Machine learning and (geo)physical modelling-1

julien.brajard@nersc.no

October 2025

NERSC

`https://github.com/brajard/MAT330`

Overview of the next lectures

1. Lecture 1 (Tuesday 21 Oct. 2023):
   Generalities and Principles of Machine Learning
2. Lecture 2 (Wednesday 23 Oct. 2023):
   Neural networks, deep learning, training
3. Lecture 3 (Tuesday 30 Oct. 2023):
   Practical work

julien.brajard@nersc.no

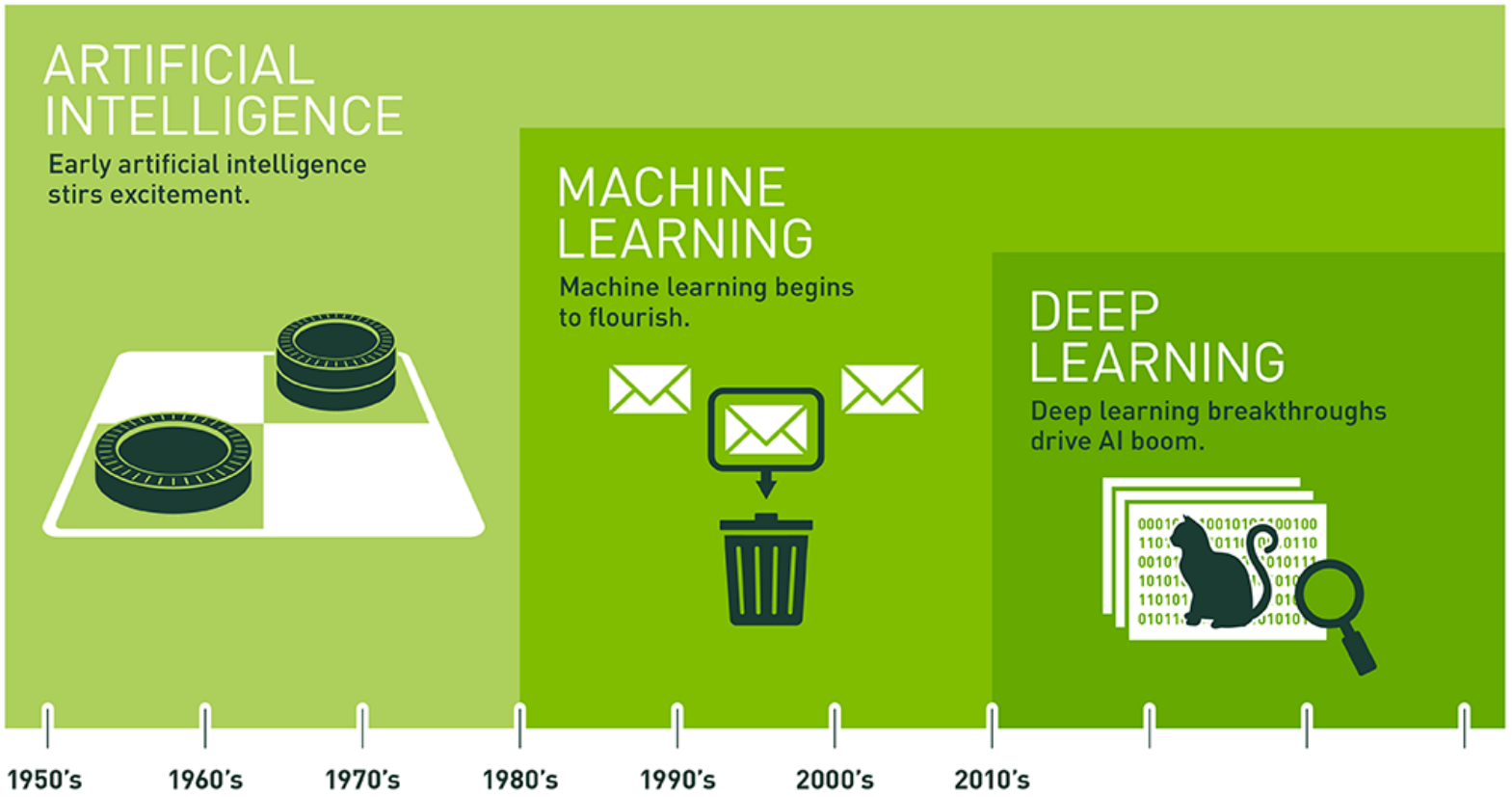
References

- Francois Chollet.
  Deep learning with Python.
  Simon and Schuster, 2021.
  https://d2l.ai/index.html.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
  Deep Learning.
  MIT Press, 2016.
  http://www.deeplearningbook.org.
- Jake VanderPlas.
  Python Data Science Handbook: Essential Tools for Working with Data.
  O'Reilly Media, Inc., 1st edition, 2016.
  https://jakevdp.github.io/PythonDataScienceHandbook/.
- Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola.
  Dive into deep learning.
  Cambridge University Press, 2023.
  https://d2l.ai/index.html.

Table of contents

Introduction

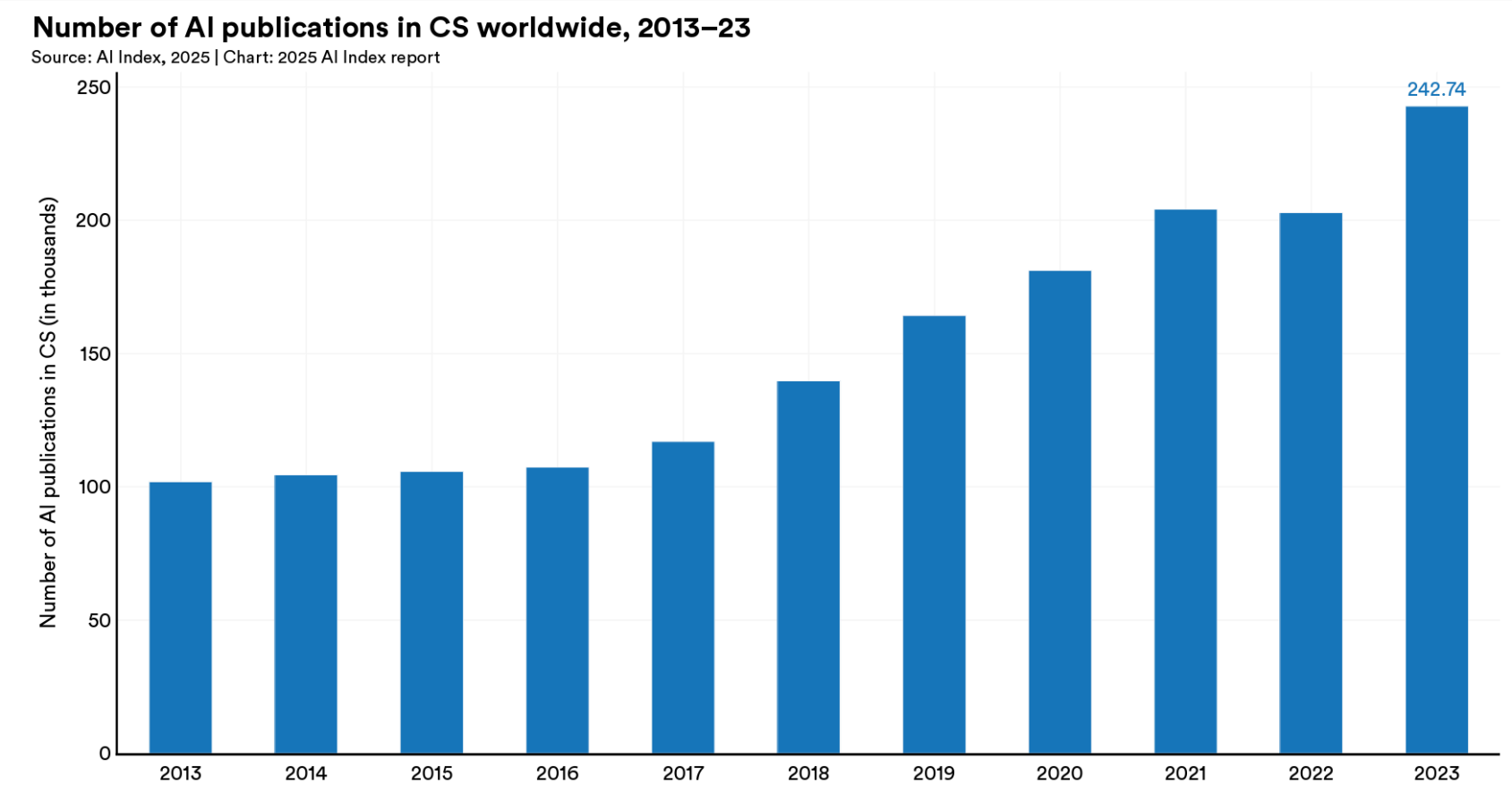
Scope of the lecture: Machine Learning



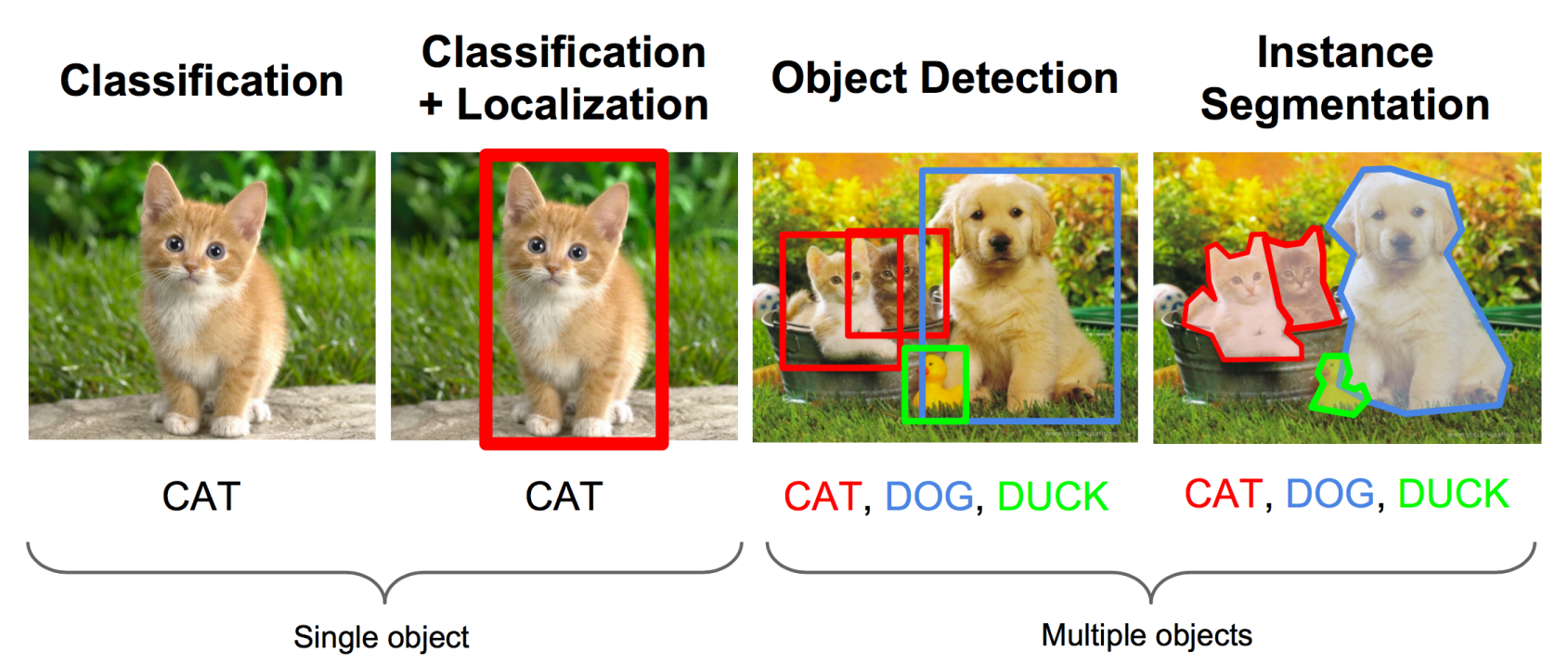
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: NVidia

A (very) active field

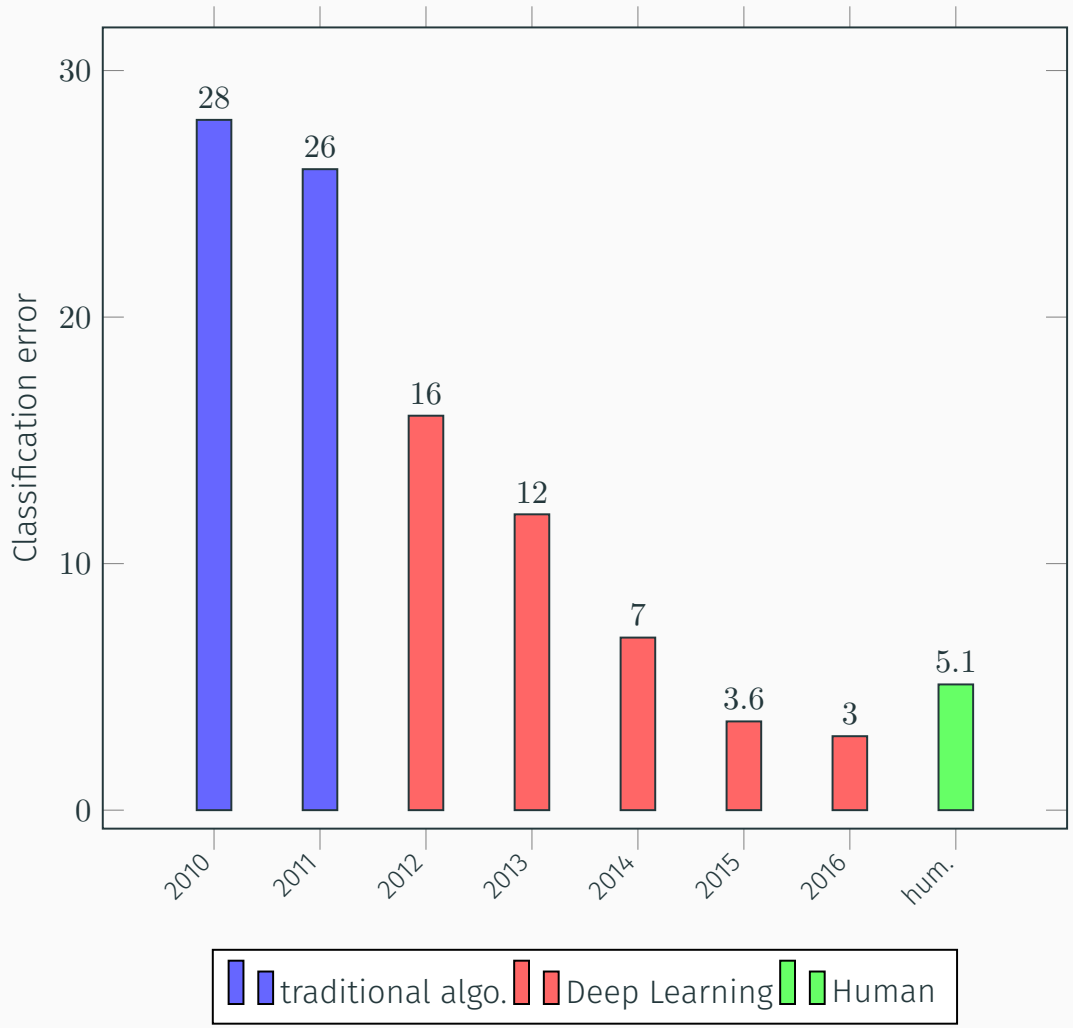


Example 1: Computer Vision



Li, Karpathy and Johnson, 2016, Stanford CS231n course

Example 1: Computer Vision



Deep learning architectures were based on Convolutional Neural Networks (CNN).

Example 2: Playing Games

- **1997**: Deep Blue defeats Kasparov at Chess.
- **2016**: AlphaGo's victory again Lee Sedol at Go.

  - **2017**: AphaGo Zero learns how to play Go only by playing against itself. It outperformed previous AlphaGo version (Reinforcement learning)
  - **2017**: DeepStack beats professional human poker players.

Example 3: Text to image

Objective: Generate an image from a textual description.

- **2021**: DALL·E
- **2022**: Midjourney and Stable diffusion
- Since **2022**: all over the place

Example 3: Text to image

Objective: Generate an image from a textual description.

- **2021**: DALL·E
- **2022**: Midjourney and Stable diffusion
- Since **2022**: all over the place

photography style with a 22mm lens of a handsome redheaded biker man with a short beard.

Text to image - Progress in free tools

A test on:

`https://deepai.org/machine-learning-model/text2img`

Prompt: *“students attending a machine learning course”*

Text to image - Progress in free tools

A test on:

`https://deepai.org/machine-learning-model/text2img`

Prompt: *“students attending a machine learning course”*

Result in Oct. 2022

Text to image - Progress in free tools

A test on:

`https://deepai.org/machine-learning-model/text2img`

Prompt: *“students attending a machine learning course”*

Result in Oct. 2022

Result in Oct. 2023

Text to image - Progress in free tools

A test on:

`https://deepai.org/machine-learning-model/text2img`

Prompt: *“students attending a machine learning course”*

Result in Oct. 2022

Result in Oct. 2023

Result in Oct. 2025

Example 4: Large language models (LLM)

Examples: GPT-5 (OpenAI), LLaMA 4(Meta AI), Gemini 2.5 (Google DeepMind), DeepSeek3.2 (DeepSeek).

How I tried to delegate my job

I have to give 6 hours of an introduction course to machine learning to master students. The objective is to undersrtand the main concepts of machine learning from a practical point of view. Can you suggest an outline?

Copilot

That sounds like a great opportunity to give students a solid foundation in machine learning! For a 6-hour introductory course aimed at master's students with a practical focus, here's a suggested outline that balances theory, intuition, and hands-on experience:

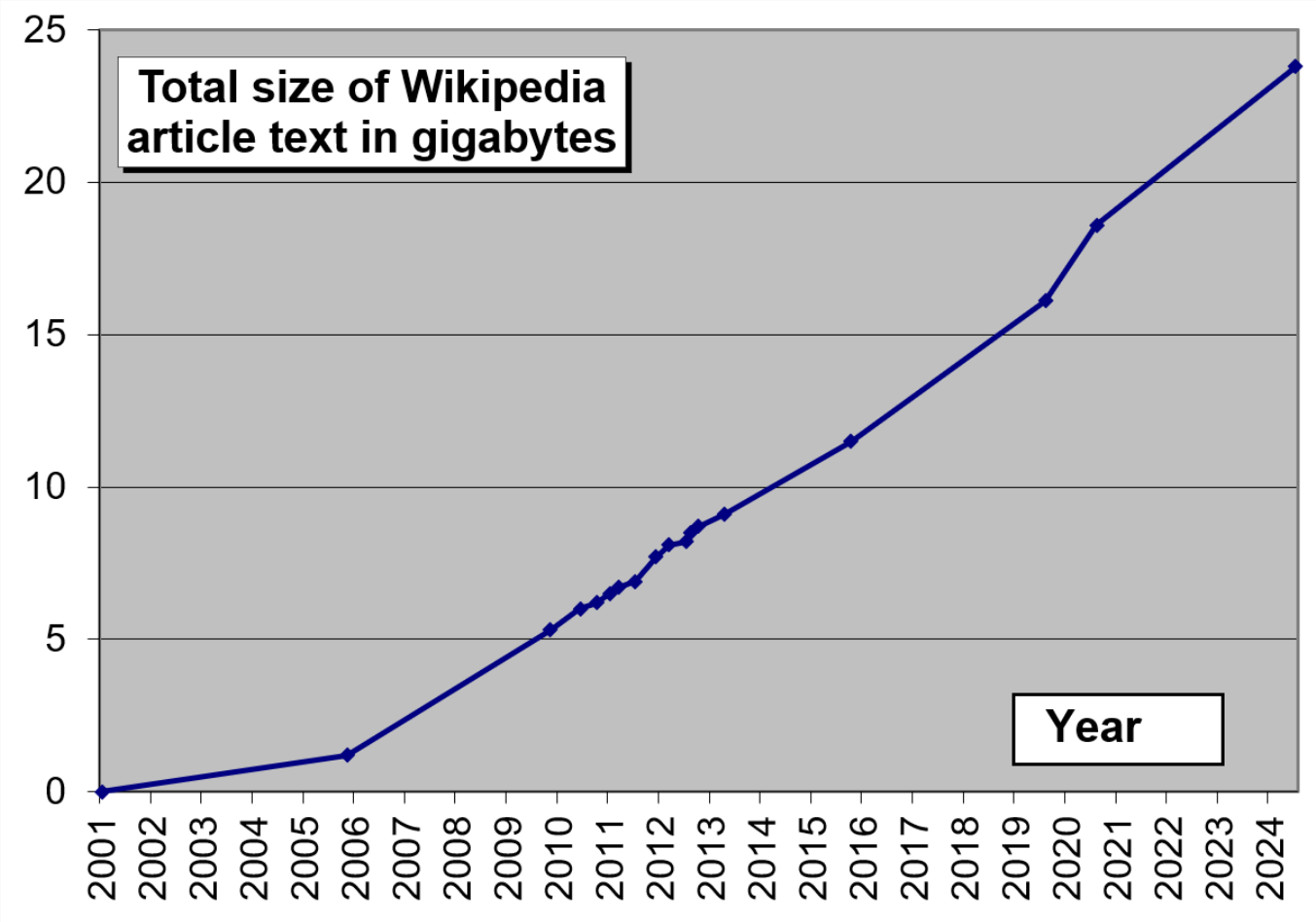
🧠 Machine Learning Introduction Course Outline (6 Hours)

Session 1: Foundations of Machine Learning (1 hour)

- **What is Machine Learning?**
  - Definitions and types: Supervised, Unsupervised, Reinforcement Learning
  - ML vs. traditional programming
- **Key Concepts**
  - Features, labels, models, training, testing
  - Overfitting vs. underfitting
- **Real-world Applications**

Reasons for these recent achievements?

