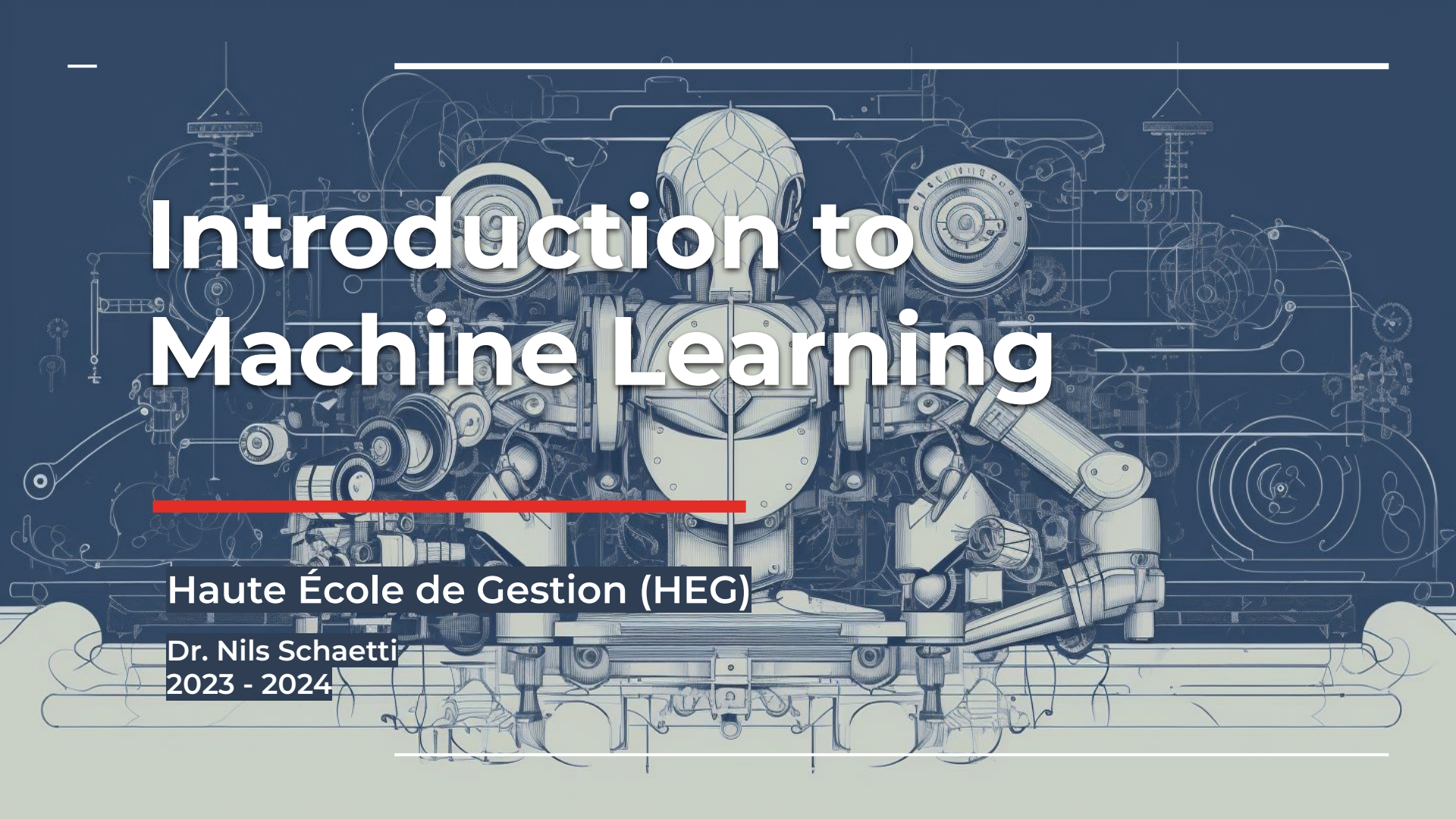


Introduction to Machine Learning

Haute École de Gestion (HEG)

Dr. Nils Schaetti

2023 - 2024

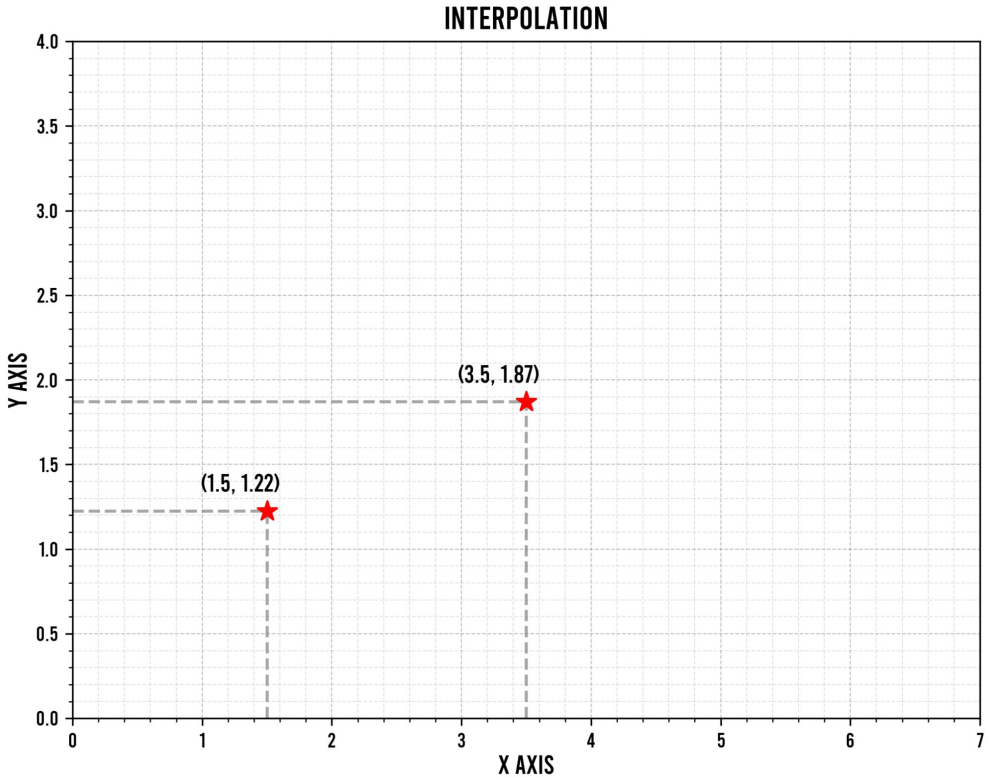
Learning Theory

“ Improving the performance of a program in solving tasks without being explicitly programmed to solve these tasks “

