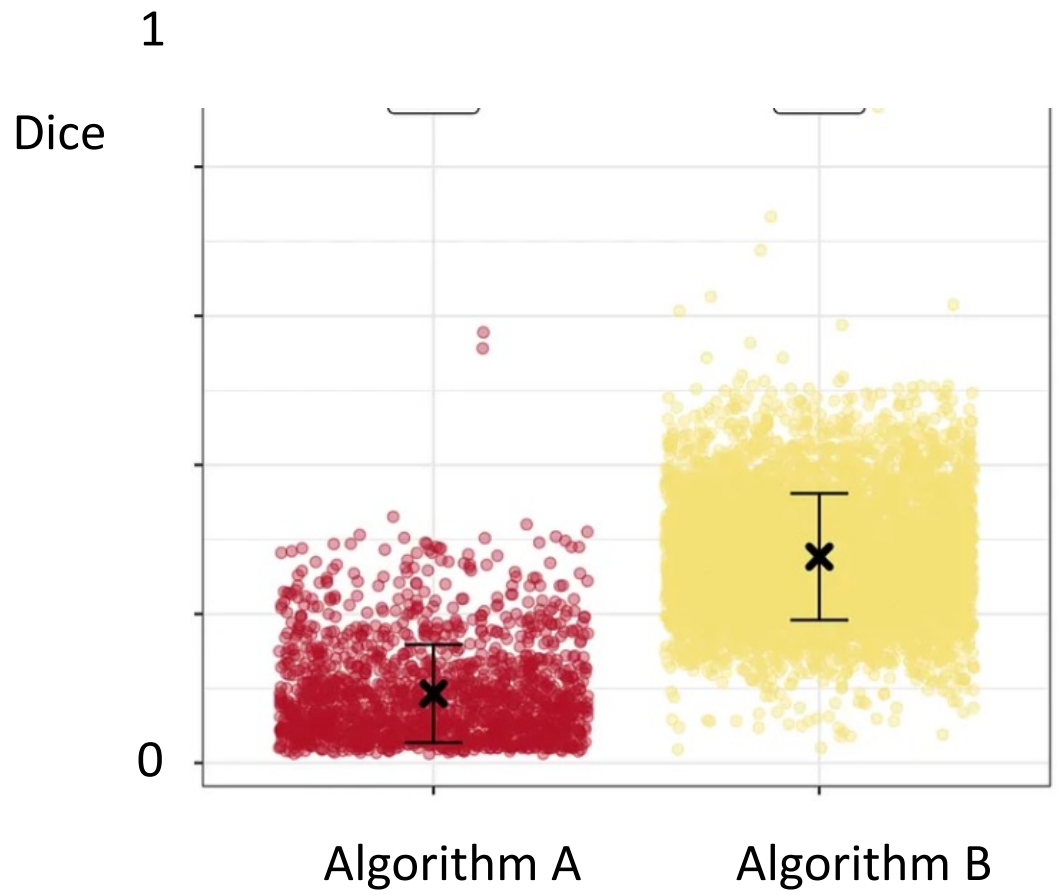


Error bars represent the standard-deviation computed on the test set

Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph

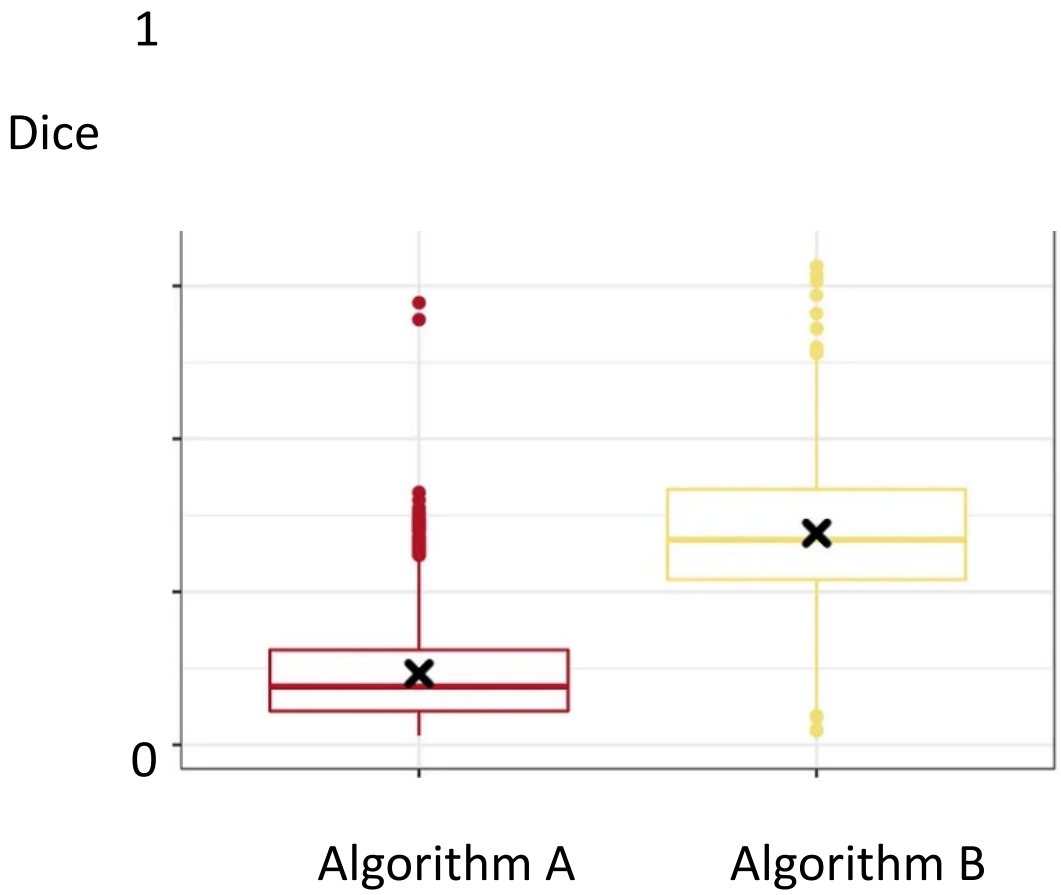


Maybe standard deviation was not enough

Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph

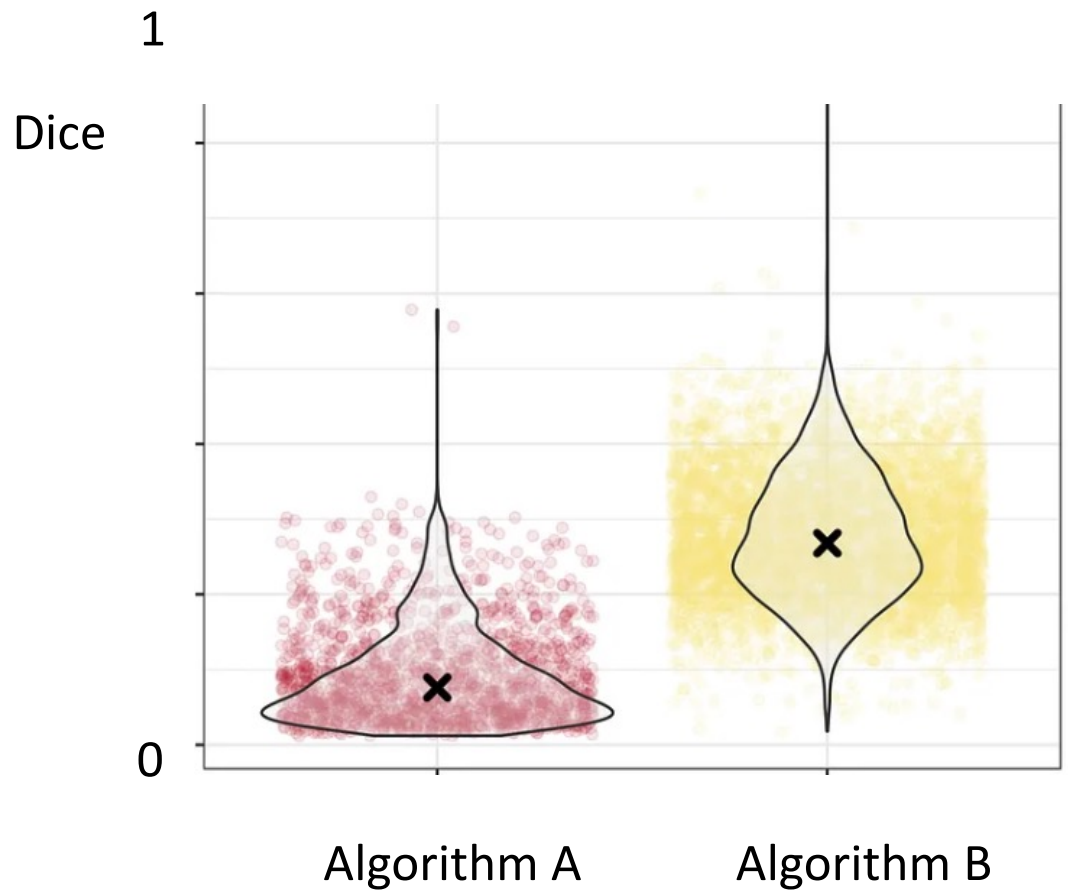


The box extends from Q1 to Q3

Descriptive statistics: trained models

- **Assess the variability of the performance**
  - Is it stable?
  - Are there extreme cases? (complete failures)

- **What should we do?**
  - Plot the distribution
    - Are mean and standard-deviation meaningful?
  - Report in tables
    - Mean and standard-deviation
      - or
    - Median and IQR
  - If enough space
    - Provide a graph



Violin plot with median and jitter points

Descriptive statistics: trained models

It is not enough to report a mean estimate of the performances.