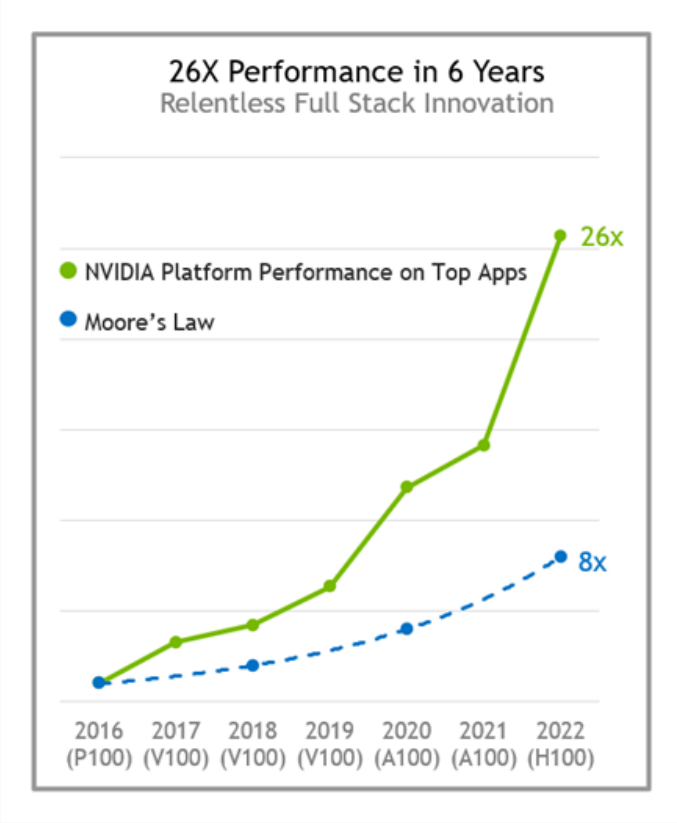
- Increasing of the datasets in size and quality



source: wikimedia.org

Reasons for these recent achievements?

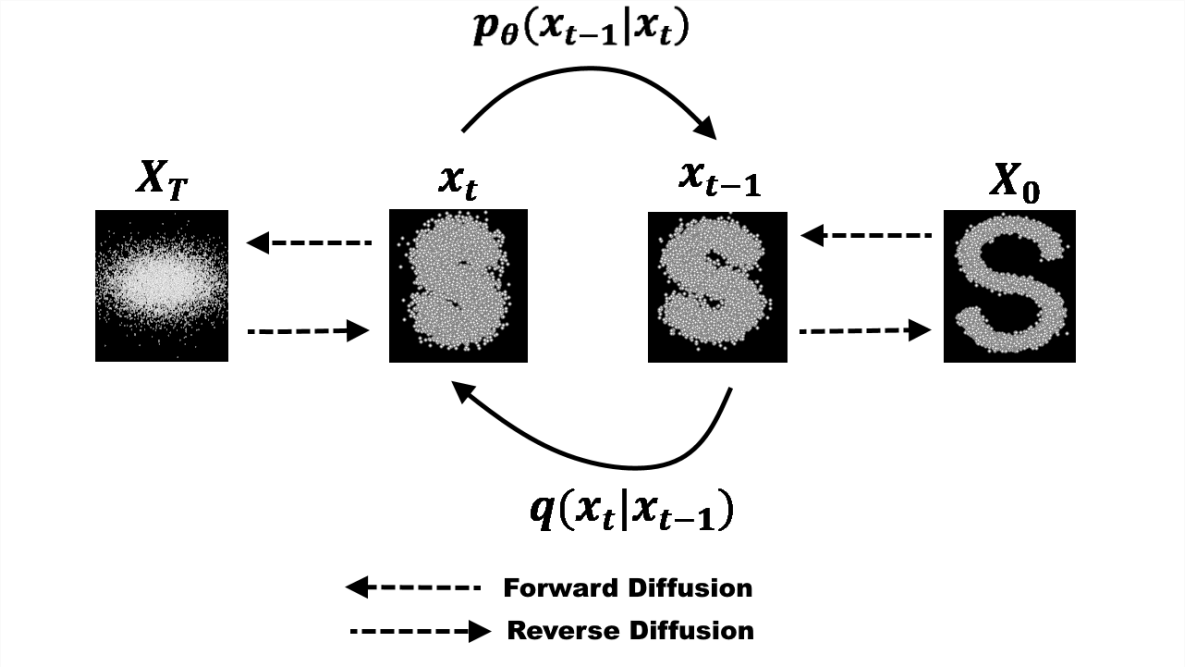
- Increasing of the datasets in size and quality
- Progress in computing resources.



source: NVDIA

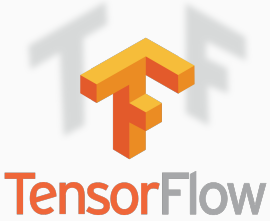
Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)



Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)
- Very efficient software (GPU, cloud computing, automatic differentiation, ...)



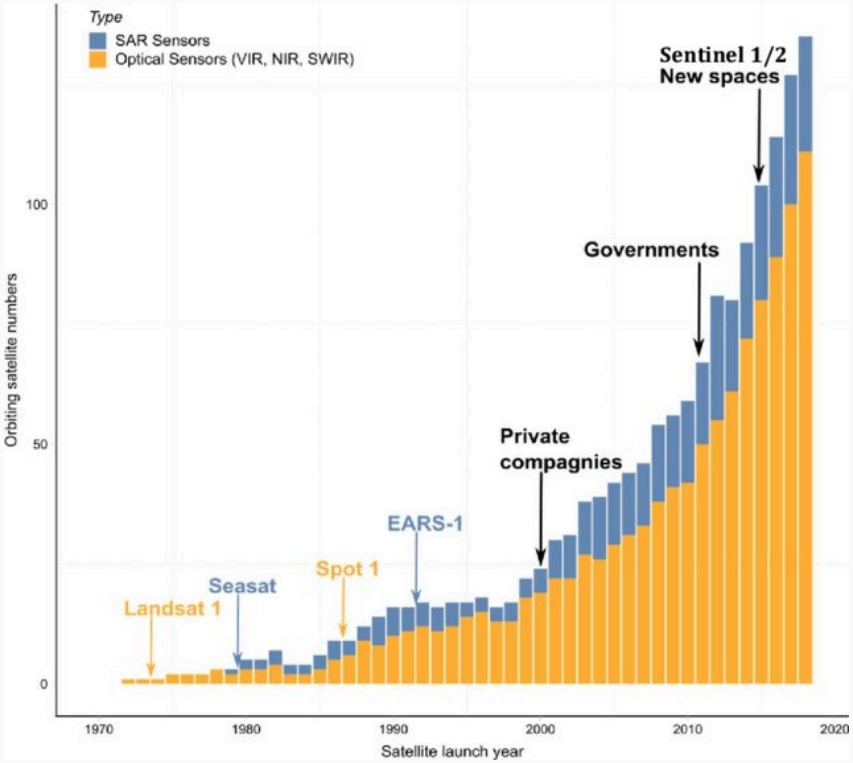
Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)
- Very efficient software (GPU, cloud computing, automatic differentiation, ...)
- Free software and open data culture.

Apply Machine-Learning to physical (Earth-system) modelling?

Why is it a good idea?

- A increasing number of geophysical data (one spatial mission: 24 TB/day)



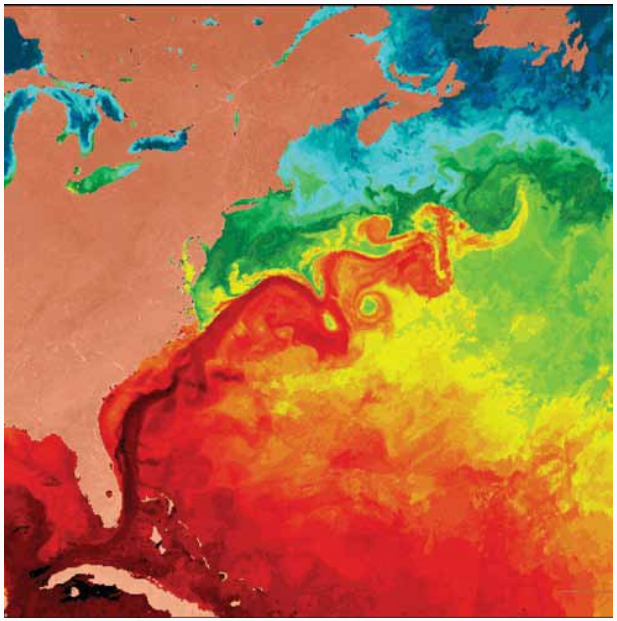
Earth System observavtion satellites

source: Tonneau et al. (2020)

Apply Machine-Learning to physical (Earth-system) modelling?

Why is it a good idea?

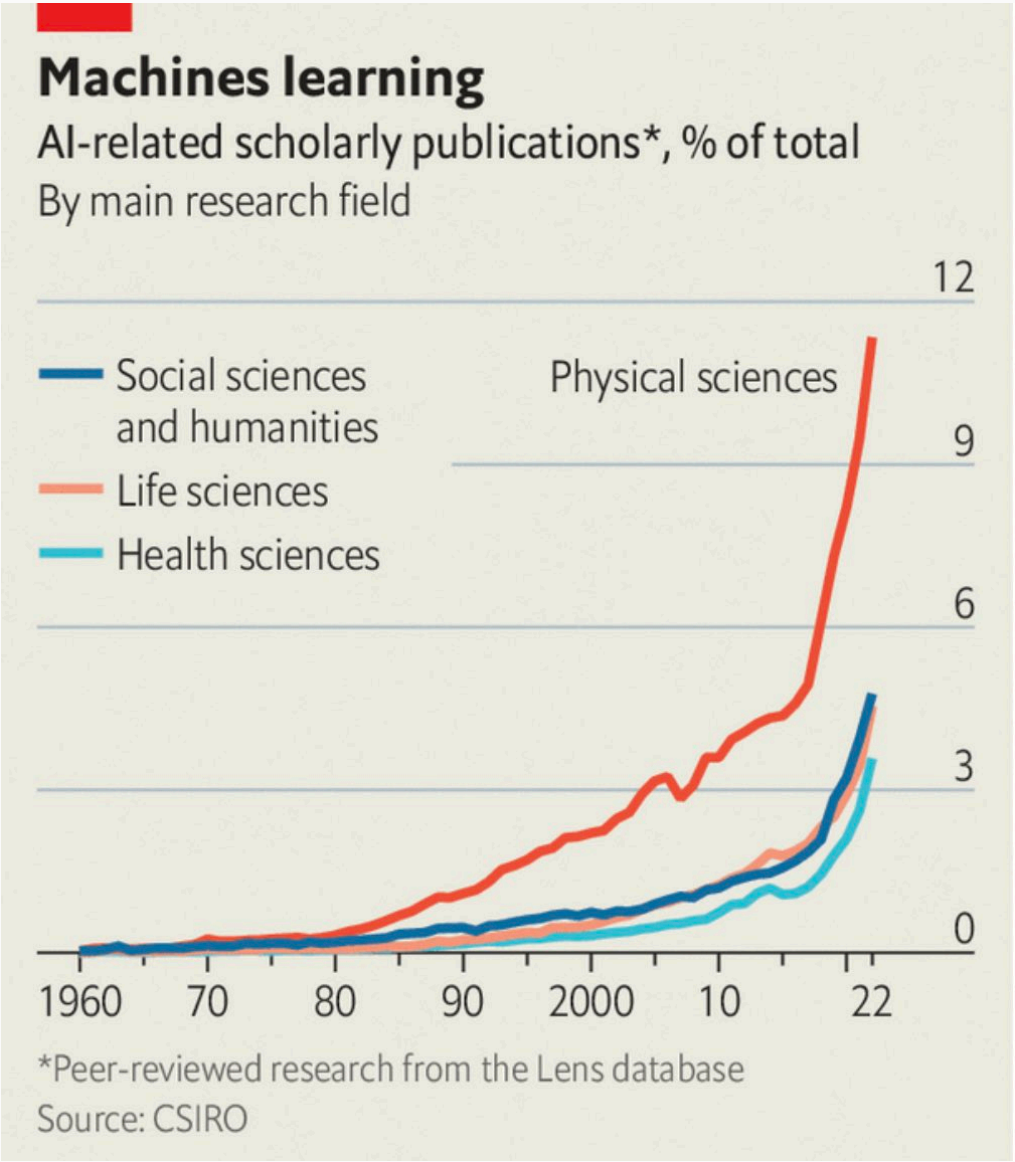
- A increasing number of geophysical data (one spatial mission: 24 TB/day)
- Data with highly significant spatial patterns



Sea Surface temperature of the gulf stream

source: Talley (2000)

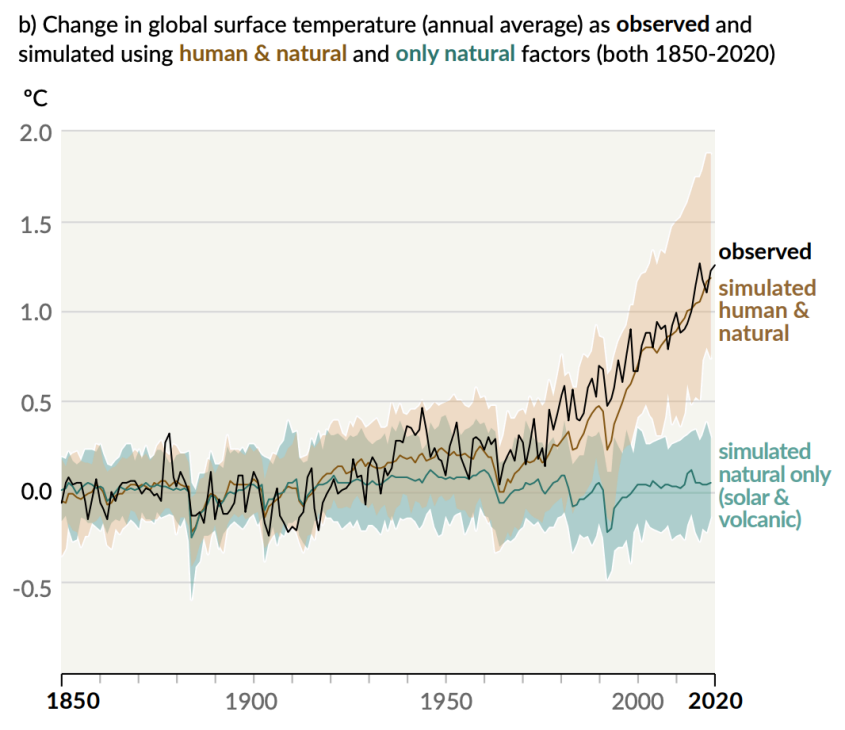
AI in physical sciences



Why is physical modelling specific?

NASDAQ Composite sock market index over the last 10 years

Figure 1: IPCC, AR6, WG1

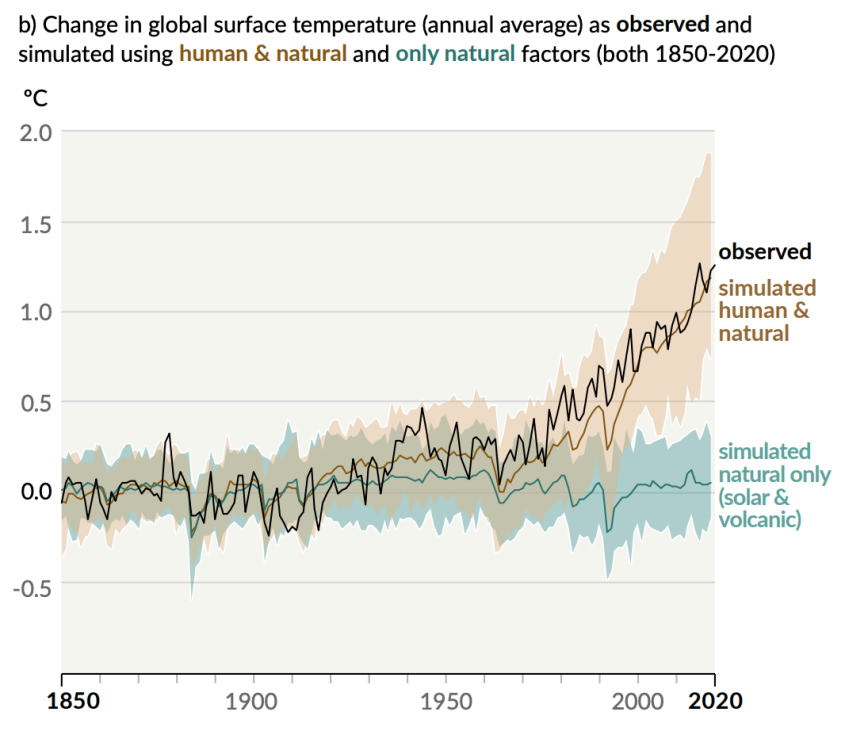


Why is physical modelling specific?

NASDAQ Composite sock market index over the last 10 years

Mostly unknown dynamical processes

Figure 1: IPCC, AR6, WG1



Mostly known dynamical processes (based on physical principles)

What about data assimilation?

Machine learning and data assimilation are closely linked.

- Cheng, Sibo, et al., 2023. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica* 10.6:1361-1387.
- Geer, A.J., 2021. Learning earth system models from observations: machine learning or data assimilation?. *Philosophical Transactions of the Royal Society A*, 379(2194)
- Brajard et al. 2019. Connections between data assimilation and machine learning to emulate a numerical model. *Proceedings of the 9th International Workshop on Climate informatics*
- Bocquet et al. 2019. Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear processes in geophysics.* 26(3).
- Abarbanel, H.D., Rozdeba, P.J. and Shirman, S., 2018. Machine learning: Deepest learning as statistical data assimilation problems. *Neural computation, 30(8).*

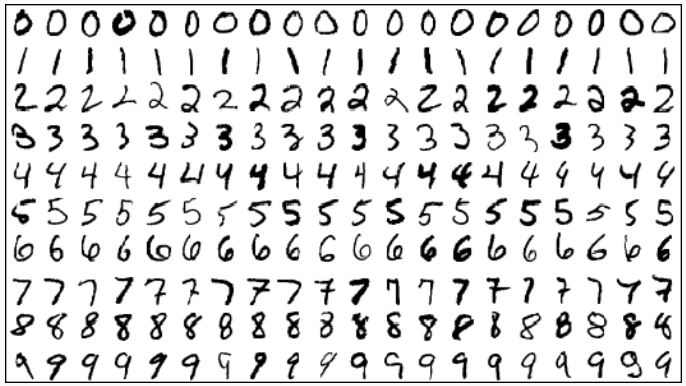
Generalities on Machine Learning

What is this about ?

Can we extract knowledge, make some predictions, determine a ”model” using this large amount of data ?

What is this about ?

Can we extract knowledge, make some predictions, determine a ”model” using this large amount of data ?

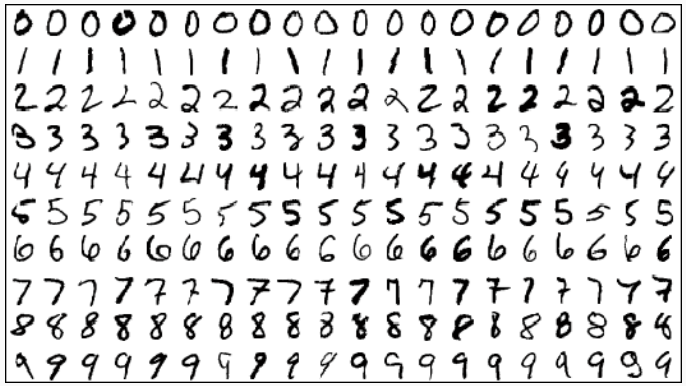


Base of images

$\longrightarrow$ Digit$\in \{0, \ldots, 9\}$

What is this about ?

Can we extract knowledge, make some predictions, determine a "model" using this large amount of data ?



Base of images

$\longrightarrow$ Digit$\in \{0, \ldots, 9\}$

- From high dimensional data (thousands to millions dimensions) to reduced dimensional data (less than 100)
- From disorganized data to organized information
- Can we teach a machine how to do that ?

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a ”useful” representation of these data or some coherent categories.

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a ”useful” representation of these data or some coherent categories.
    - Determine typical behaviours of clients in a supermarket knowing what they have bought.

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a ”useful” representation of these data or some coherent categories.
    - Determine typical behaviours of clients in a supermarket knowing what they have bought.
3. Semi-Supervised Learning: The training is supervised by the data itself

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a ”useful” representation of these data or some coherent categories.
    - Determine typical behaviours of clients in a supermarket knowing what they have bought.
3. Semi-Supervised Learning: The training is supervised by the data itself
    - time series forecasting.

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a "useful" representation of these data or some coherent categories.
    - Determine typical behaviours of clients in a supermarket knowing what they have bought.
3. Semi-Supervised Learning: The training is supervised by the data itself
    - time series forecasting.
4. Reinforcement Learning: We can initiate and observe the interaction of an agent with its environment. We want to optimize the behavior of the agent.

Two types of settings

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a "useful" representation of these data or some coherent categories.
    - Determine typical behaviours of clients in a supermarket knowing what they have bought.
3. Semi-Supervised Learning: The training is supervised by the data itself
    - time series forecasting.
4. Reinforcement Learning: We can initiate and observe the interaction of an agent with its environment. We want to optimize the behavior of the agent.
    - Playing a chess game.

Two classes of (self)-supervised learning problems

1. Regression: Determination of a quantitative variable from a set of data
    - The price of a building from various predictors (Surface, ...)
    - A physical value (Temperature, humidity, ...) in the future knowing the past
    - ...

Two classes of (self)-supervised learning problems

1. Regression: Determination of a quantitative variable from a set of data
   - The price of a building from various predictors (Surface, ...)
   - A physical value (Temperature, humidity, ...) in the future knowing the past
   - ...
2. Classification: Determination of a class
   - A digit from a image
   - Identification of the content of an image
   - ...

Formally

A Machine

$$y = \mathcal{M}(x, \theta)$$

- x: input
- y: output
- $\mathcal{M}$: a model (named "machine")
- θ : parameters of the model $\mathcal{M}$.

Machine learning consists in optimizing θ using a set of data.
This is the training process.

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - y is a subset of x:
    self-supervised learning

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - y is a subset of x:
    self-supervised learning
- An objective
  - y is quantitative: regression
  - y is a class: classification

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - y is a subset of x: self-supervised learning
- An objective
  - y is quantitative: regression
  - y is a class: classification
- A computational architecture (the machine)
  - linear
  - non-linear
  - neural networks, random forest, ...