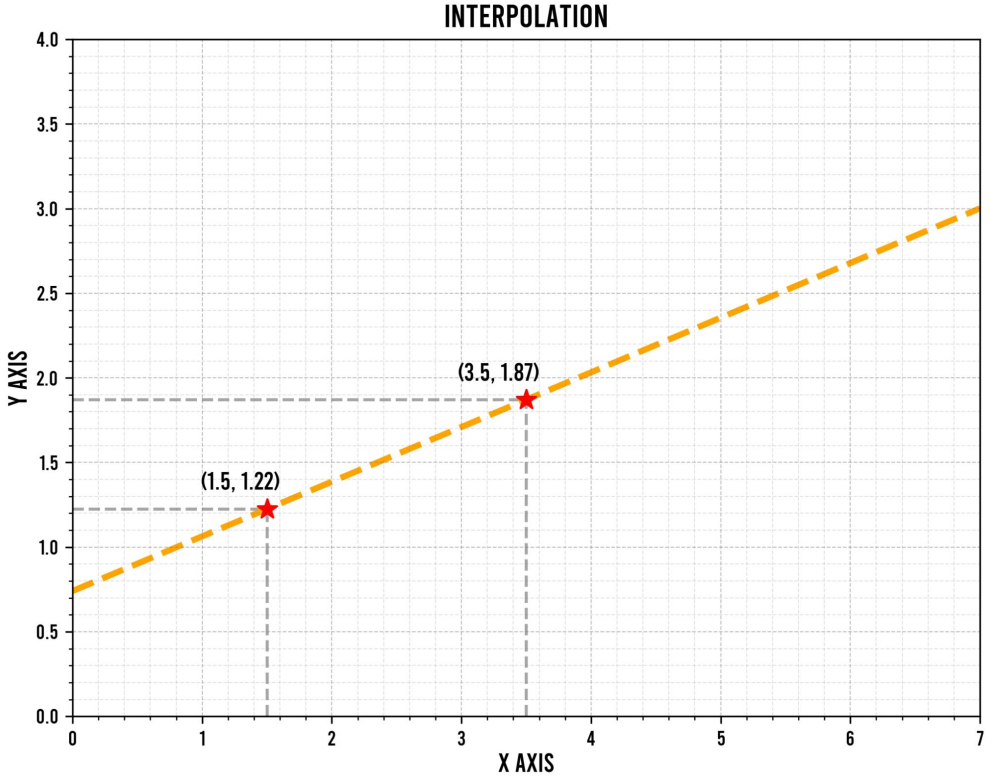
What is learning ?

- Find a function **X → Y**
- We only have some observations:
  - **(x_1, y_1), (x_2, y_2), (x_3, y_3)**
  - Or learning examples
- We want to predict y for a given x (**y = f(x)**)
- **Learning theory**: what allows a machine to learn?

