



UNIVERSITY  
OF AMSTERDAM

# Machine Learning for Physics and Astronomy

Juan Rojo

VU Amsterdam & Theory group, Nikhef

*Natuur- en Sterrenkunde BSc (Joint Degree), Honours Track*  
*Lecture 1, 03/09/2020*

# Today's lecture

- ✿ Why Machine Learning? Basic concepts and terminology
- ✿ Supervised Learning: model fitting and polynomial regression
- ✿ The need for regularisation in ML (cross-validation)
- ✿ The bias/variance tradeoff

# Why Machine Learning?

input from *Lectures on Machine Learning*, S. Carrazza, TAE2018

# Why this course?

Machine Learning rightly deserves to be part of the **toolbox of a modern scientist**

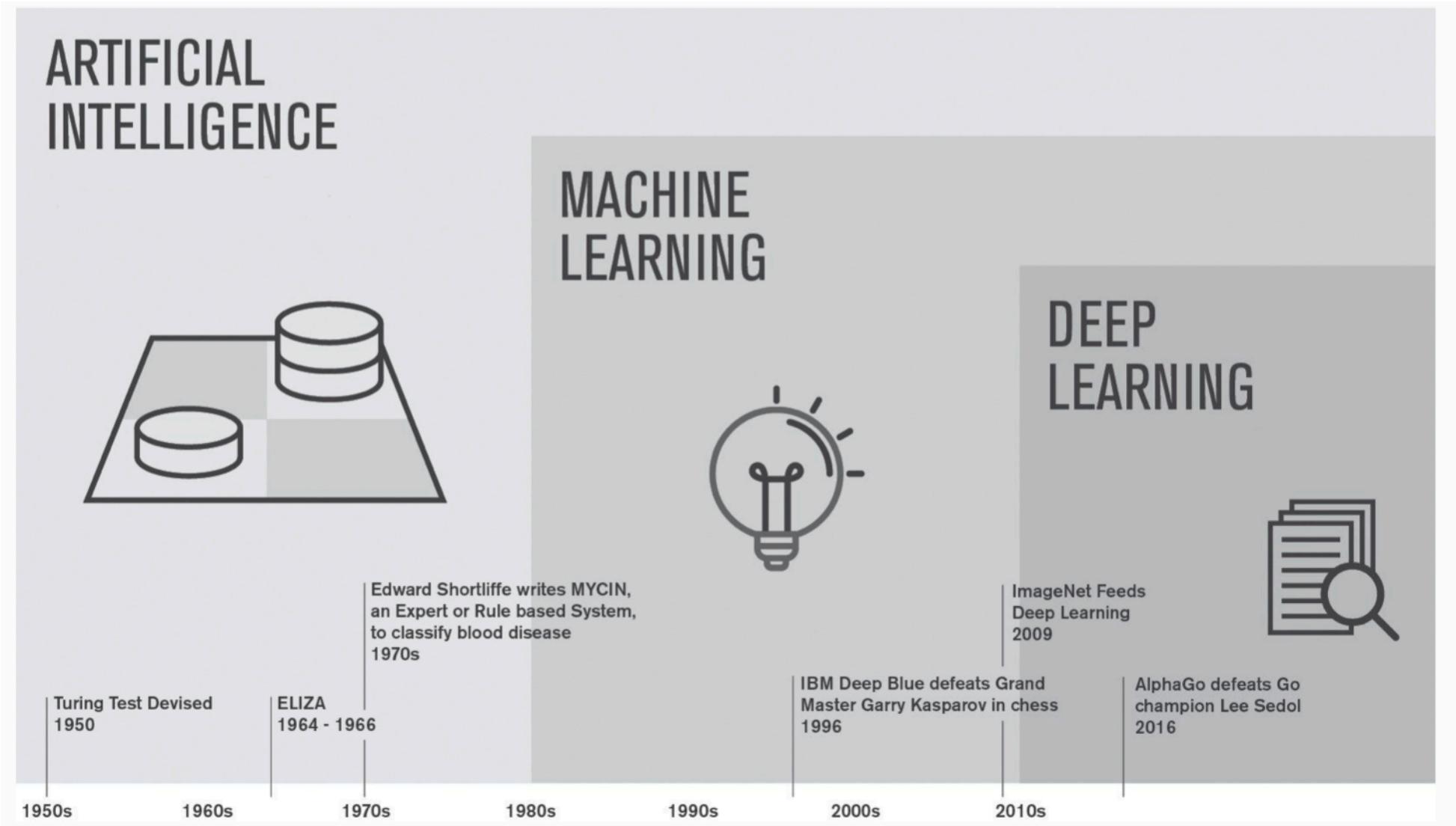
- 💡 Essential for building **modern models and algorithms** in various areas of physics and astronomy
- 💡 Very **fast developments** both in algorithms and in computing platforms have significantly extended the breadth of problems that can be tackled with ML
- 💡 Deep **physical connections** with many problems in physics and astronomy, e.g. quantum computation, condensed matter systems, conservation laws, ....
- 💡 Applied to problems even in very **formal fields**, e.g. string theory
- 💡 Large interested in the community, **societal implications** (AI hype)

Furthermore, **expertise in ML/AI** is powerful asset for also for careers outside academia

# AI vs ML

Machine Learning is part of the **Artificial Intelligence paradigm**

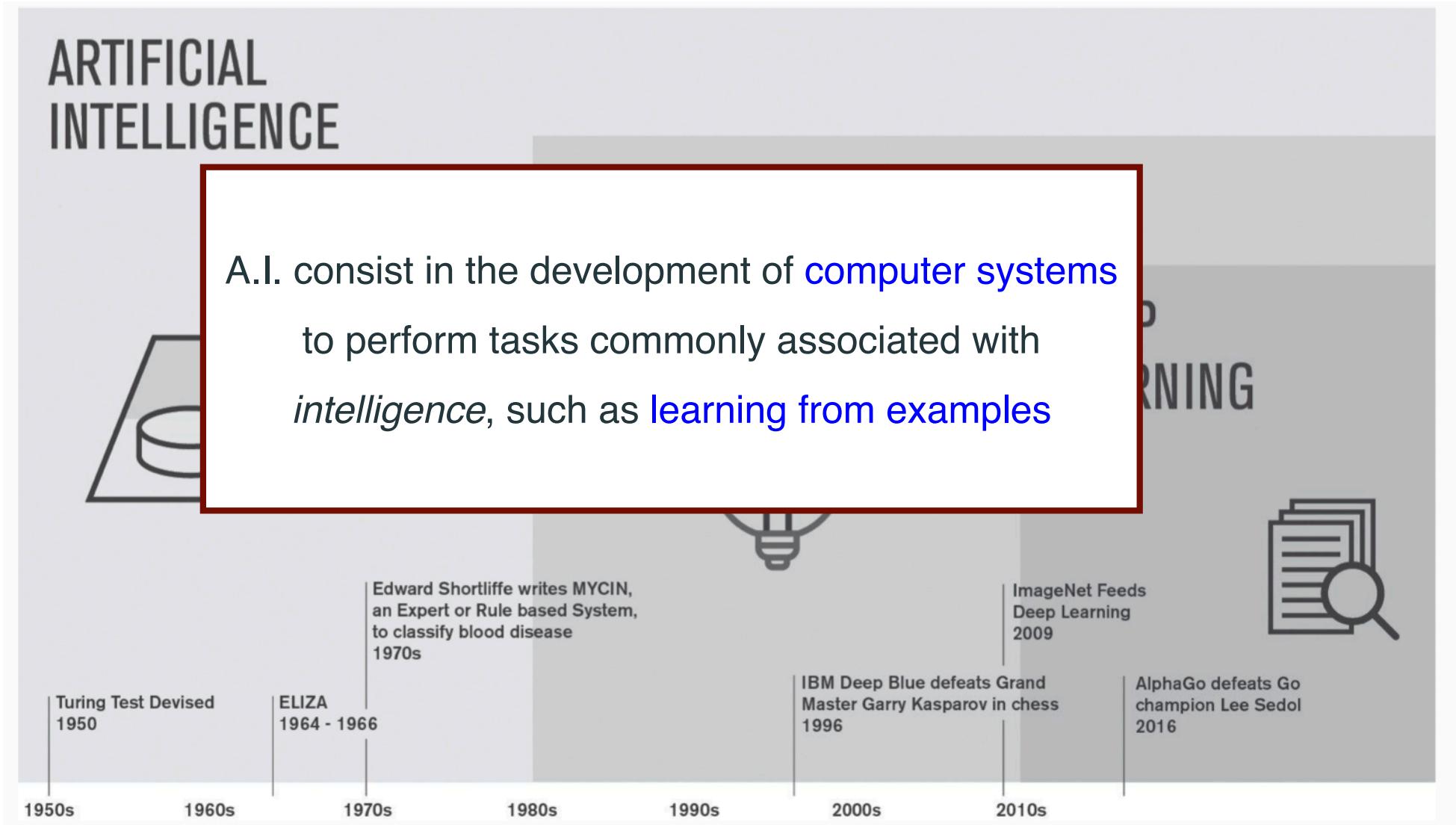
*AI is the science and engineering of making intelligent machines (McCarthy '56)*



# AI vs ML

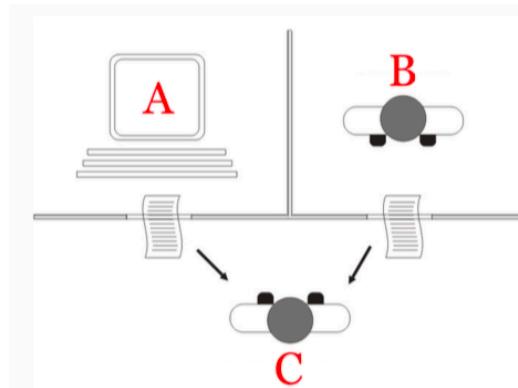
Machine Learning is part of the **Artificial Intelligence paradigm**

*AI is the science and engineering of making intelligent machines (McCarthy '56)*



# Problems in AI

Most problems tackled with Artificial Intelligence fall in **two categories**



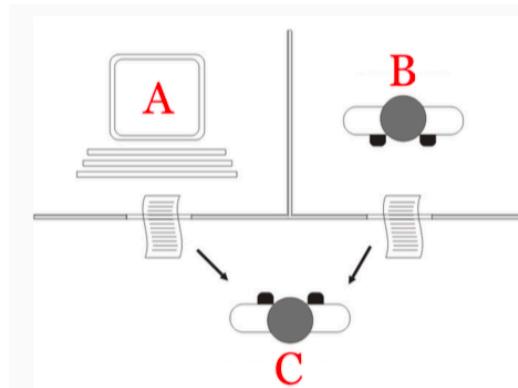
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9



# Problems in AI

Most problems tackled with Artificial Intelligence fall in **two categories**

- (1) abstract and formal: *easy for computers* but *difficult for humans*  
**knowledge-based approach**



e.g. chess (DeepBlue)

(chess is deterministic game with finite number of options )

- (2) intuitive, hard to formalize: *easy for humans* but *difficult for machines*  
**concept capture and generalisation**

0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9



e.g. pattern  
recognition

# Problems in AI

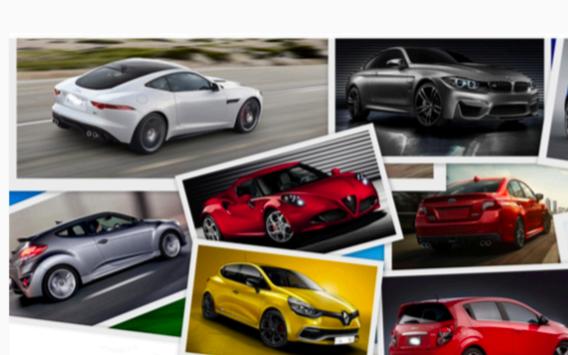
Most problems tackled with Artificial Intelligence fall in **two categories**