The Machine Learning recipe

A Machine

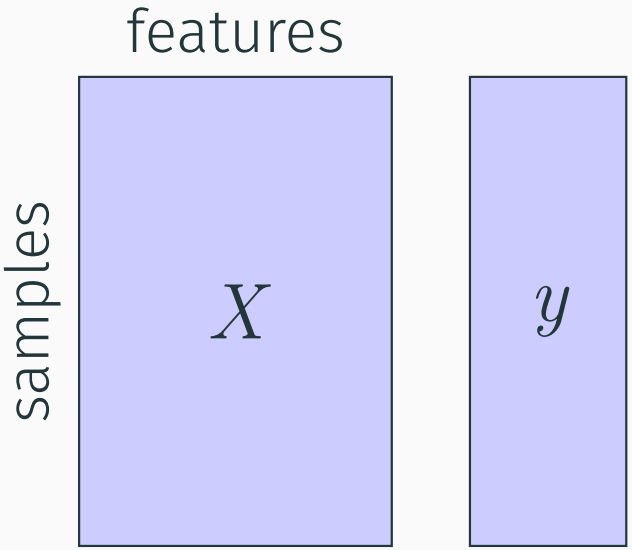
$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - y is a subset of x: self-supervised learning
- An objective
  - y is quantitative: regression
  - y is a class: classification
- A computational architecture (the machine)
  - linear
  - non-linear
  - neural networks, random forest, ...
- A learning process
  - Estimation of θ

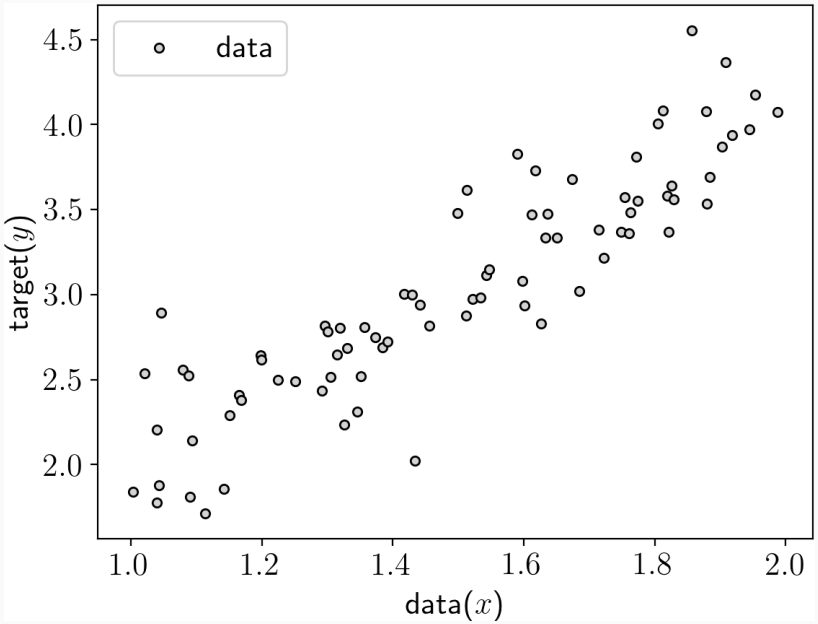
Multidimensional data

Generally, we have multidimensional data X and a one-dimensional target y.



An illustration

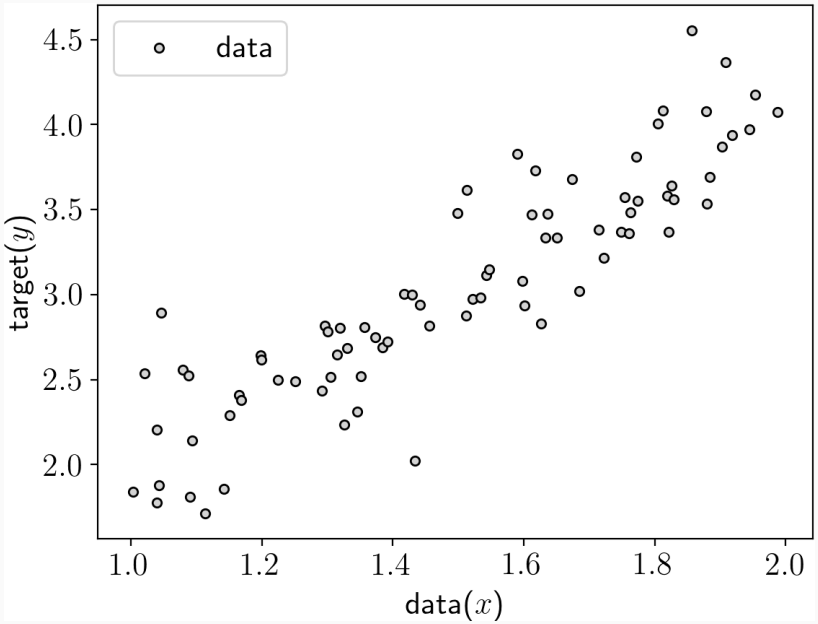
- Some Data



  - some values of y are known: supervised learning
  - y is quantitative: regression

An illustration

- Some Data

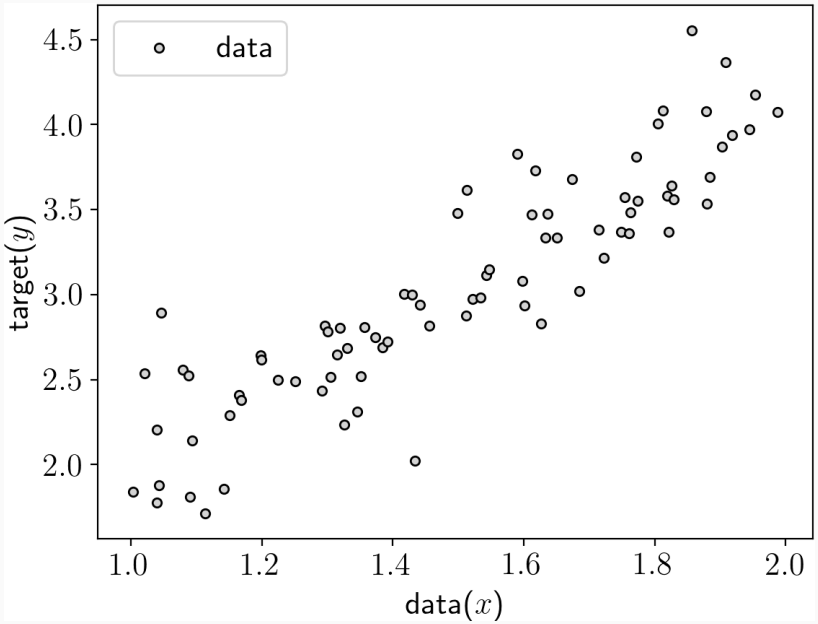


  - some values of y are known: supervised learning
  - y is quantitative: regression

- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

An illustration

- Some Data



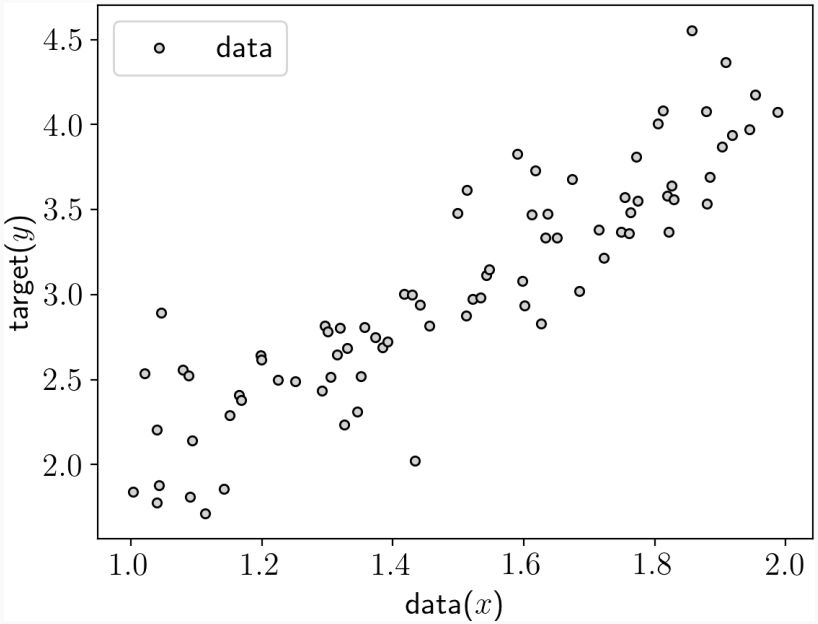
  - some values of y are known: supervised learning
  - y is quantitative: regression

- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

- A model: $y = \theta_1 X + \theta_0$ (linear)

An illustration

- Some Data

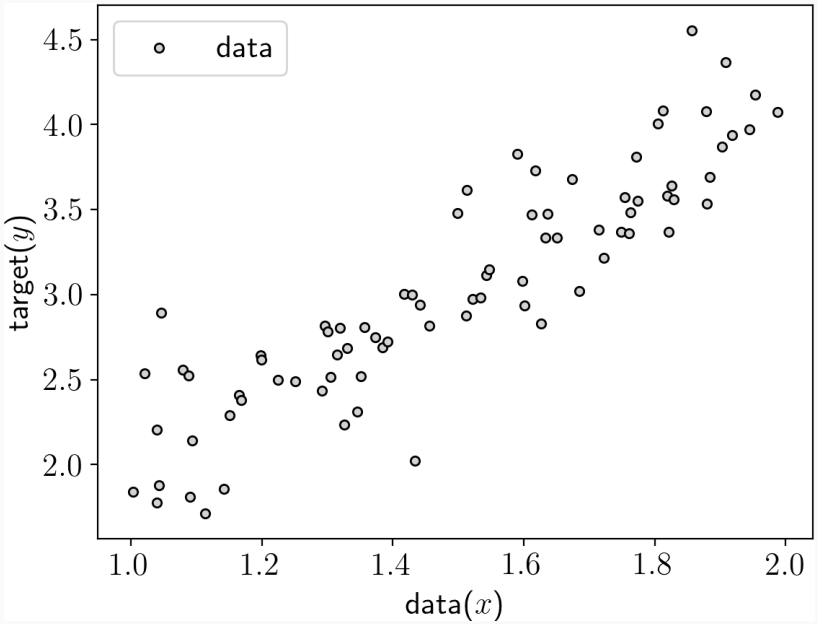


  - some values of y are known: supervised learning
  - y is quantitative: regression
- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

- A model: $y = \theta_1 X + \theta_0$ (linear)
- A learning process: $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$

An illustration

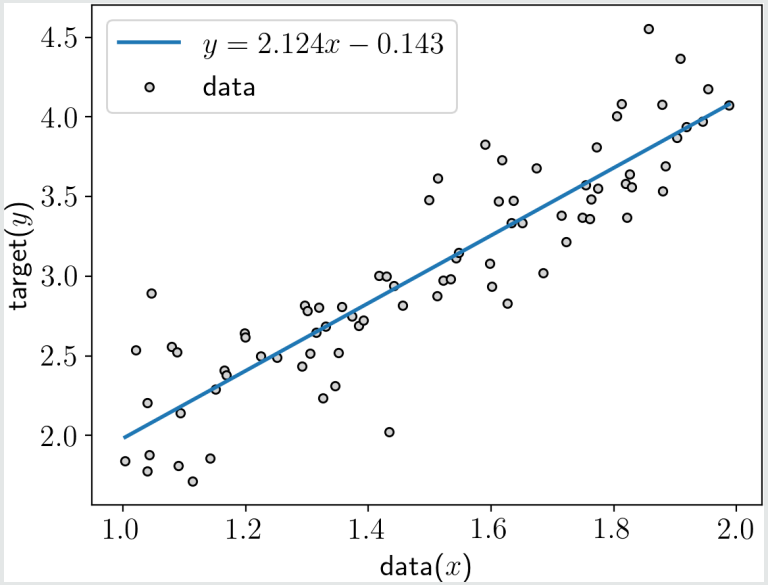
- Some Data



  - some values of y are known: supervised learning
  - y is quantitative: regression
- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

- A model: $y = \theta_1 X + \theta_0$ (linear)
- A learning process: $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$

Result



Model selection/validation

Choice of the model

Polynomial regression

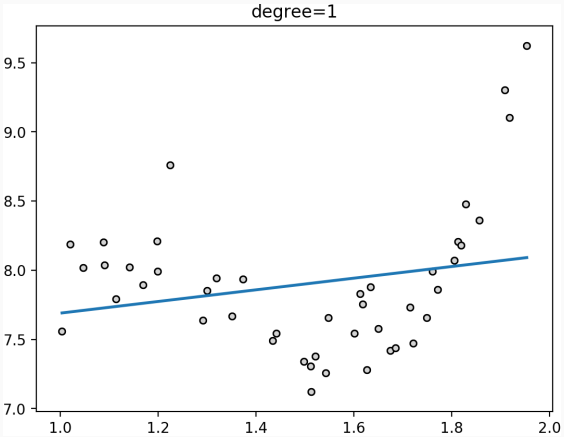
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

Choice of the model

Polynomial regression

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

degree = 1 (linear)

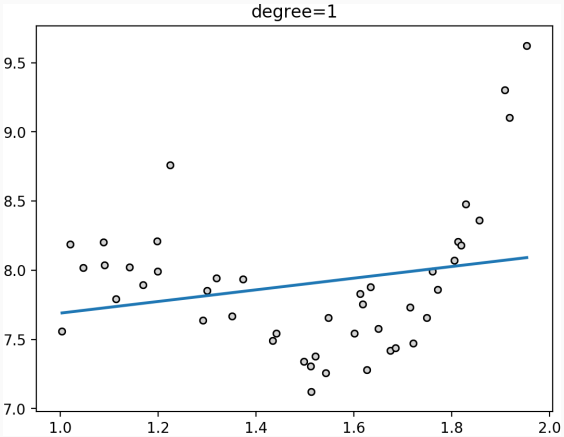


Choice of the model

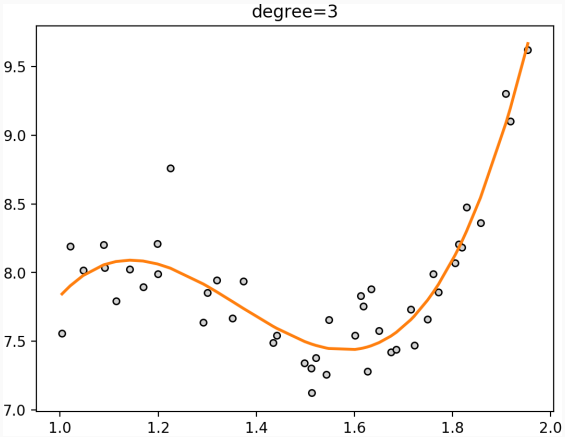
Polynomial regression

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

degree = 1 (linear)



degree = 3

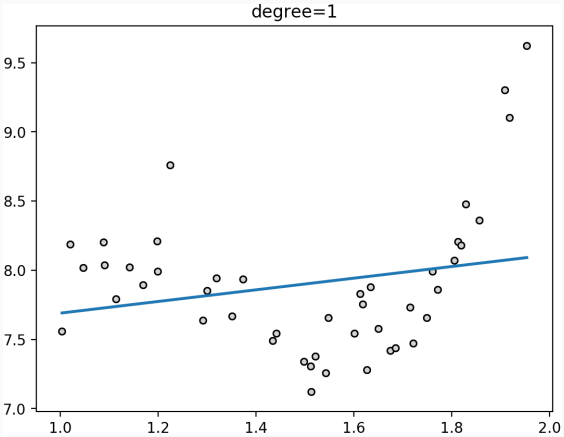


Choice of the model

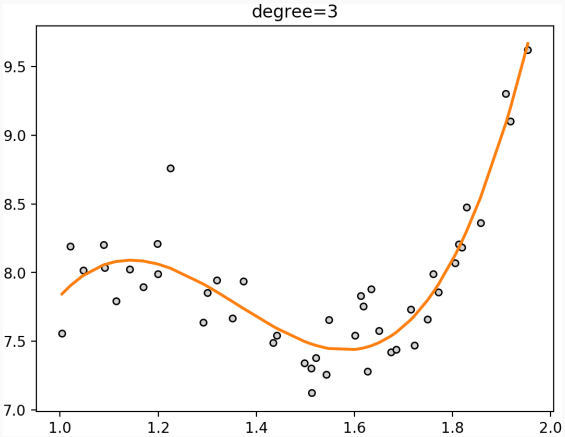
Polynomial regression

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

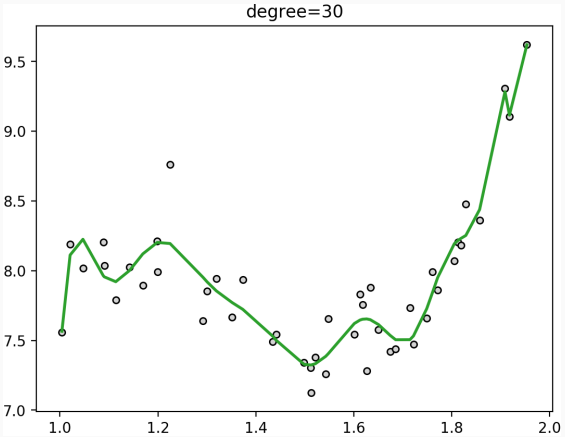
degree = 1 (linear)



degree = 3



degree = 30



What is the best model?

Train/Validation split

The idea

Evaluate a score on an independent dataset

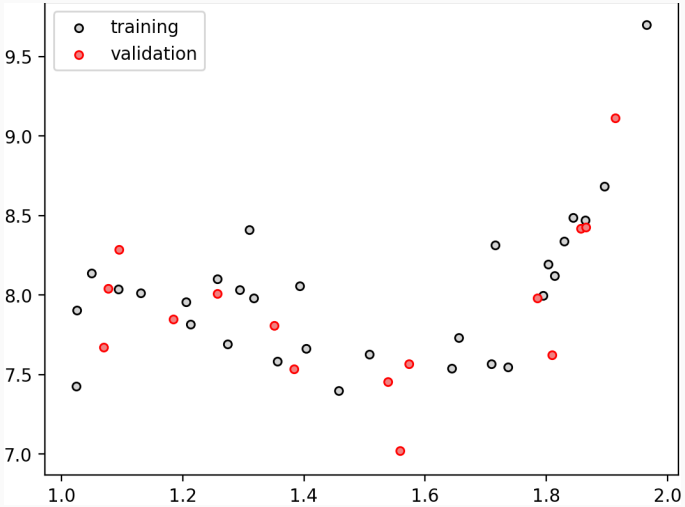
Train/Validation split

The idea

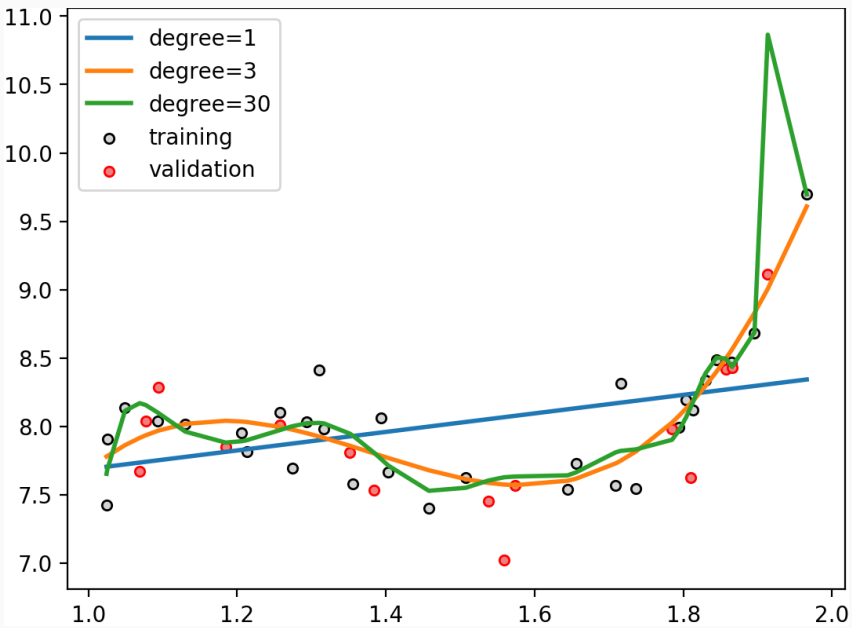
Evaluate a score on an independent dataset

In our example we can randomly divide (X, y) in two datasets:

- The training dataset X_{train}, y_{train} used to fit the model.
- The validation dataset X_{val}, y_{val} used to compute the score (e.g., correlation, mean-squared error)



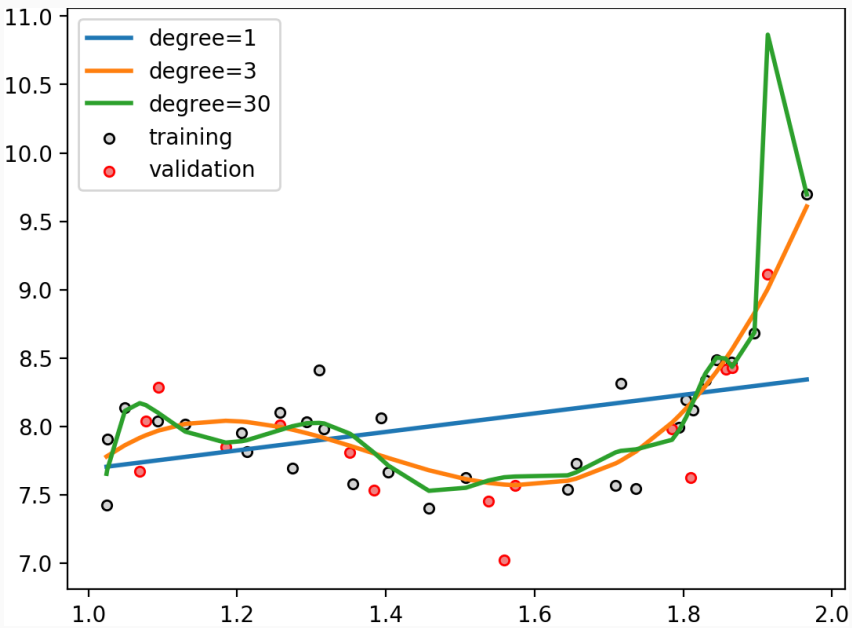
Choice of the model



Score: Mean Square Error (MSE)