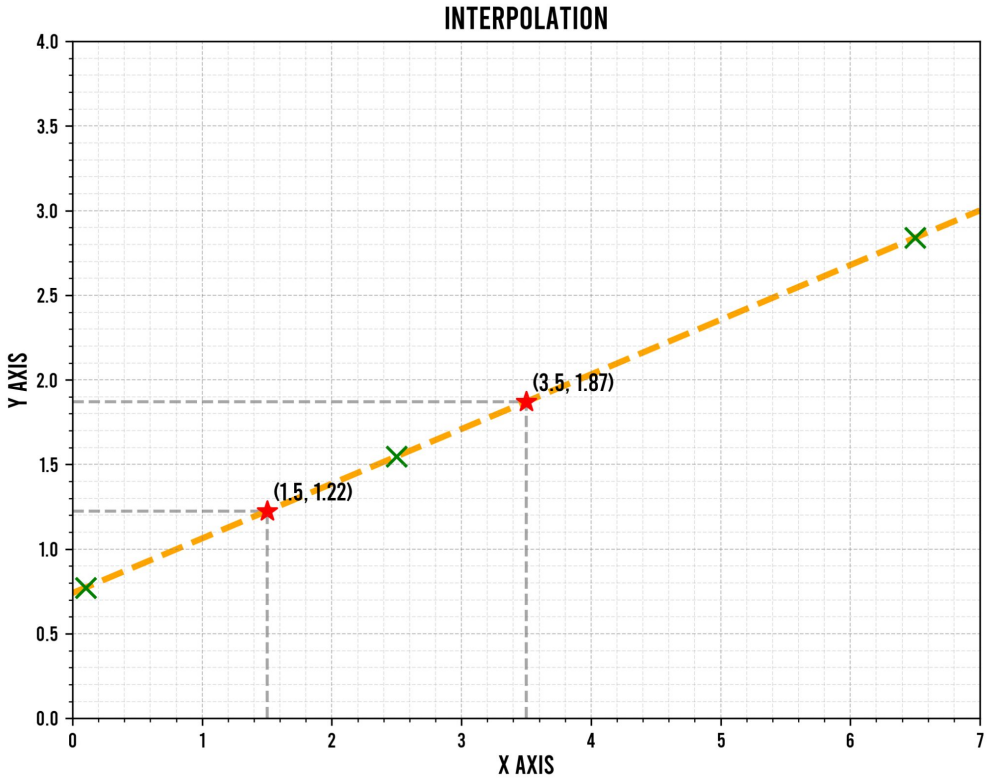
Find a function



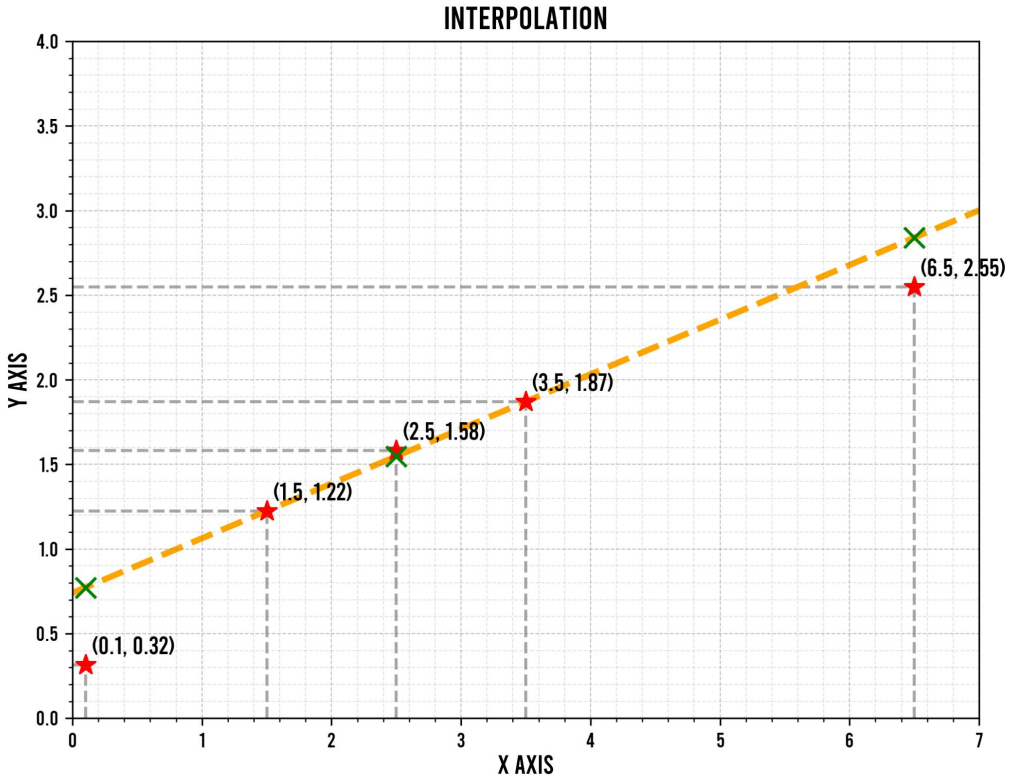
Find a function



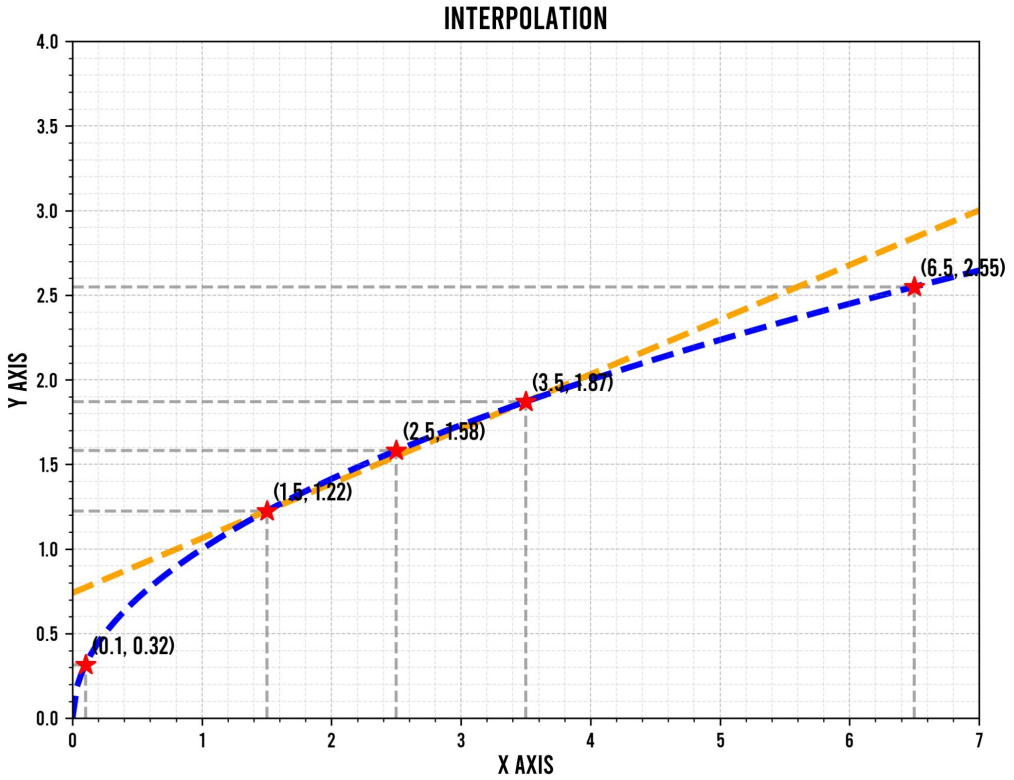
Find a function



Find a function



Find a function

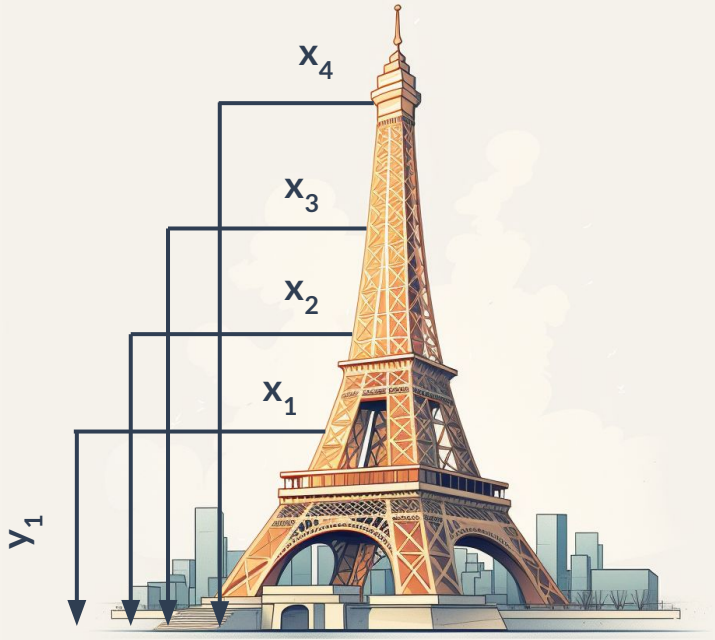


Inductive reasoning

Find the underlying principle (model, function) that fits the observations (data).

- **Search space** : the set of all possible curves. Can also be seen as the set of all possible functions (**X → Y**).
- **Example** : the enormous, yet finite, set of all possible classification rules (boundaries).
- **Training** : find the best possible combination in the search space.
  - Test (enumerate) the whole search space.
  - Eliminate points in the search space which do not fit data.
  - Remaining points contain the underlying principle.

Inductive reasoning



- We have a set of observations under the form (x, y).
- x is the height, y the time to reach the ground.
- **Goal**: find an function h that is a good approximation of underlying law.
- **Good approximation**: f(x) ≈ h(x)
- **Basic form of learning**:
  - We know nothing about f (prior knowledge.
  - How do we know how good h is?

Inductive and deductive reasoning

Model
(general)

Observations
(particular)

Induction

Going from observations to models
(general rule)

Model

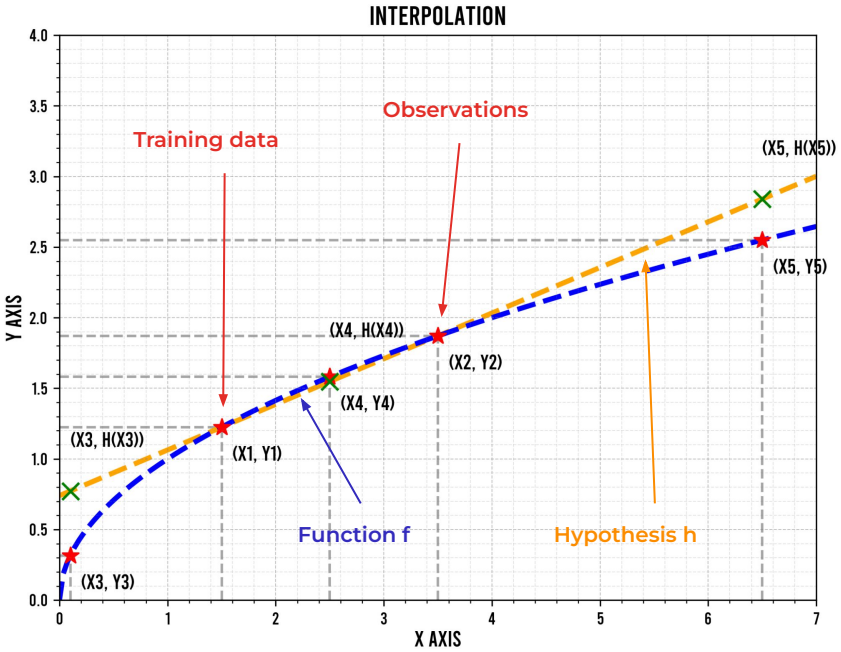
Observations

Inference

Going from models to predictions

Curve fitting

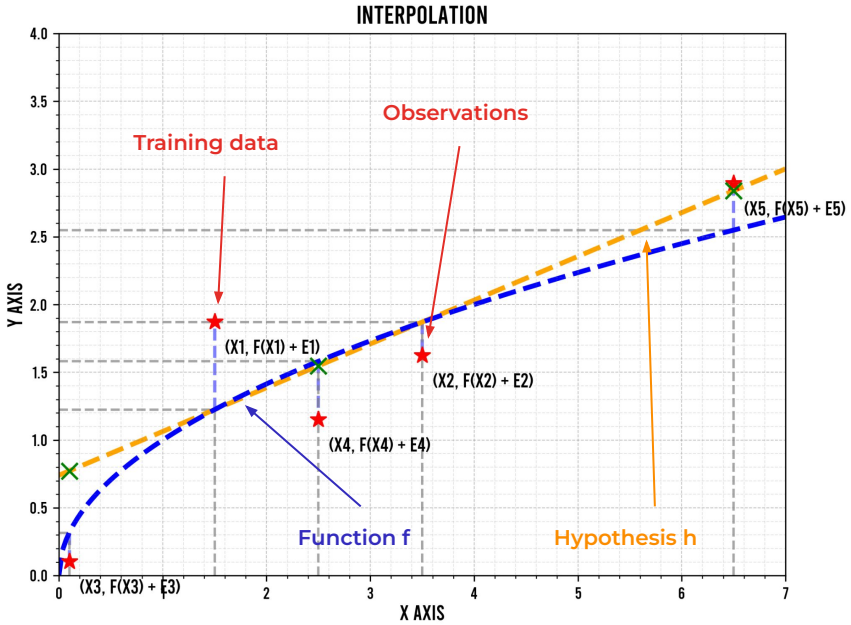
A little bit of formalism



Learn a function from training examples (x, f(x))

