

# Machine learning and physical (Earth system) modelling

---

julien.brajard@nersc.no

June 2021

NERSC

slides+notebook:<https://github.com/nansencenter/ml-crashcourse>

# References

-  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.

# Table of contents

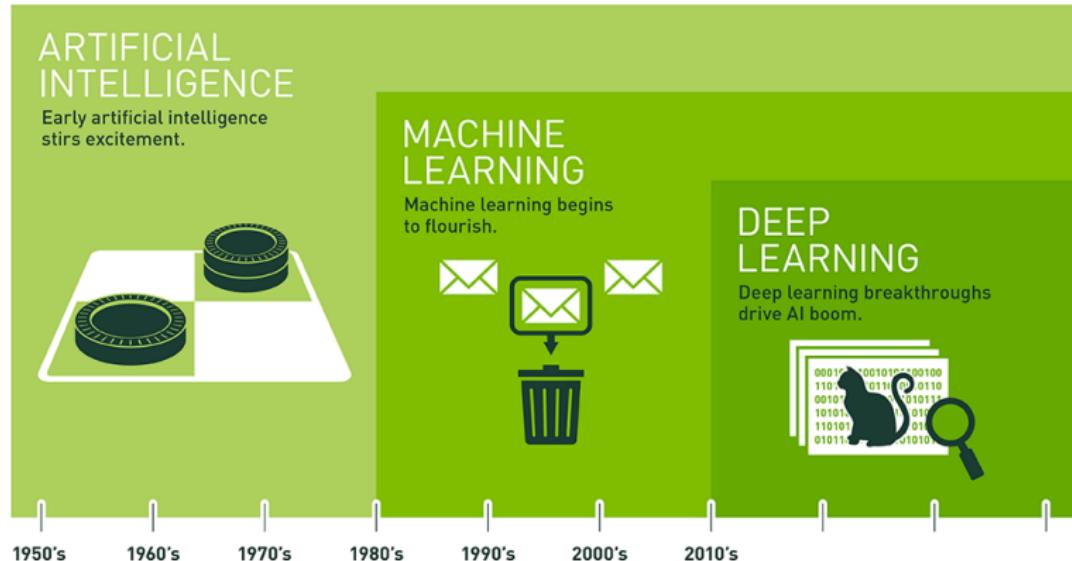
---

1. Introduction
2. Generalities on Machine Learning
3. Model selection/validation
4. Steps of a machine learning process
5. Neural Networks
6. Gradient backpropagation
7. Optimizing a machine learning (gradient method)

# Introduction

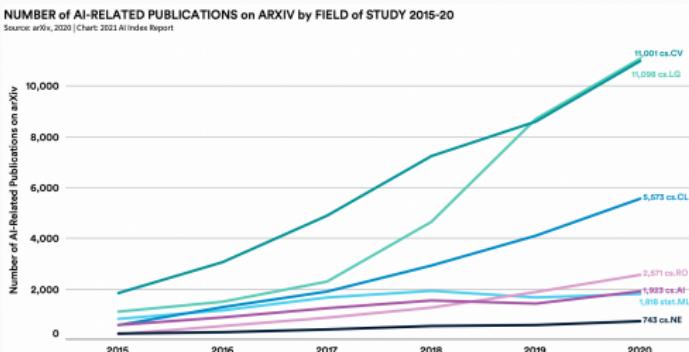
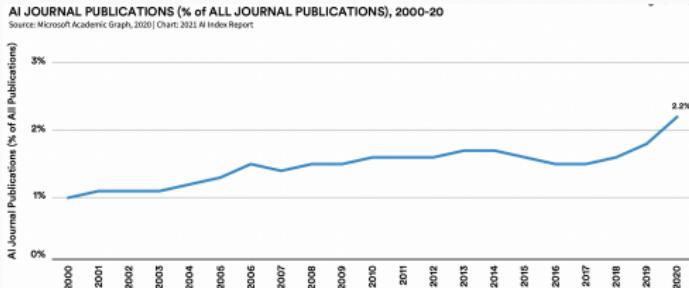
---