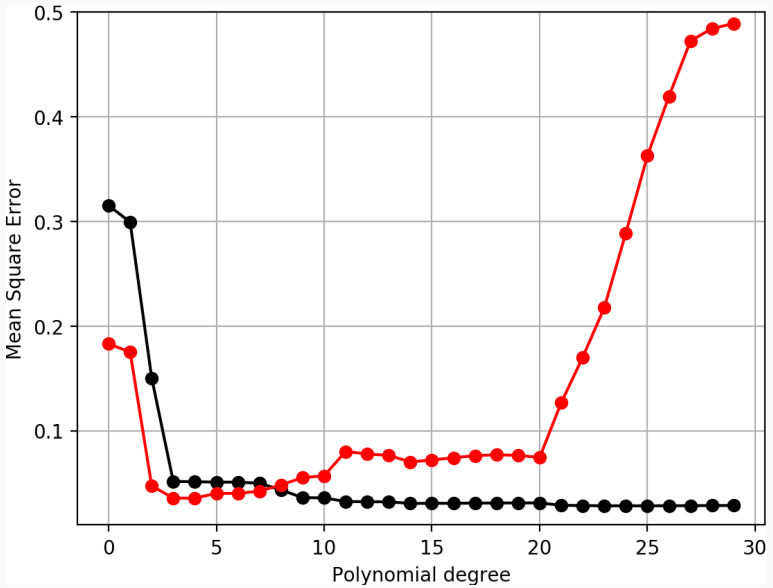
Deg.	Train Score	Val. Score
1	0.17	0.23
3	0.045	0.062
30	0.035	0.27

Choice of the model



Score: Mean Square Error (MSE)

Deg.	Train Score	Val. Score
1	0.17	0.23
3	0.045	0.062
30	0.035	0.27

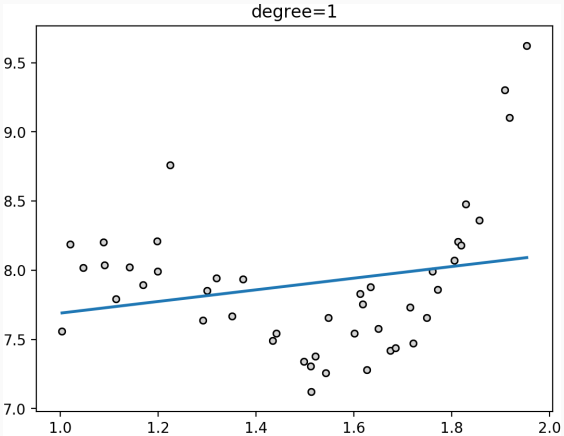


Choice of the model

Polynomial regression

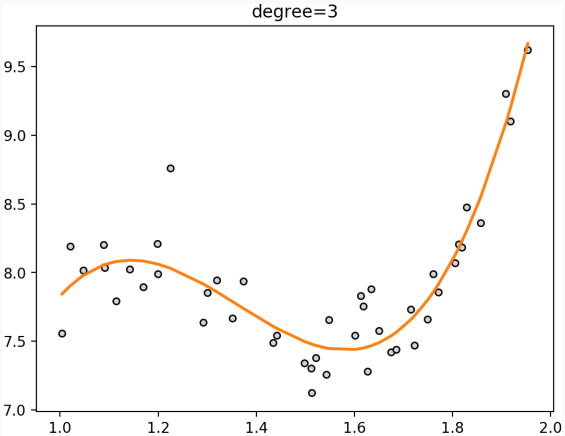
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

degree = 1 (linear)



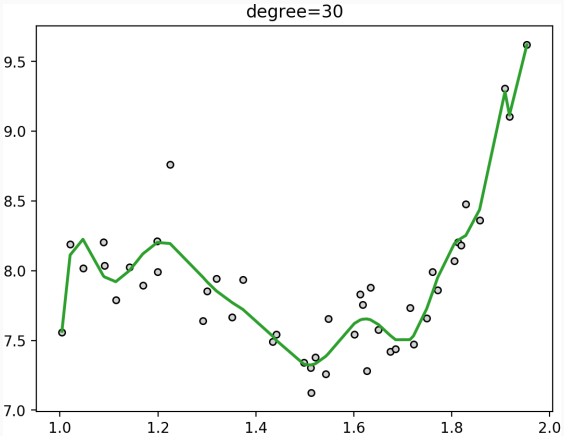
underfitting

degree = 3



good fit

degree = 30



overfitting

Train/Validation split

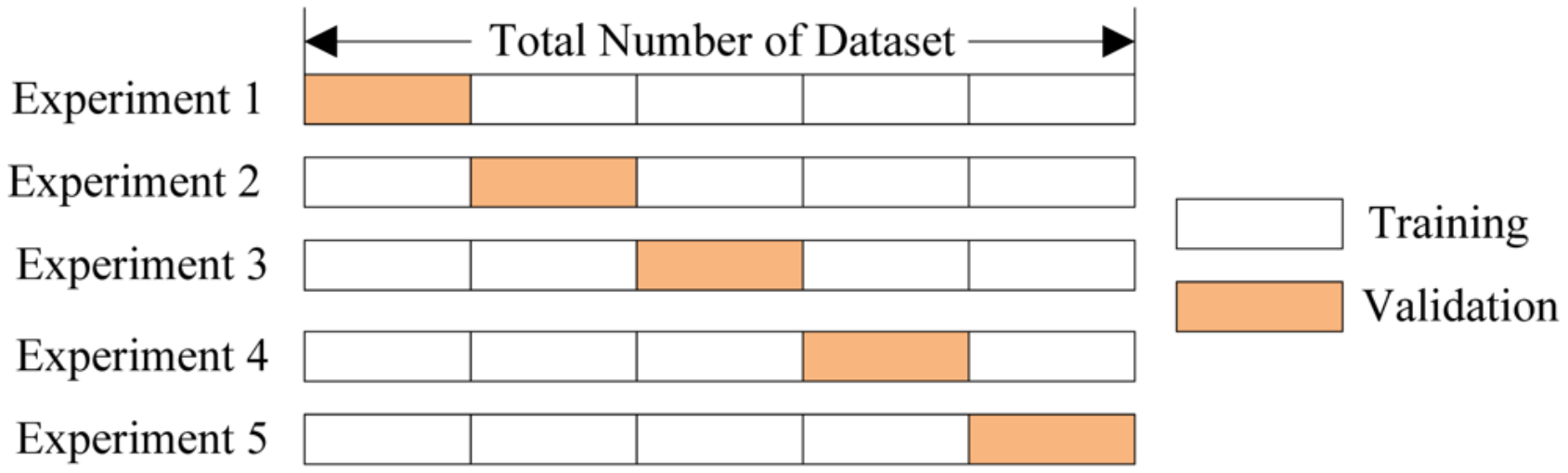
Drawbacks

- drastically reduces the number of samples that can be used for learning the model
- Results can depend on a particular random choice for the pair of (train, validation) sets.

More Robust: cross validation

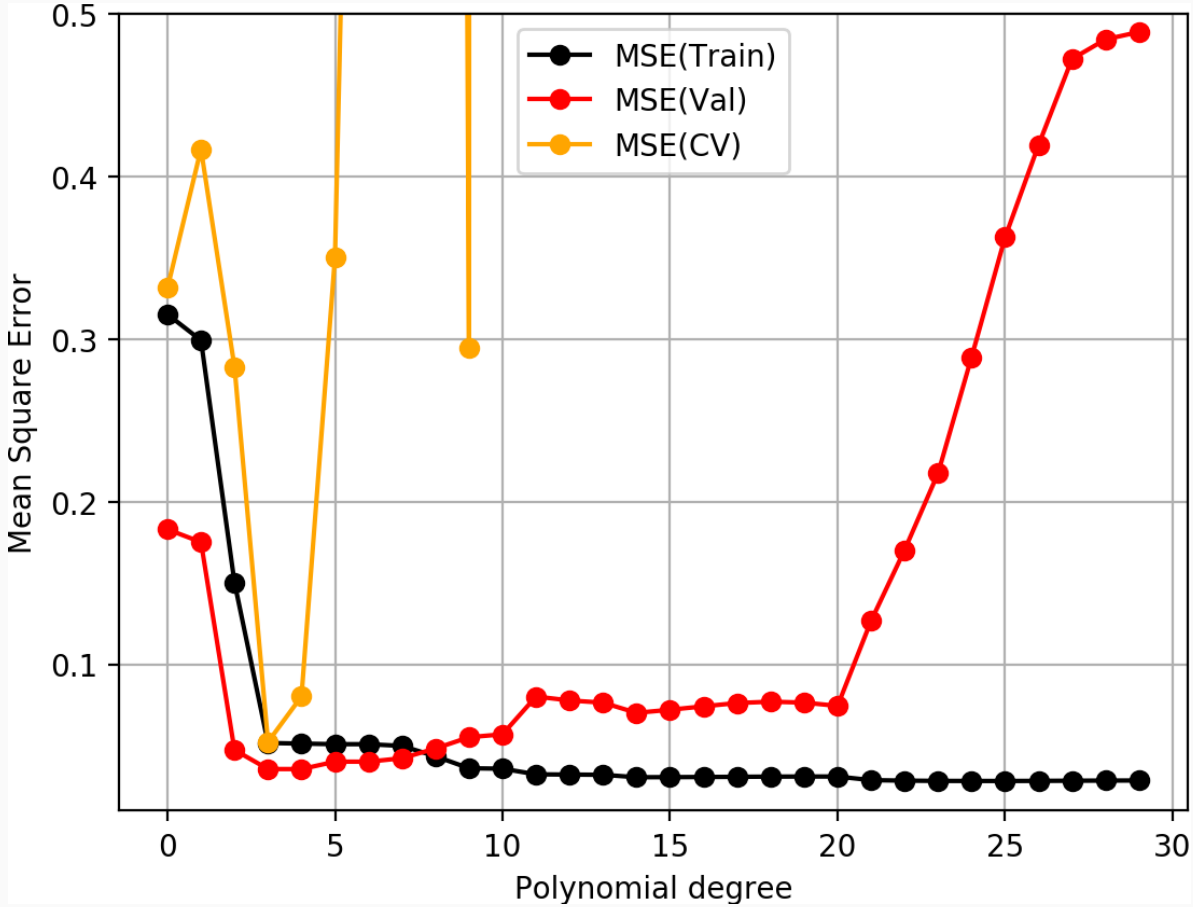
The idea

- Dividing the data in n folds,
- Learning n model (each time with a different training set),
- Compute the mean score over n validation set.



Cross-Validation

Fold	MSE
1	0.052
2	0.043
3	0.137
4	0.025
5	0.048
6	0.144
7	0.011
8	0.025
9	0.010
10	0.028
Mean	0.05



Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).

Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.

Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.
3. But then... the validation set was used to determine the best machine learning process

Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.
3. But then... the validation set was used to determine the best machine learning process
4. To evaluate independently the performance of our model, we should compute the score on a third independent dataset: The test dataset.

Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.
3. But then... the validation set was used to determine the best machine learning process
4. To evaluate independently the performance of our model, we should compute the score on a third independent dataset: The test dataset.

Wrapping up

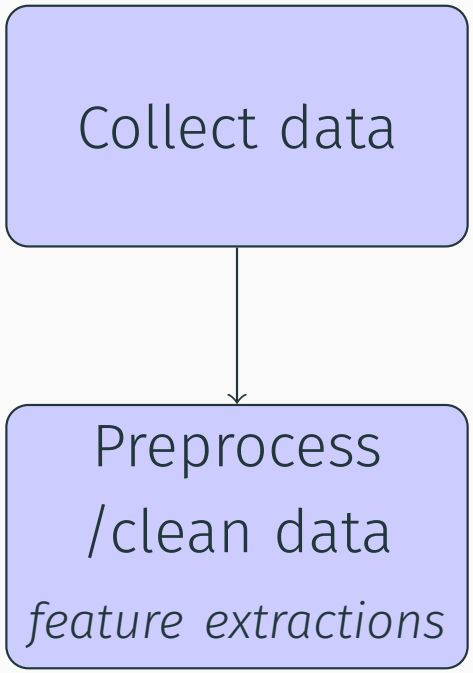
1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.
3. But then... the validation set was used to determine the best machine learning process
4. To evaluate independently the performance of our model, we should compute the score on a third independent dataset: The test dataset.

More on that in next lecture...