- Underlying function f(x) : we know nothing about it.
- Hypothesis function h(x) : our hypothesis, as good as possible.
- Good approximation: f(x) ≈ h(x)

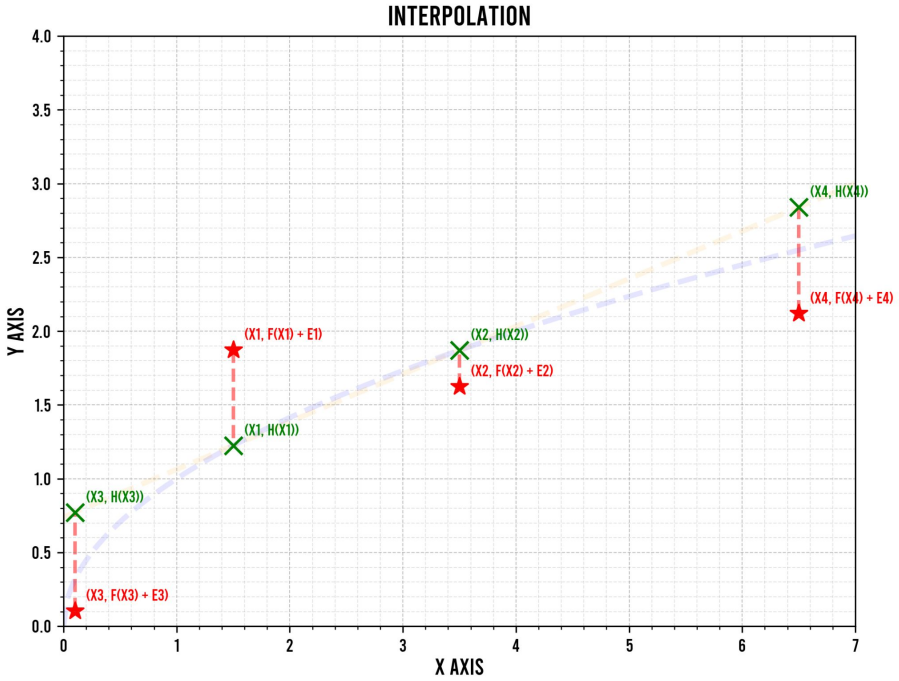
A little bit of formalism



Learn a function from training examples (x, f(x))

- Underlying function f(x) : we know nothing about it.
- Hypothesis function h(x) : our hypothesis, as good as possible.
- Good approximation: f(x) ≈ h(x)
- **But**: we have only noisy observations !

What’s a good hypothesis function ?

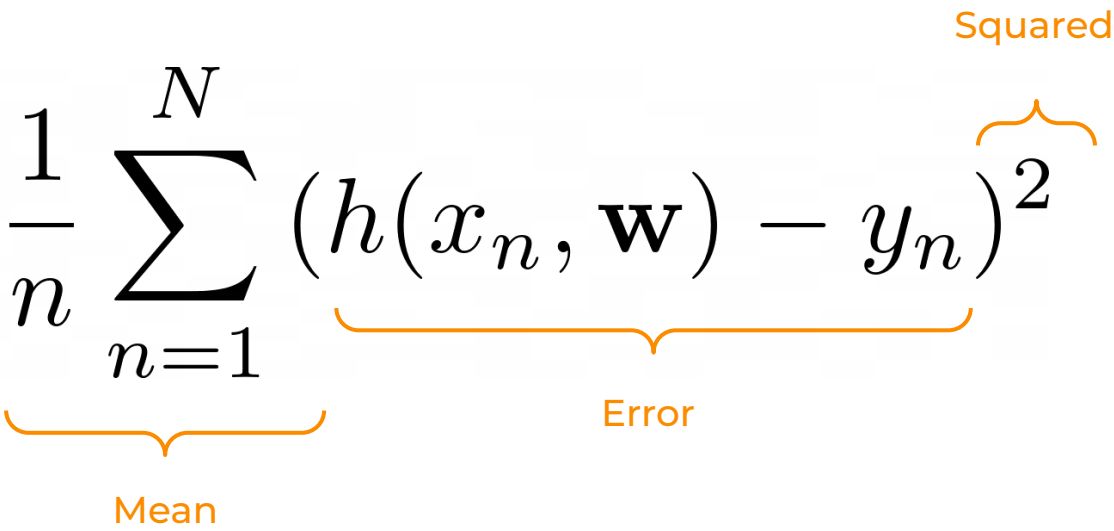


Training error: we define an error function measuring how h(x, w) fits the training set.

Number of training samples

$$\frac{1}{n}\sum_{n=1}^{N}\left(h(x_n, \mathbf{w}) - y_n\right)^2$$

Curve fitting: error function



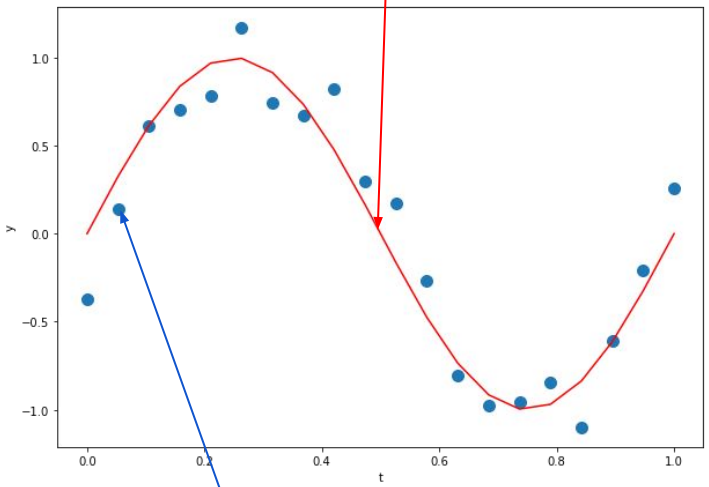
Curve fitting: example

M: degree of polynomial

$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

The real process

$$f = sin(2\pi x)$$



$$x = sin(2\pi x) + \epsilon$$

Observation = regularity + noise

- The components of the **vector w** are the parameters of the model.
- We want to find the vector w that results in **hypothesis h** to match the **real process f**.
- **Training**: adjust **w** to agree with **f** on the training set.
- How do we measure the quality of the model ?

Learning a curve with polynomials

M: degree of polynomial

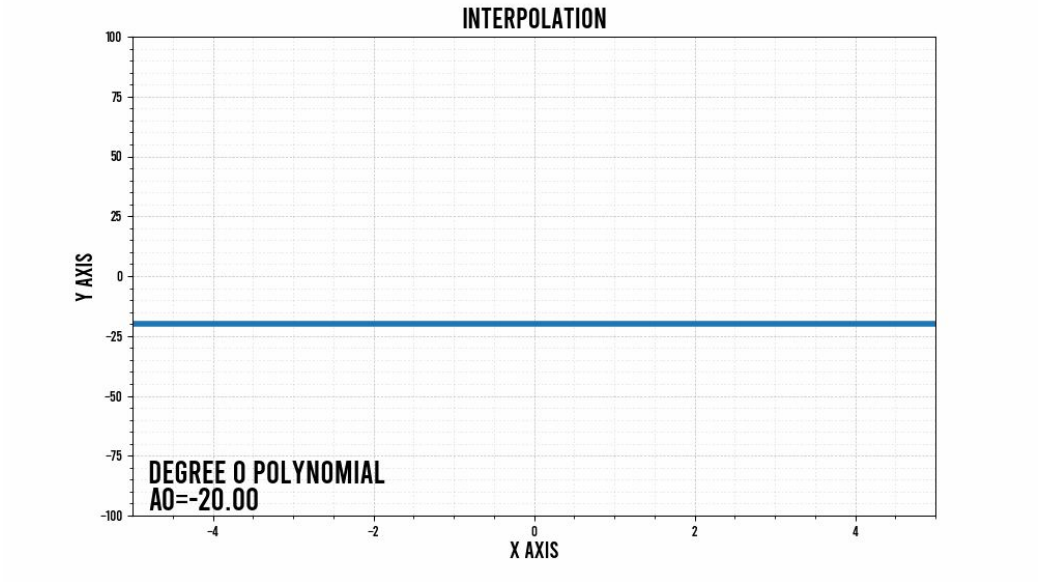
$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- The components of the **vector w(x)** are the parameters of the model.
- We want to find the vector w that results in **hypothesis h(x)** to match the **real process f(x)**.
- **Training**: adjust w to agree with f on the training set.
- How do we measure the quality of the model ?