To excel in tasks which are intuitive to humans but difficult to machines, an A.I. system needs to **acquire its own knowledge**: the Machine is Learning

*unlike in chess, where there is no new knowledge to acquire once rules are spelled out*

Machine Learning algorithms allow computers the ability to **carry out a task without being explicitly programmed how to do it by learning from examples**

(2) intuitive, hard to formalize: *easy for humans but difficult for machines*  
**concept capture and generalisation**



e.g. pattern  
recognition

# ML is everywhere

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records
- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)
- **Human learning:**
  - Concept/human recognition
  - Computer vision
  - Product recommendation

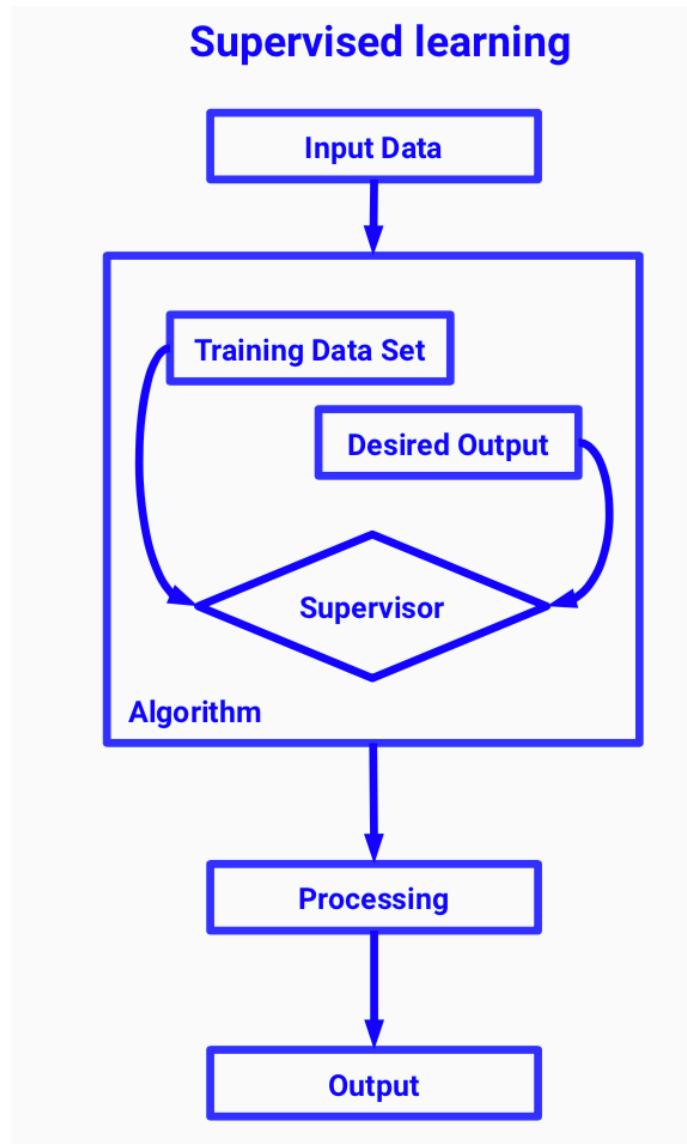
*Amazon, Netflix, Google, ....*



# Learning to learn

**Machine Learning** algorithms can be divided into several classes, including

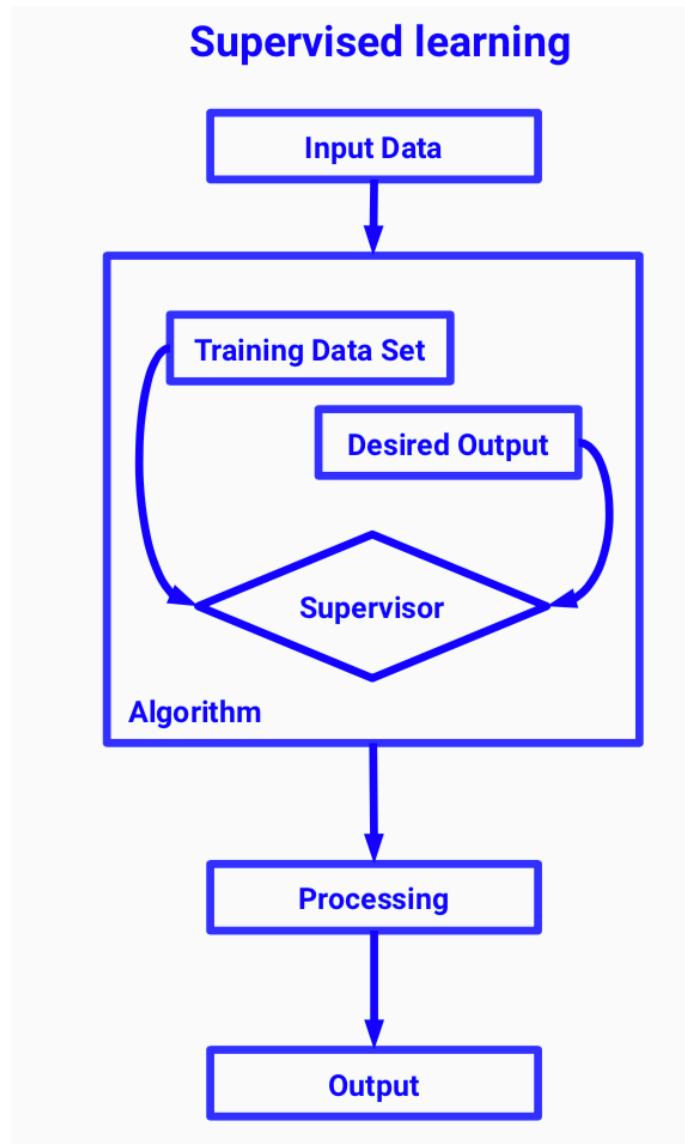
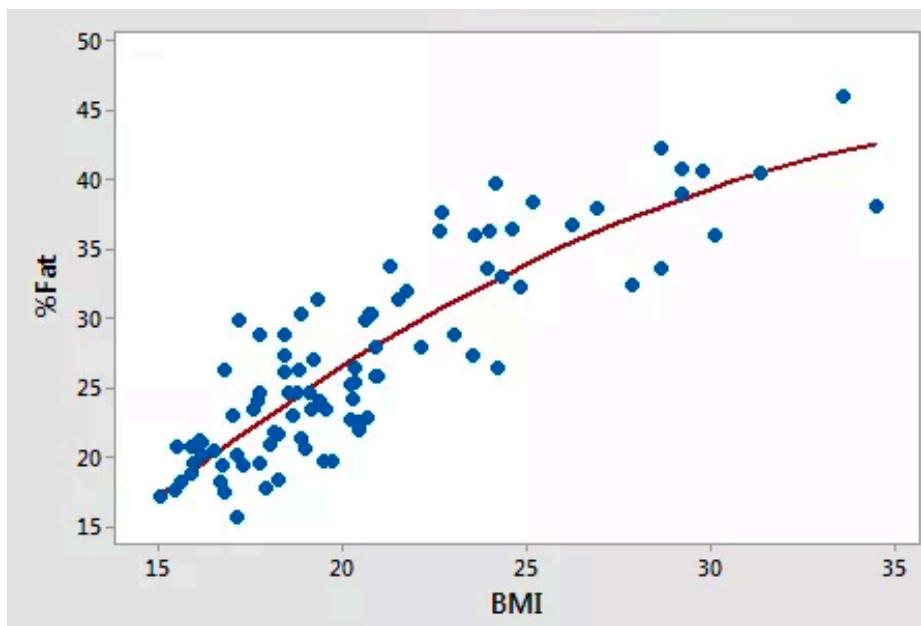
- 💡 **Supervised Learning:**  
regression, classification, ...



# Learning to learn

**Machine Learning** algorithms can be divided into several classes, including

- ⌚ **Supervised Learning:**  
regression, classification, ...



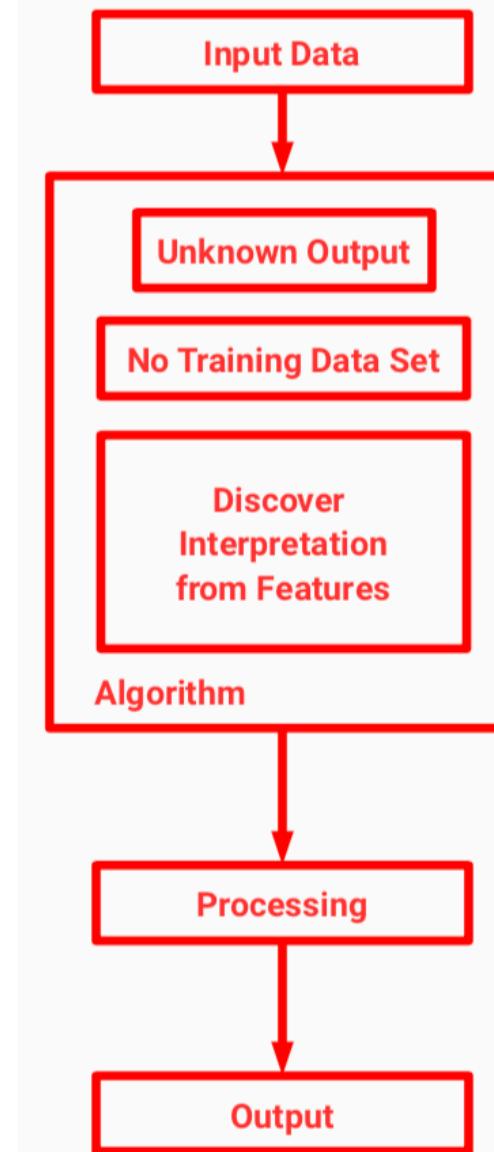
# Learning to learn

**Machine Learning** algorithms can be divided into several classes, including

- **Supervised Learning:**  
regression, classification, ...

- **Unsupervised Learning:**  
clustering, data dimensional  
reduction, ....