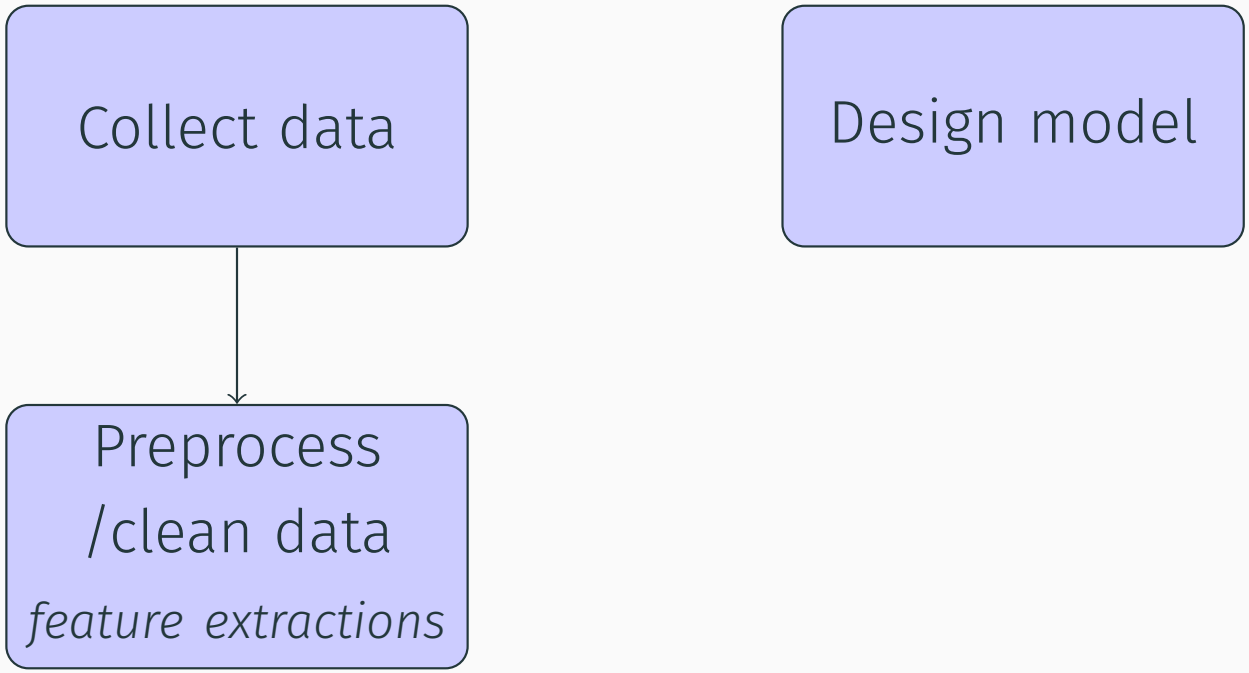
Steps of a machine learning process

Steps

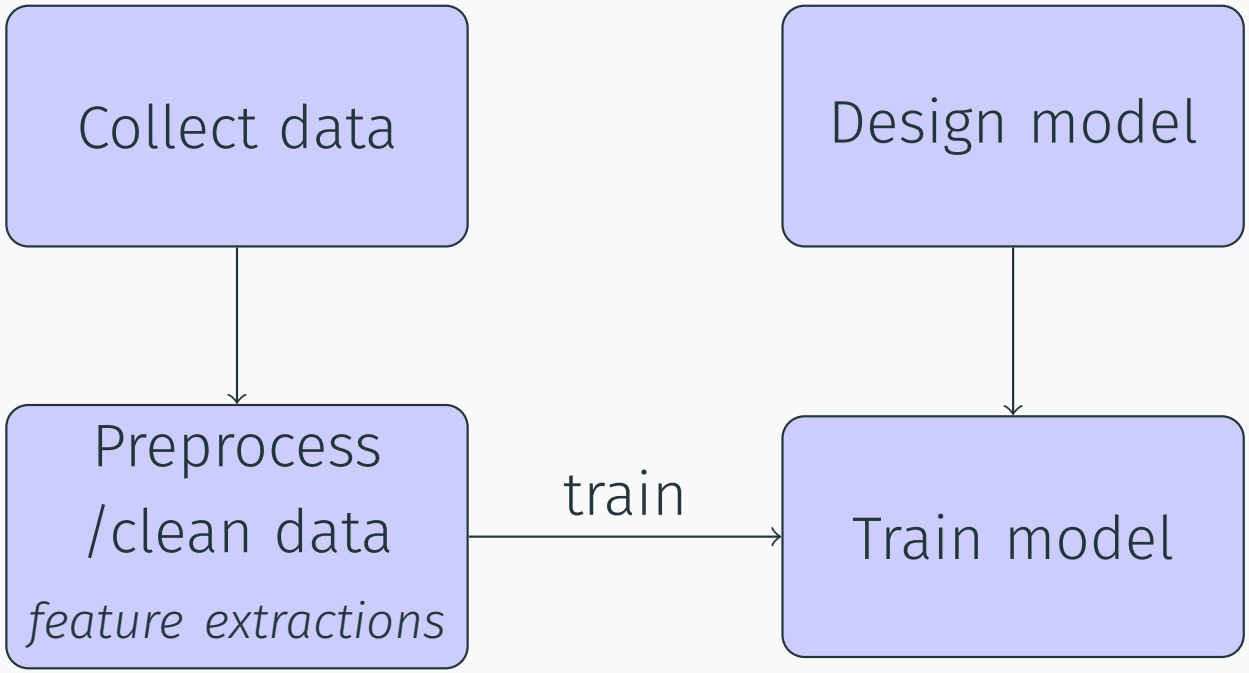
Steps



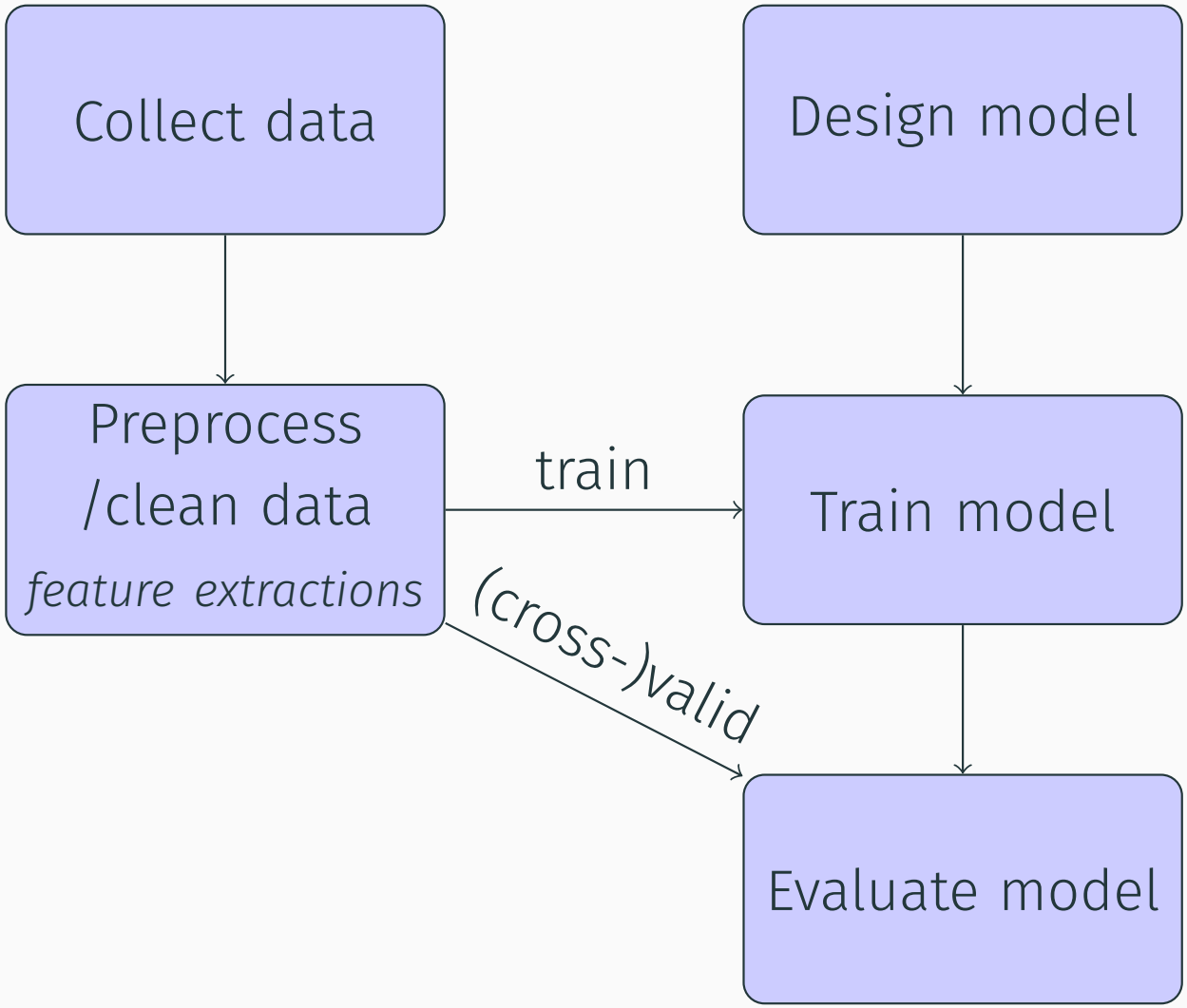
Steps


Collect data
Preprocess
/clean data
feature extractions
Design model

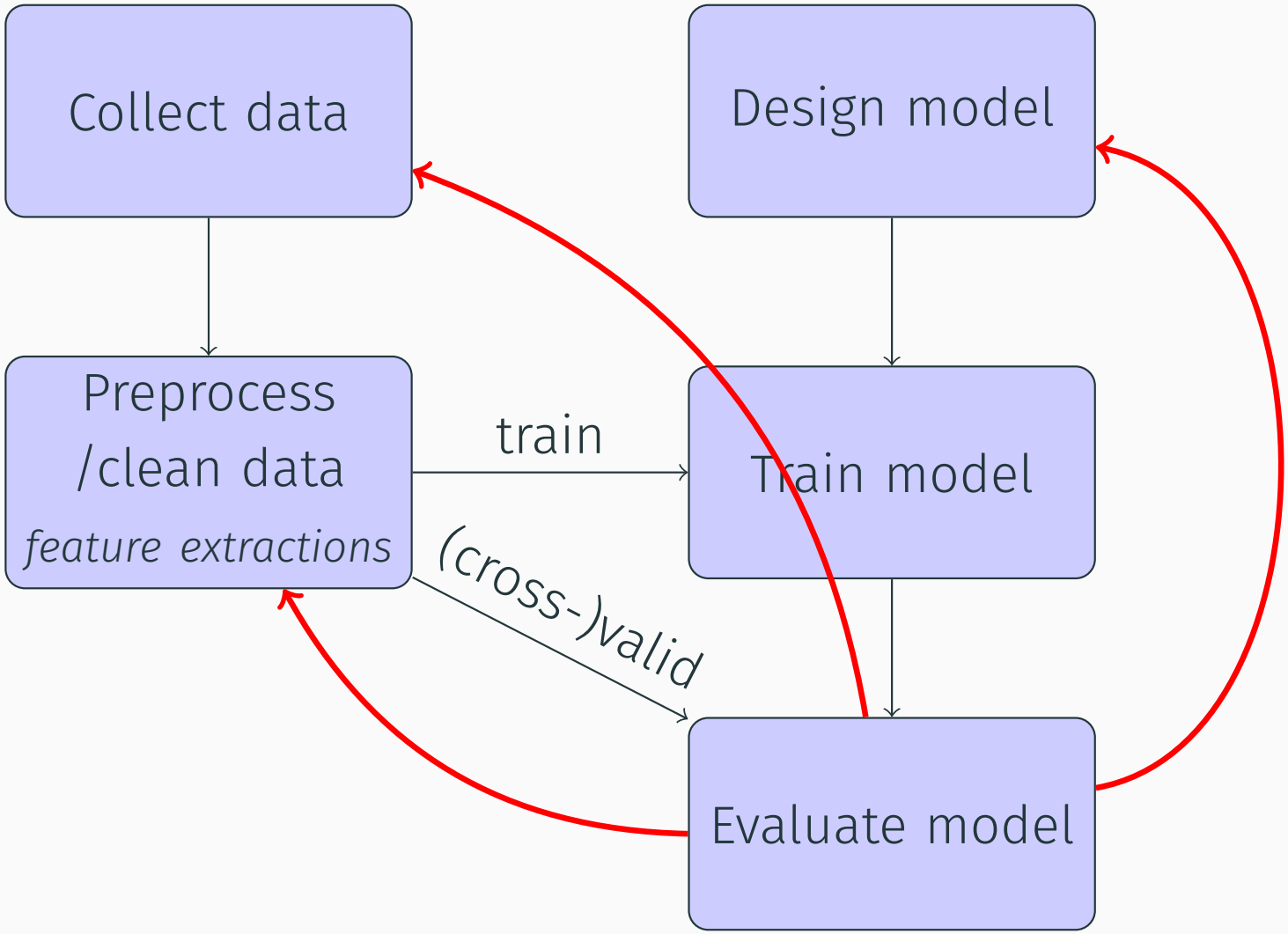
Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model

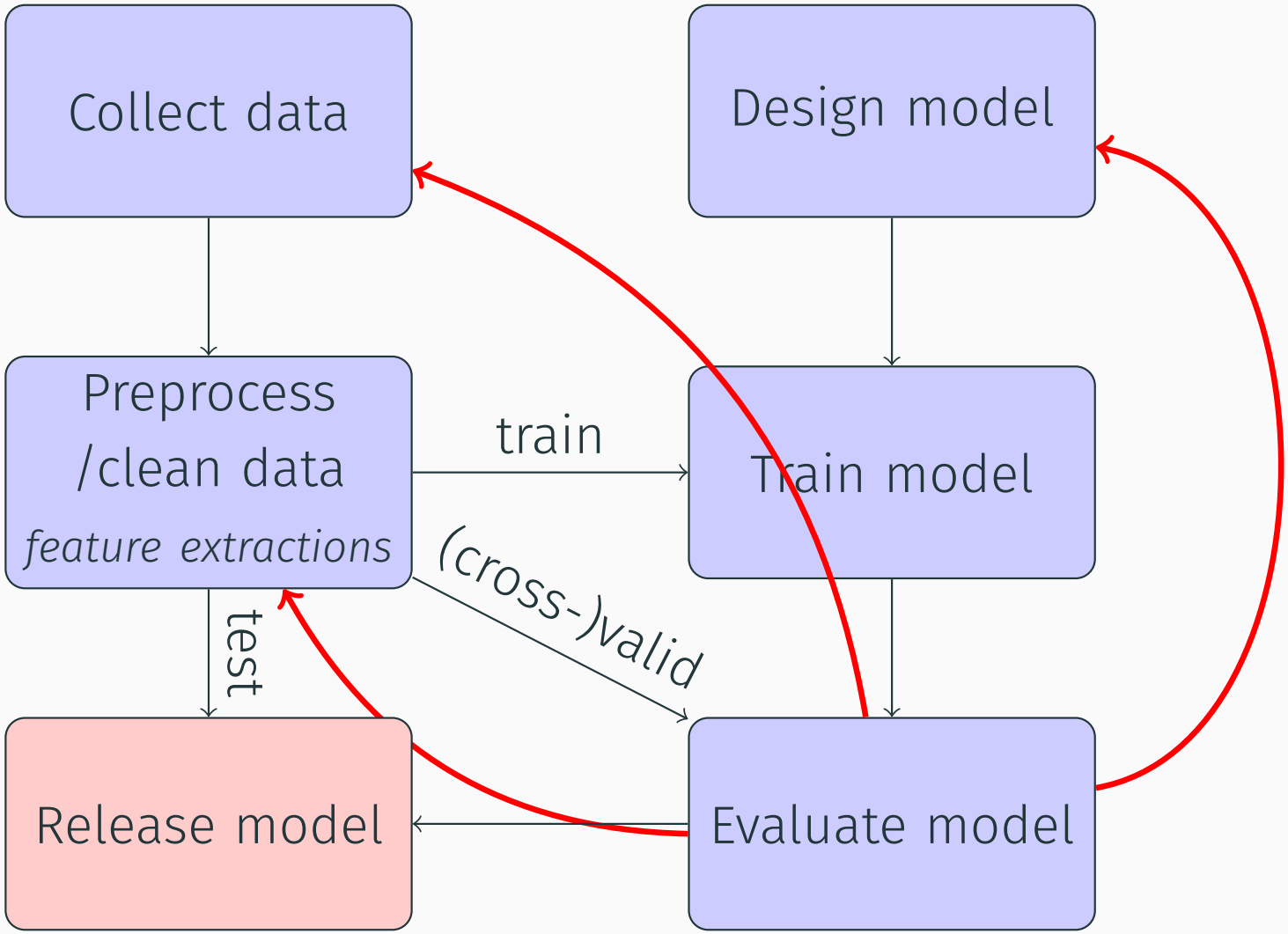
Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model
(cross-)valid
Evaluate model

Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model
(cross-)valid
Evaluate model

Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model
(cross-)valid
test
Release model
Evaluate model

In summary

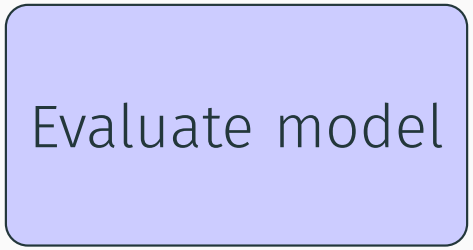
From one dataset, 3 sub-datasets have to be extracted:

- A training dataset
- A validation dataset

Can be done iteratively in a cross-validation procedure.
Some parameters of the model (e.g. polynomial order in a polynomial regression) were determined from the validation dataset.

- A test dataset (independent from the two others) to estimate the final performance of the model.

Additional Comments



Model evaluation is a critical step in the machine learning pipeline. Key considerations include:

- Is the dataset and evaluation metric unbiased? Do they accurately reflect the desired model performance?
- Is there a simpler model that performs sufficiently well? Avoid proposing complex machine learning solutions when a simpler approach is more effective.

Adequate metric

A typical metric for classification problems is the **accuracy**:

$$\text{acc} = \frac{TP + TN}{T + N}$$

TP: True Positive (number of positive results correctly classified)

TN: True Negative (number negative results correctly classified)

P: Number of positive examples

N: Number of negative examples

Adequate metric

A typical metric for classification problems is the **accuracy**:

$$\text{acc} = \frac{TP + TN}{T + N}$$

TP: True Positive (number of positive results correctly classified)

TN: True Negative (number negative results correctly classified)

P: Number of positive examples

N: Number of negative examples

Example of Credit Card Fraud Detection

Objective: Predict if a credit card operation is fraudulent

In the dataset, frauds (positive class) are **0.172%** of the cases.

`https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`

Adequate metric

A typical metric for classification problems is the **accuracy**:

$$\text{acc} = \frac{TP + TN}{T + N}$$

TP: True Positive (number of positive results correctly classified)

TN: True Negative (number negative results correctly classified)

P: Number of positive examples

N: Number of negative examples

Example of Credit Card Fraud Detection

Objective: Predict if a credit card operation is fraudulent

In the dataset, frauds (positive class) are **0.172%** of the cases.

`https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`

Predicting always the negative class: acc=**99.828%** !!!

Accuracy is not a adequat metric in this case

Adequate metric

A typical metric for classification problems is the **accuracy**:

$$\text{acc} = \frac{TP + TN}{T + N}$$

TP: True Positive (number of positive results correctly classified)

TN: True Negative (number negative results correctly classified)

P: Number of positive examples

N: Number of negative examples

Example of Credit Card Fraud Detection

Objective: Predict if a credit card operation is fraudulent

In the dataset, frauds (positive class) are **0.172%** of the cases.

`https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`

Predicting always the negative class: acc=**99.828%** !!!

Accuracy is not a adequat metric in this case

Other metric:

$$\text{recall} = \frac{TP}{P}$$

Always define a baseline

Example: Weather Forecast

Imagine we have a model predicting ”rain” or ”no-rain” the day after with an accuracy of 75%.

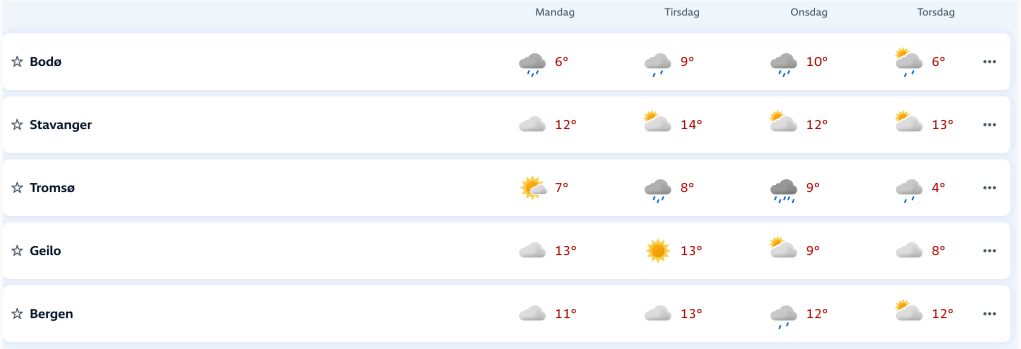
Is it good?

Always define a baseline

Example: Weather Forecast

Imagine we have a model predicting ”rain” or ”no-rain” the day after with an accuracy of 75%.

Is it good?



A very simple baseline: the persistence.

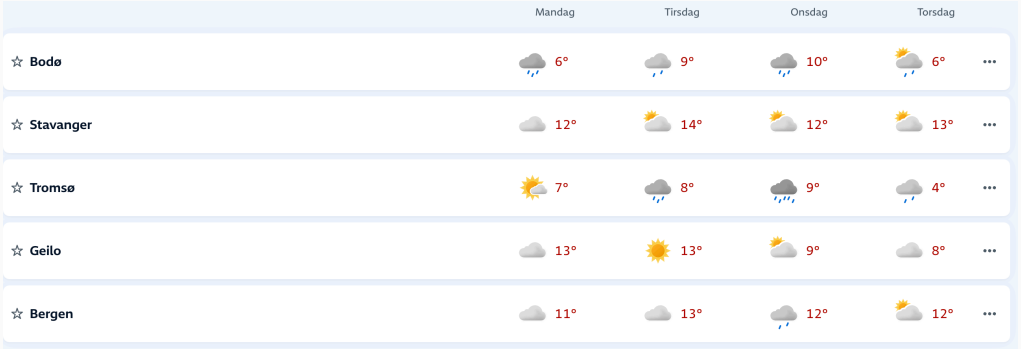
Predict that the day after is the same as the day before

Always define a baseline

Example: Weather Forecast

Imagine we have a model predicting ”rain” or ”no-rain” the day after with an accuracy of 75%.

Is it good?



A very simple baseline: the persistence.

Predict that the day after is the same as the day before

Accuracy of persistence: 80%

In this case, a ML model should do better than 80% to be considered.


Kahoot!

Feature processing

Structure of the features

What is a feature?

It is a measurable property of the data.

Orgnisation of the features:

Scalar	Image/Tensor	Sequence	Unstructured
number of room of a house, profession, ...	values of a pixel in a image	time series, text	available paths in a city map

Type of features

- quantitative/continuous features (e.g. distance to employment centers, temperature)

Type of features

- quantitative/continuous features (e.g. distance to employment centers, temperature)
- ordinal/discrete features (e.g. number of rooms, category of a hurricane)

Type of features

- quantitative/continuous features (e.g. distance to employment centers, temperature)
- ordinal/discrete features (e.g. number of rooms, category of a hurricane)
- categorical features (e.g. name of the neighborhood, name of the ocean)

Type of features

- quantitative/continuous features (e.g. distance to employment centers, temperature)
- ordinal/discrete features (e.g. number of rooms, category of a hurricane)
- categorical features (e.g. name of the neighborhood, name of the ocean)

Type of features

- quantitative/continuous features (e.g. distance to employment centers, temperature)
- ordinal/discrete features (e.g. number of rooms, category of a hurricane)
- categorical features (e.g. name of the neighborhood, name of the ocean)

Encoding of the features?

Feature encoding

Type	Examples		Encoding
Quantitative	distance	$\{1.2, 2.3, 0.1\}$	$\{1.2, 2.3, 0.1\}$
Ordinal	rooms	$\{2, 3, 4\}$	$\{2, 3, 4\}$
Qualitative	Ocean	$\{\text{Atlantic}, \text{Indian}, \text{Pacific}\}$	$\{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$

Feature encoding

Type	Examples		Encoding
Quantitative	distance	$\{1.2, 2.3, 0.1\}$	$\{1.2, 2.3, 0.1\}$
Ordinal	rooms	$\{2, 3, 4\}$	$\{2, 3, 4\}$
Qualitative	Ocean	$\{\text{Atlantic}, \text{Indian}, \text{Pacific}\}$	$\{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$

Qualitative variable: one-hot encoding.

- You must not encode qualitative features with integer $1, 2, 3$, it would mean that Pacific > Indian > Atlantic.
- In $\mathrm{sklearn}$ there is a function that makes the one-hot encoding: $\mathrm{OneHotEncoder}()$
- If the number of modalities (number of different features) is high, encoding qualitative feature produces a big-sized vector.

Embedding

A common way to deal with features with a lot of modalities: Embedding

Principle of embedding

Let's consider a qualitative variable with n modalities, represented by the n-dimensional binary vector $\mathbf{x}$

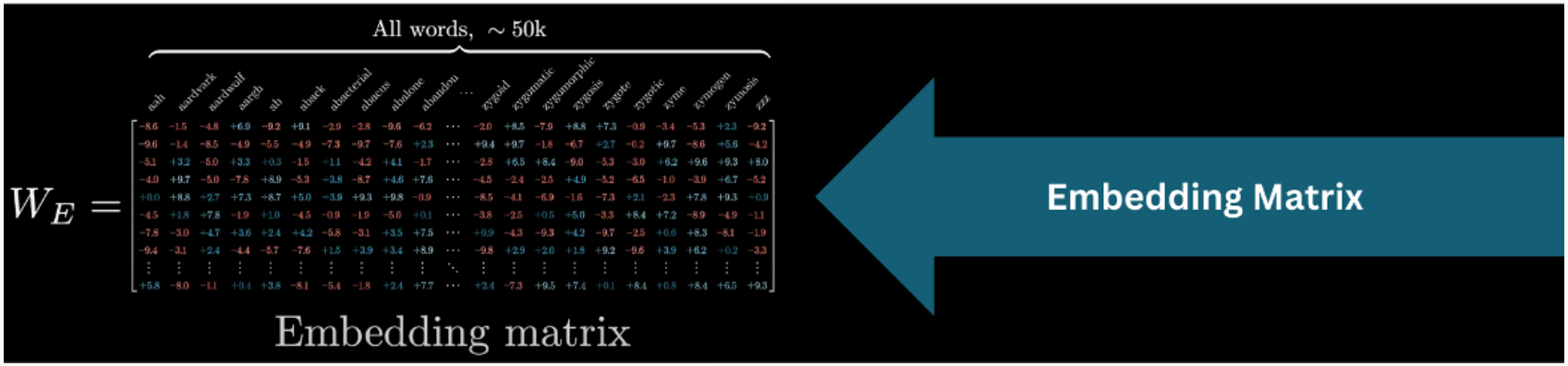
Embedding consists in representing this variable by a vector $\mathbf{v} \in \mathbb{R}^p, p << n$

The embedding is represented by a matrix $\mathbf{M} \in \mathbb{R}^{n \times p}$ such as:

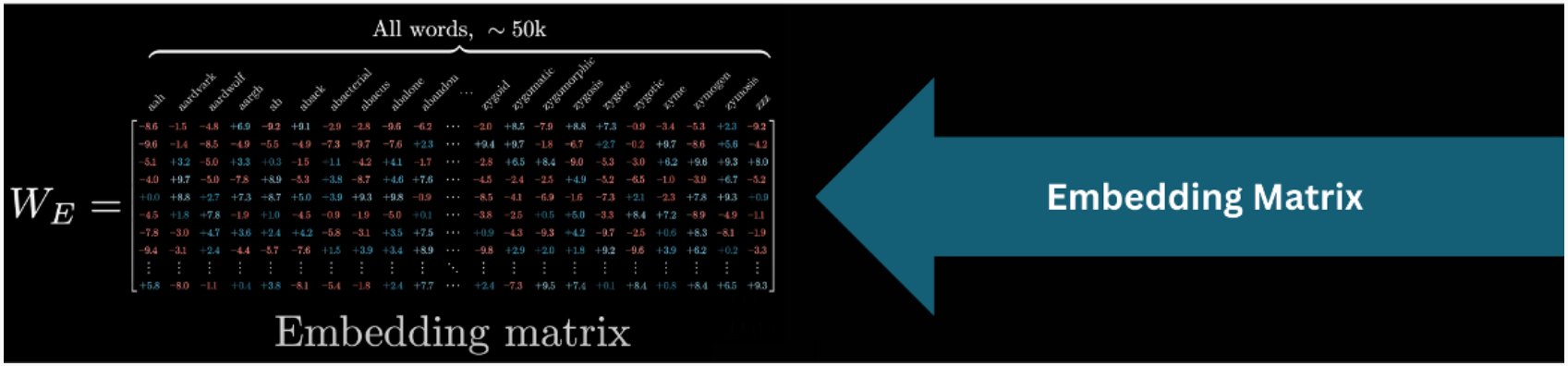
$$\mathbf{v} = \mathbf{M} \cdot \mathbf{x}$$

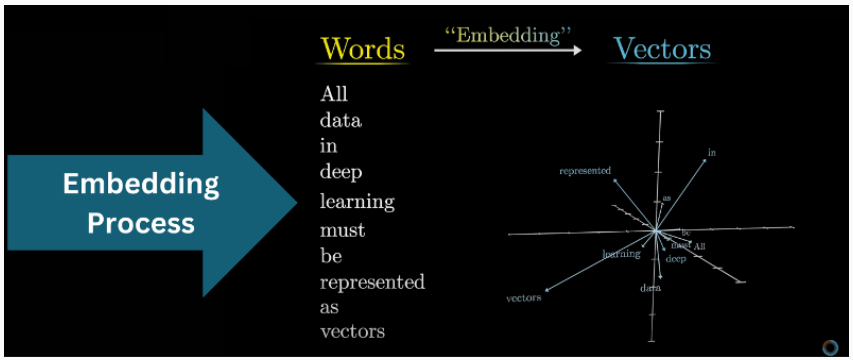
Coefficients of $\mathbf{M}$ have to be optimized given an objective criteria (that depends on your problem)

Embedding for Large Language Models (LLM)