# Scope of the lecture: Machine Learning



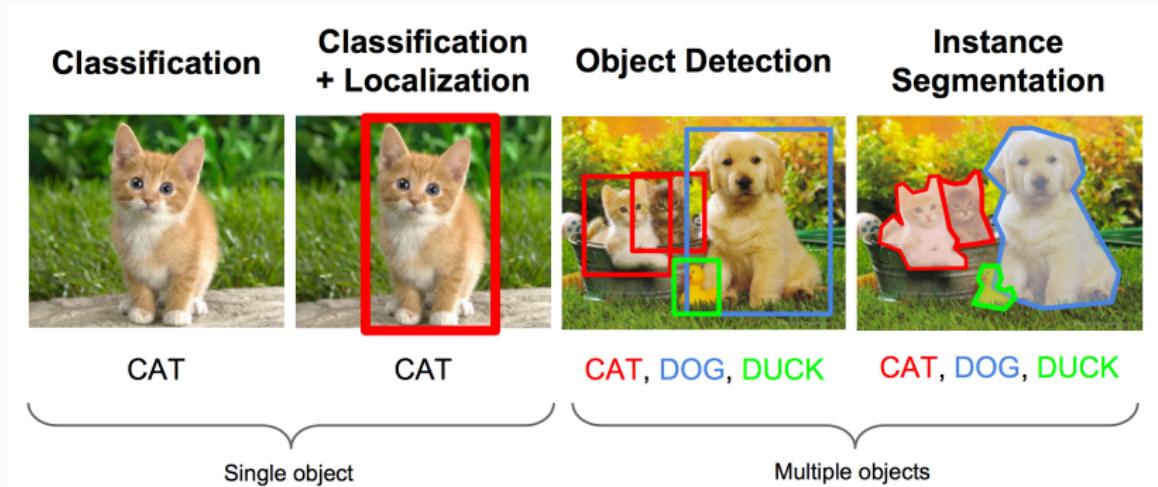
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

# A (very) active field



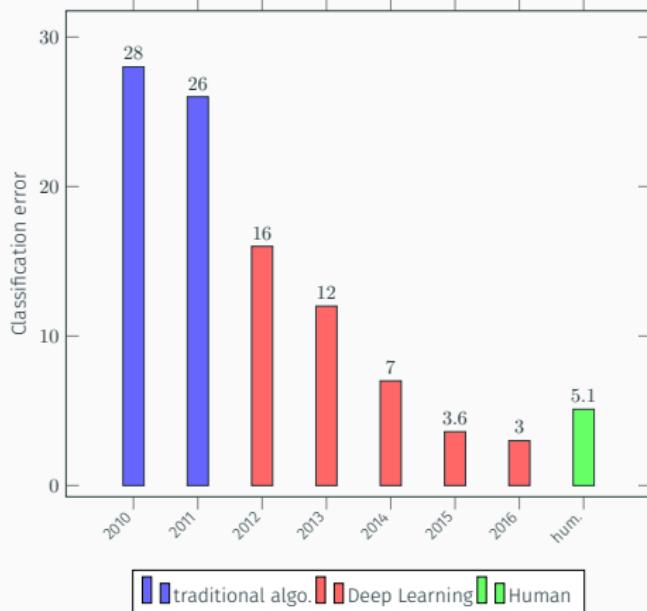
Zhang et al., "The AI Index 2021 Annual Report"

# Example 1: Computer Vision



*Li, Karpathy and Johnson, 2016, Stanford CS231n course*

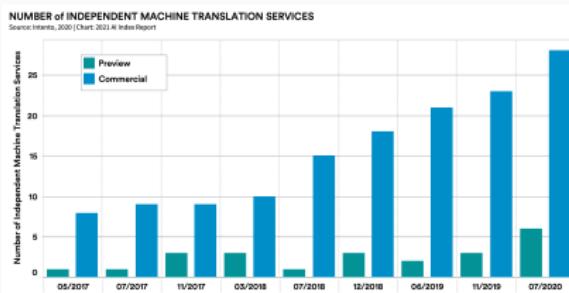
## Example 1: Computer Vision



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

## Example 2: Machine Translation

Objective : translate a text from a language to another.