One also need to know how variable these performances are

Learning procedures

Statistical analysis

- **We are going to consider two cases**
  - **Trained models**
    - The model has been trained, the only source of variation is the test set
  - **Learning procedure**
    - You want to know how variable is the performance with respect to different sources
      - Test set
      - Training set
      - Hyperparameters
      - Initialization
      - ….

Statistical analysis

- **We are going to consider two cases**
  - **Trained models**
    - The model has been trained, the only source of variation is the test set
  - **Learning procedure**
    - You want to know how variable is the performance with respect to different sources
      - Test set
      - Training set
      - Hyperparameters
      - Initialization
      - ….

Descriptive statistics: learning procedures

It is not as trivial as it looks

Descriptive statistics: learning procedures

SD from cross-validation (CV): a common practice

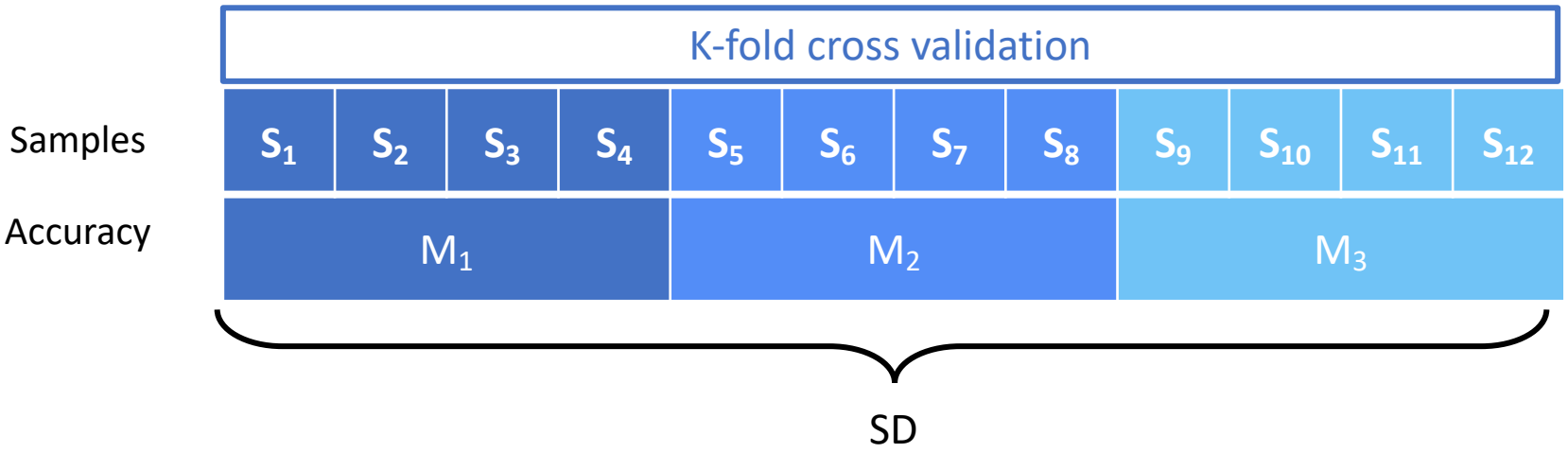
	Accuracy
Doe et al. [31]	86.9 ± 2.1
Due et al. [27]	87.6 ± 1.1
Blah et al [21]	88.0 ± 1.9
Zorglub et al [11]	88.3 ± 2.3
Proposed method	**89.7** ± 1.8

Results are reported as mean ± standard-deviation of the accuracy
which is computed over 5-folds of cross validation

Optimistic scenario: often the legend won't be that clear

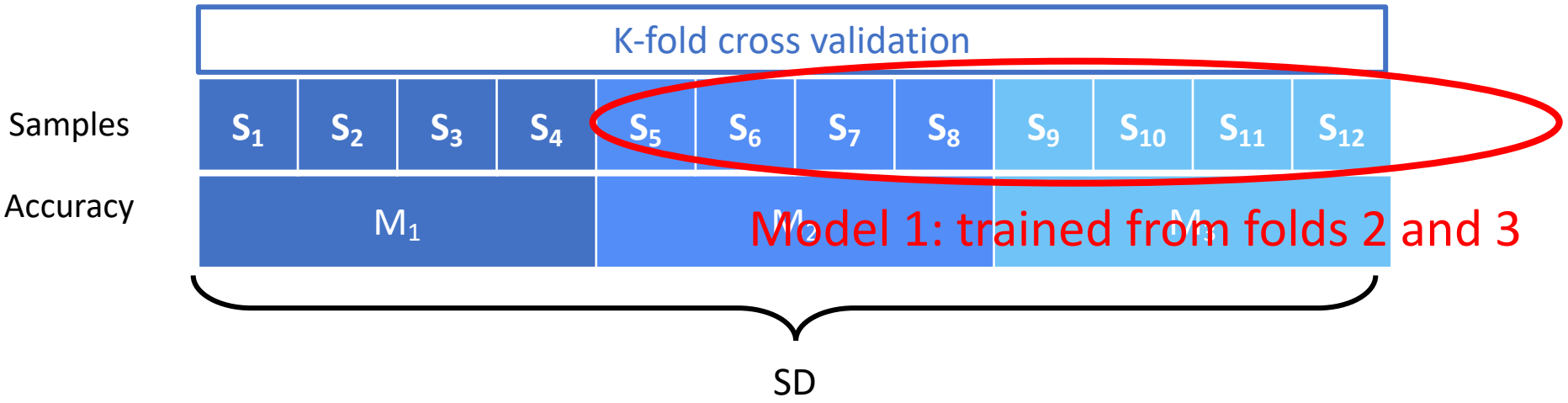
Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?



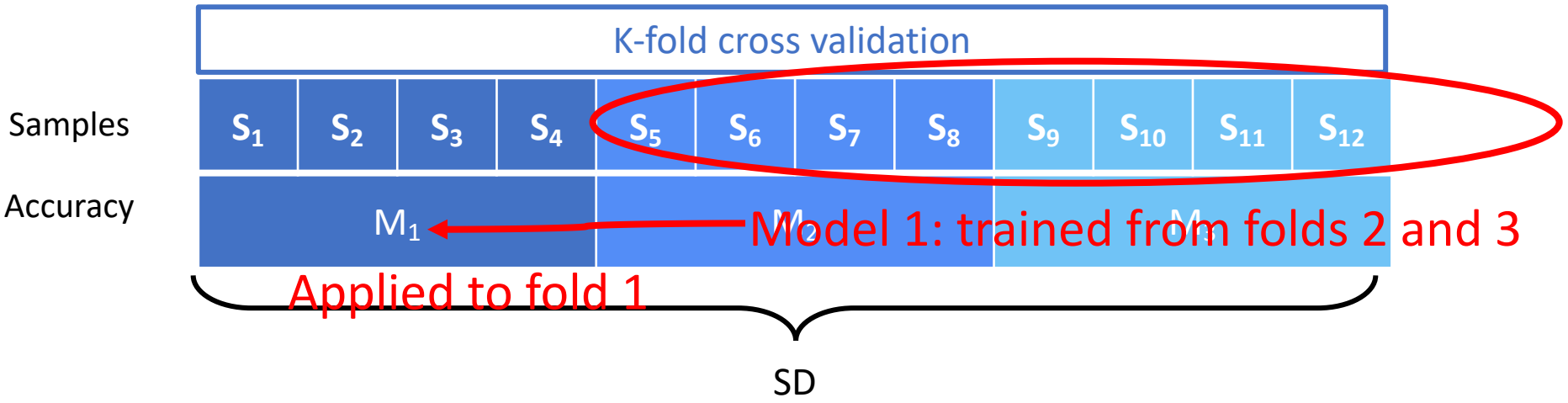
Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?