Examples of polynomials

A polynomial of degree 0 :

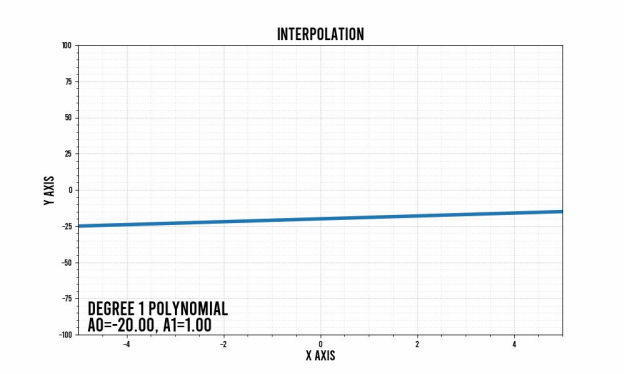
$$h(x) = w_0$$

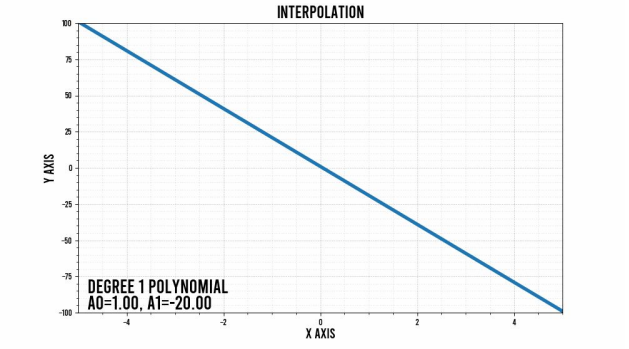


Examples of polynomials

A polynomial of degree 1 :

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

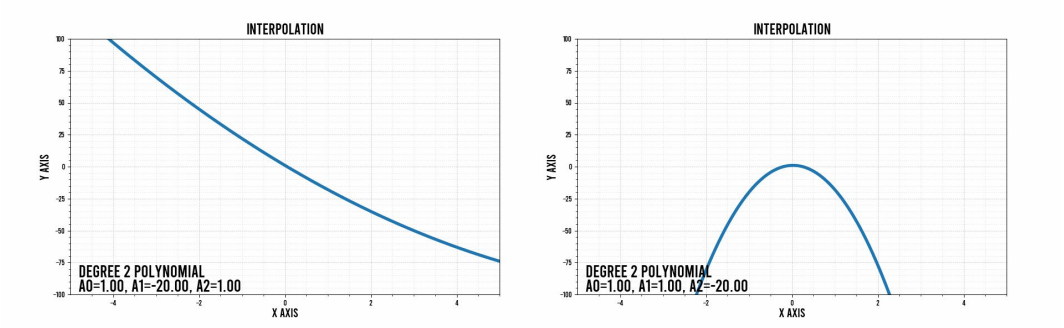


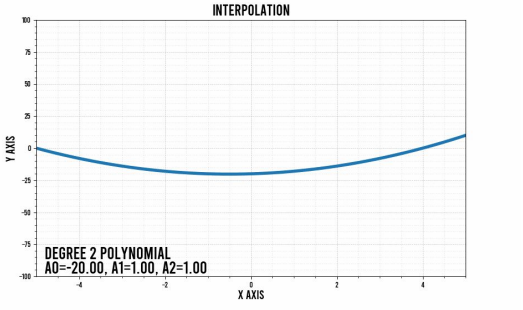


Examples of polynomials

A polynomial of degree 2 :

$$h(x) = w_0 + w_1 x + w_2 x^2$$

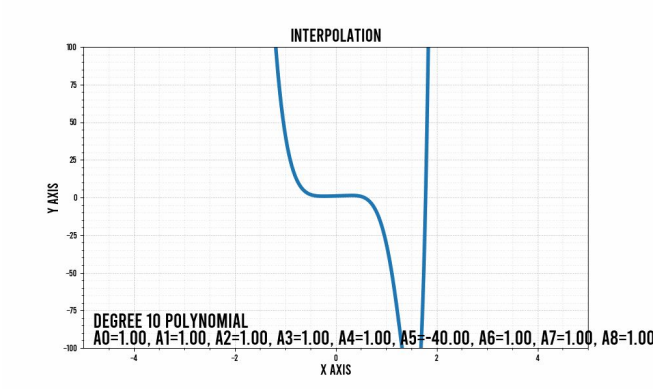


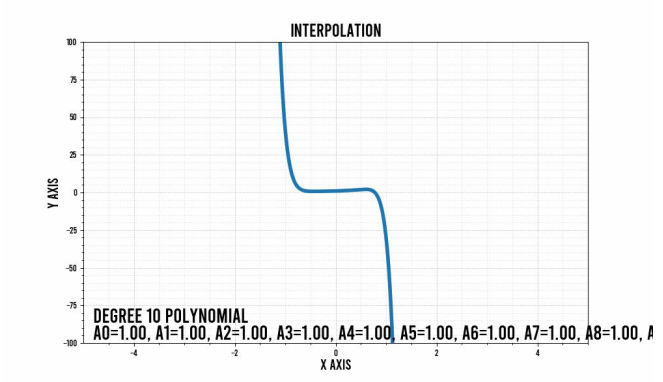


Examples of polynomials

A polynomial of degree 10 :