## Unsupervised learning

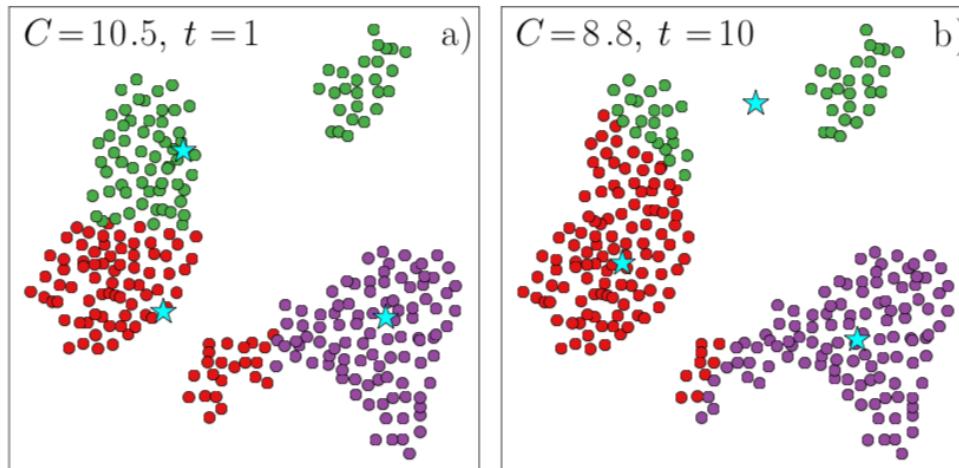


# Learning to learn

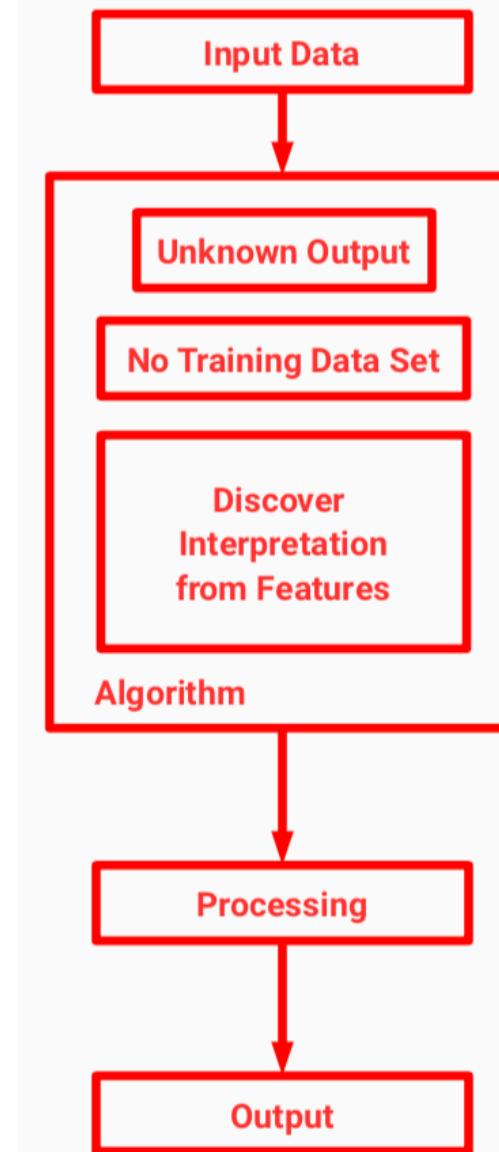
**Machine Learning** algorithms can be divided into several classes, including

⌚ **Supervised Learning:**  
regression, classification, ...

⌚ **Unsupervised Learning:**  
clustering, data dimensional  
reduction, ....



## Unsupervised learning



# Learning to learn

**Machine Learning** algorithms can be divided into several classes, including

- ⌚ **Supervised Learning:**  
regression, classification, ...
- ⌚ **Unsupervised Learning:**  
clustering, data dimensional reduction, ....
- ⌚ **Reinforcement learning:**  
efficiently react to changing environment



# Learning to learn

**Machine Learning** algorithms can be divided into several classes, including



📌 **Reinforcement learning:**  
efficiently react to changing  
environment

## Reinforcement learning

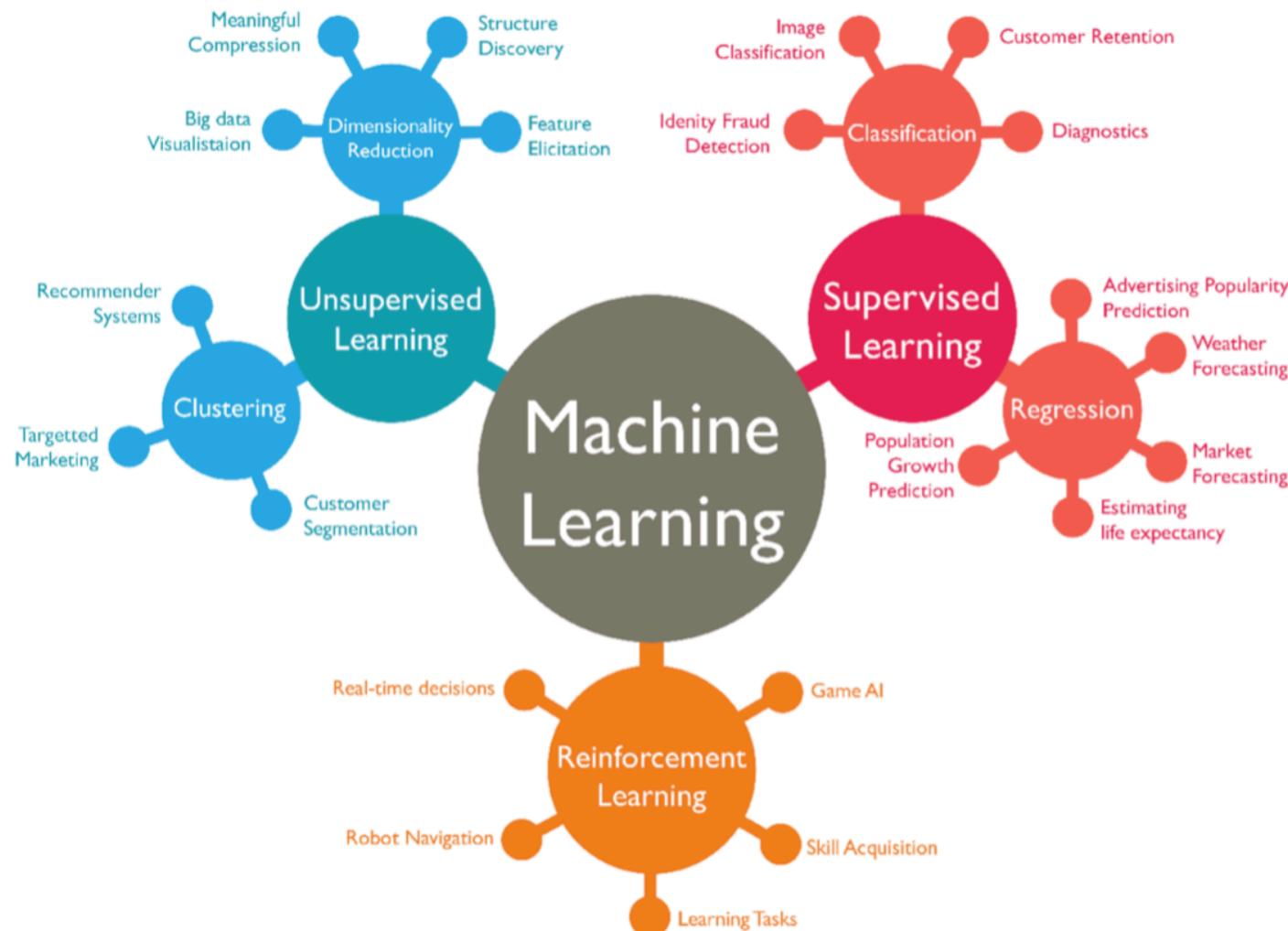


# The Machine Learning Galaxy I



*In this course we will have time to cover only subset of ML algorithms and applications!*

# The Machine Learning Galaxy II



*Image via [Abdul Rahid](#)*

*In this course we will have time to cover only subset of ML algorithms and applications!*

# **Supervised Learning: Model Fitting and Regression**

# Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

main property of supervised learning: the **training samples are labeled**

# Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

main property of supervised learning: the **training samples are labeled**

$$\mathbf{x}_i = \left( x_{i,1}, x_{i,2}, \dots, x_{i,p} \right) \rightarrow y_i$$

*data point (with p features)*   *label*

the label can be **discrete** (signal/noise, cat/dog) or **continuous** (output of function)

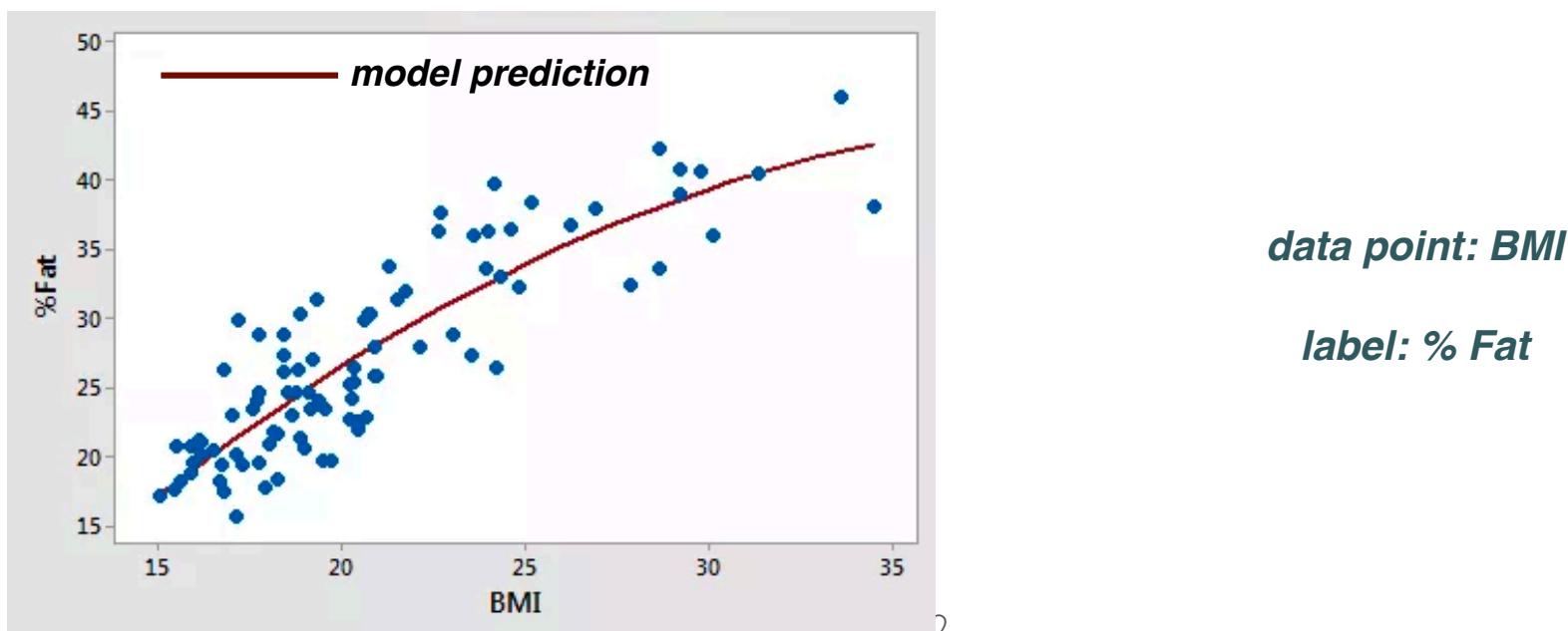
# Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

main property of supervised learning: the **training samples are labeled**

*continuous outputs:*  
*regression*



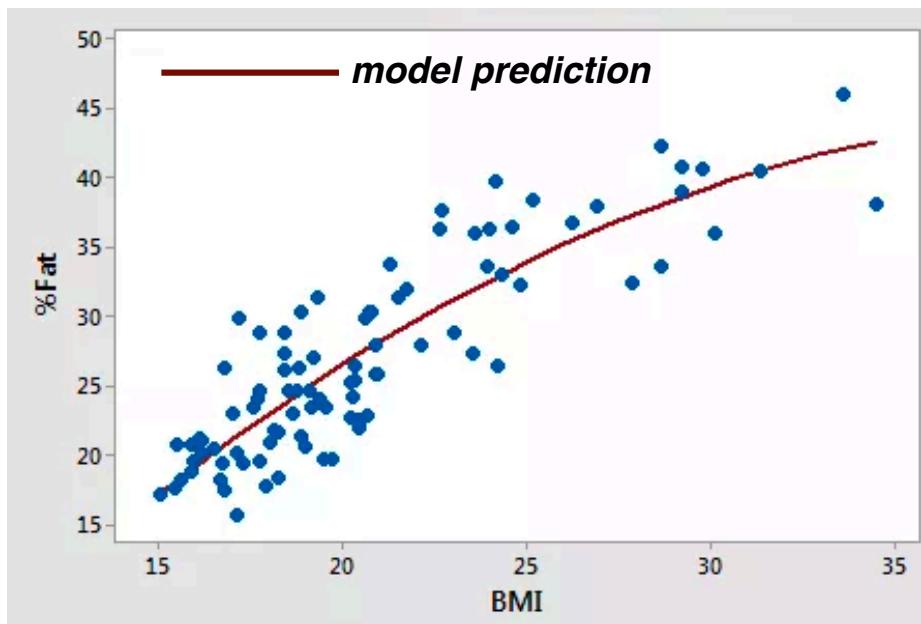
# Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