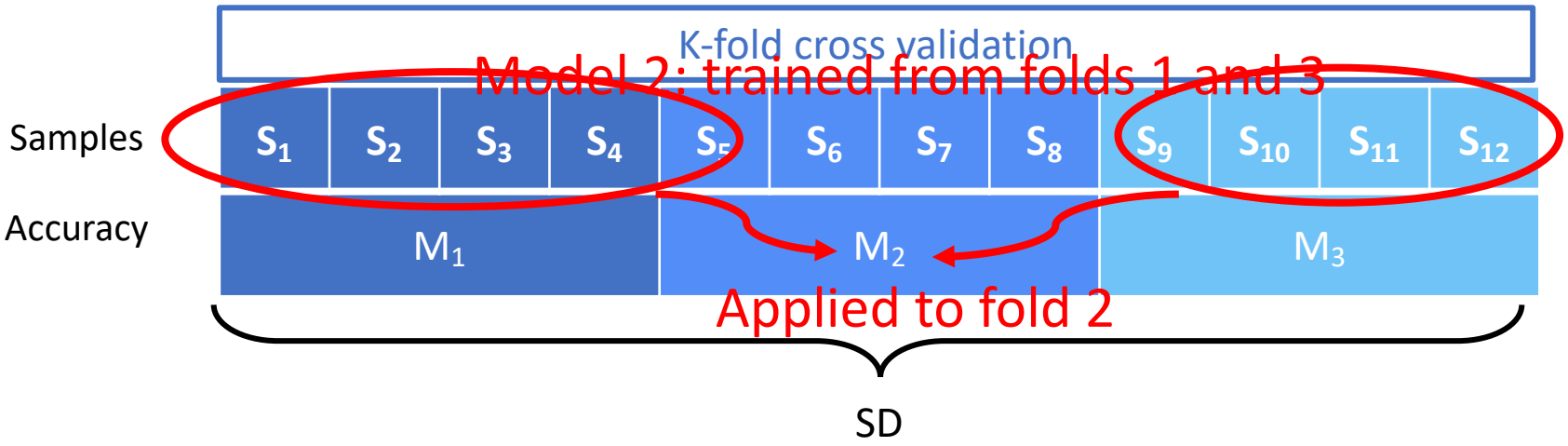
Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?



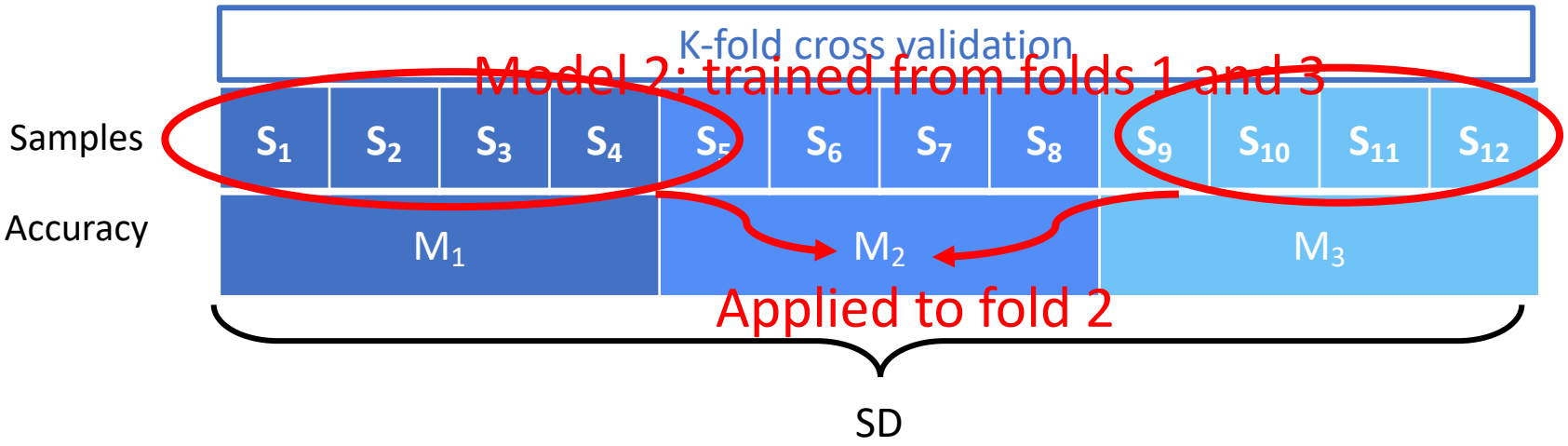
Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?



Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?

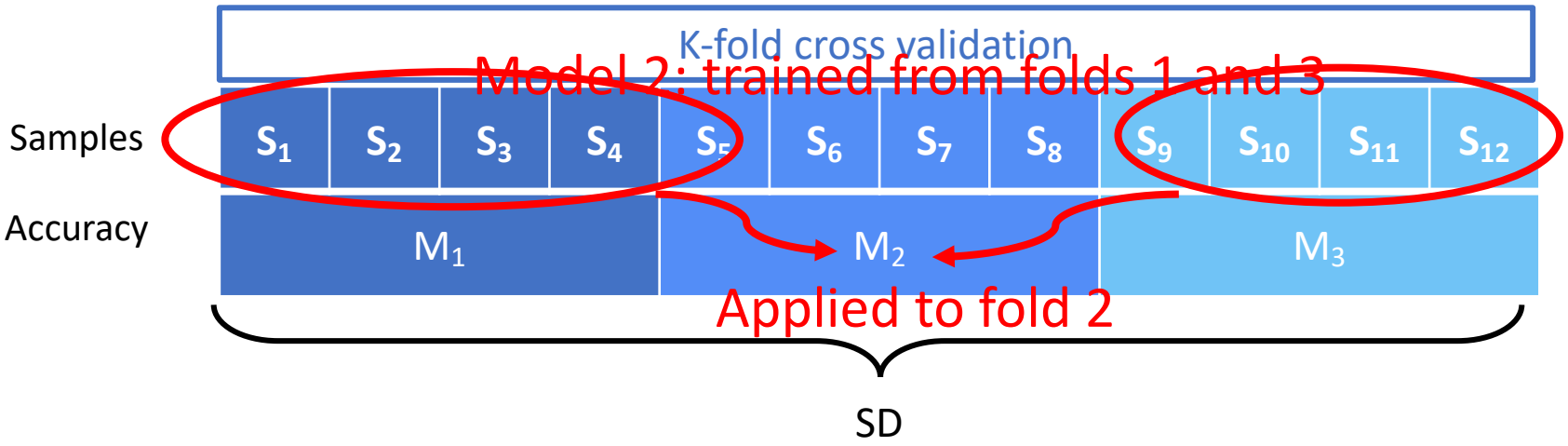


We are mixing two sources of variance:

- Training set
- Test set

Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?

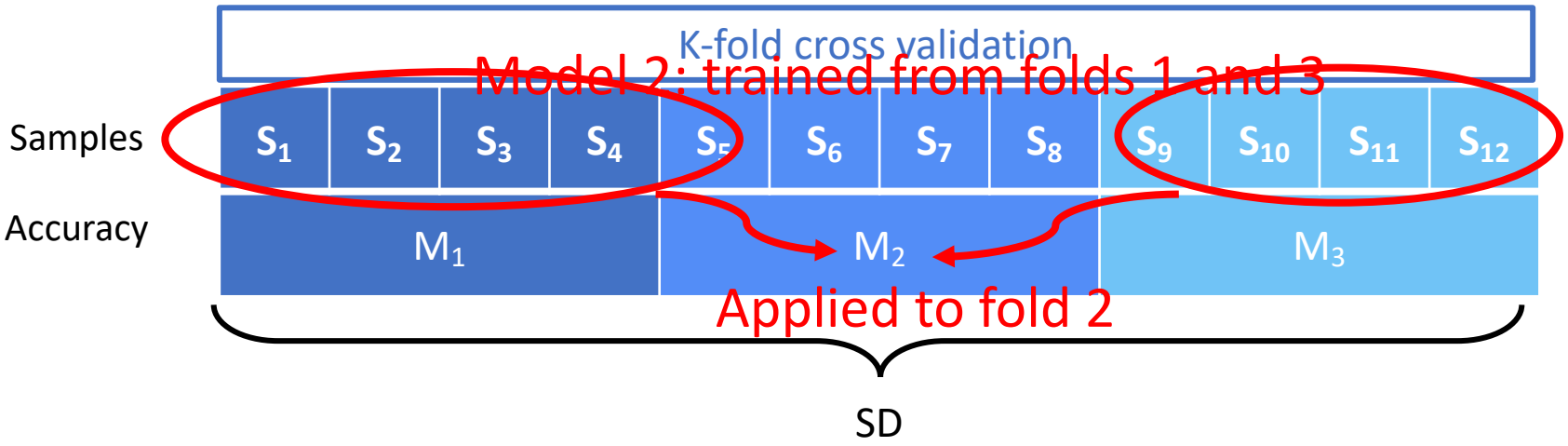


We are mixing **several** sources of variance:

- Training set
- Test set
- Initialization (unless you explicitly fixed it)
- Data order (unless it is always the same…)
- ….