$$h(x) = w_0 + w_1 x + w_2 x^2 \ldots + w_{10} x^{10}$$

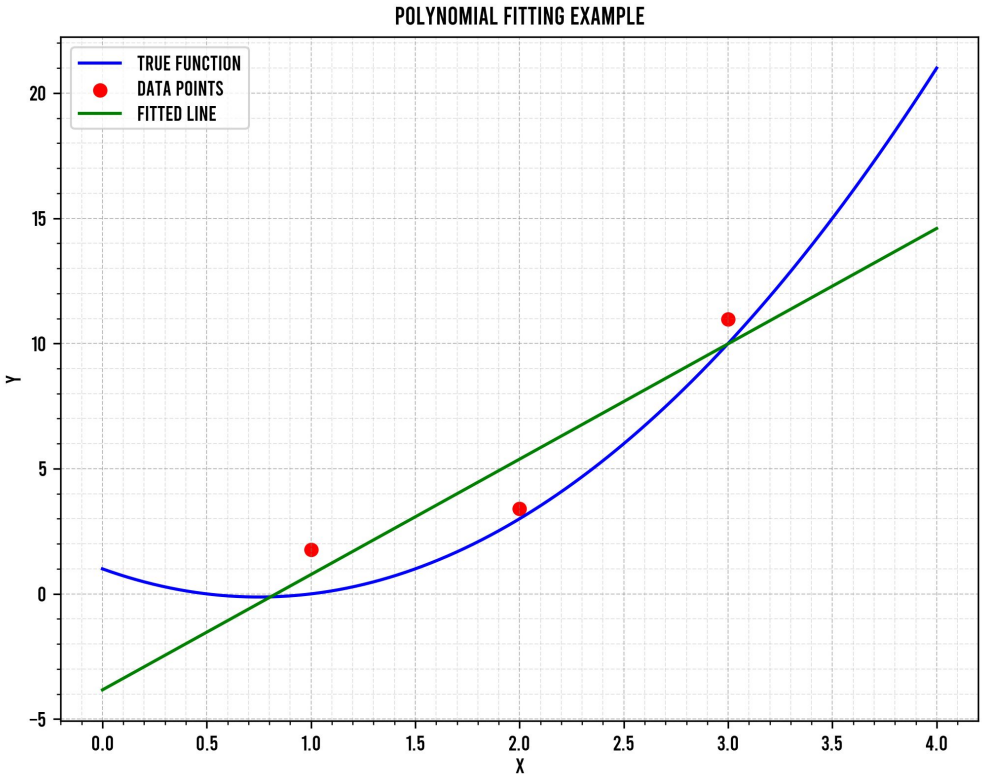




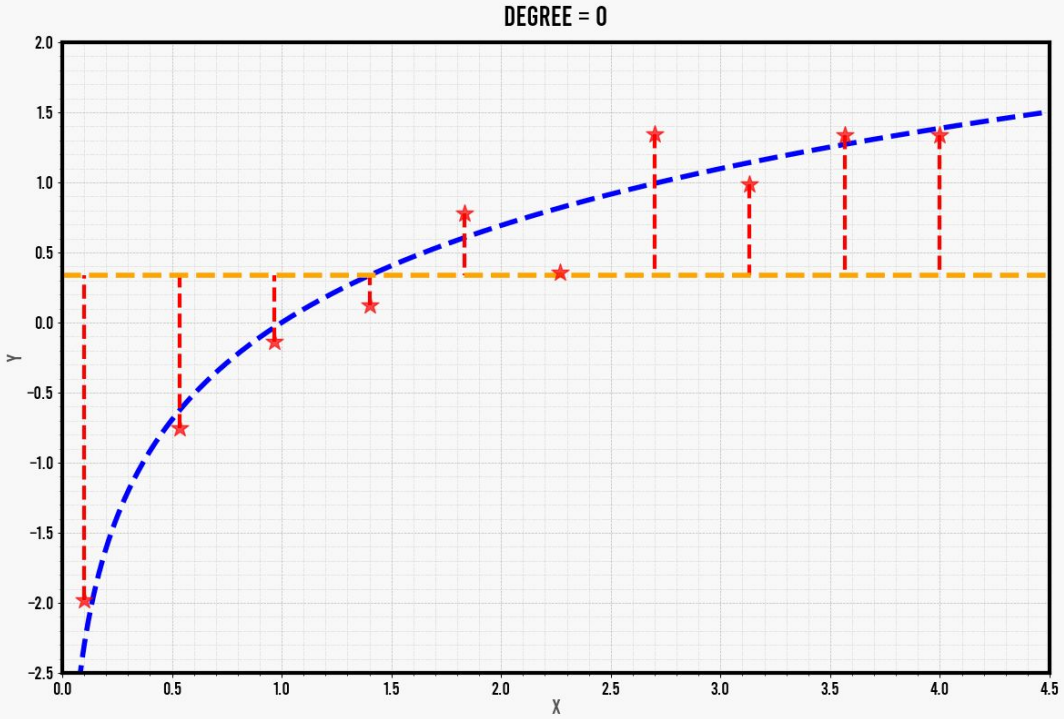
Learning curve with polynomials

```
import numpy as np
import matplotlib.pyplot as plt

# Perform polynomial fitting
degree = 1  # Linear fit
coefficients = np.polyfit(x_points, y_points, degree)

# Create a polynomial function from the coefficients
poly = np.poly1d(coefficients)

y_pred = poly(x_values)
```