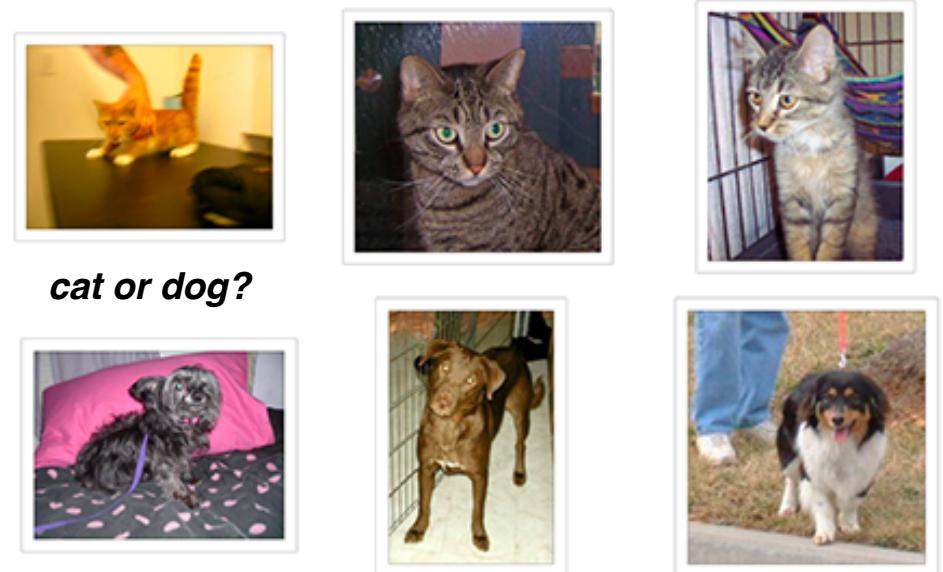
note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

main property of supervised learning: the **training samples are labeled**

*continuous outputs:*  
*regression*



*discrete outputs:*  
*classification*



# Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

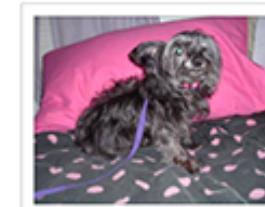
main property of supervised learning: the **training samples are labeled**

*discrete outputs:  
classification*

*data point: RGB values of each pixel*



*label: car or dog*



# Setting up the problem

problems in **Supervised Machine Learning** are defined by the following ingredients:

(1) *Input dataset:*  $\mathcal{D} = (X, Y)$

*array of **independent** variables*

*array of **dependent** variables*

$$X = (x_1, x_2, \dots, x_N)$$

$$Y = (y_1, y_2, \dots, y_N)$$

$$x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,p})$$

*each independent variable contains  $p$  features*

# Setting up the problem

problems in **Supervised Machine Learning** are defined by the following ingredients:

(1) **Input dataset:**  $\mathcal{D} = (X, Y)$

array of **independent**  
variables

$$X = (x_1, x_2, \dots, x_n)$$

$$x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,p})$$

array of **dependent**  
variables

$$Y = (y_1, y_2, \dots, y_n)$$

each **independent variable contains  $p$  features**

(2) **Model:**

$$f(X, \theta)$$

mapping between dependent  
and independent variables

$$f : X \rightarrow Y$$

model parameters

$$\theta = (\theta_1, \theta_2, \dots, \theta_m)$$

the more complex the problem, the more flexible the model

# Setting up the problem

problems in **Supervised Machine Learning** are defined by the following ingredients:

(1) **Input dataset:**  $\mathcal{D} = (X, Y)$

(2) **Model:**  $f(X, \theta)$

(3) **Cost function:**  $C(Y; f(X; \theta))$

The cost function measures how well the model (for a specific choice of its parameters) is able to **describe the input dataset**

*example of cost function for single dependent variable: sum of residuals squared*

$$C(Y; f(X; \theta)) = \frac{1}{n} \sum (y_i - f(x_i, \theta))^2$$

# Setting up the problem

problems in **Supervised Machine Learning** are defined by the following ingredients:

$$(1) \text{ Input dataset: } \mathcal{D} = (X, Y)$$

$$(2) \text{ Model: } f(X, \theta)$$

$$(3) \text{ Cost function: } C(Y; f(X; \theta))$$

The cost function measures how well the model (for a specific choice of its parameters) is able to describe the input dataset

Fitting the model means determining the values of its parameters which **minimise the cost function**

$$\frac{\partial C(Y; f(X; \theta))}{\partial \theta_i} \Bigg|_{\theta=\theta_{\text{opt}}} = 0$$

# Model fitting

before dealing with more sophisticated samples of Machine Learning, let's illustrate the main aspects of model fitting with a simple example: **polynomial regression in 1D**

First of all we can generate data following a **known underlying law** and then adding **stochastic noise**, to emulate a realistic situation

$$y_i = f(x_i) + \eta_i, \quad i = 1, \dots, n$$

*dependent variable*      *independent variable*      *uncorrelated Gaussian variable*

*the physical law  
(might be known)*

The larger the value of the **noise  $\sigma$**   
compared to the variations of the **law  $f$** ,  
the noisier the data

$$\langle \eta_i \rangle = 0$$
$$\langle \eta_i \eta_j \rangle = \sigma^2 \delta_{ij}$$

*noise strength*

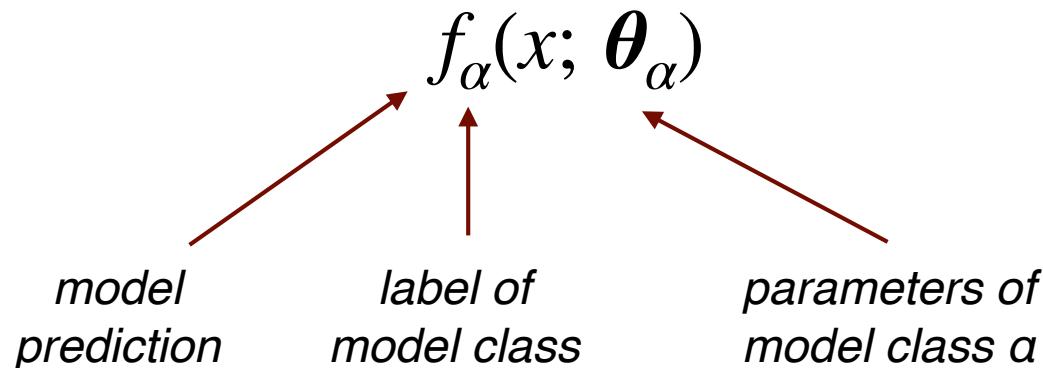
# Model fitting

in the real world we would have only the information about the ``measurements''

$$\mathcal{D} = (X, Y)$$

and our goal is to extract the **underlying physical law** from this data

We need to define **classes of models** that might provide a good description of the data, in a way that its parameters encode the main features of the physical law



we want to compare the performance of models with **different complexities**: different functional forms, number of parameters, intrinsic flexibility, ....

# Model fitting

The simplest possible model is a polynomial: **polynomial regression**

$$f_\alpha(x; \theta_\alpha) = \sum_{j=0}^{N_\alpha} \theta_{\alpha,j} x^j$$

for example the model class that contains all possible cubic polynomials is

$$f_3(x; \theta_3) = \sum_{j=0}^3 \theta_{3,j} x^j$$

a more complex model does not necessarily imply a more predictive one: the appropriate amount of complexity depends on the **features of the data sample** (e.g. size, variability)

in polynomial regression the model parameter are given by **least-squares method**

$$\hat{\theta} = \arg \min_{\theta} \left\{ \sum_{i=1}^n (y_i - f_\alpha(x_i; \theta_\alpha))^2 \right\}$$

*minimise sum of residuals squared*

# Model fitting

Simples case: **least-squares method** for order-one polynomials

$$f_q(x; \theta_1) = \sum_{j=0}^1 \theta_{1,j} x^j = \theta_{1,0} + \theta_{1,1} x$$

which is nothing but the **linear regression** taught in first-year statistics

$$\hat{\theta} = \arg \min_{\theta} \left\{ \sum_{i=1}^n (y_i - \theta_{1,0} - \theta_{1,1} x)^2 \right\}$$

where you can compute analytically the **best-fit values of the coefficients**

# Model fitting

Simples case: **least-squares method** for order-one polynomials

$$\frac{\partial}{\partial \theta_0} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i)^2 \Big|_{\widehat{\theta}_0} = 0$$

$$\frac{\partial}{\partial \theta_1} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i)^2 \Big|_{\widehat{\theta}_1} = 0$$

and by solving this linear system of equations one obtains the **model parameters**

$$\widehat{\theta}_1 = \frac{\langle xy \rangle - \langle x \rangle \langle y \rangle}{\langle x^2 \rangle - \langle x \rangle^2}$$

$$\widehat{\theta}_0 = \langle y \rangle - \widehat{\theta}_1 \langle x \rangle$$

*averages computed over the  $n$  elements of the data sample*

# **The need for regularisation**

# Best-fit model

What is the best strategy to **determine the model parameters?**

# Best-fit model

What is the best strategy to **determine the model parameters**?

Seems a silly question, surely those are simply **minimum of cost function**?

$$\hat{\theta} = \arg \min_{\theta} \left\{ \sum_{i=1}^n (y_i - f_{\alpha}(x_i; \theta_{\alpha}))^2 \right\}$$

# Best-fit model

What is the best strategy to determine the model parameters?

Seems a silly question, surely those are simply **minimum of cost function**?