Descriptive statistics: learning procedures

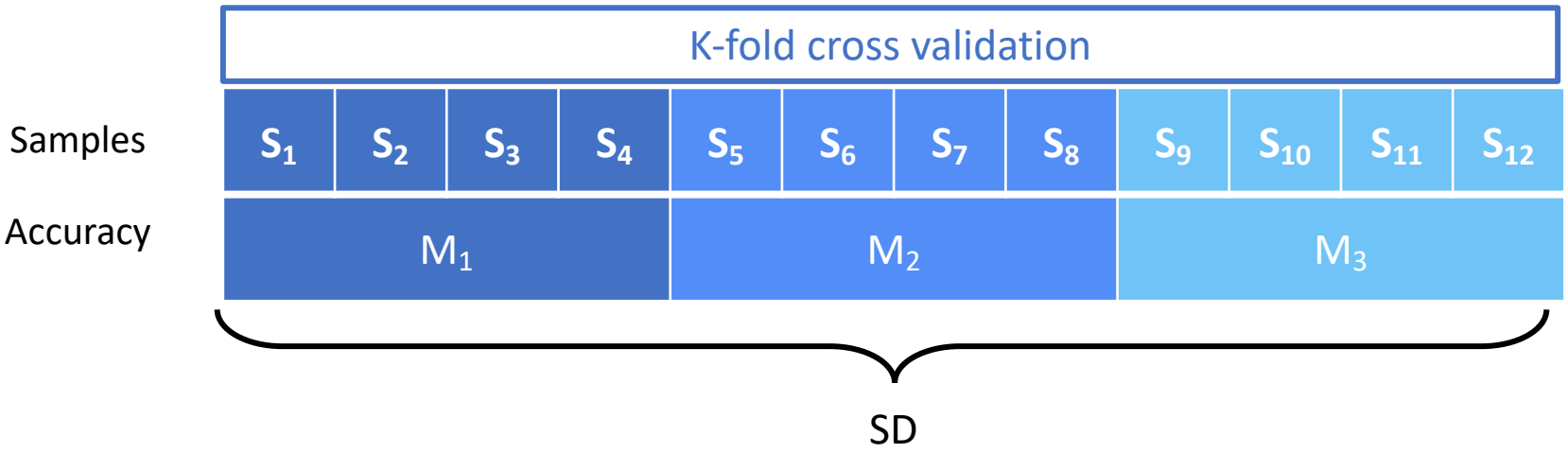
SD from cross-validation (CV): what does it mean?



Models 1 and 2 are not independent

Descriptive statistics: learning procedures

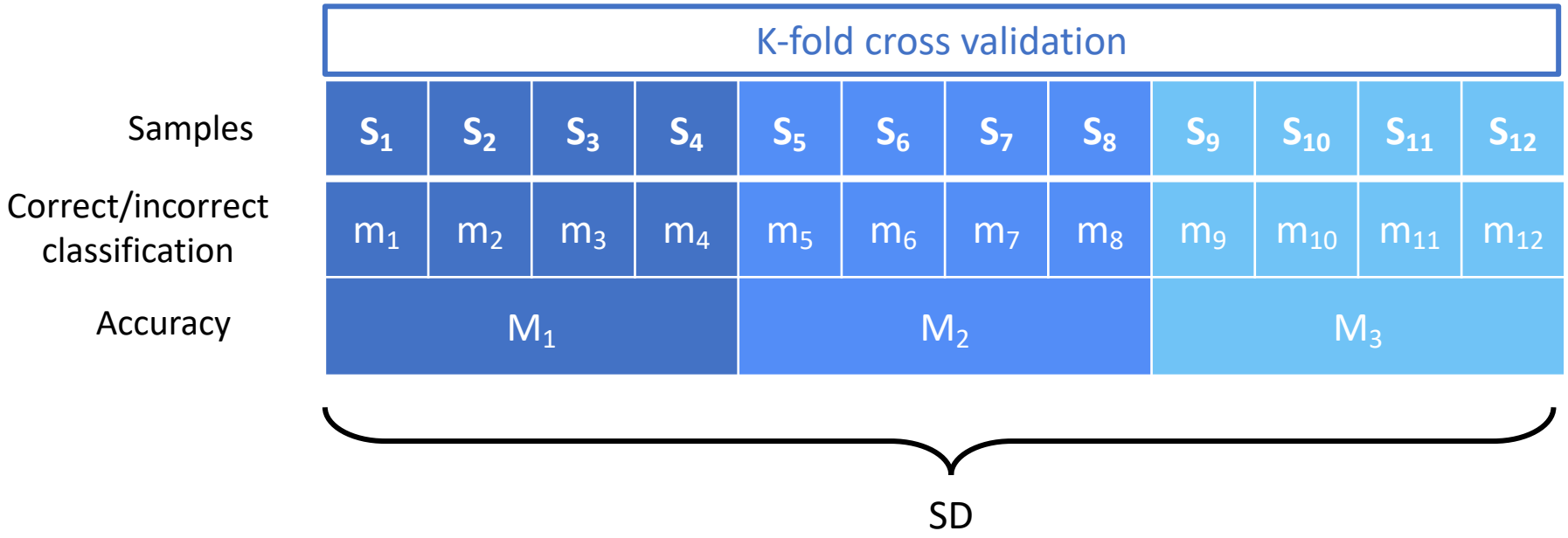
SD from cross-validation (CV): what does it mean?



What does this SD mean?

Descriptive statistics: learning procedures

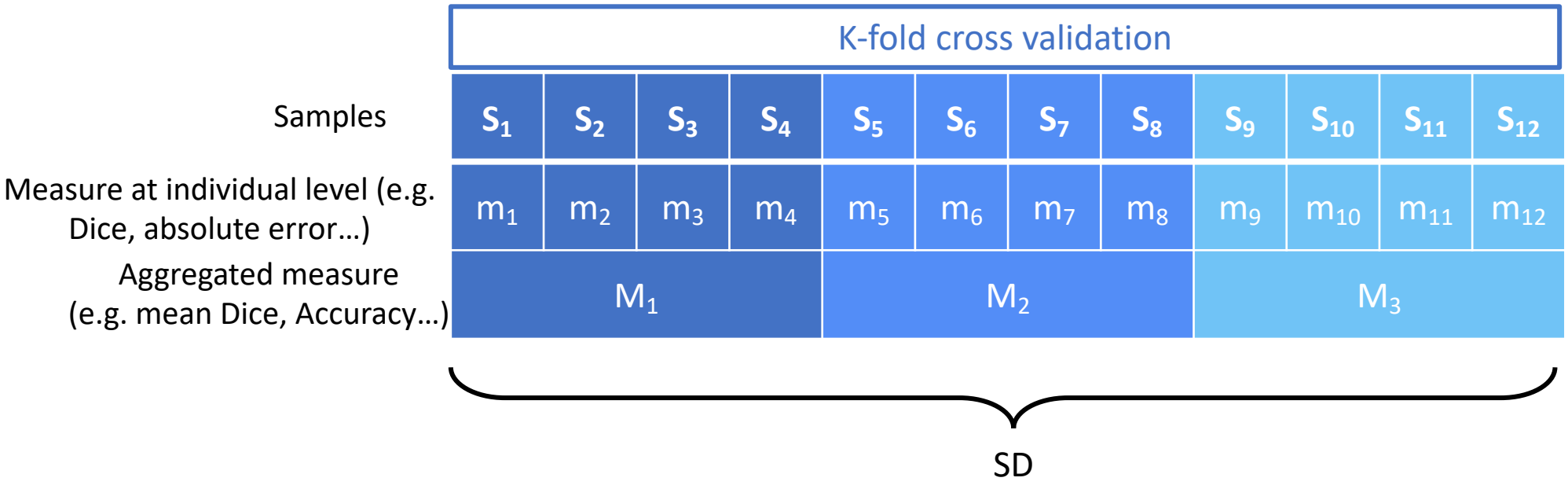
SD from cross-validation (CV): what does it mean?



What is behind this computation?

Descriptive statistics: learning procedures

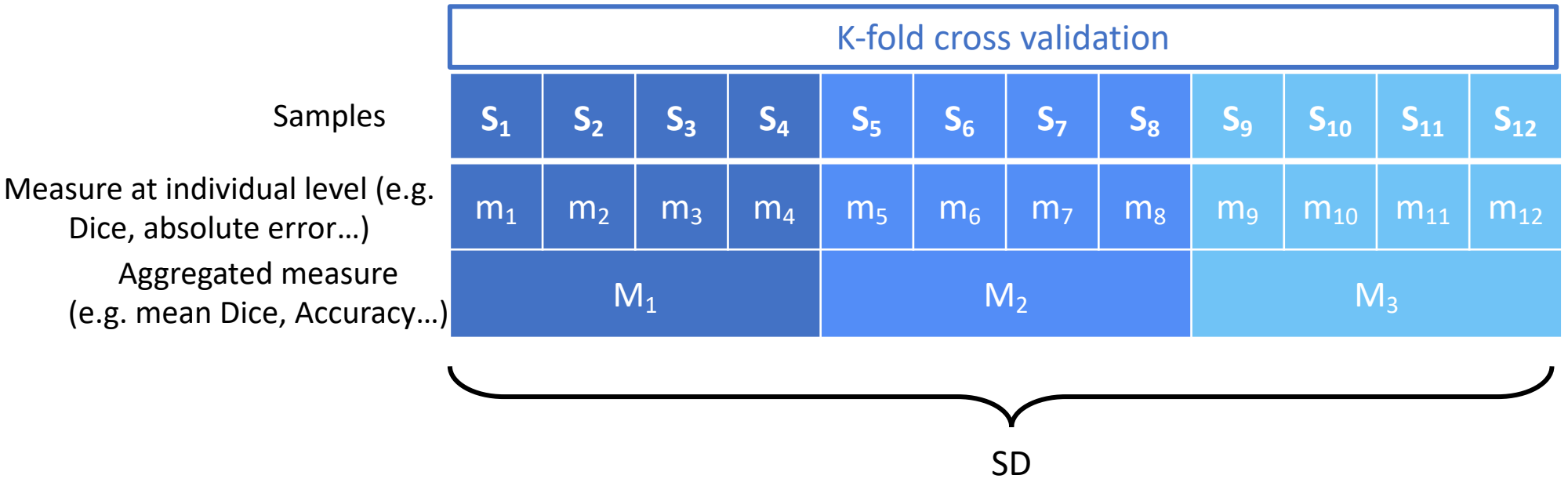
SD from cross-validation (CV): what does it mean?