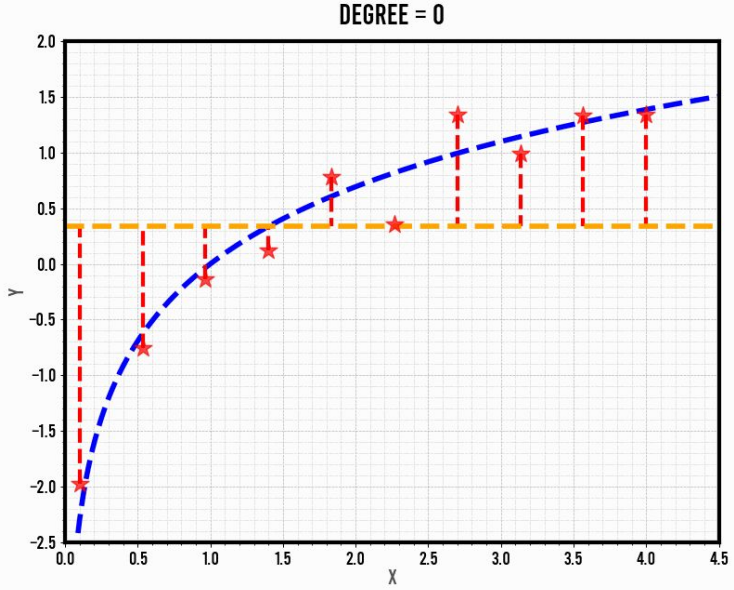
Learning a curve with polynomials

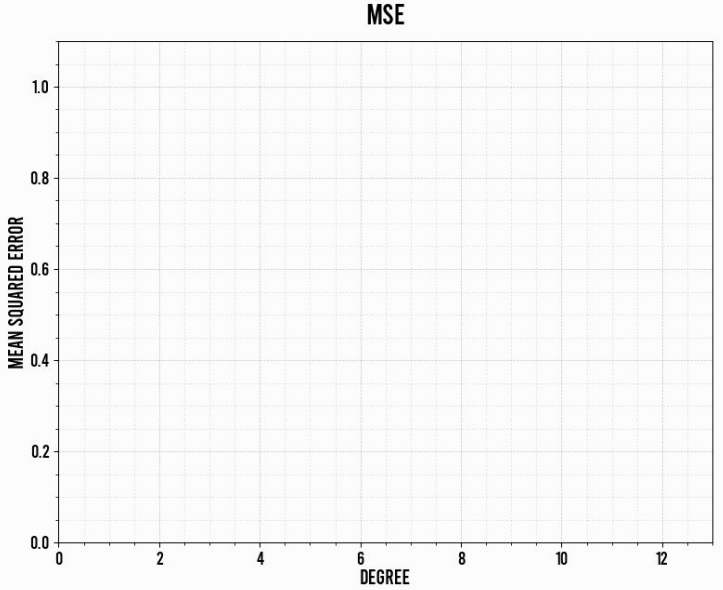


Learning a curve with polynomials

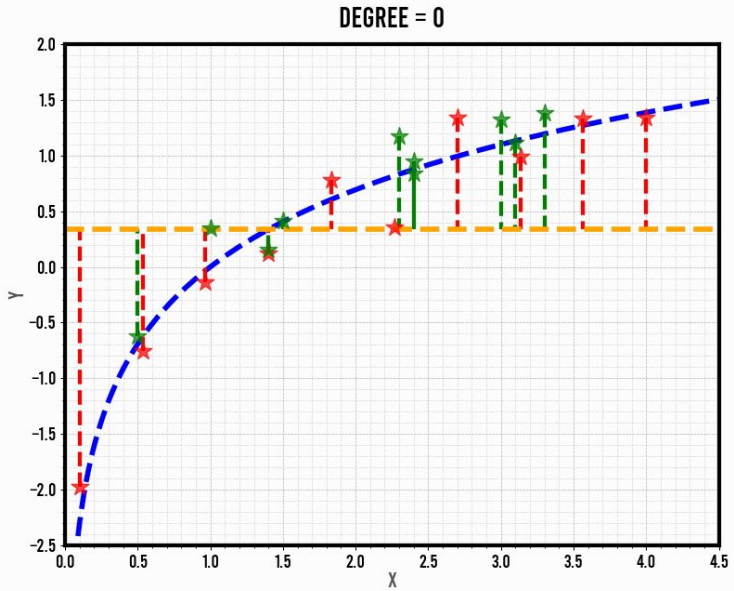


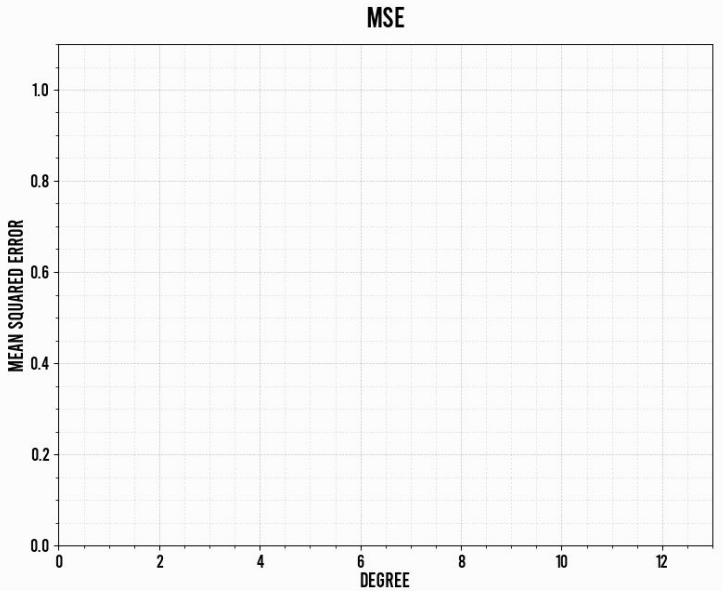
Learning a curve with polynomials





Learning a curve with polynomials





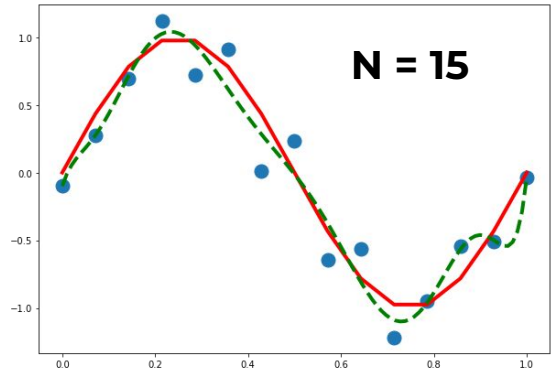
Learning a curve with polynomials

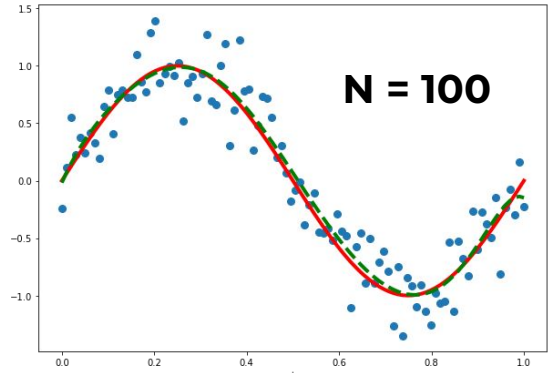
Empirical risk (R_{emp}) / training error : error on the training samples (according to loss L).

Generalization error / test error : error on test samples, not used for training !

The complexity of the model depends on the size of the training set !

Curve fitting: overfitting and training set



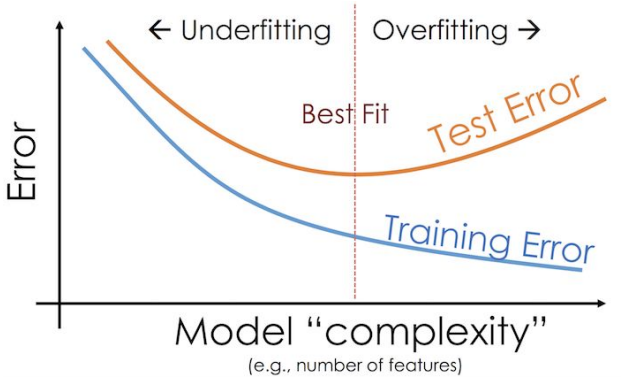


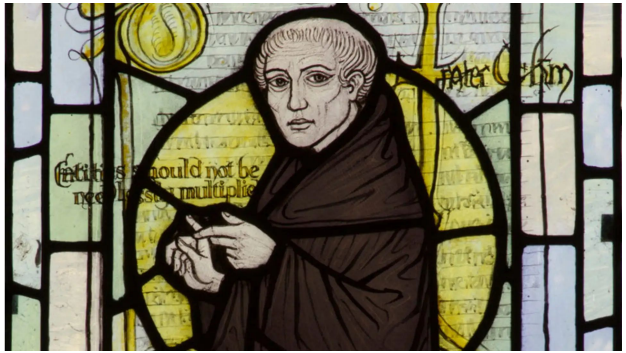
- We can reduce overfitting by collecting more data (increasing the training set).
- **Example** : Hypothetical function h of order 9.

Rule of thumb

5M ≤ N or 10M ≤ N

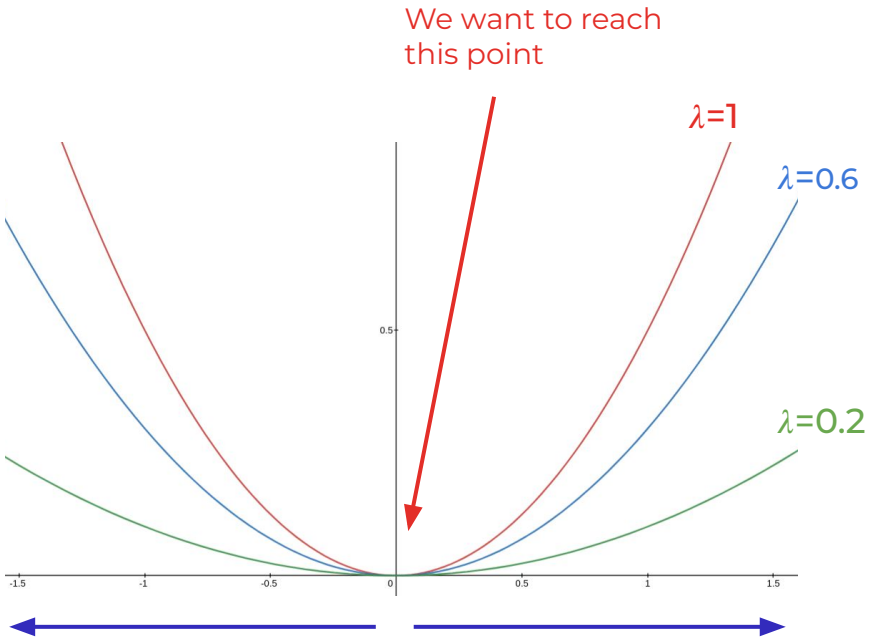
Curve fitting: Ockham’s razor





- **Ockham’s razor** : “entities should not be multiplied beyond necessity”.
- **Swinburne** : “the simplest hypothesis proposed as an explanation of phenomena is more likely to be the true one than is any other available hypothesis”.
- **Machine Learning** : prefer the simplest hypothesis (model) consistent with (training) data.