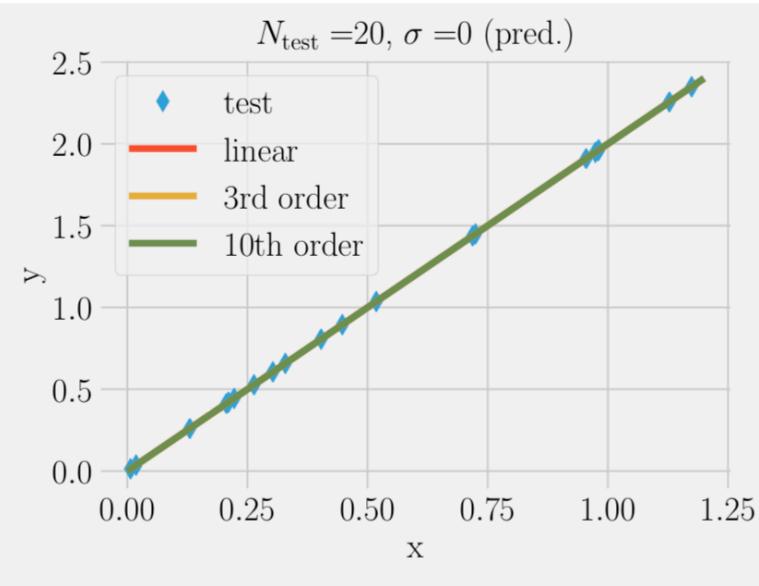
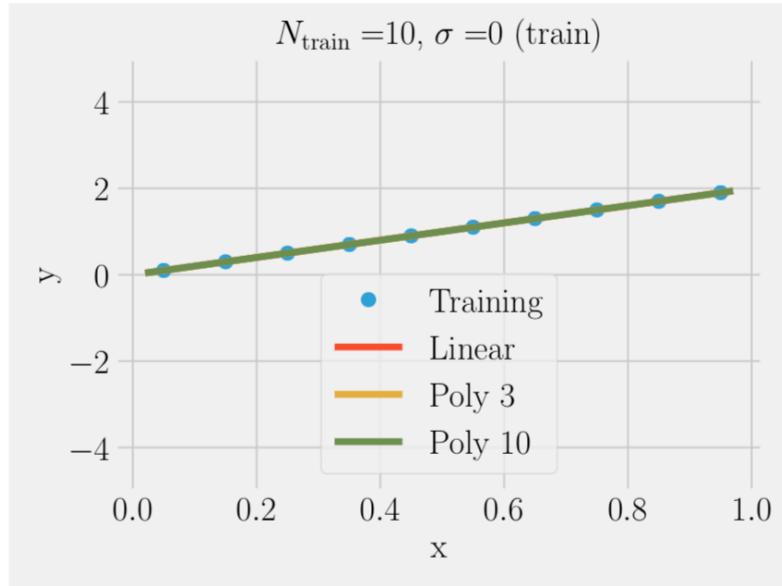
$$\hat{\theta} = \arg \min_{\theta} \left\{ \sum_{i=1}^n (y_i - f_{\alpha}(x_i; \theta_{\alpha}))^2 \right\}$$

However this is in general **not the case**, because:

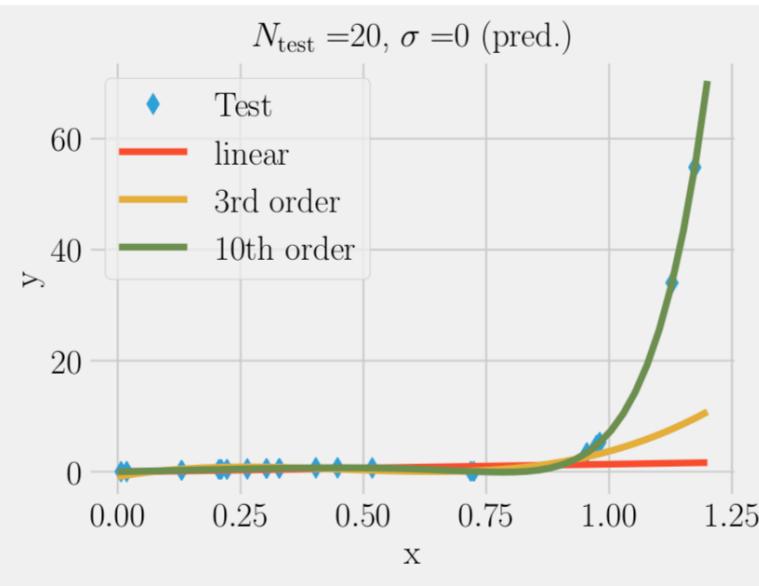
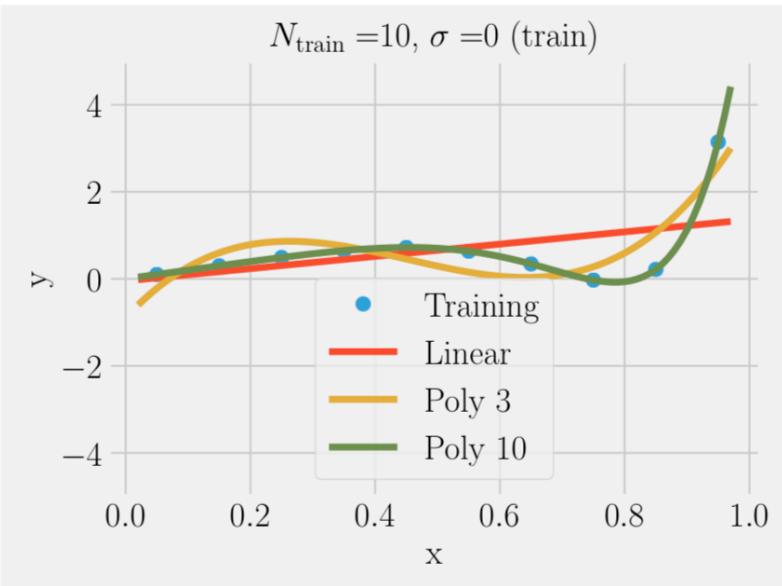
- ➊ Real world data is **noisy**: we want to **learn the underlying law**, not the statistical fluctuations
- ➋ More than fitting the data, our real goal is to create a model that **predicts future/different data**: we need figures of merit outside the training dataset!
- ➌ To ensure that our model describes the underlying law (and thus one can safely generalise) rather than the noise, a **regularisation procedure** needs to be used

# Model fitting

$$f(x) = 2x, \quad x \in [0,1]$$



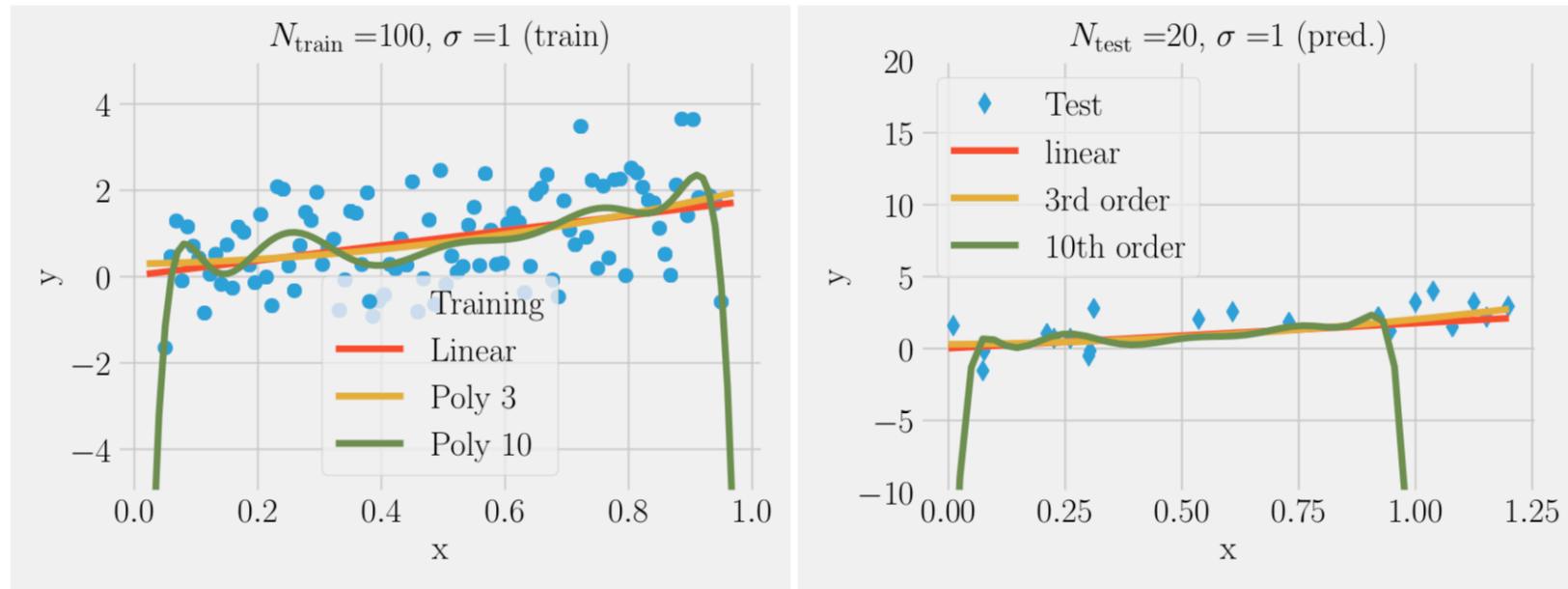
$$f(x) = 2x - 10x^5 + 15x^{10}, \quad x \in [0,1]$$



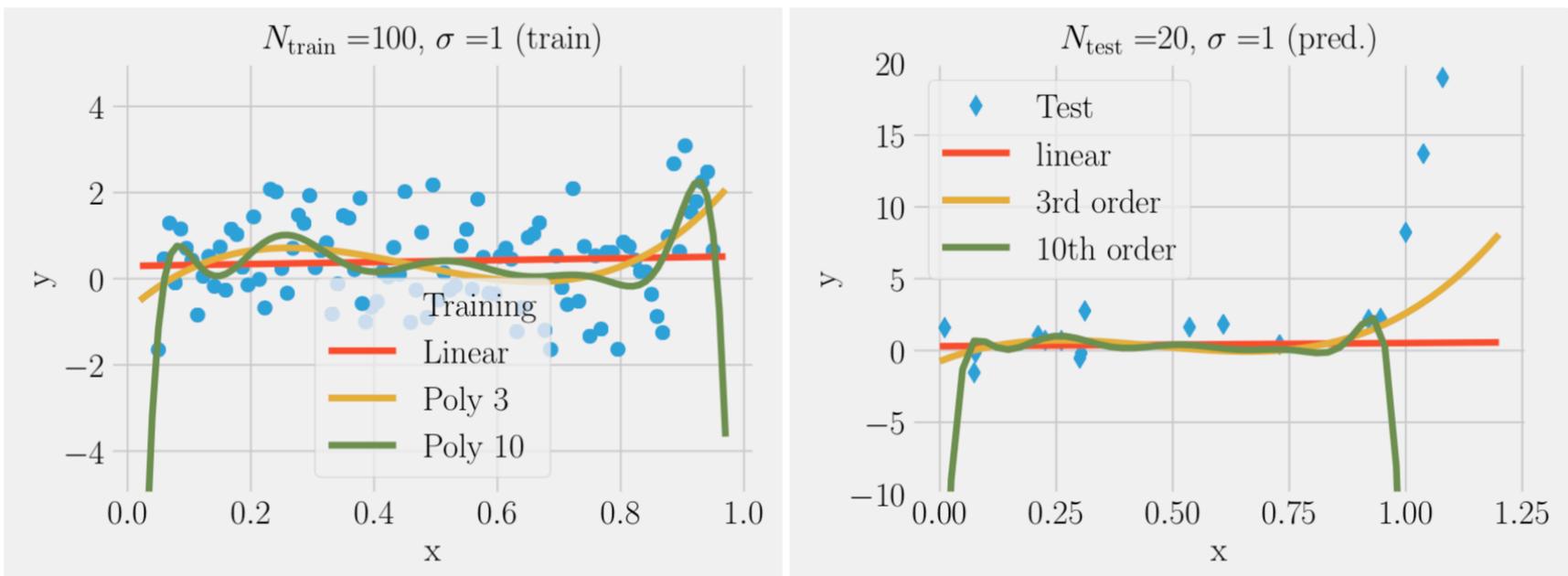
***in the absence of noise, fitting and predicting are identical, provided model has enough flexibility***