More generally

Descriptive statistics: learning procedures

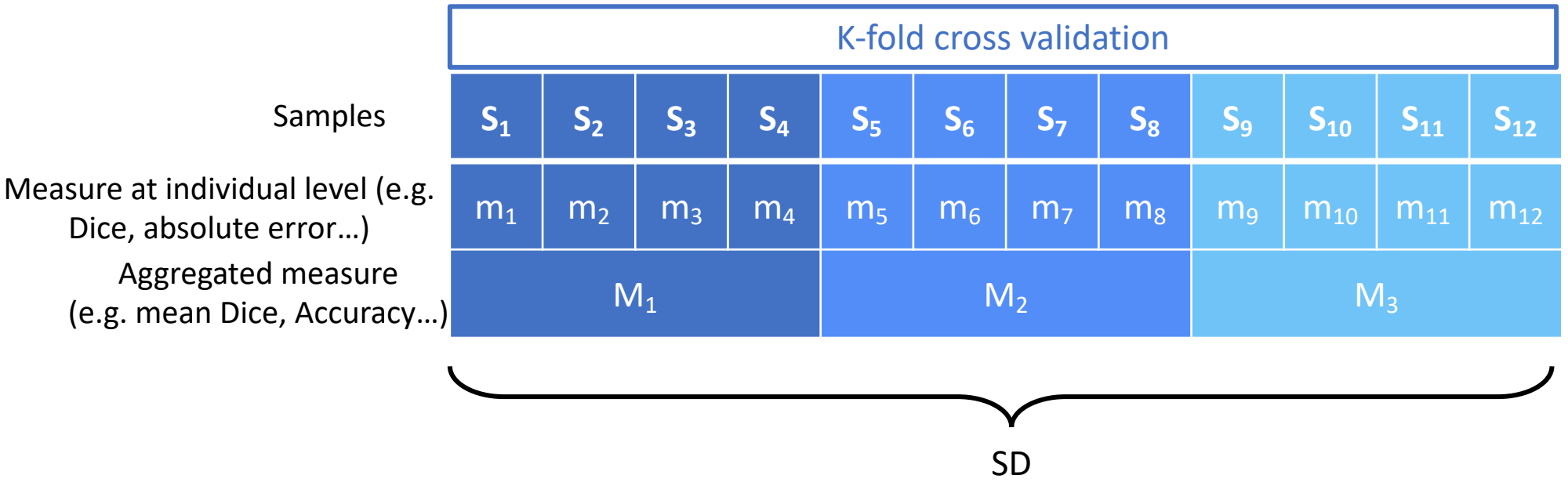
SD from cross-validation (CV): what does it mean?



Looks like we are doing SD of the sampling distribution… What does it reminds you of?

Descriptive statistics: learning procedures

SD from cross-validation (CV): what does it mean?



Looks like we are doing SD of the sampling distribution… What does it reminds you of?

Standard Error

Descriptive statistics: learning procedures

Take home:

- Statistical assessment of learning procedures is not trivial
- It does not mean that you should do nothing!
  - It is still good to know how variable is your procedure with respect to
    - Training splits
    - Initialization
    - Hyperparameters
  - But do not think that this is an unbiased estimate of the standard-deviation or the standard-error
    - Do not use it for statistical testing or other inferential statistics

Statistical analysis: inferential statistics

Trained models

Intuition on precision

Intuitively, the precision with which the performances are estimated **will depend on two factors** :

- **The size of the test set**
  - The larger, the more precise
- **The variability of the performance**
  - i.e. the standard-deviation of the metric on the test set
  - It is often linked to the performance itself
    - Better performance -> lower variability
    - For instance for classifiers
      - Binomial proportion, variance is ***p(1-p)*** where p is the proportion of correctly classified samples

How precise is the estimation of a trained model?

Two (related) tools:

- **standard error** of a given statistic (e.g. the mean)
  - standard deviation of the **sampling** distribution
    - Not to be confused with distribution of the metric
- **confidence intervals** (e.g. at 95%)

How to compute these?

Approach 1: bootstrap

Bootstrap: approximate the sampling distribution

Given a test set of size n, M (e.g. $M = 5000$) bootstrap samples of size n are drawn with replacement. We denote a given bootstrap sample as S^*_m and its mean as μ^*_m where $m \in \{1, \ldots, M\}$. The average of μ^*_m across bootstrap samples is denoted as μ^* The set of values $\{\mu^*_m\}$ are an approximation of the sampling distribution of the statistic (here the mean).

One can then obtain the standard-error

$$\mathrm{SE}^* = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (\mu^*_m - \mu^*)^2}$$

and the 95% confidence interval

$$\mathrm{CI}^* = [a^*, b^*]$$

is the set of values between the 2.5% and 97.5% percentiles of the sorted bootstrap means $(\mu^*_1, \mu^*_2, \ldots, \mu^*_m, \ldots \mu^*_M)$.

Approach 1: bootstrap

Bootstrap:

- easy to perform
- no assumption on the distribution of the statistic
- also ok for small samples

Approach 2: deriving from SD of test set

If the metric comes from a distribution with standard-deviation σ

$$\mathrm{SE} = \frac{\sigma}{\sqrt{n}}$$

In practice, of course, one only has access to the sample standard deviation $\hat{\sigma}$

$$\mathrm{SE} \approx \frac{\hat{\sigma}}{\sqrt{n}}$$

Then the confidence interval at 95% is

$$\mathrm{CI} \approx [\hat{\mu} - 1.96 \times \mathrm{SE}, \hat{\mu} + 1.96 \times \mathrm{SE}]$$

where $\hat{\mu}$ denotes the sample mean.

- OK when n is sufficiently large
- Can be used to simulate confidence intervals for various sizes of n

Approach 2: deriving from SD of test set

$$\text{CI} \approx [\hat{\mu} - 1.96 \times \text{SE}, \hat{\mu} + 1.96 \times \text{SE}]$$

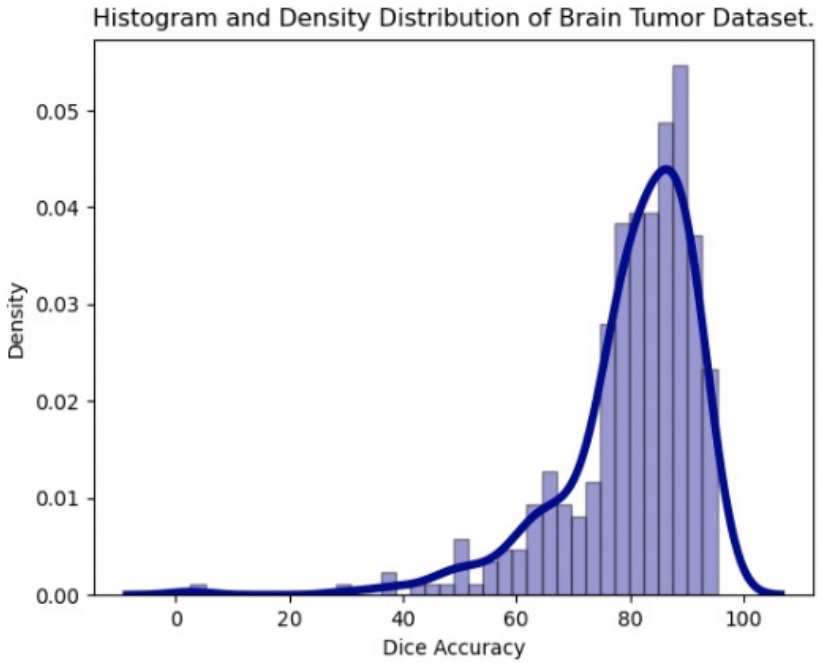
Relies on normal distribution: $z_{1-\alpha/2} = 1.96$ for $\alpha = 0.05$ (95% CI).
It is a bit conservative since is $\hat{\sigma}$ is not known but estimated from the test set.
A better estimate is given by using the t distribution.