Zhang et al., "The AI Index 2021 Annual Report"

- Oct. 2013: Pionneering scientific paper (Kalchbrenner, N., and Blunsom, P).
- 2016: Neural machine translation outperform traditional approaches on public benchmarks
- 2017: Major systems switch to neural machine translation (using deep recurrent neural networks)

## Example 3: Playing Games

- 1997: Deep Blue defeats Kasparov at Chess.
- 2016: AlphaGo's victory again Lee Sedol at Go.
- 2017: AlphaGo 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.



# AI Art?

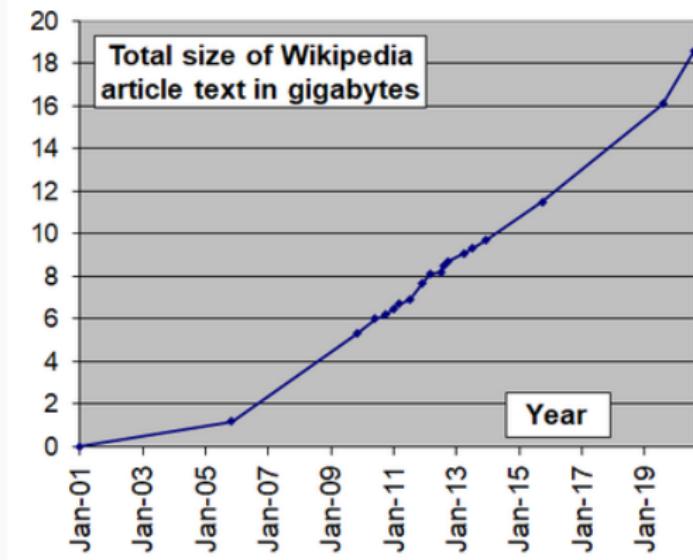


*Edmond de Bellamy* by Obvious(collective)

Generated using a Generative Adversarial Network.  
Selling price (Oct. 2018): \$432,000

## Reasons for these recent achievements?

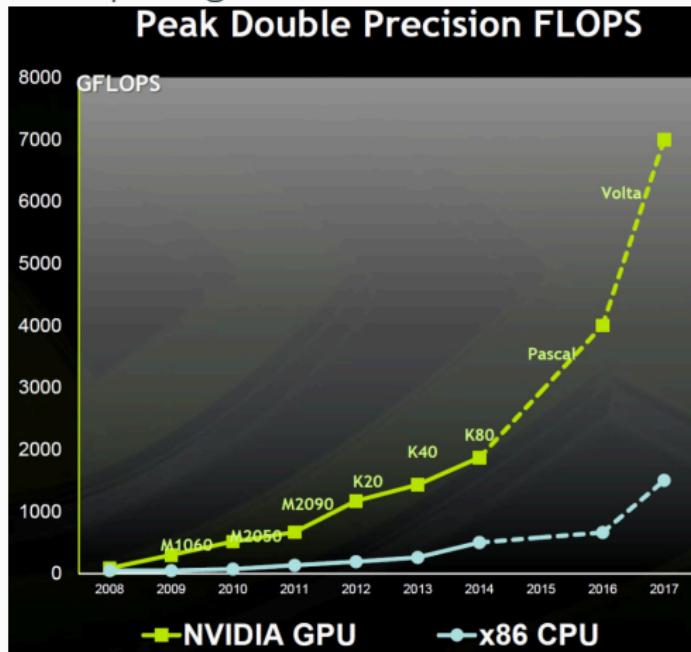
- Increasing of the datasets in size and quality



source: *Wikipedia*

# Reasons for these recent achievements?

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



# 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?

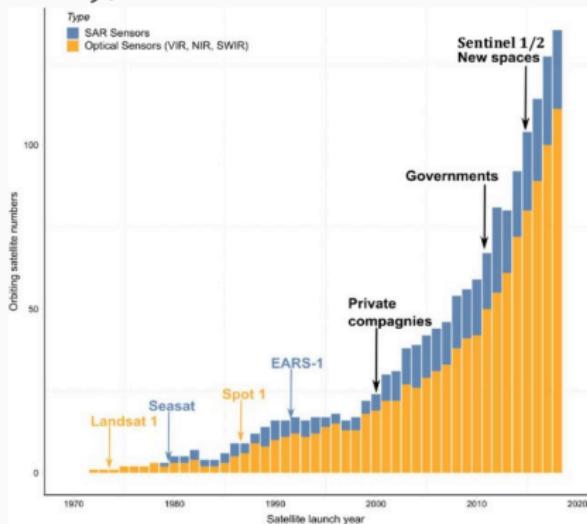
- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)
- Very efficient software (GPU, cloud computing, automatic differentiation, ...)
- Free software and open data culture.