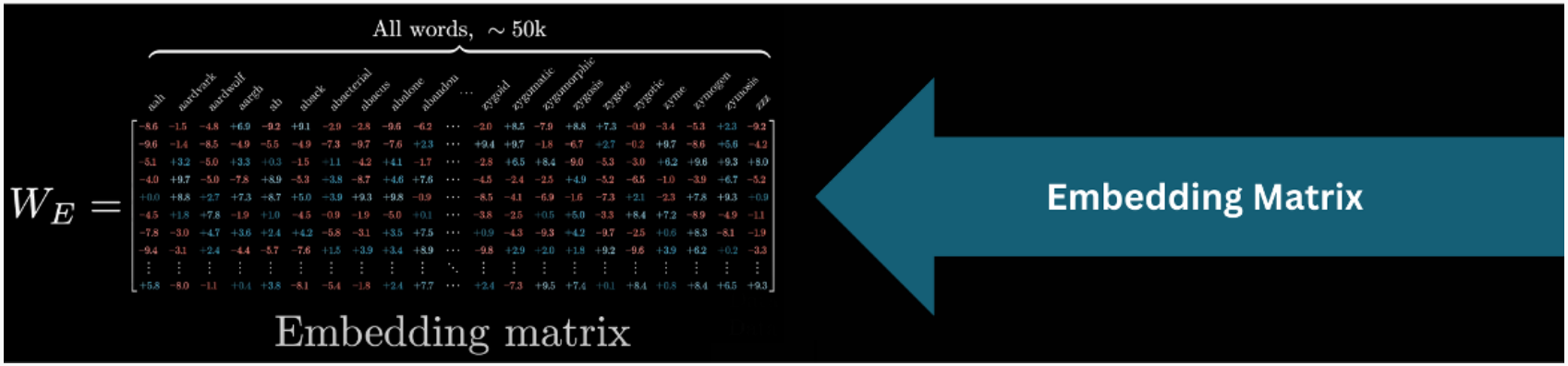


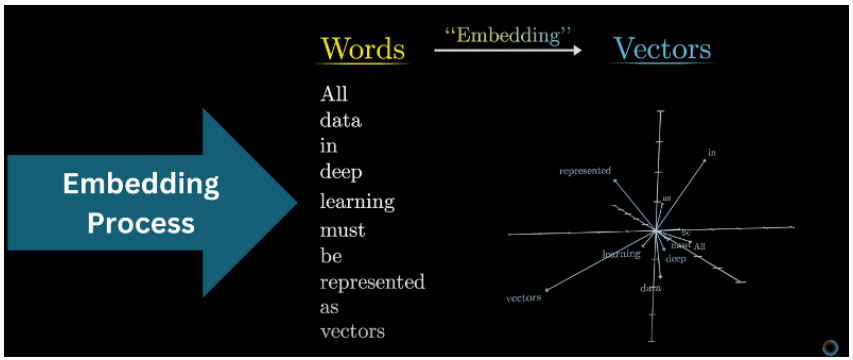
Embedding for Large Language Models (LLM)



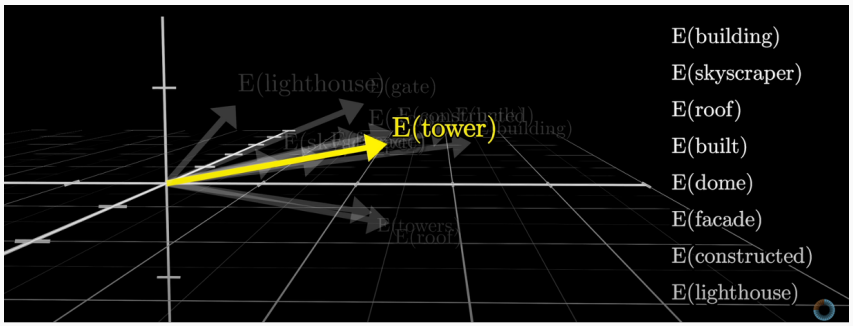


Embedding for Large Language Models (LLM)



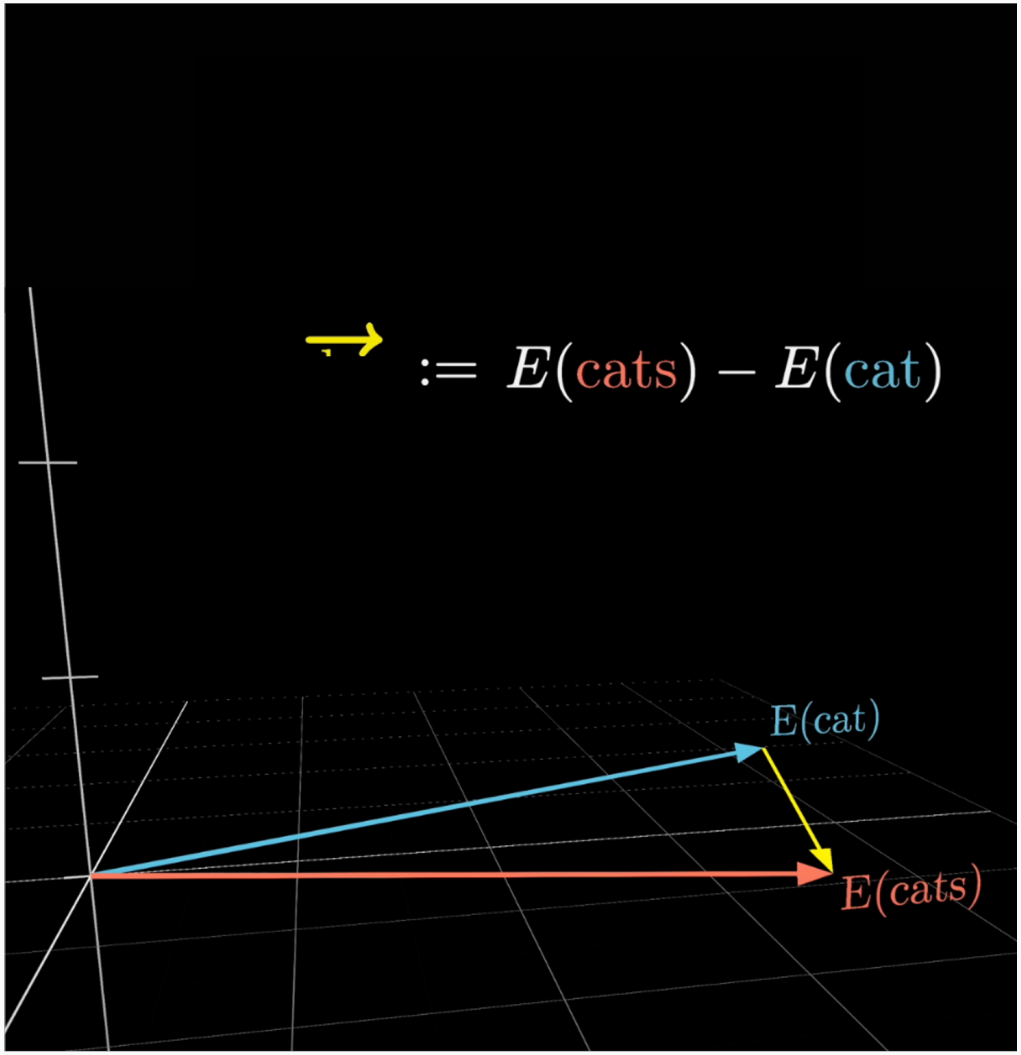


Embeddings of similar words are close:



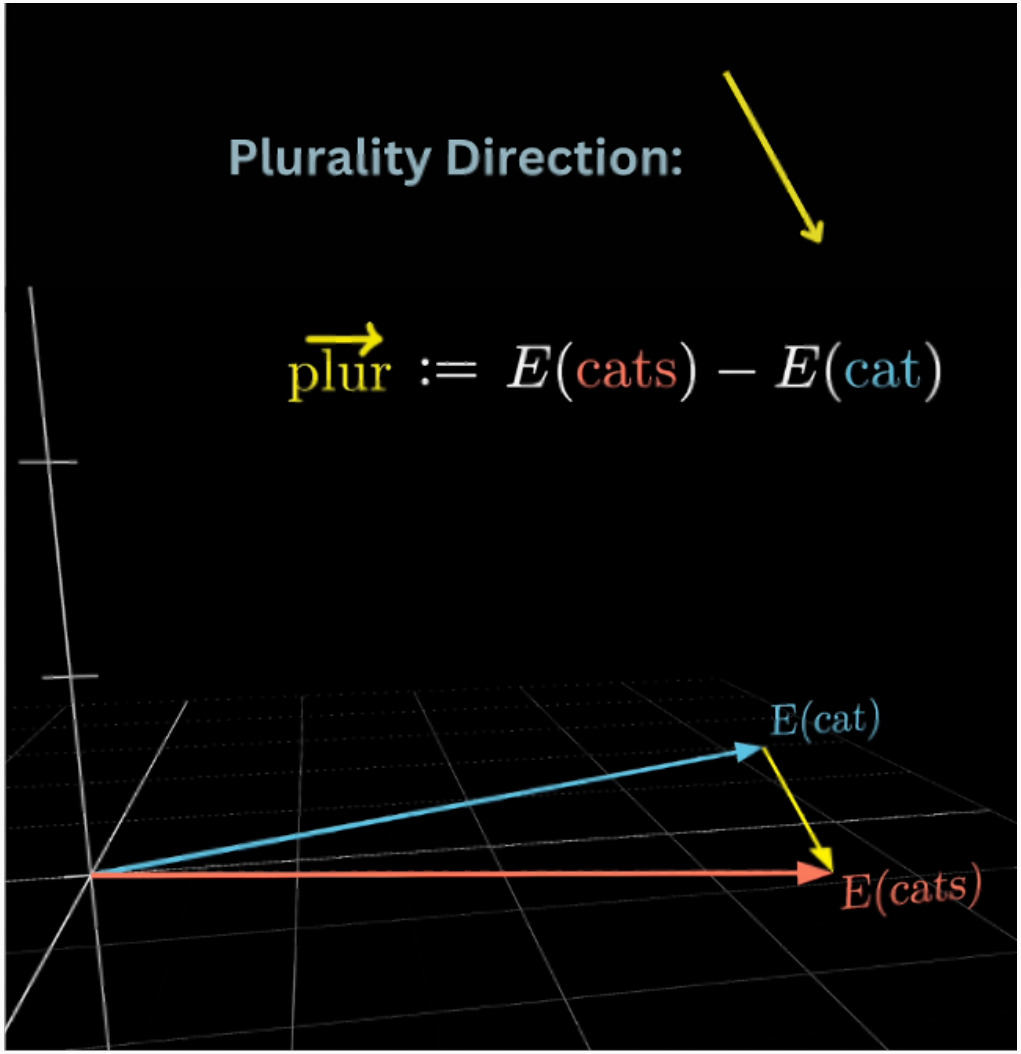
https://www.3blue1brown.com/lessons/gpt

The notion of direction



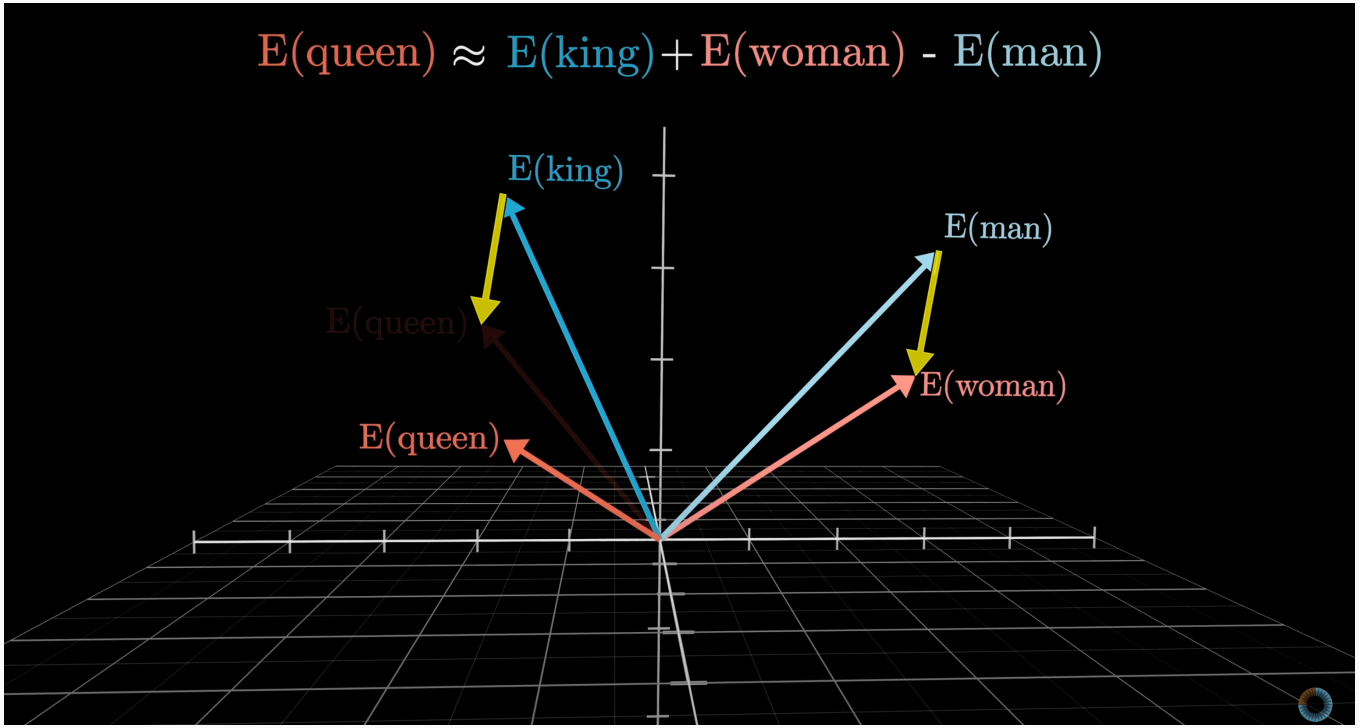
https://www.3blue1brown.com/lessons/gpt

The notion of direction

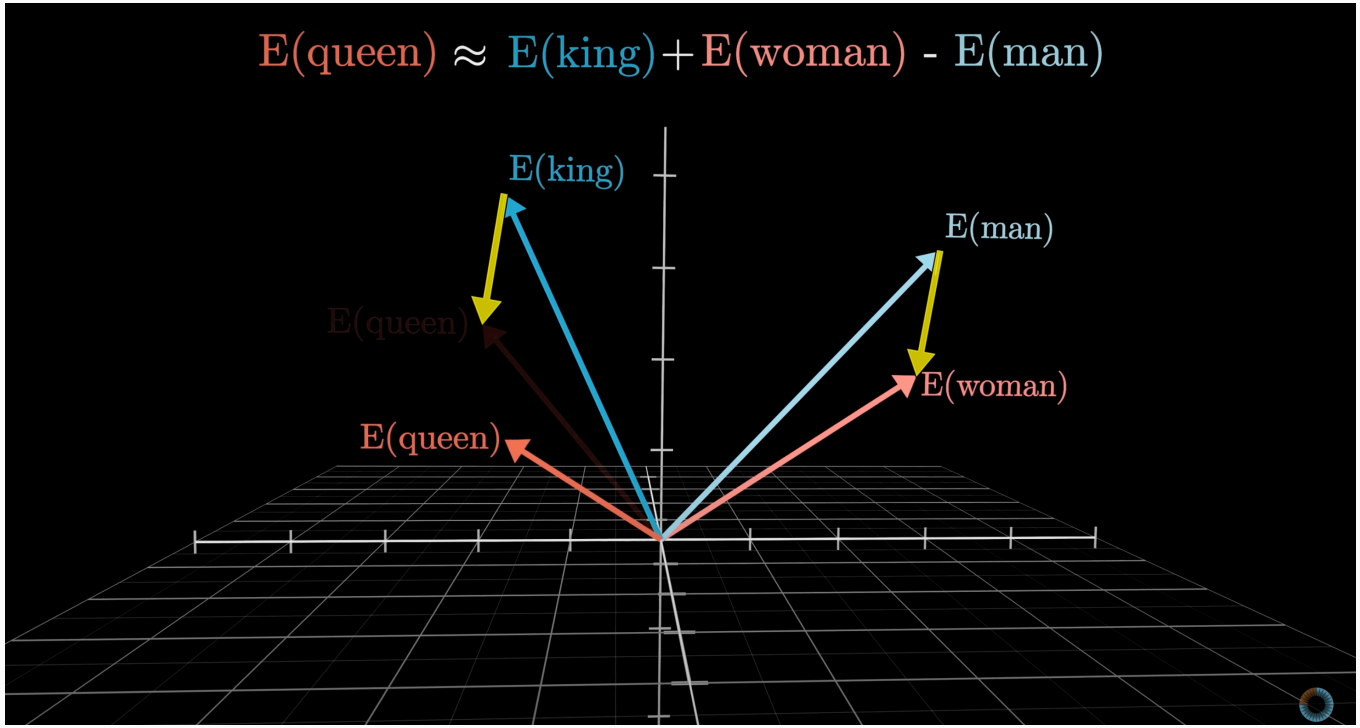


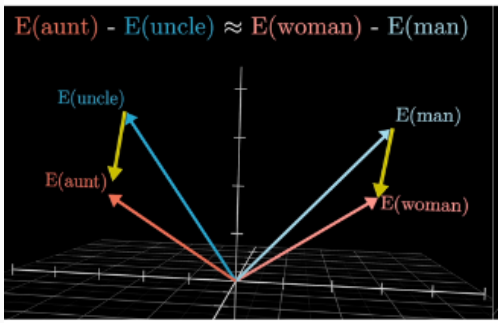
https://www.3blue1brown.com/lessons/gpt

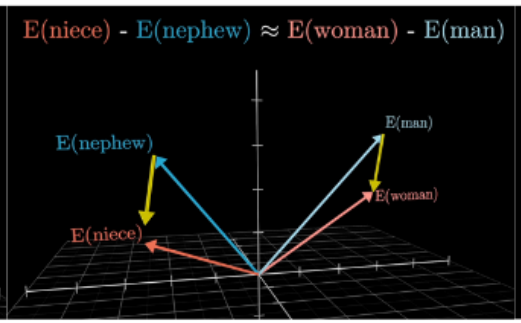
The space of embedding encodes some ”meaning”

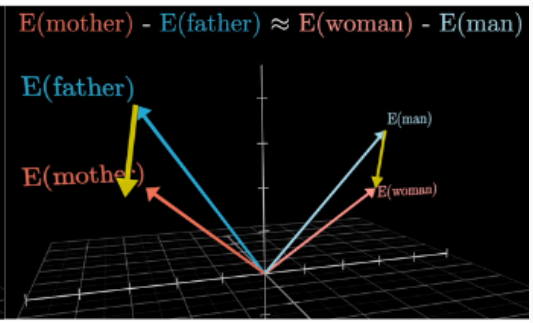


The space of embedding encodes some ”meaning”



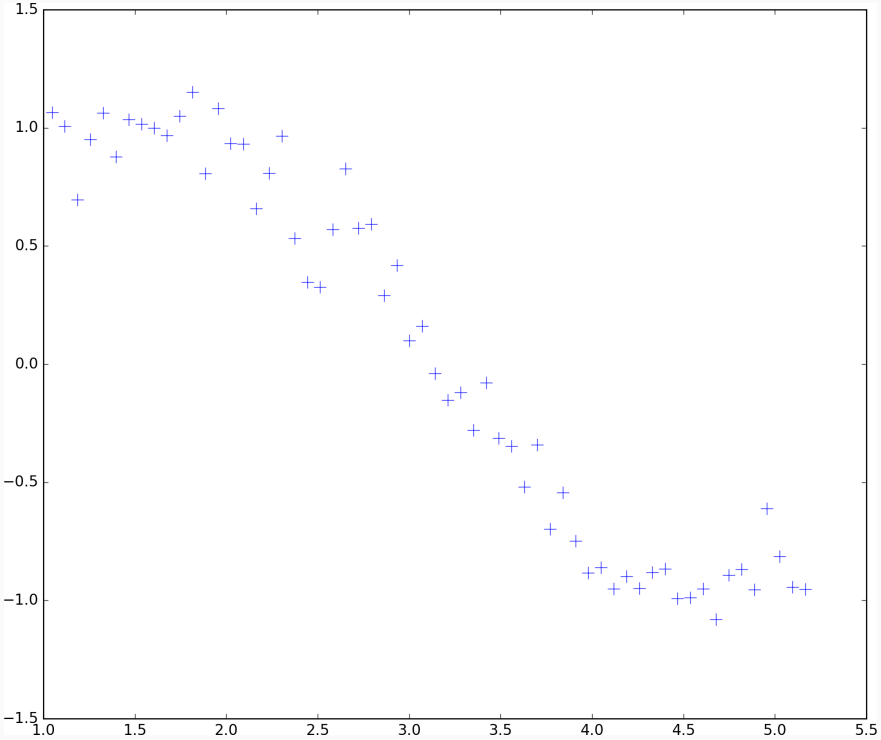






L1/L2 regularization

Example of a non-linear relationship



An idea

We could take a polynomial hypothesis model:

$$h_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \ldots + \theta_p x^p$$

Example

$\{(x_1, y_1), \ldots, (x_n, y_n)\}$ is the learning dataset.

For a given polynomial degree p, parameters $\boldsymbol{\theta}$ are determined minimizing the least-mean square cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum (y_i - h_{\boldsymbol{\theta}}(x_i))^2$$

with $h_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \ldots + \theta_p x^p$

- It can be determined using a gradient descent method

Example

$\{(x_1, y_1), \ldots, (x_n, y_n)\}$ is the learning dataset.