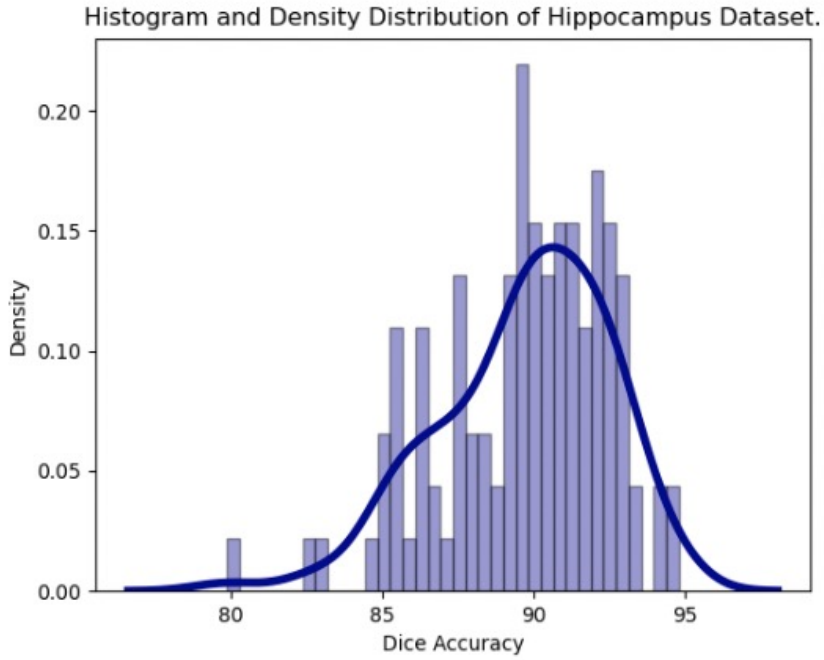
$$\text{CI} \approx [\hat{\mu} - t_{n-1,1-\alpha/2} \times \text{SE}, \hat{\mu} + t_{n-1,1-\alpha/2} \times \text{SE}]$$

However, t quickly converges to z.

Some typical metric distributions

Dice metric

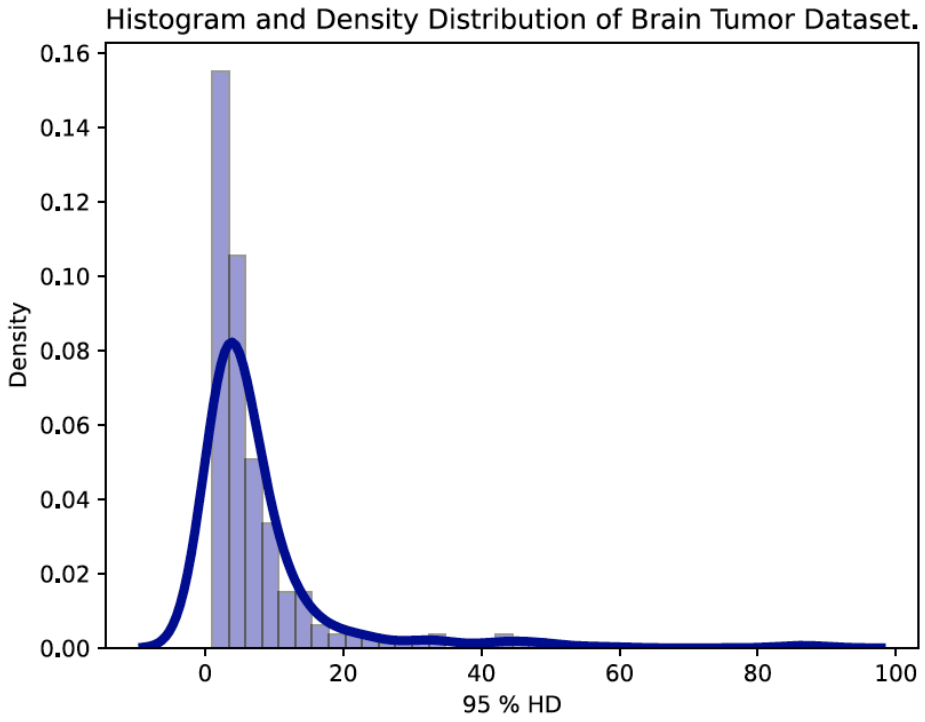


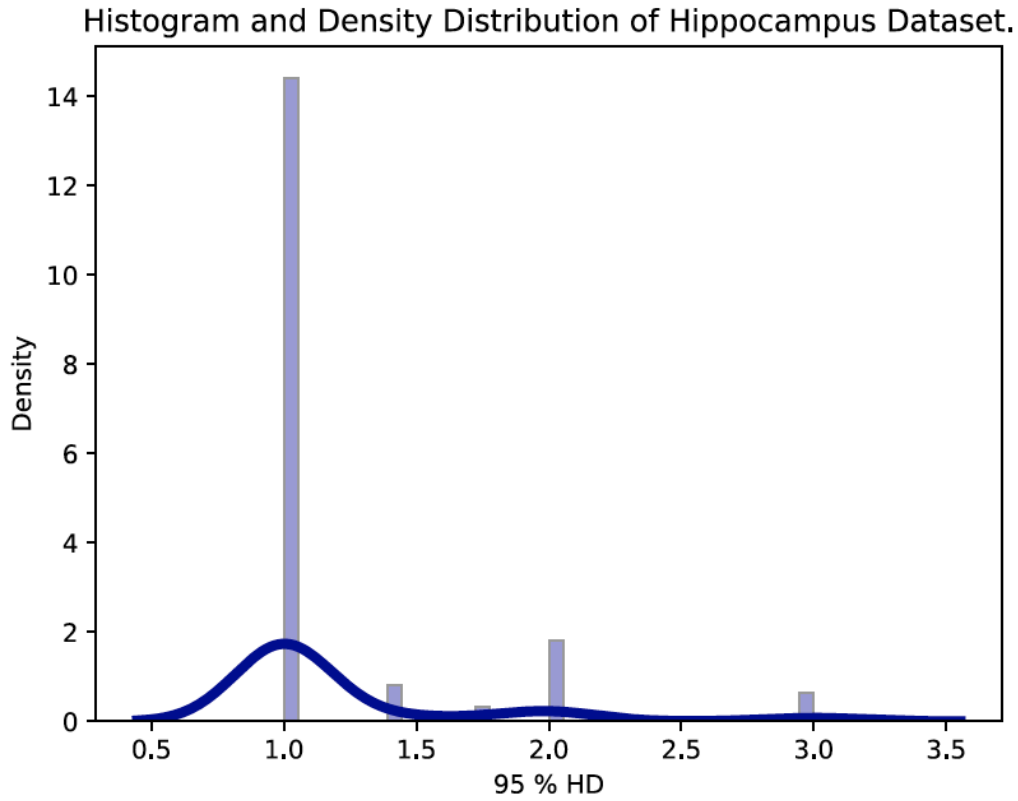


(El Jurdi, Varoquaux and Colliot, https://arxiv.org/pdf/2307.10926.pdf)

Some typical metric distributions

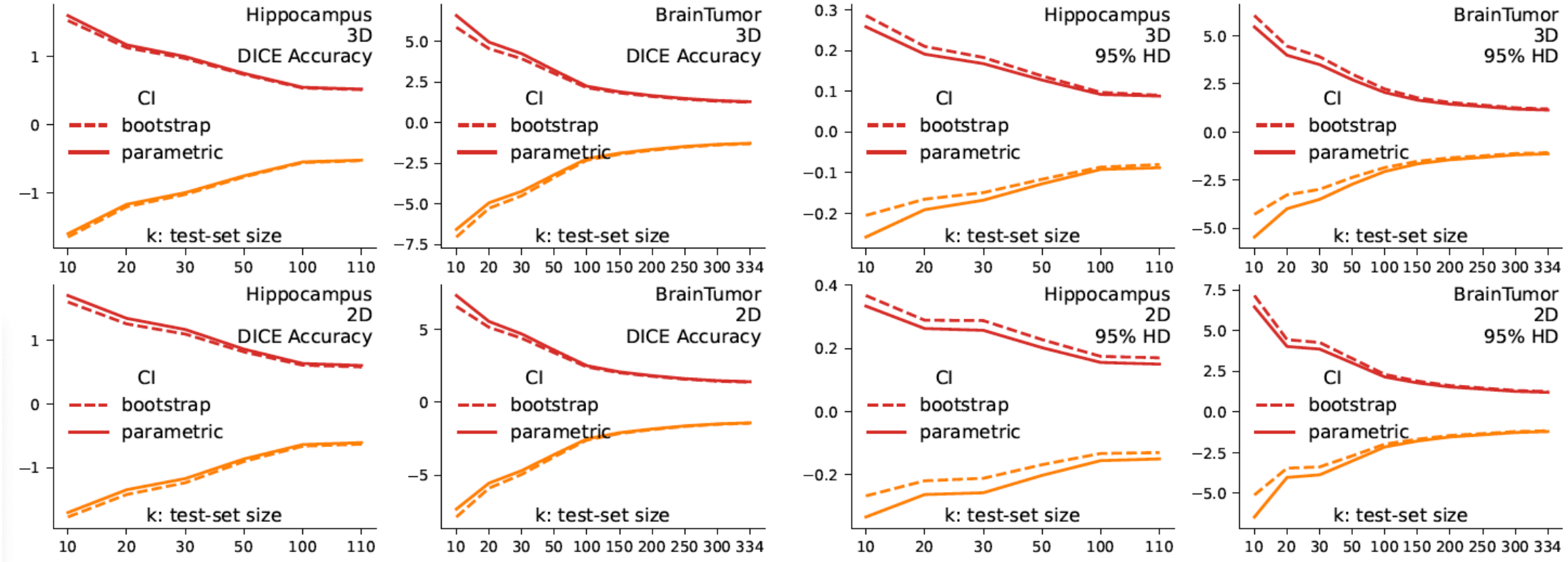
95% Hausdorff distance metric





(El Jurdi, Varoquaux and Colliot, https://arxiv.org/pdf/2307.10926.pdf)