# 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)

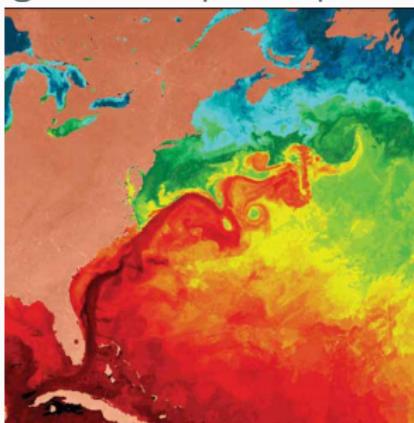


Earth System observavtion satellites

# 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

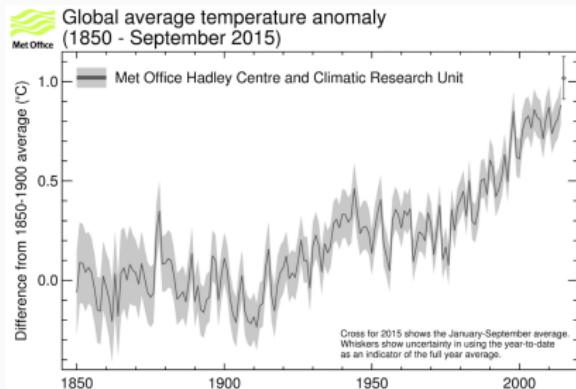


Sea Surface temperature of the gulf stream

source: *Talley (2000)*

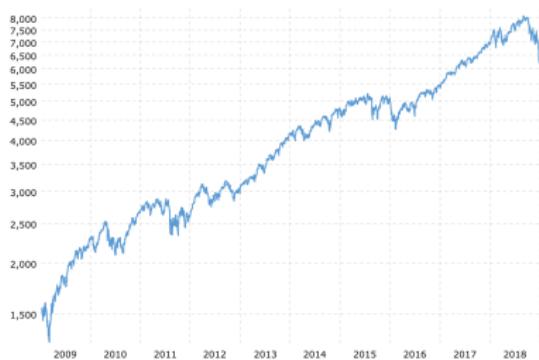
# Why is physical modelling specific?

NASDAQ Composite stock market index over the last 10 years

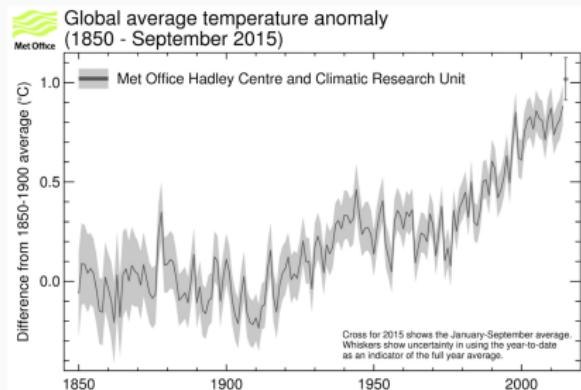


# Why is physical modelling specific?

NASDAQ Composite sock market index over the last 10 years



Mostly unknown dynamical processes



Mostly known dynamical processes (based on physical principles)

# What about data assimilation?

Machine learning and data assimilation are closely linked.

Some references:

- 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).

During this summer school:

Rossella Arcucci talk on Friday June 4th, 9:40 (CEST).

# 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 ?



→  $\text{Digit} \in \{0, \dots, 9\}$

Base of images

# What is this about ?

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



→ Digit ∈ {0, ..., 9}

Base of images

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

# Two classes of Machine 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 Machine 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
  - ...

## Two types of objectives

---

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

## Two types of objectives

---

1. **Supervised learning**: we have a set of labeled data with examples of targets.
2. **Unsupervised learning**: we only have unlabeled 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 objectives

1. **Supervised learning**: we have a set of labeled data with examples of targets.
2. **Unsupervised learning**: we only have unlabeled 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 behaviors of clients in a supermarket knowing what they have bought.