For a given polynomial degree p, parameters $\boldsymbol{\theta}$ are determined minimizing the least-mean square cost function:

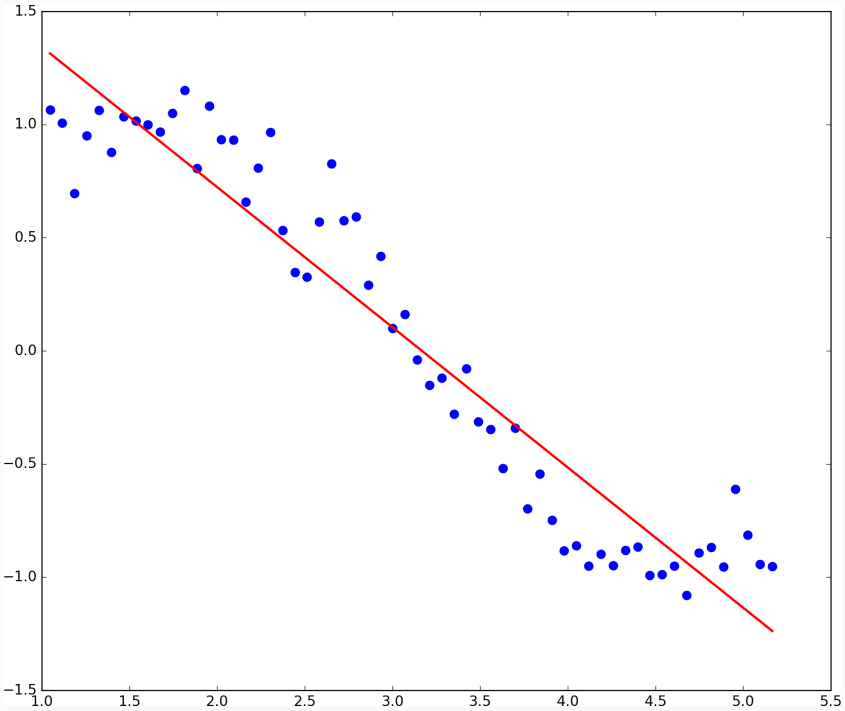
$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum (y_i - h_{\boldsymbol{\theta}}(x_i))^2$$

with $h_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \ldots + \theta_p x^p$

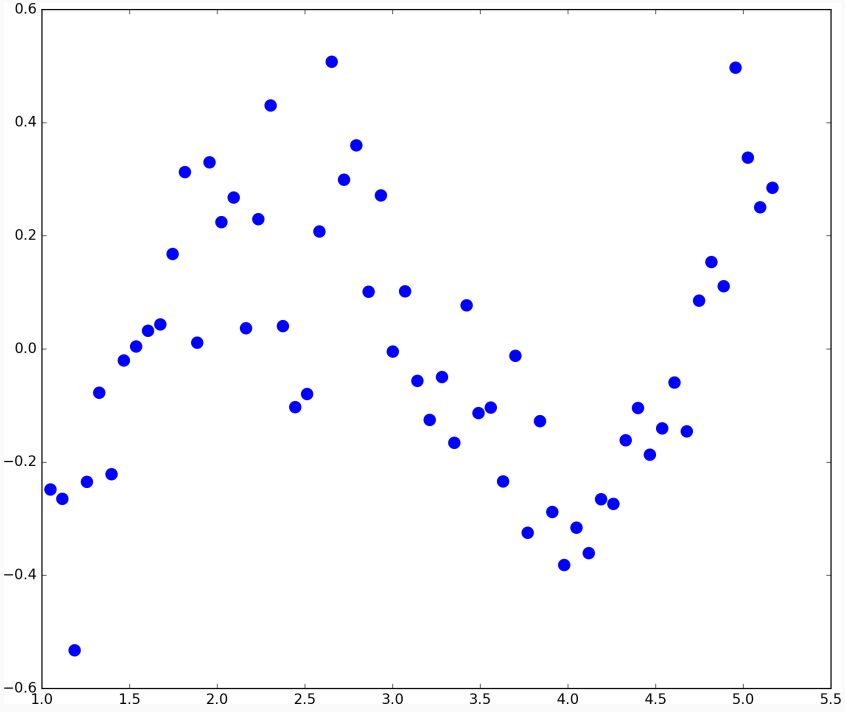
- It can be determined using a gradient descent method
- If the degree of the polynomial $p = 1$, it is a simple linear regression

A first result

$p = 1$ (linear regression)



residuals

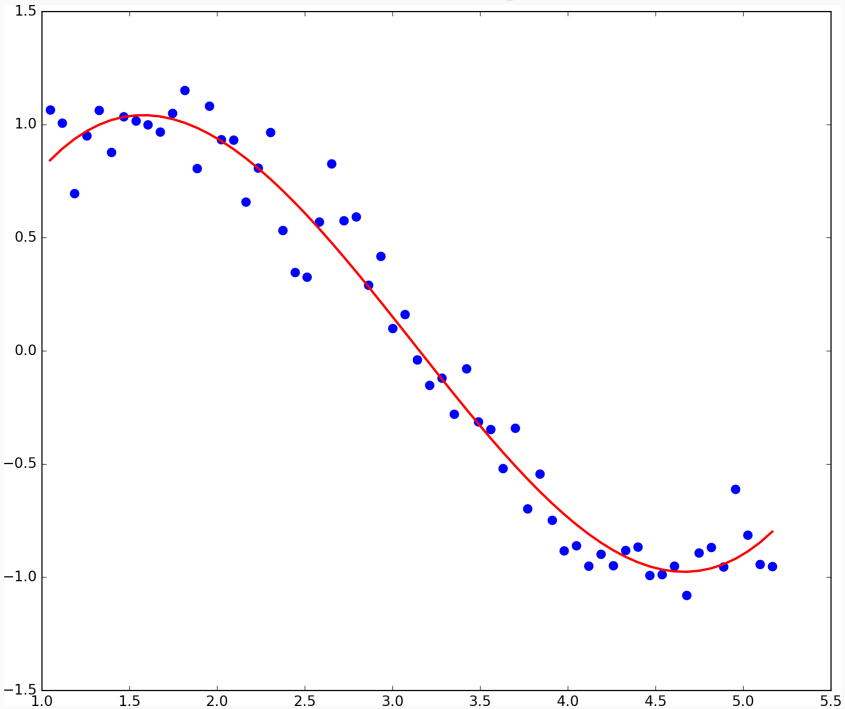


Prediction error

$$\text{err} = \frac{1}{n}\sum \text{res}^2 = 5.46e-2$$

Increasing the polynomial degree ?

$p = 3$ (cubic regression)



Prediction error

$$\text{err} = \frac{1}{n}\sum \text{res}^2 = 1.80e-2$$

Is it different from the linear regression ?

Let's consider :

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ x^1 \\ x^2 \\ \vdots \\ x^p \end{pmatrix}$$

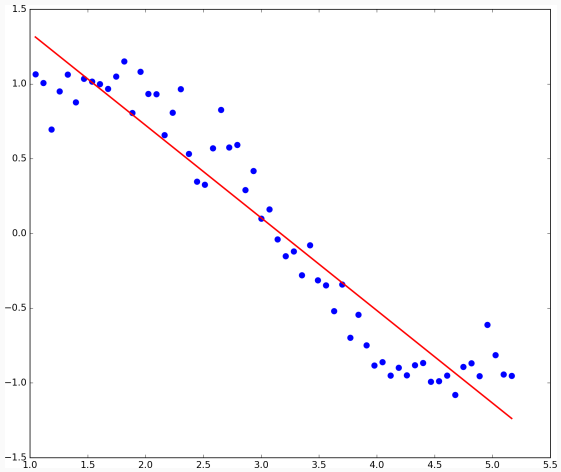
then

$$h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x^1 + \ldots + \theta_p x^p = \boldsymbol{\theta}^T \boldsymbol{x}$$

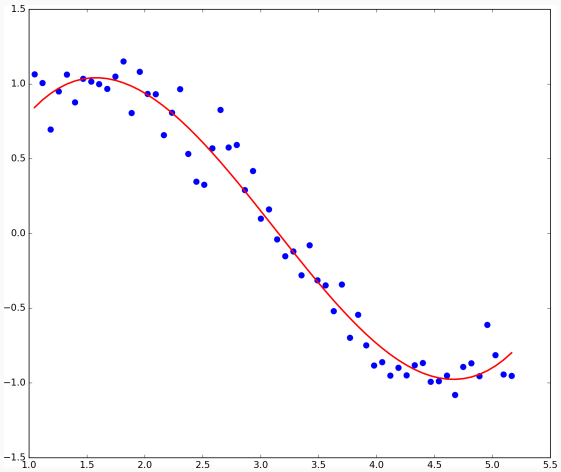
By extending a scalar predictor to a vector, polynomial regression is equivalent to linear regression.

Increasing the degree ?

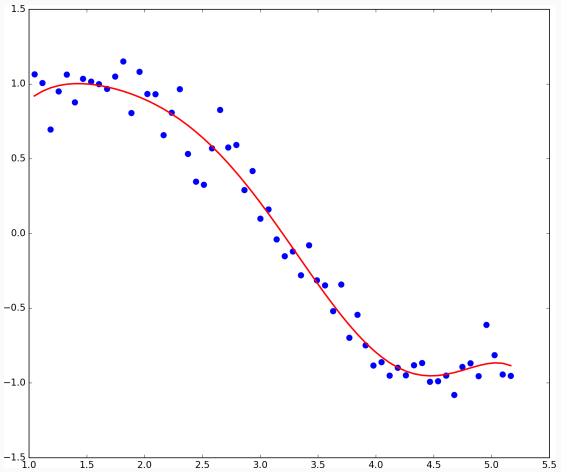
$p = 1$



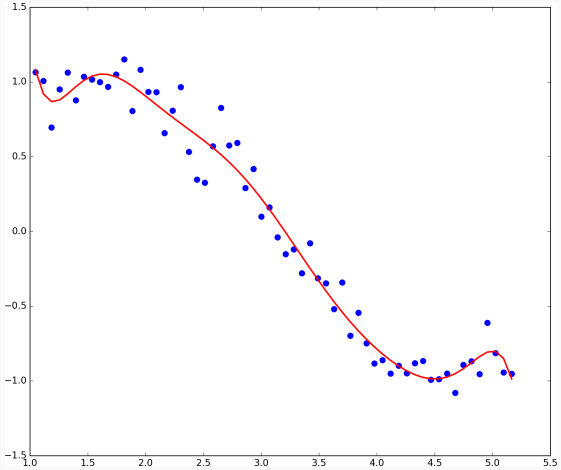
$p = 3$



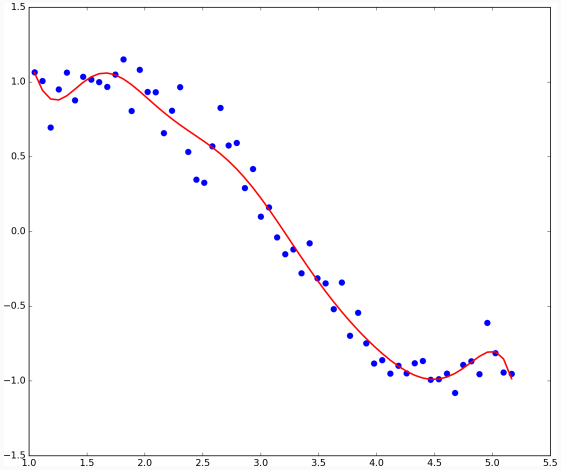
$p = 4$



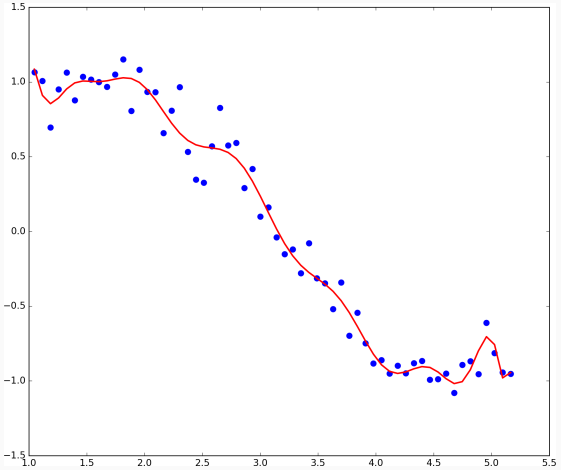
$p = 9$



$p = 12$



$p = 15$



Overfitting

When there are too many parameters to fit, the model can reproduce a random noise, it is called overfitting

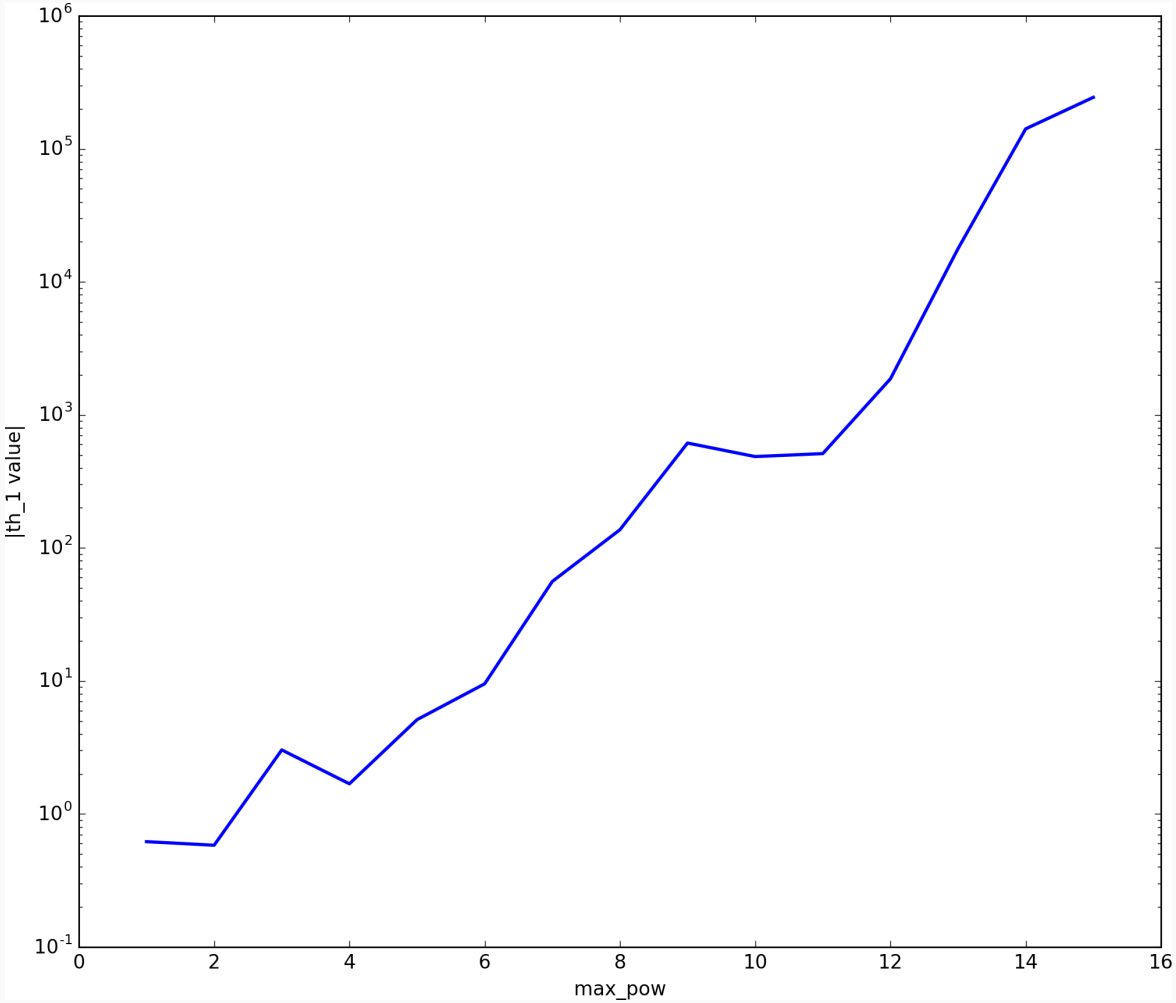
Overfitting

When there are too many parameters to fit, the model can reproduce a random noise, it is called overfitting

	rmse	th_0	th_1	th_2	th_3	th_4	th_5	th_6	th_7	th_8
max_pow_1	+5.47e-02	+1.96e+00	-6.20e-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max_pow_2	+5.46e-02	+1.91e+00	-5.83e-01	-5.96e-03	NaN	NaN	NaN	NaN	NaN	NaN
max_pow_3	+1.84e-02	-1.08e+00	+3.03e+00	-1.29e+00	+1.37e-01	NaN	NaN	NaN	NaN	NaN
max_pow_4	+1.80e-02	-2.66e-01	+1.69e+00	-5.32e-01	-3.57e-02	+1.39e-02	NaN	NaN	NaN	NaN
max_pow_5	+1.70e-02	+2.99e+00	-5.12e+00	+4.72e+00	-1.93e+00	+3.35e-01	-2.07e-02	NaN	NaN	NaN
max_pow_6	+1.65e-02	-2.80e+00	+9.52e+00	-9.71e+00	+5.23e+00	-1.55e+00	+2.33e-01	-1.36e-02	NaN	NaN
max_pow_7	+1.55e-02	+1.93e+01	-5.60e+01	+6.90e+01	-4.46e+01	+1.65e+01	-3.53e+00	+4.05e-01	-1.92e-02	NaN
max_pow_8	+1.53e-02	+4.32e+01	-1.37e+02	+1.84e+02	-1.33e+02	+5.77e+01	-1.53e+01	+2.42e+00	-2.10e-01	+7.68e-03
max_pow_9	+1.46e-02	+1.68e+02	-6.15e+02	+9.63e+02	-8.46e+02	+4.61e+02	-1.62e+02	+3.68e+01	-5.22e+00	+4.22e-01
max_pow_10	+1.46e-02	+1.38e+02	-4.86e+02	+7.26e+02	-5.96e+02	+2.93e+02	-8.75e+01	+1.45e+01	-8.06e-01	-1.38e-01
max_pow_11	+1.45e-02	-7.49e+01	+5.12e+02	-1.33e+03	+1.87e+03	-1.61e+03	+9.14e+02	-3.50e+02	+9.14e+01	-1.61e+01
max_pow_12	+1.45e-02	-3.39e+02	+1.87e+03	-4.42e+03	+6.01e+03	-5.25e+03	+3.12e+03	-1.30e+03	+3.84e+02	-8.03e+01
max_pow_13	+1.43e-02	+3.20e+03	-1.78e+04	+4.46e+04	-6.66e+04	+6.61e+04	-4.61e+04	+2.32e+04	-8.55e+03	+2.30e+03
max_pow_14	+1.31e-02	+2.38e+04	-1.41e+05	+3.79e+05	-6.10e+05	+6.57e+05	-5.03e+05	+2.82e+05	-1.17e+05	+3.66e+04
max_pow_15	+1.17e-02	-3.62e+04	+2.44e+05	-7.46e+05	+1.38e+06	-1.71e+06	+1.53e+06	-1.00e+06	+4.98e+05	-1.88e+05

High parameters values

Value of the parameters $|\theta_1|$ with respect with the degree of the polynomial regression



Regularization

The idea

The idea of regularization is to perform a regression minimizing a cost function that includes a term to penalize "big" values for the parameters:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum(y_i - h_{\boldsymbol{\theta}}(x_i))^2 + \alpha P(\boldsymbol{\theta})$$

Regularization

The idea

The idea of regularization is to perform a regression minimizing a cost function that includes a term to penalize "big" values for the parameters:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum(y_i - h_{\boldsymbol{\theta}}(x_i))^2 + \alpha P(\boldsymbol{\theta})$$

We consider two penalty terms:

- Ridge Regularization (L2): $P(\boldsymbol{\theta}) = \sum_{i=0}^{p} \theta_i^2$

Regularization

The idea

The idea of regularization is to perform a regression minimizing a cost function that includes a term to penalize "big" values for the parameters:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum(y_i - h_{\boldsymbol{\theta}}(x_i))^2 + \alpha P(\boldsymbol{\theta})$$

We consider two penalty terms:

- Ridge Regularization (L2): $P(\boldsymbol{\theta}) = \sum_{i=0}^{p} \theta_i^2$
- Lasso Regularization (L1): $P(\boldsymbol{\theta}) = \sum_{i=0}^{p} |\theta_i|$

Regularization

The idea

The idea of regularization is to perform a regression minimizing a cost function that includes a term to penalize "big" values for the parameters:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum(y_i - h_{\boldsymbol{\theta}}(x_i))^2 + \alpha P(\boldsymbol{\theta})$$

We consider two penalty terms:

- Ridge Regularization (L2): $P(\boldsymbol{\theta}) = \sum_{i=0}^{p} \theta_i^2$
- Lasso Regularization (L1): $P(\boldsymbol{\theta}) = \sum_{i=0}^{p} |\theta_i|$
- Elastic Net combines both regularizations

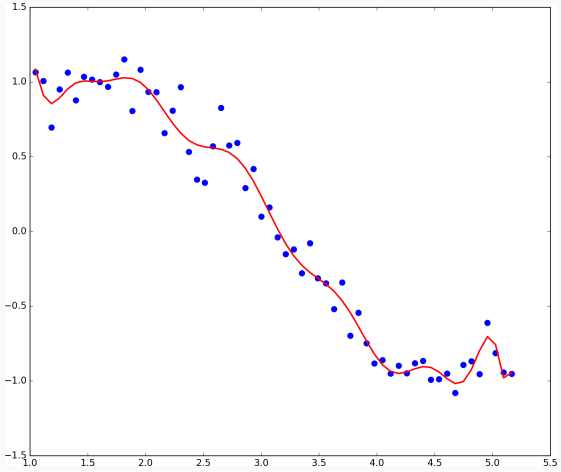
Ridge regression (L2)

Ridge regression is a linear regression with a Ridge regularization:

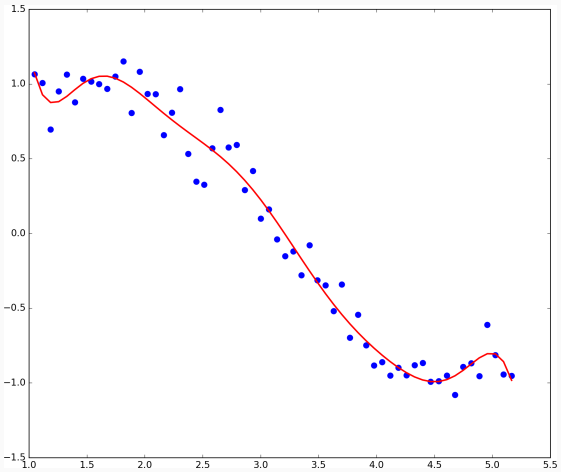
$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum(y_i - h_{\boldsymbol{\theta}}(x_i))^2 + \alpha\sum_{i=0}^{p}\theta_i^2$$