Parametric vs bootstrap CIs



Parametric estimates are reasonable even for small size

(El Jurdi, Varoquaux and Colliot, https://arxiv.org/pdf/2307.10926.pdf)

Some typical confidence intervals

Classification (accuracy)

Table 2: ***Binomial confidence intervals on accuracy*** *(95% CI) for different values of ground-truth accuracy.*

N	65%	80%	90%	95%
100	[-9.0% 9.0%]	[-8.0% 8.0%]	[-6.0% 5.0%]	[-5.0% 4.0%]
1000	[-3.0% 2.9%]	[-2.5% 2.4%]	[-1.9% 1.8%]	[-1.4% 1.3%]
10000	[-0.9% 0.9%]	[-0.8% 0.8%]	[-0.6% 0.6%]	[-0.4% 0.4%]
100000	[-0.3% 0.3%]	[-0.2% 0.2%]	[-0.2% 0.2%]	[-0.1% 0.1%]

From (Varoquaux and Colliot, 2023- https://hal.science/hal-03682454/)

Some typical confidence intervals

Segmentation (Dice)

Typical value obtained for hippocampus segmentation

Typical value obtained for brain tumor segmentation

$\sigma \downarrow$	$n \rightarrow$	10	20						500	1000	2000
0.47	*SE*	0.15	0.11						0.02	0.01	0.01
	CI	[-0.29, 0.29]	[-0.21, 0.21]						[-0.04, 0.04]	[-0.03, 0.03]	[-0.02, 0.02]
0.81	*SE*	0.26							0.04	0.03	0.02
	CI	[-0.5, 0.5]	[-0.35, 0.35]						[-0.07, 0.07]	[-0.05, 0.05]	[-0.04, 0.04]
1	*SE*		0.22						0.04	0.03	0.02
	CI	[-0.62, 0.62]	[-0.44, 0.44]						[-0.09, 0.09]	[-0.06, 0.06]	[-0.04, 0.04]
2.79	*SE*	0.88	0.62	0.51	0.39	0.28	0.2	0.16	0.12	0.09	0.06
	CI	[-1.73, 1.73]	[-1.22, 1.22]	[-1.0, 1.0]	[-0.77, 0.77]	[-0.55, 0.55]	[-0.39, 0.39]	[-0.32, 0.32]	[-0.24, 0.24]	[-0.17, 0.17]	[-0.12, 0.12]
3.26	*SE*	1.03	0.73	0.6	0.46	0.33	0.23	0.19	0.15	0.1	0.07
	CI	[-2.02, 2.02]	[-1.43, 1.43]	[-1.17, 1.17]	[-0.9, 0.9]	[-0.64, 0.64]	[-0.45, 0.45]	[-0.37, 0.37]	[-0.29, 0.29]	[-0.2, 0.2]	[-0.14, 0.14]
5	*SE*	1.58	1.12	0.91	0.71	0.5	0.35	0.29	0.22	0.16	0.11
	CI	[-3.1, 3.1]	[-2.19, 2.19]	[-1.79, 1.79]	[-1.39, 1.39]	[-0.98, 0.98]	[-0.69, 0.69]	[-0.57, 0.57]	[-0.44, 0.44]	[-0.31, 0.31]	[-0.22, 0.22]
10.63	*SE*	3.36	2.38	1.94	1.5	1.06	0.75	0.61	0.48	0.34	0.24
	CI	[-6.59, 6.59]	[-4.66, 4.66]	[-3.8, 3.8]	[-2.95, 2.95]	[-2.08, 2.08]	[-1.47, 1.47]	[-1.2, 1.2]	[-0.93, 0.93]	[-0.54, 0.54]	[-0.47, 0.47]
11.26	*SE*	3.56	2.52	2.06	1.59	1.13	0.8	0.65	0.5	0.29	0.25
	CI	[-6.98, 6.98]	[-4.93, 4.93]	[-4.03, 4.03]	[-3.12, 3.12]	[-2.21, 2.21]	[-1.56, 1.56]	[-1.27, 1.27]	[-0.99, 0.99]	[-0.7, 0.7]	[-0.49, 0.49]
12	*SE*	3.79	2.68	2.19	1.7	1.2	0.85	0.69	0.54	0.38	0.27
	CI	[-7.44,	[-5.26, 5.26]	[-				.36]	[-1.05, 1.05]	[-0.74, 0.74]	[-0.53, 0.53]
13.13	*SE*	4.15	2.94						0.59	0.42	0.29
	CI	[-8.14, 8.14]	[-5.7 .75]	[-				.49]	[-1.15, 1.15]	[-0.81, 0.81]	[-0.58, 0.58]
20	*SE*	6.32	4.47						0.89	0.63	0.45
	CI	[-12.4, 12.4]	[-8.77, 8.77]	[-7				.26]	[-1.75, 1.75]	[-1.24, 1.24]	[-0.88, 0.88]

Table 1: Simulation of standard r different sizes n of the test set and for different values of σ. G needed to obtain a CI narrower than 1 %

(El Jurdi, Varoquaux and Colliot, https://arxiv.org/pdf/2307.10926.pdf)

Confidence intervals: take home

- **Always report confidence intervals for your metrics**
- **Do this on an independent test set**
- **Bootstrap is a reasonable approach (even though parametric approximations are not crazy)**
  - Easy to perform
  - No assumptions (except independence of course)
- **Typically intervals are wider than people think**
  - In particular for classification
- **Parametric methods are useful to derive a "rule-of-thumb" when one does not have access to the individual values of the test set**
  - E.g. when reviewing a paper

Statistical testing

- **Testing for trained models does not present major difficulties**
  - Non-parametric testing
  - McNemar test for classifiers
  - Paired t-test for continuous metrics (e.g. segmentation)

- **Always do this on an independent test set**
  - Not over cross-validation

Learning procedures

Comparing learning procedures

We just saw how to compare trained models

A different question is to compare learning procedures

- E.g. you want to know if ResNet is better than SVM for a task, not if your trained ResNet is better than your trained SVM

Trained models

- Only one source of variance: test set

Learning procedures

- A complex problem because
  - Different sources of variance
  - Difficult to compute unbiased variance estimates

Some sources of variance

Choice of test data. A given test set is an arbitrary sample of the actual population that we are trying to generalize to (see previous section)

Choice of data splits. Will change the trained models.

Hyper-parameter optimization. Another attempt to tune hyper-parameter would lead to slightly different choice. Thus benchmarks do not give an absolute characterization of a learning procedure, but are muddied by imperfect hyper-parameters.

Random seeds. Random choices in a learning procedure –initial weights, random drop-out for neural networks or bootstraps in bagging– lead to uncontrolled fluctuations in benchmarking results that do not characterize the procedure's ability to generalize to new data.

(Bouthillier et al, Proc MLS, 2021)

Some sources of variance

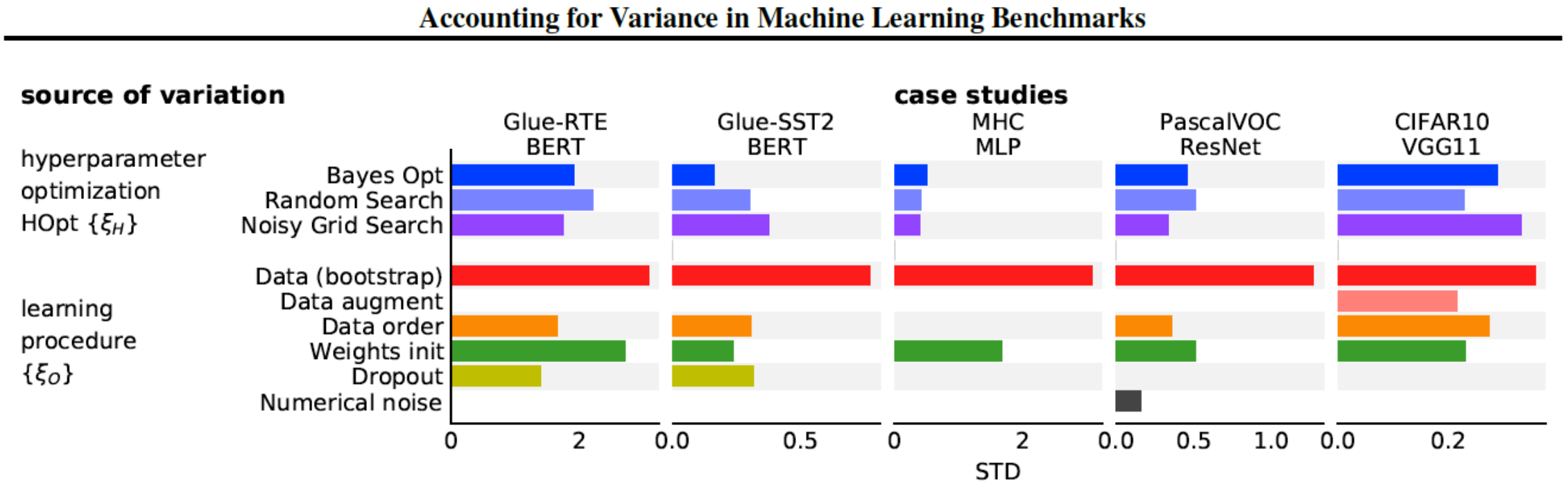


Figure 1. **Different sources of variation of the measured performance**: across our different case studies, as a fraction of the variance induced by bootstrapping the data. For hyperparameter optimization, we studied several algorithms.