## Two types of objectives

1. **Supervised learning**: we have a set of labeled data with examples of targets.
2. **Unsupervised learning**: we only have unlabeled 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 behaviors of clients in a supermarket knowing what they have bought.
3. **Semi-Supervised Learning**: Only a few subset of the data are labeled

## Two types of objectives

1. **Supervised learning:** we have a set of labeled data with examples of targets.
2. **Unsupervised learning:** we only have unlabeled 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 behaviors of clients in a supermarket knowing what they have bought.
3. **Semi-Supervised Learning:** Only a few subset of the data are labeled
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 objectives

1. **Supervised learning:** we have a set of labeled data with examples of targets.
2. **Unsupervised learning:** we only have unlabeled 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 behaviors of clients in a supermarket knowing what they have bought.
3. **Semi-Supervised Learning:** Only a few subset of the data are labeled
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.

## A Machine

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

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

Machine learning consists in optimizing  $\theta$  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
  - $x$  and some subset of  $y$ :  
semi-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
  - $x$  and some subset of  $y$ : semi-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
  - $x$  and some subset of  $y$ : semi-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

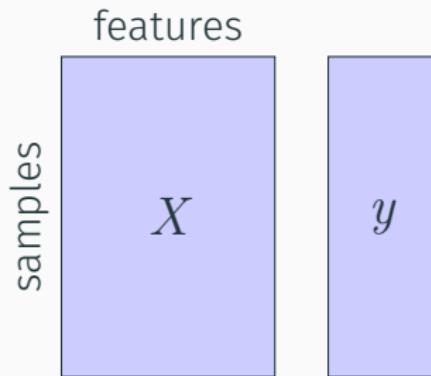
$$y = \mathcal{M}(x, \theta)$$

What are **the ingredients?**

- Some **data**
  - $x, y$ : supervised learning
  - only  $x$ : unsupervised learning
  - $x$  and some subset of  $y$ : semi-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  $\theta$

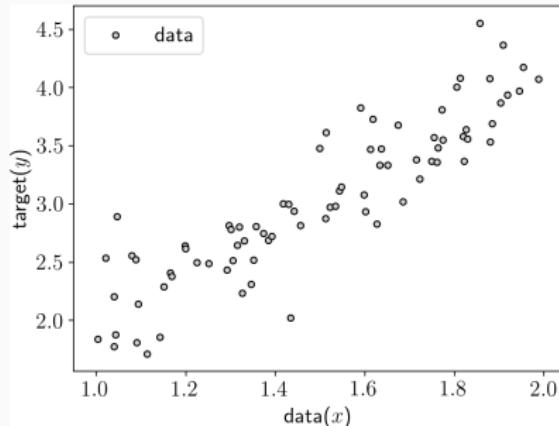
# Multidimensional data

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



# An illustration

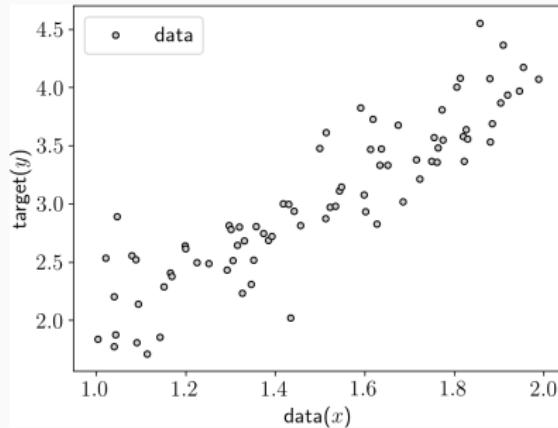
- Some Data



- there are labeled  $y$ : supervised learning
- $y$  is quantitative: regression