# Model fitting

$$f(x) = 2x, \quad x \in [0,1]$$



$$f(x) = 2x - 10x^5 + 15x^{10}, \quad x \in [0,1]$$



*in the presence of noise, models with less complexity can exhibit improved predictive power*

# Cross-validation: regularisation

What is the best strategy to determine the model parameters?

first of all, divide input dataset into **two disjoint sets**

$$\mathcal{D} = \mathcal{D}_{\text{tr}} + \mathcal{D}_{\text{val}}$$

*training dataset:*  
*used to extract model parameters*

*validation dataset:*  
*used to monitor generalisation power*

the model parameters are determined using information from training dataset

$$\hat{\theta} = \arg \min_{\theta} \{ C(Y_{\text{th}}, f(X_{\text{tr}}; \theta)) \}$$

while the **performance of the model** (generalisation, extrapolation) is evaluated by computing the cost function on the validation dataset

$$C(Y_{\text{val}}, f(X_{\text{val}}; \hat{\theta}))$$

# Cross-validation (regularisation)

*In-sample (training) error*  $\longrightarrow E_{\text{tr}} \equiv C(Y_{\text{tr}}, f(X_{\text{tr}}; \hat{\theta}))$

*Out-of-sample (validation) error*  $\longrightarrow E_{\text{val}} \equiv C(Y_{\text{val}}, f(X_{\text{val}}; \hat{\theta}))$

Splitting the data into mutually exclusive training and validation sets provides an unbiased estimate for the **predictive performance of the model**

In ML problems one should select the model that **minimises** the out-of-sample error  $E_{\text{val}}$ , since this is the model that generalises in the most efficient way

*note that choices on how to partition  
the dataset will affect the conclusion  
about which is best model*

# Cross-validation (regularisation)

*In-sample (training) error*  $\longrightarrow E_{\text{tr}} \equiv C(Y_{\text{tr}}, f(X_{\text{tr}}; \hat{\theta}))$

*Out-of-sample (validation) error*  $\longrightarrow E_{\text{val}} \equiv C(Y_{\text{val}}, f(X_{\text{val}}; \hat{\theta}))$

Splitting the data into mutually exclusive training and validation sets provides an unbiased estimate for the **predictive performance of the model**

In ML problems one should select the model that **minimises** the out-of-sample error  $E_{\text{val}}$ , since this is the model that generalises in the most efficient way

**Fitting is not predicting:** in general the model that describes better a given set of data will not be the one that generalises and predicts better related datasets

# Regularisation

In ML, the common goal of regularisation strategies is to modify the learning algorithms to **reduce its generalisation error without increasing its training error**

$$C(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \theta))^2 \quad \textit{unregularised cost function}$$

$$C(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \theta))^2 + \lambda \theta^T \theta \quad \textit{regularised cost function}$$

*regularisation hyperparameter,  
to be tuned*

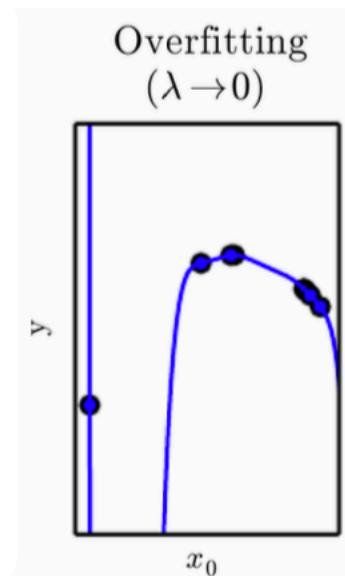
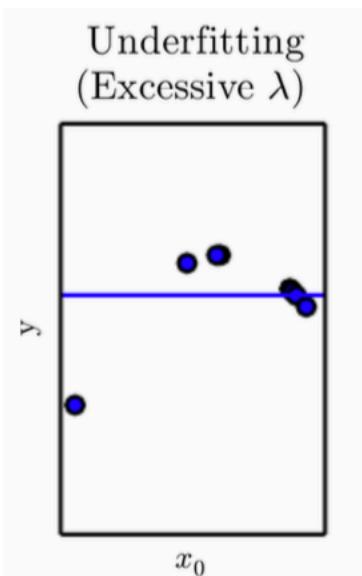
*weight-decay regularisation:  
model parameters with smaller  
L2-norms are preferred*

# Regularisation

In ML, the common goal of regularisation strategies is to modify the learning algorithms to **reduce its generalisation error without increasing its training error**

$$C(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \theta))^2 \quad \text{unregularised cost function}$$

$$C(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \theta))^2 + \lambda \theta^T \theta \quad \text{regularised cost function}$$



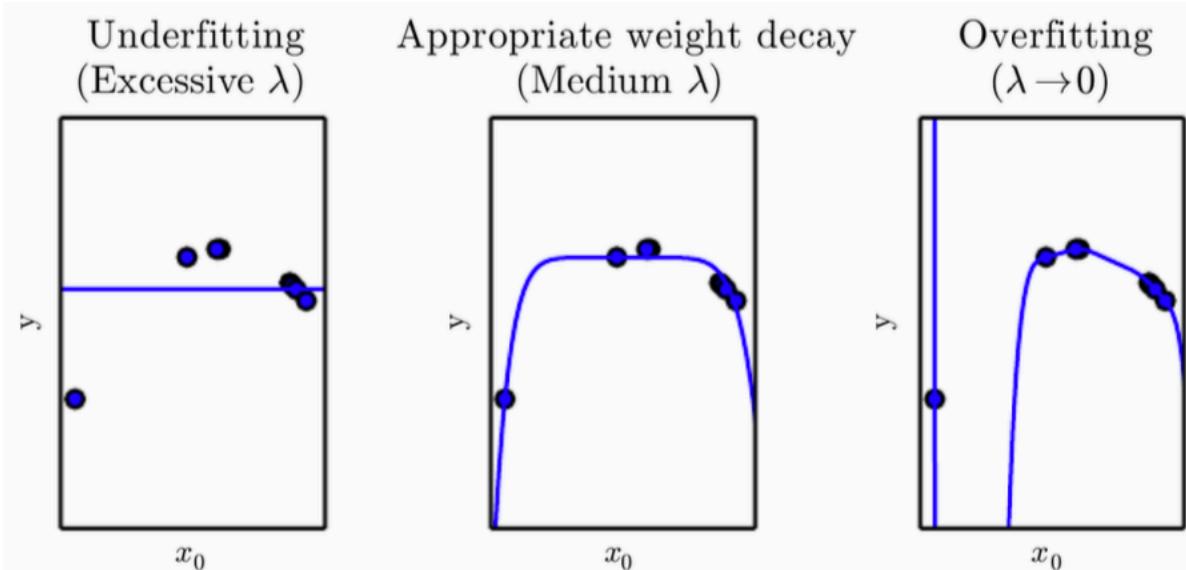
*no free lunch,  
careful tuning of  
regularisation  
required*

# Regularisation

In ML, the common goal of regularisation strategies is to modify the learning algorithms to **reduce its generalisation error without increasing its training error**

$$C(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \theta))^2 \quad \text{unregularised cost function}$$

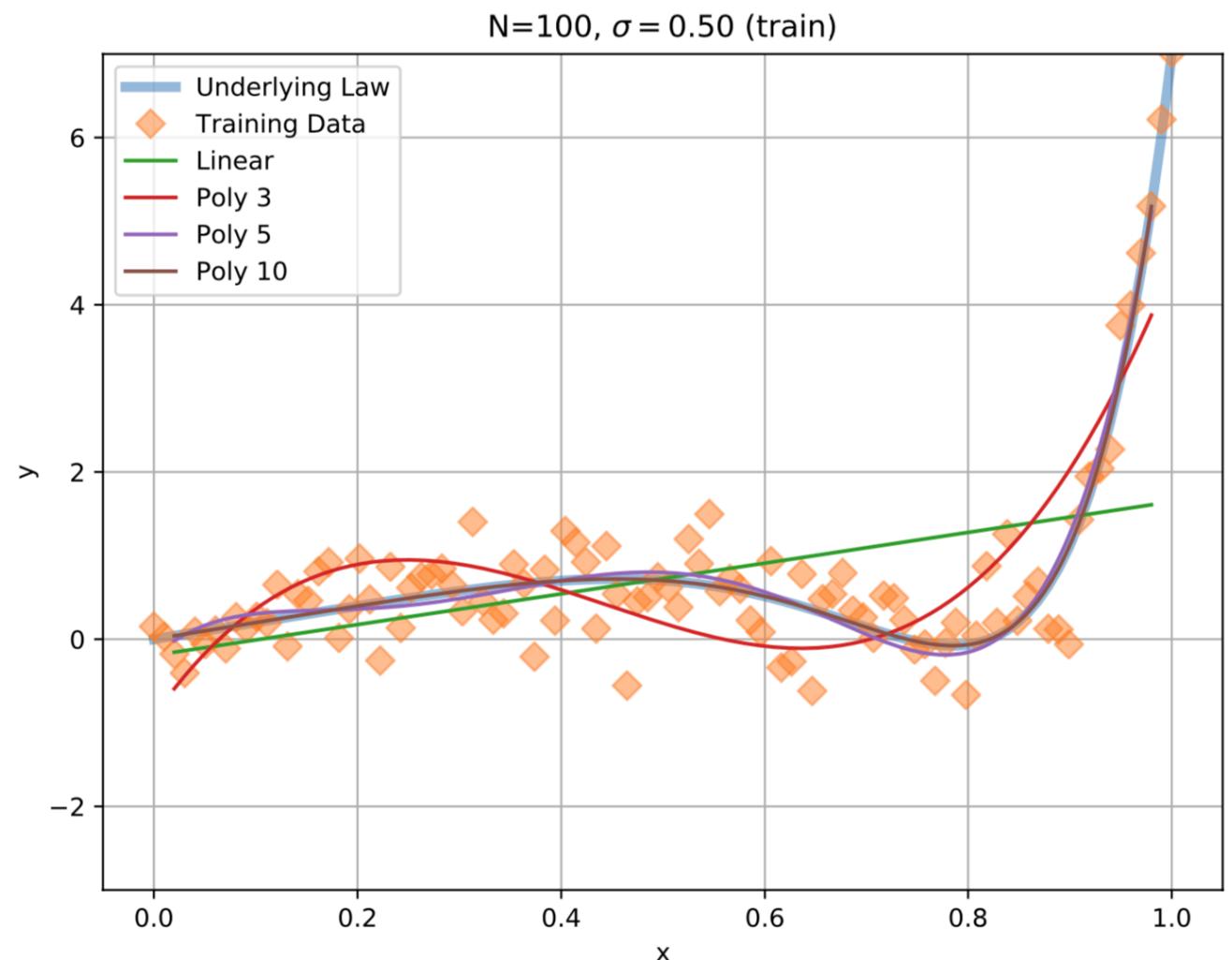
$$C(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \theta))^2 + \lambda \theta^T \theta \quad \text{regularised cost function}$$



*no free lunch,  
careful tuning of  
regularisation  
required*

# Tutorial 1 Exercise 1a

- Generate data with different underlying laws and try to model it using **polynomial regression**
- Study how the cost function to the training data depends on *i*) the complexity of the underlying law, *ii*) the flexibility of the model, *iii*) the number of training points
- Now generate a validation dataset, either within the training data range  $[0,1]$  or outside it. Study which model is **more predictive** as a function of the same parameters as before in the two cases
- Can you **identify overfitting**? Can you think of a way of avoiding it?