(Bouthillier et al, Proc MLS, 2021)

How to proceed?

Open question. No perfect procedure yet.

Assessing all sources of variances is costly.

Some solutions have been proposed.

Bouthillier et al (Proc MLS, 2021)

- Perform many runs, varying sources of variance
- How frequently procedure A outperforms procedure B?

$$P(A > B) = \frac{1}{k} \sum_{i=1}^{k} I_{\{M_i^A > M_i^B\}}$$

where M_i^A (resp. M_i^B) is the performance metric measured on the i^{th} split for procedure A (resp. B)

Conclusion on validation

Conclusion

- **Metrics**
  - **Use the appropriate metrics**
    - E.g, never use accuracy with unbalanced datasets
  - Several metrics are necessary to get a comprehensive view of the performance
    - E.g. Do not only report accuracy!

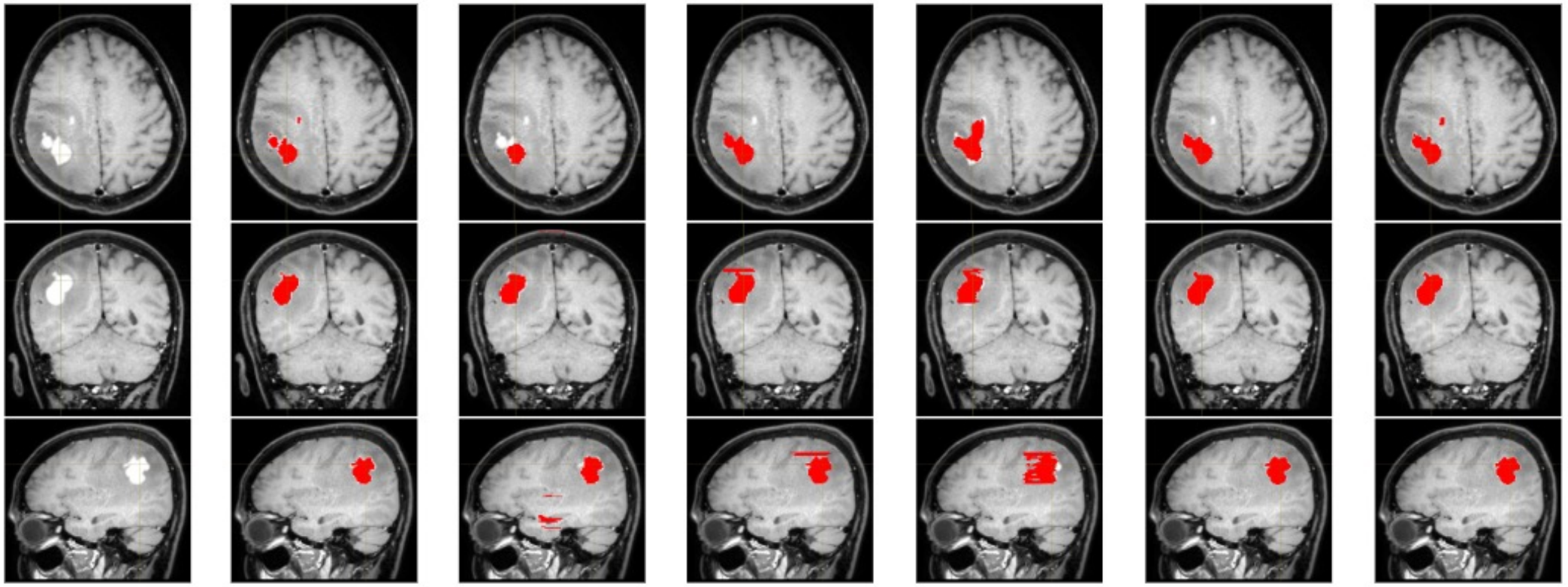
Conclusion

- **Validation**
  - Beware of data leakage
  - If you do deep learning, always use a separate test set, in addition to training and validation sets
    - Leave the test set untouched until the end

- **Do a thorough statistiscal analysis**
  - Trained models
    - Descriptive statistics, graphs
    - Inferential statistics: CIs, tests
  - Learning procedures
    - No universal approach but…
    - … not a good reason to do nothing

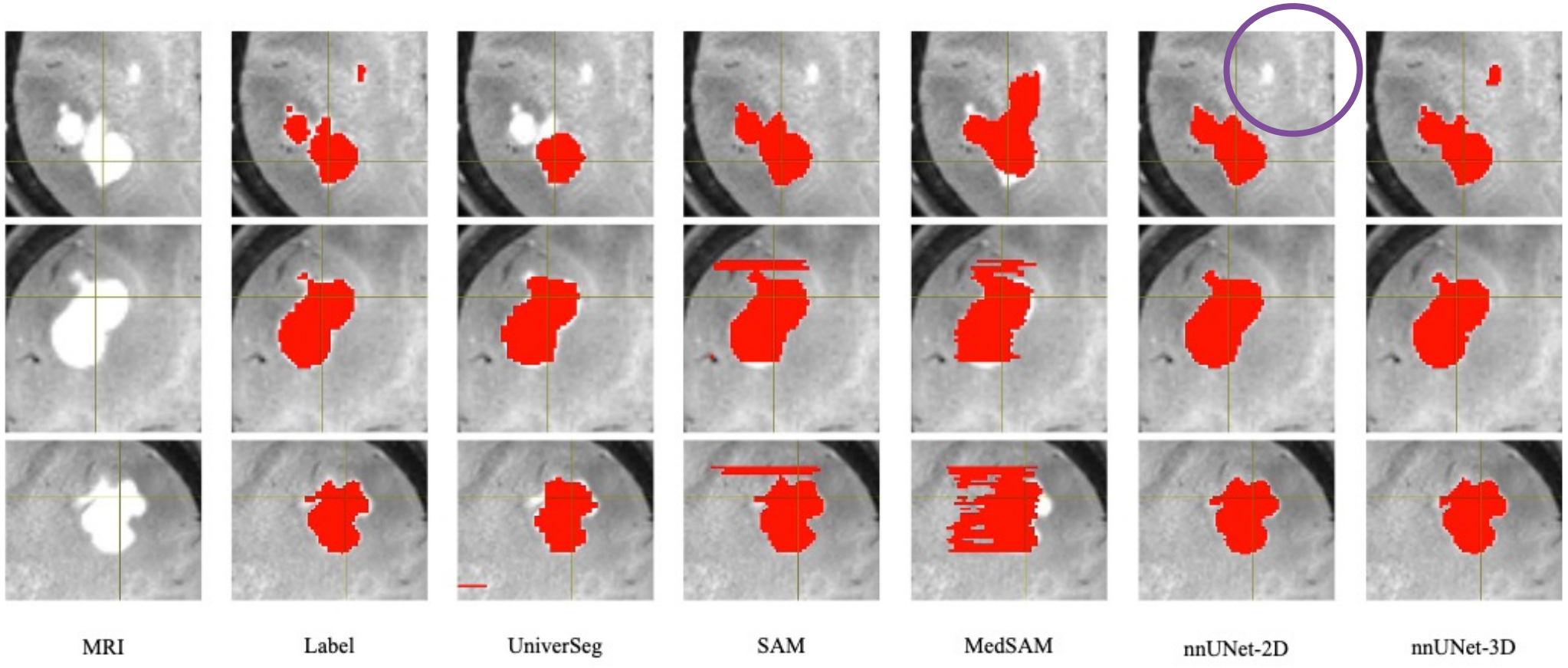
Conclusion

Always look at your data!



Conclusion

Always look at your data!



Summary

- **1. Define your task**

Summary

- **2. What data do you have?**

Summary

- **3. Design your evaluation**

- **A good ML project always start from the end!**

- **Evaluation metrics**
- **Test set**
  - Enough?
- **Training/validation set**
  - Enough?

Summary

- **4. Candidate approaches**

Summary

- **5. Cross-validation (train validation, no test)**
  - Assess different approaches
  - Train models
  - Hyperparameter tuning
  - Feature selection
  - Study stability of the learning procedure (but be very careful with inferential statistics)

Summary

- **6. Testing**
  - Descriptive statistics
  - Inferential statistics