# An illustration

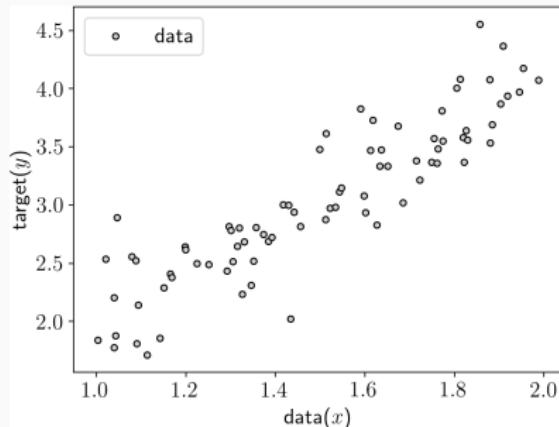
- Some Data



- there are labeled  $y$ : 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

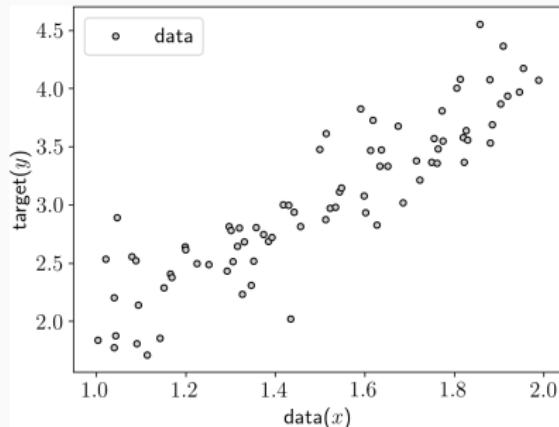


- A model:  $y = \theta_1 X + \theta_0$  (linear)

- there are labeled  $y$ : 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

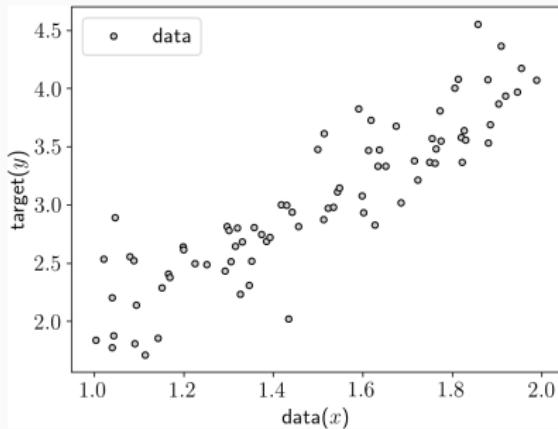


- A **model**:  $y = \theta_1 X + \theta_0$  (linear)
- A **learning** process:  
$$\theta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y$$

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

# An illustration

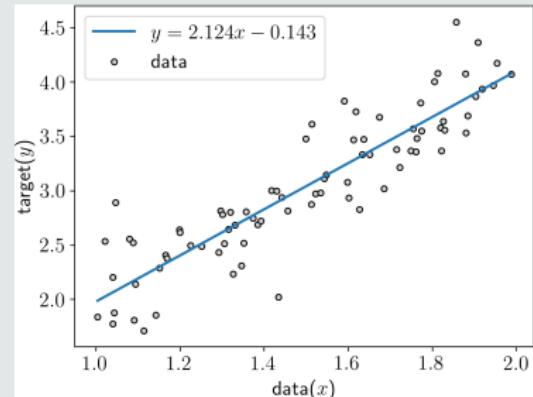
- Some Data



- there are labeled  $y$ : 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

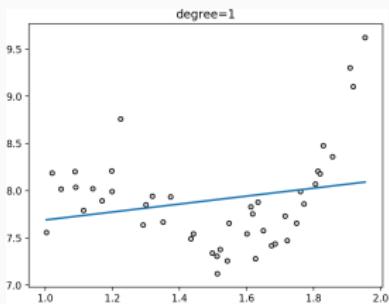
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^d \theta_i X^i$$