Results for $p = 15$ and varying α

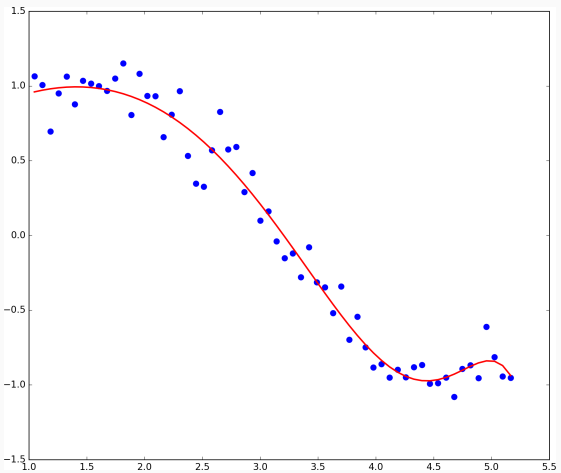
$\alpha = 0$



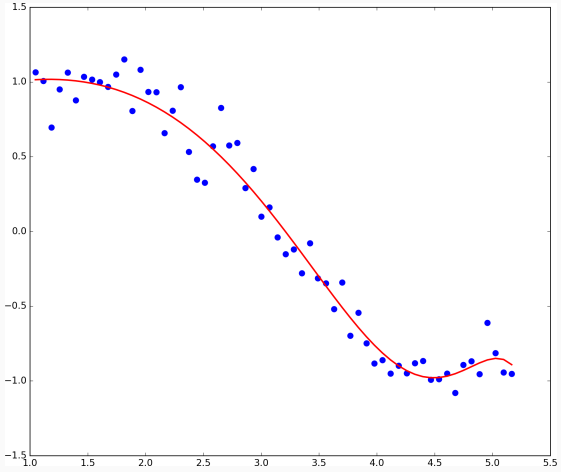
$\alpha = 1e-15$



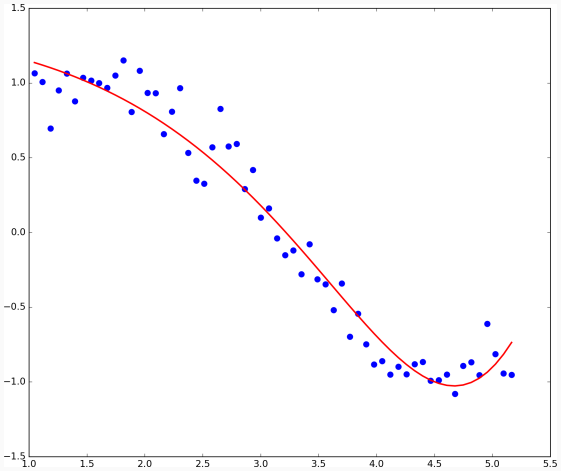
$\alpha = 1e-4$



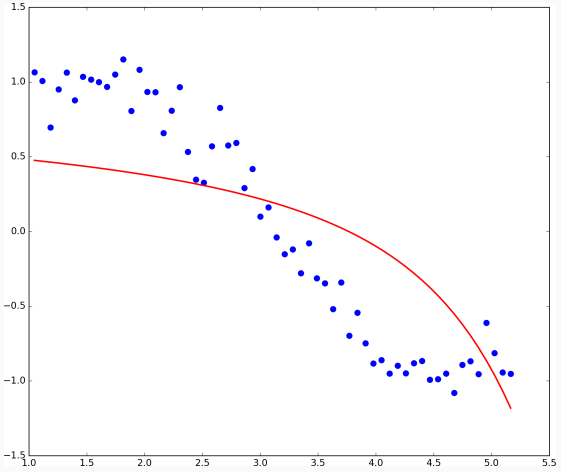
$\alpha = 1e-3$



$\alpha = 1e-2$

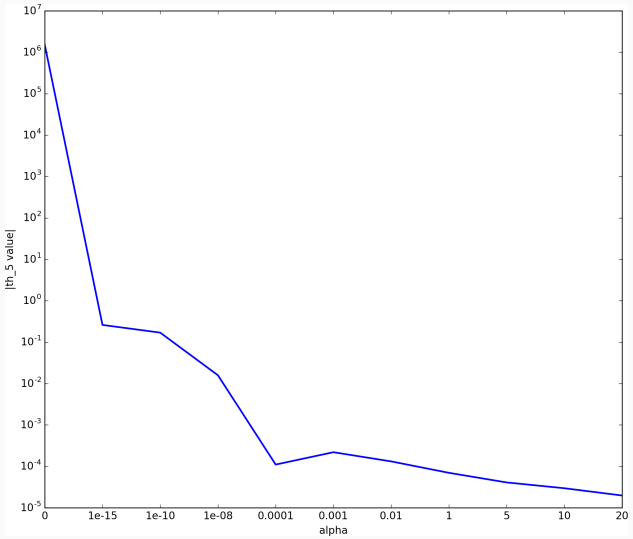


$\alpha = 5$



Values of coefficients

	rmse	th_0	th_1	th_2	th_3	th_4	th_5	th_6	th_7	th_8
alpha_0	+1.17e-02	-3.62e+04	+2.44e+05	-7.46e+05	+1.38e+06	-1.71e+06	+1.53e+06	-1.00e+06	+4.98e+05	-1.88e+05
alpha_1e-15	+1.46e-02	+9.43e+01	-2.98e+02	+3.79e+02	-2.37e+02	+6.72e+01	-2.60e-01	-4.35e+00	+5.62e-01	+1.42e-01
alpha_1e-10	+1.54e-02	+1.12e+01	-2.90e+01	+3.11e+01	-1.52e+01	+2.89e+00	+1.69e-01	-9.10e-02	-1.08e-02	+1.98e-03
alpha_1e-08	+1.58e-02	+1.34e+00	-1.53e+00	+1.75e+00	-6.80e-01	+3.88e-02	+1.58e-02	+1.59e-04	-3.60e-04	-5.37e-05
alpha_0.0001	+1.60e-02	+5.61e-01	+5.47e-01	-1.28e-01	-2.57e-02	-2.82e-03	-1.10e-04	+4.06e-05	+1.52e-05	+3.65e-06
alpha_0.001	+1.67e-02	+8.18e-01	+3.05e-01	-8.67e-02	-2.05e-02	-2.84e-03	-2.19e-04	+1.81e-05	+1.24e-05	+3.43e-06
alpha_0.01	+2.39e-02	+1.30e+00	-8.84e-02	-5.15e-02	-1.01e-02	-1.41e-03	-1.32e-04	+7.23e-07	+4.14e-06	+1.30e-06
alpha_1	+9.41e-02	+9.69e-01	-1.39e-01	-1.93e-02	-3.00e-03	-4.66e-04	-6.97e-05	-9.90e-06	-1.29e-06	-1.43e-07
alpha_5	+2.31e-01	+5.48e-01	-5.89e-02	-8.52e-03	-1.42e-03	-2.41e-04	-4.08e-05	-6.87e-06	-1.15e-06	-1.91e-07
alpha_10	+3.00e-01	+4.00e-01	-3.72e-02	-5.53e-03	-9.50e-04	-1.67e-04	-2.96e-05	-5.23e-06	-9.25e-07	-1.63e-07
alpha_20	+3.79e-01	+2.77e-01	-2.25e-02	-3.40e-03	-5.99e-04	-1.08e-04	-1.97e-05	-3.60e-06	-6.58e-07	-1.20e-07



Value of θ_5

Determination of the parameters in the ridge regression

Considering the cost function J:

$$J(\boldsymbol{\theta}) = J_{lms}(\boldsymbol{\theta}) + \alpha \sum_{i=0}^{p} \theta_i^2$$

In a gradient algorithm, update of the parameters:

$\theta_i^{k+1} = \theta_i^k - \nu.(\frac{\partial J_{lms}}{D\theta_i} - 2\alpha\theta_i^k)$

So the update rule is:

$$\theta_i^{k+1} = \theta_i^k(1 - 2\nu\alpha) - \Delta_{lms}$$

where Δ_{lms} is the update in case of non-regularized regression

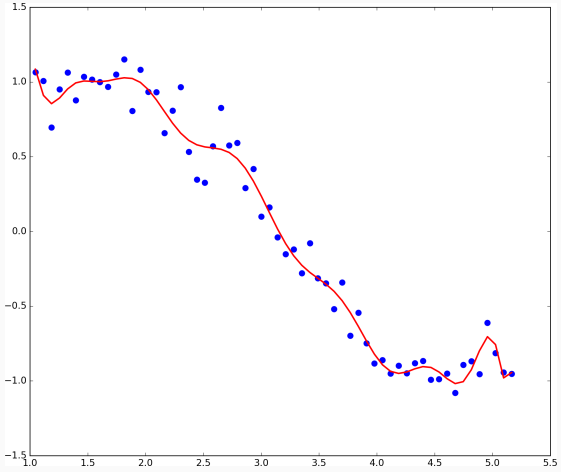
Lasso regression (L1)

Lasso regression is a linear regression with a Lasso regularization:

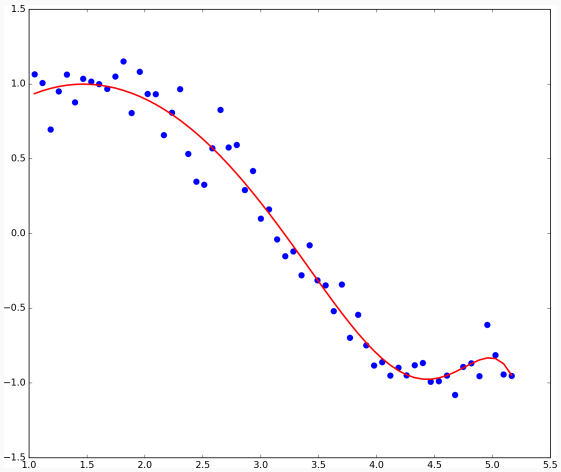
$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum(y_i - h_{\boldsymbol{\theta}}(x_i))^2 + \alpha\sum_{i=0}^{p}|\theta_i|$$

Results for $p = 15$ and varying α

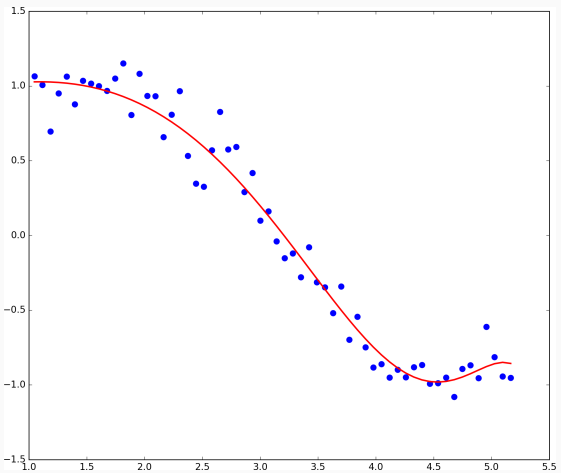
$\alpha = 0$



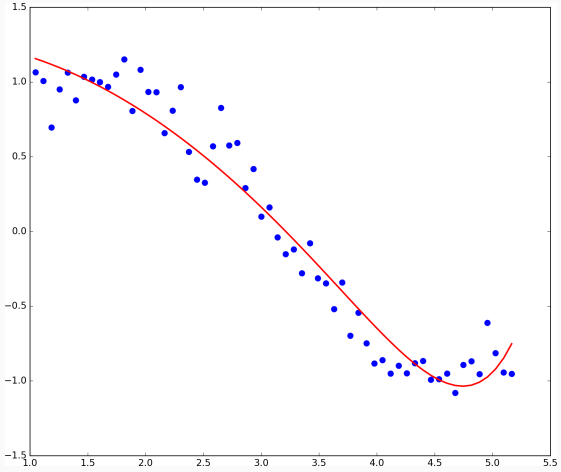
$\alpha = 1e-15$



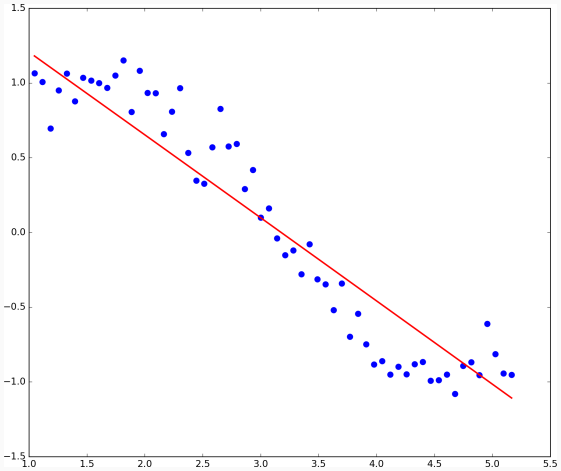
$\alpha = 1e-4$



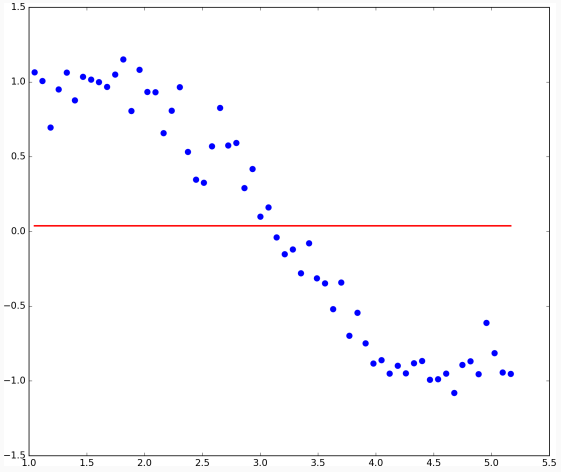
$\alpha = 1e-3$



$\alpha = 1e-2$

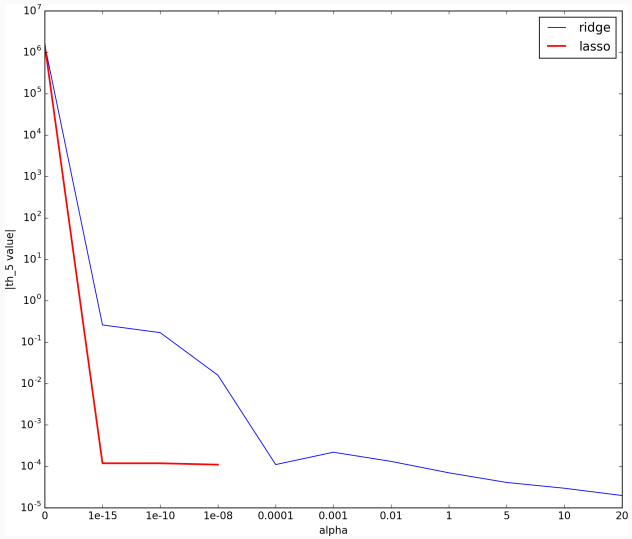


$\alpha = 5$



Values of coefficients

	rmse	th_0	th_1	th_2	th_3	th_4	th_5	th_6	th_7	th_8
alpha_0	+1.17e-02	-3.62e+04	+2.44e+05	-7.46e+05	+1.38e+06	-1.71e+06	+1.53e+06	-1.00e+06	+4.98e+05	-1.88e+05
alpha_1e-15	+1.59e-02	+2.22e-01	+1.06e+00	-3.69e-01	+8.85e-04	+1.63e-03	-1.19e-04	-6.44e-05	-6.28e-06	+1.45e-06
alpha_1e-10	+1.59e-02	+2.22e-01	+1.06e+00	-3.69e-01	+8.84e-04	+1.63e-03	-1.18e-04	-6.44e-05	-6.28e-06	+1.45e-06
alpha_1e-08	+1.59e-02	+2.22e-01	+1.06e+00	-3.69e-01	+7.69e-04	+1.62e-03	-1.10e-04	-6.45e-05	-6.32e-06	+1.43e-06
alpha_0.0001	+1.72e-02	+9.03e-01	+1.71e-01	-0.00e+00	-4.78e-02	-0.00e+00	-0.00e+00	+0.00e+00	+0.00e+00	+9.47e-06
alpha_0.001	+2.80e-02	+1.29e+00	-0.00e+00	-1.26e-01	-0.00e+00	-0.00e+00	-0.00e+00	+0.00e+00	+0.00e+00	+0.00e+00
alpha_0.01	+6.07e-02	+1.76e+00	-5.52e-01	-5.62e-04	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00
alpha_1	+6.16e-01	+3.80e-02	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00
alpha_5	+6.16e-01	+3.80e-02	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00
alpha_10	+6.16e-01	+3.80e-02	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00
alpha_20	+6.16e-01	+3.80e-02	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00	-0.00e+00



Value of θ_5

Determination of the parameters in the lasso regression

Considering the cost function J:

$$J(\boldsymbol{\theta}) = J_{lms}(\boldsymbol{\theta}) + \alpha \sum_{i=0}^{p} |\theta_i|$$

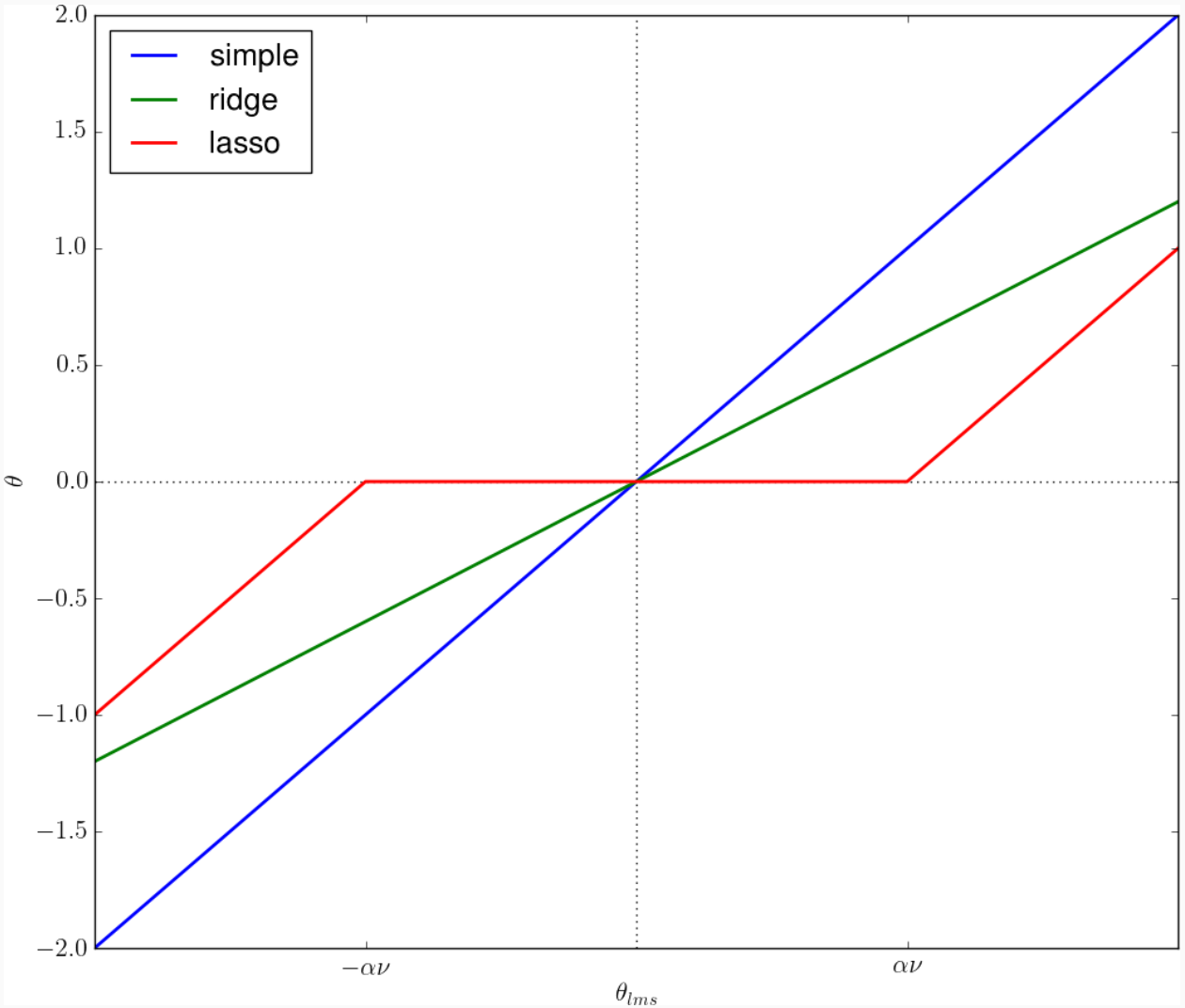
In a gradient algorithm, update of the parameters would be:

J is not differentiable. If we consider $\theta_{lms} = \theta_i^k - \nu . \frac{\partial J_{lms}}{D\theta_i}$

The update rule is:

- $\theta_i^{k+1} = \theta_{lms} + \alpha\nu$ if $\theta_{lms} < -\alpha\nu$
- $\theta_i^k = 0$ if $-\alpha\nu < \theta_{lms} < \alpha\nu$
- $\theta_i^{k+1} = \theta_{lms} - \alpha\nu$ if $\theta_{lms} > \alpha\nu$

Summary of parameters updates



Comparison Ridge/Lasso

Ridge (L2)

- Prevents the overfitting
- includes all the features (dimensions) of the predictor, so it can be useless for high-dimensional predictors

Comparison Ridge/Lasso

Ridge (L2)

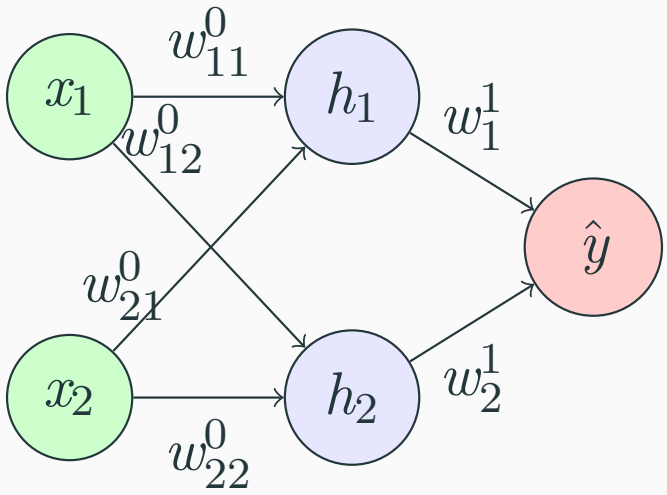
- Prevents the overfitting
- includes all the features (dimensions) of the predictor, so it can be useless for high-dimensional predictors

Lasso (L1)

- Provides sparse solutions and reduces the dimension of the predictor
- If some features in the predictor are correlated, arbitrarily select one from the others.

In a neural network

L2 (Ridge) and L1 (Lasso) regularization are widely used in Neural Network architectures.



- Non-regularized loss function: $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
- L2-regularized loss function:
  $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2 + \alpha \sum_{i=0}^{p} |w_i|$.
- L1-regularized loss function:
  $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2 + \alpha \sum_{i=0}^{p} w_i^2$.

Wrapping-up

Advantage

L1/L2 regularization prevents overfitting even for large architecture (a big weight vector to optimize)

Wrapping-up

Advantage

L1/L2 regularization prevents overfitting even for large architecture (a big weight vector to optimize)

But...

It comes with an extra hyperparameter to tune (using, e.g., cross-validation): α

An example using scikit-learn

California House Pricing