# Why Machine Learning is difficult

- 💡 **Fitting existing data** is conceptually different from **making predictions about new data**
- 💡 Increasing the model complexity can improve the description of the training data but **reduce the predictive power** of the model due to overfitting, unless a suitable regularisation strategy is implemented
- 💡 For complex and/or small datasets, simple models can be better at prediction than complex models. The **“right” amount of complexity** cannot in general be determined from first principles
- 💡 It is difficult to **generalise** beyond the situations encountered in the training set: the model cannot learn what it has not seen

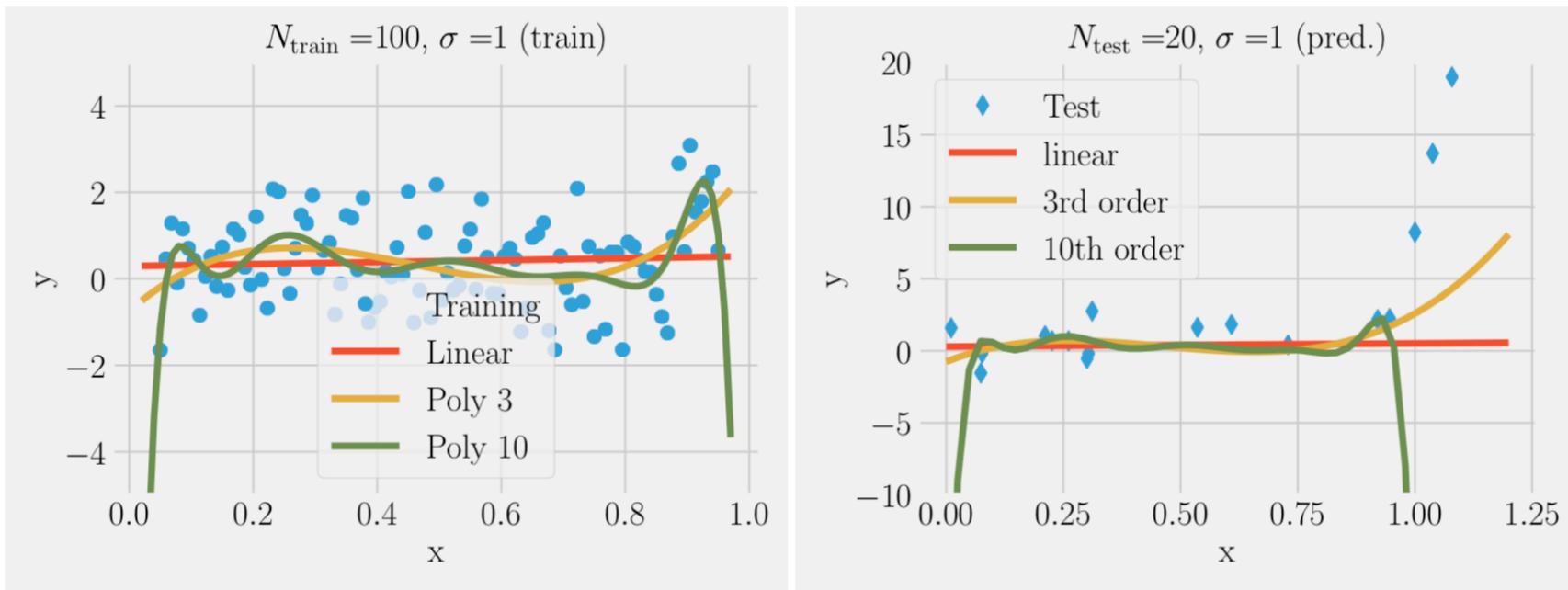
# Why Machine Learning is difficult

- 💡 **Fitting existing data** is conceptually different from **making predictions about new data**
- 💡 Increasing the model complexity can improve the description of the training data but **reduce the predictive power** of the model due to overfitting, unless a suitable regularisation strategy is implemented
- 💡 For complex and/or small datasets, simple models can be better at prediction than complex models. The **“right” amount of complexity** cannot in general be determined from first principles
- 💡 It is difficult to **generalise** beyond the situations encountered in the training set: the model cannot learn what it has not seen
- 💡 Many problems that are **approachable in principle** can become **unfeasible in practice**, e.g. due to computational limitations, lack of convergence, instability .....

*Deep Learning successes boosted by hardware developments*

# The bias-variance tradeoff

# The bias-variance trade-off



increasing the model complexity can improve the description of the training data  
but **reduce the predictive power** of the model due to overfitting

# The bias-variance trade-off

Our starting point is the **underlying law**  $y=f(x)$  which we aim to learn from a dataset  $(x_i, y_i), i=1, \dots, N$ . We will also need an **hypothesis set**  $H$  containing all functions that we consider to be good candidates for the underlying law

The goal of **Statistical Learning Theory** is to determine a function from the hypothesis set  $H$  that approximates  $f(x)$  as best as possible, ideally in a strict mathematical limit

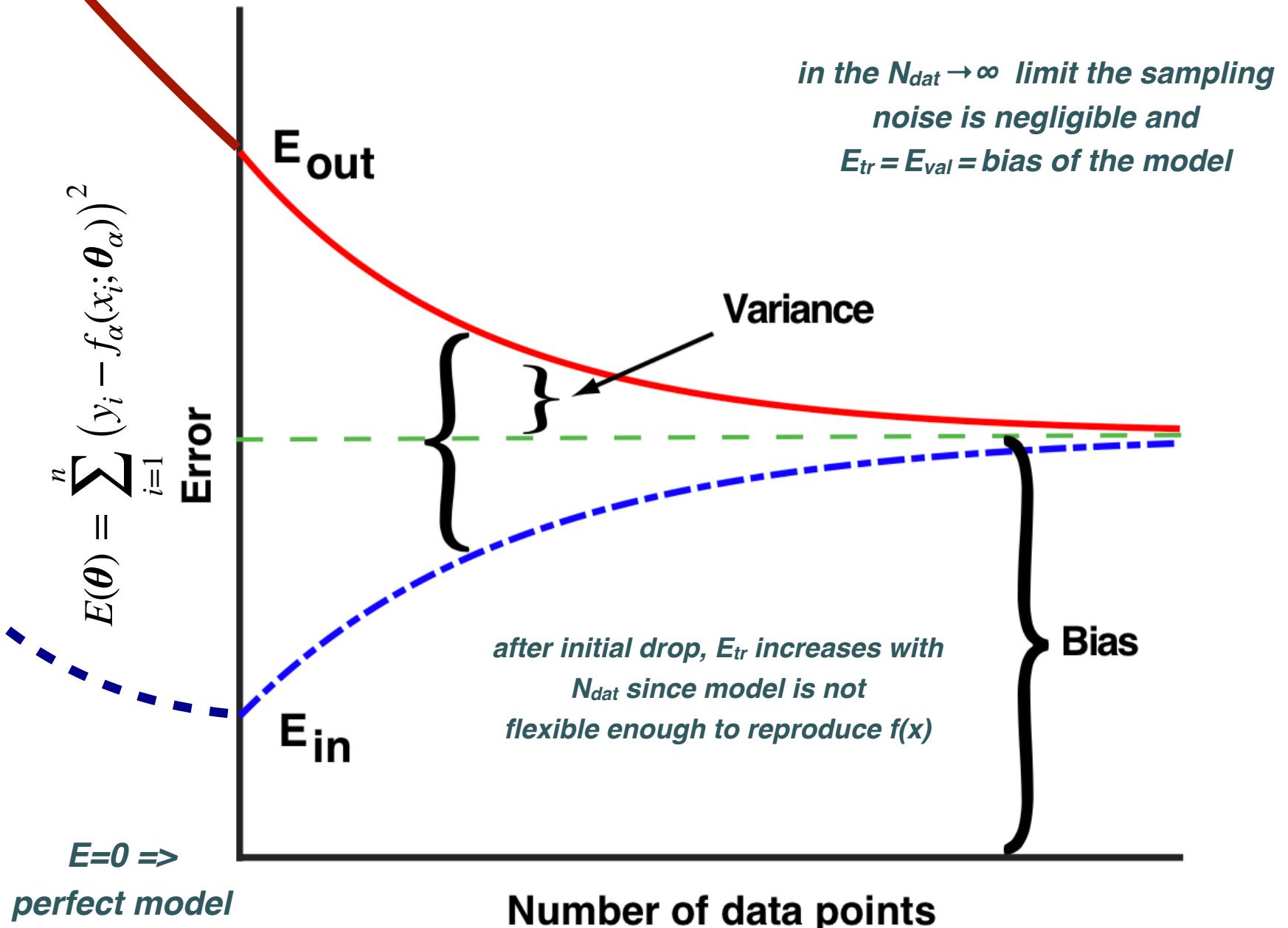
Here we will provide an **intuitive picture** of how Statistical Learning works

Consider the following situation:

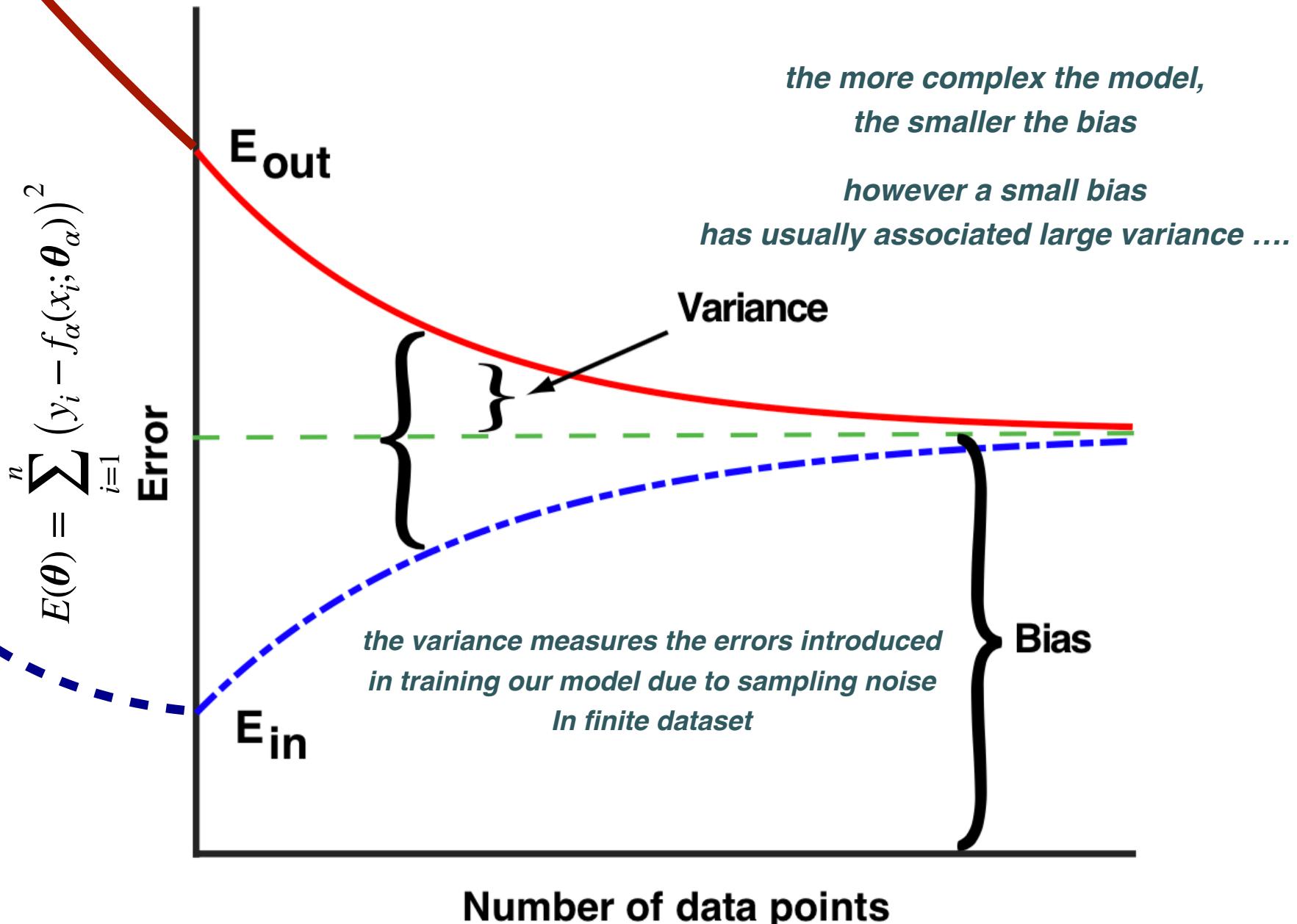
- The **underlying law** is so complex that we cannot aim to exactly reproduce  $f(x)$
- We want to study the dependence of  $E_{tr}$  and  $E_{val}$  with the number of data points