Curve fitting: Ockham's razor applied



Applying Ockham's razor: we modify how we explore the search space by penalizing complex models :

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(h(x_n, \mathbf{w}) - y_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

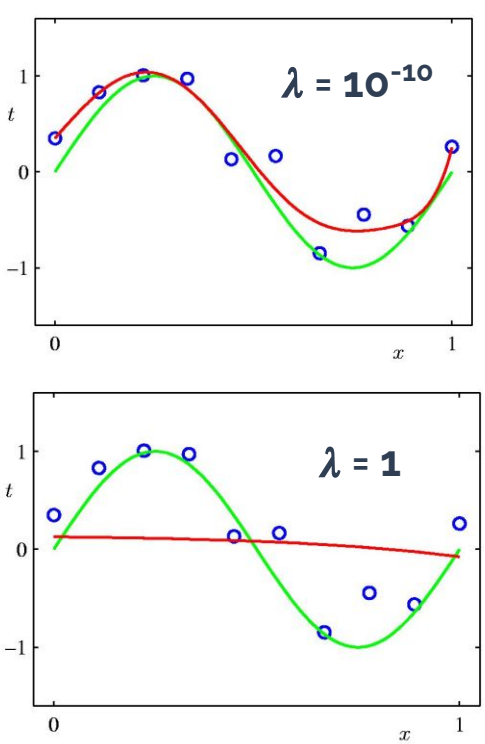
Regularisation

with

$$||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$

Weights with high value are penalized in the selection process.

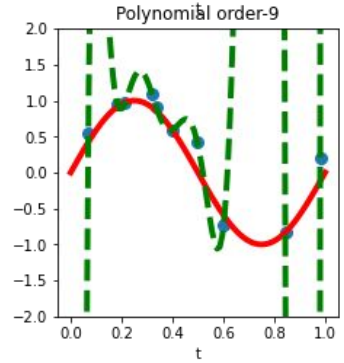
Curve fitting: Ockham's razor applied



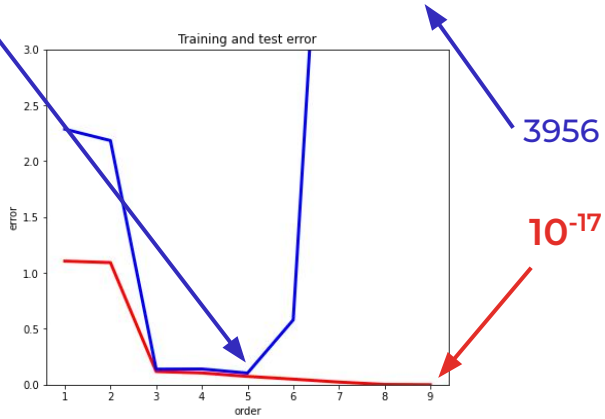
- We give an advantage to simpler model by penalizing large values for **weights w** and we obtain a better curve estimate.

- We penalize complex model too much and only very simple models are selected (order 1 or 2) as only very small values are possible for **weights w**.

Curve fitting: Ockham's razor applied



The best model with 0.104



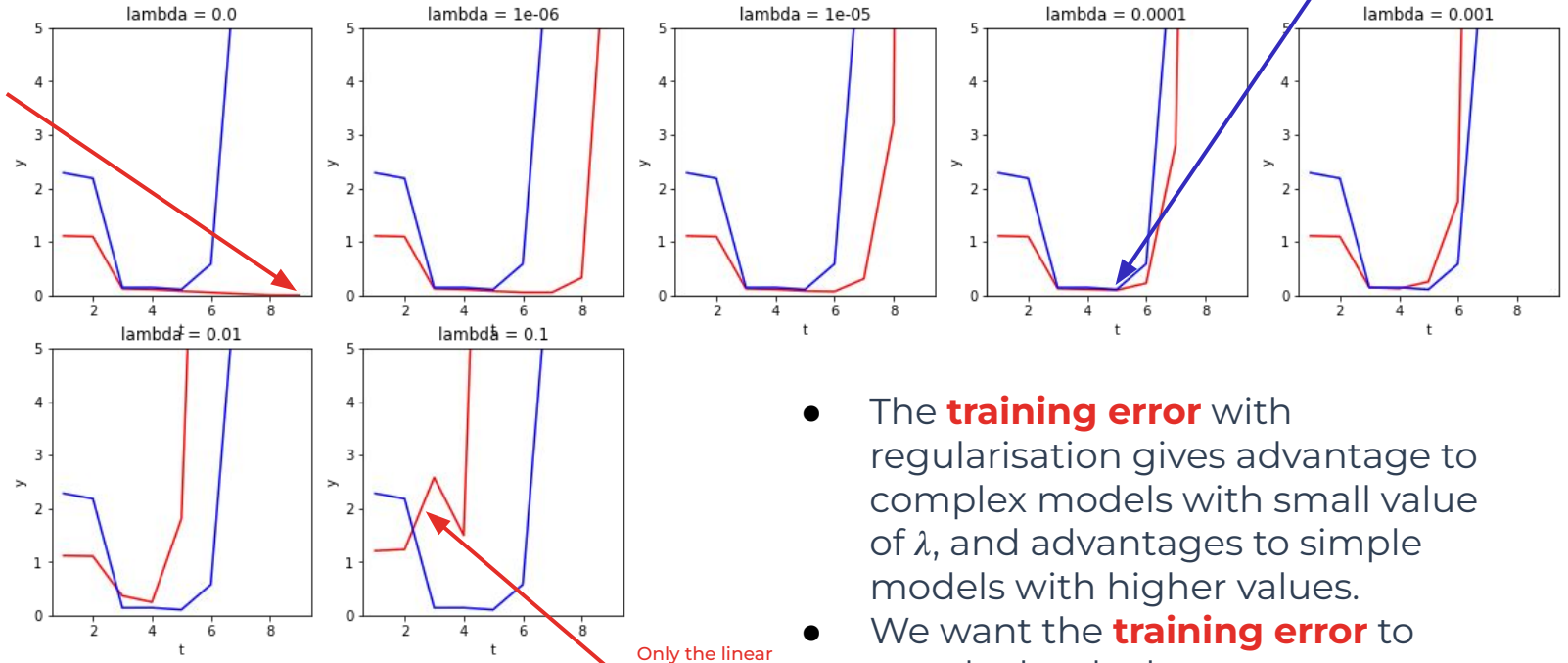
- With a complex model (polynomial of order 9), we fit the training data perfectly. The **training error** is then very low (10^{-17})

- But if we test this model on unseen data (test set), we obtain a very high **test error** of 3956. The best model is in fact a polynomial of order 5 with 0.104 !

Curve fitting: error with regularisation

The best model on the test set has also the lowest training error.

Complex model matches the data perfectly



Only the linear model will be selected !

- The **training error** with regularisation gives advantage to complex models with small value of λ, and advantages to simple models with higher values.
- We want the **training error** to match closely the **test error**.

Curve fitting: conclusion

- **Hyper-parameters vs parameter** : the regularisation parameter λ is an hyper parameter, a parameter of the mode. While w_0, etc are parameters, they are learned during training.
- **Which hypothesis function h is the best** ? We are looking for a tradeoff between complexity of h and the degree of fit to the data.
- **Regularisation modify the search space** : by adding a weight on models complexity, we can embed this property in the optimisation process.
    - Stronger regularisation advantages simple models (such as linear regression).
- **Overfitting** : a big difference between training error and test error is a sign of overfitting.

Curve fitting: notebook

https://colab.research.google.com/drive/1Qw5YMr5rDHXi4O_su7nB4-7bCQA-V0_L?usp=sharing