This setting allows us to present one of the most important concepts in the theory of Machine Learning: **the bias-variance tradeoff**

# The bias-variance tradeoff

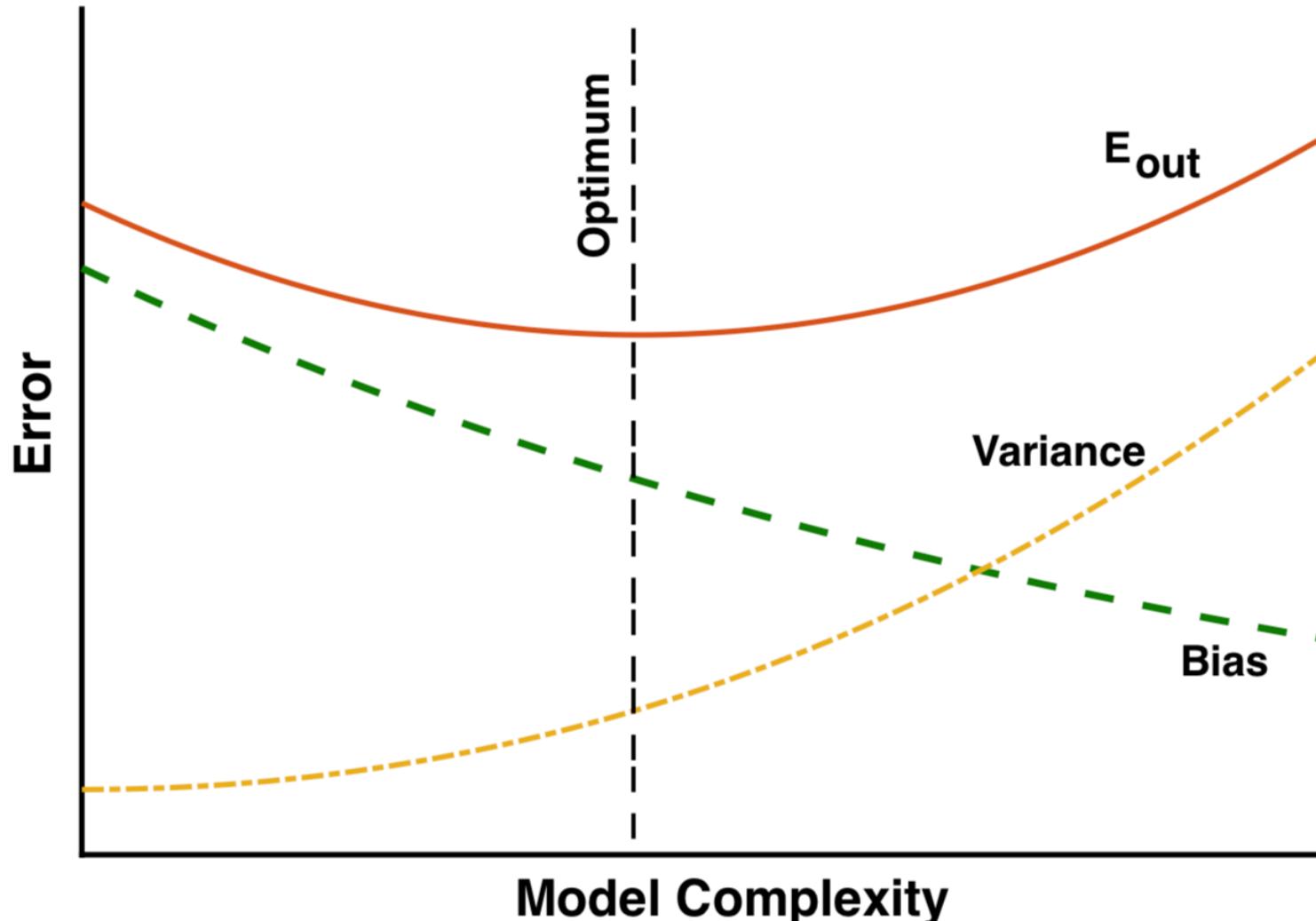


# The bias-variance tradeoff



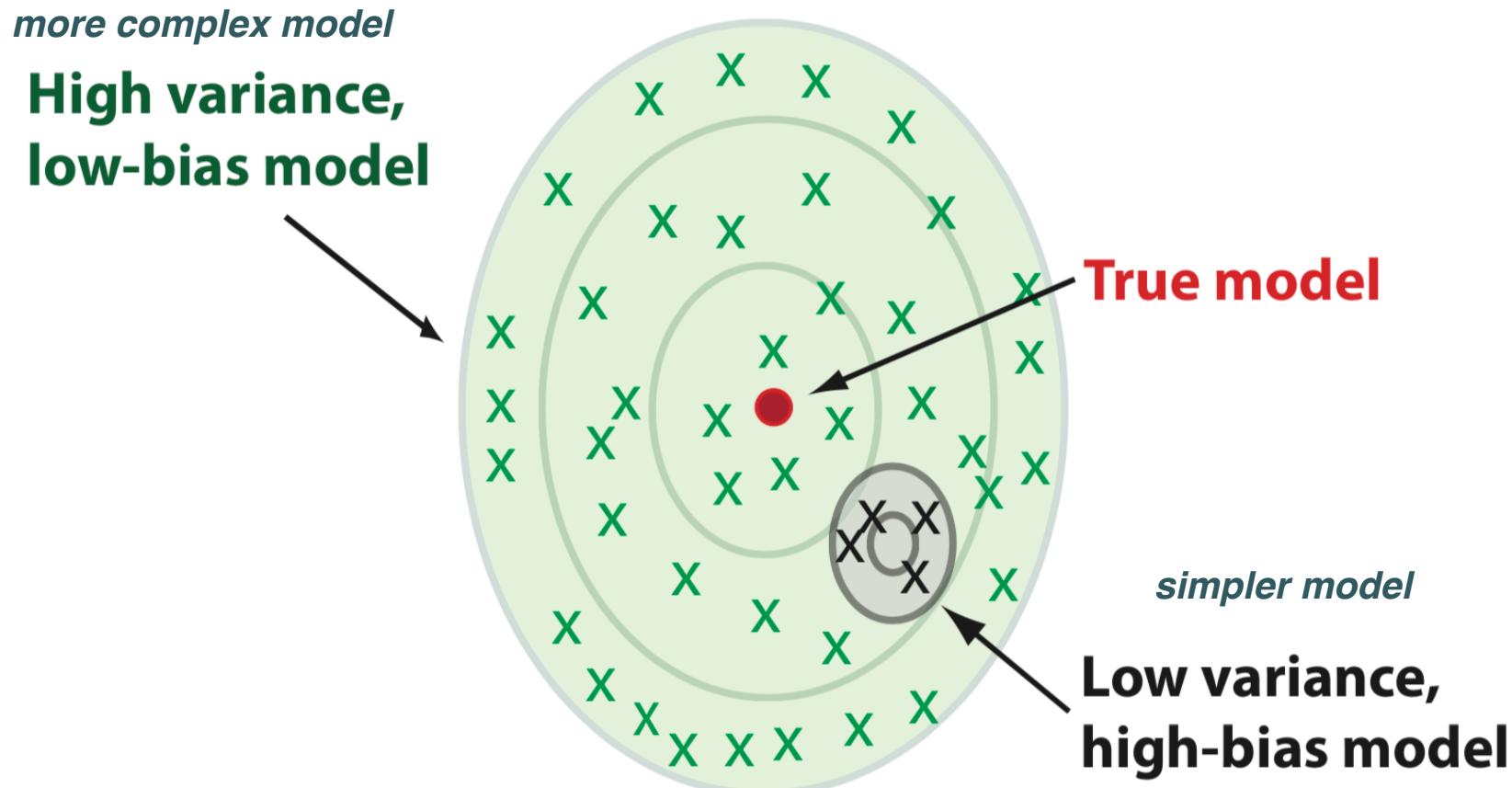
# The bias-variance tradeoff

Optimal model performance (measured by minimising the generalisation error  $E_{\text{val}}$ ) is typically achieved at intermediate levels of model complexity: the **bias-variance tradeoff**



# The bias-variance tradeoff

Optimal model performance (measured by minimising the generalisation error  $E_{\text{val}}$ ) is typically achieved at intermediate levels of model complexity: the **bias-variance tradeoff**



# The bias-variance tradeoff

we can provide further insight on the bias-variance tradeoff with a simple scenario

assume a **model** extracted from a **training set**  $\longrightarrow \hat{y}(\mathbf{x}_{\text{tr}})$

and that the true underlying law is given by

$$Y = y(\mathbf{X}) + \epsilon, \quad y(\mathbf{x}_{\text{tr}}) = E(Y, \mathbf{X} = \mathbf{x}_{\text{tr}})$$

  
*zero-mean, fixed variance noise*

take a data point **outside the training set**,  $(\mathbf{x}_0, y_0)$ , and compute **its variance** in our model

$$E[(Y(x_0) - \hat{y}(x_0))^2] = (y_0^2 - 2y_0 E[\hat{y}(x_0)] + E[\hat{y}^2(x_0)] + E[\epsilon^2])$$

  
*underlying law*      *model*

where we set to zero terms linear in the stochastic noise

# The bias-variance tradeoff

we can provide further insight on the bias-variance tradeoff with a simple scenario

assume a **model** extracted from a **training set**  $\longrightarrow \hat{y}(\mathbf{x}_{\text{tr}})$

and that the true underlying law is given by

$$Y = y(\mathbf{X}) + \epsilon, \quad y(\mathbf{x}_{\text{tr}}) = E(Y, \mathbf{X} = \mathbf{x}_{\text{tr}})$$

  
*zero-mean, fixed variance noise*

take a data point **outside the training set**,  $(\mathbf{x}_0, y_0)$ , and compute **its variance** in our model

$$E[(Y(x_0) - \hat{y}(x_0))^2] = (\text{Bias}[\hat{y}(x_0)])^2 + \text{Var}[\hat{y}(x_0)] + \text{Var}(\epsilon)$$

$$\text{Bias}[\hat{y}(x_0)] = E[\hat{y}(x_0)] - y_0$$

$$\text{Var}[\hat{y}(x_0)] = E[\hat{y}^2(x_0)] - E[\hat{y}(x_0)]^2$$

the **model generalisation power** decreases both with its bias and its variance