# 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)

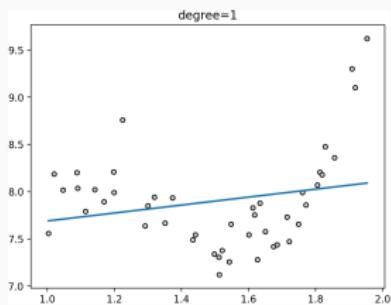


# Choice of the model

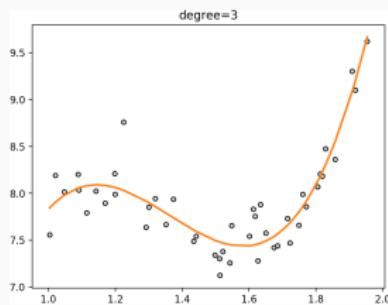
## Polynomial regression

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

degree = 1 (linear)



degree = 3

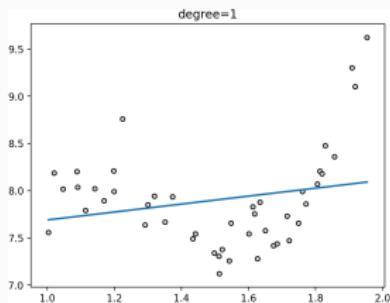


# Choice of the model

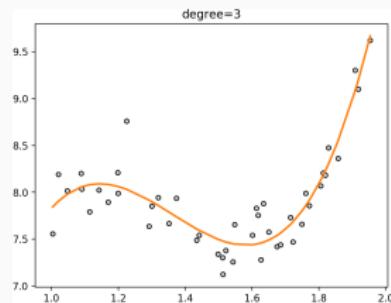
## Polynomial regression

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

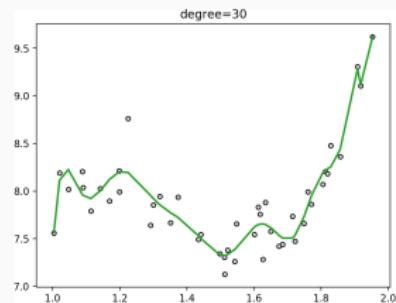
degree = 1 (linear)



degree = 3



degree = 30



What is the best model?

# Train/Validation split

## The idea

Evaluate a score on a independent dataset

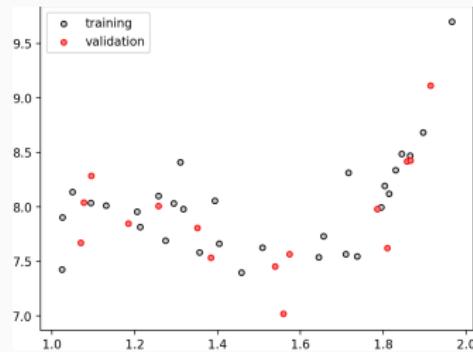
# Train/Validation split

## The idea

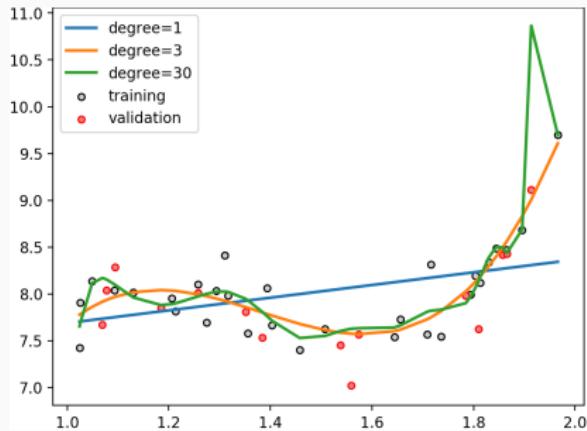
Evaluate a score on a independent dataset

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)



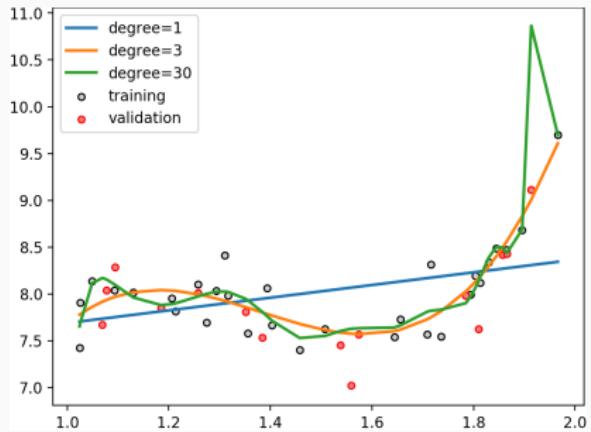
# Choice of the model



Score: Mean Square Error (MSE)

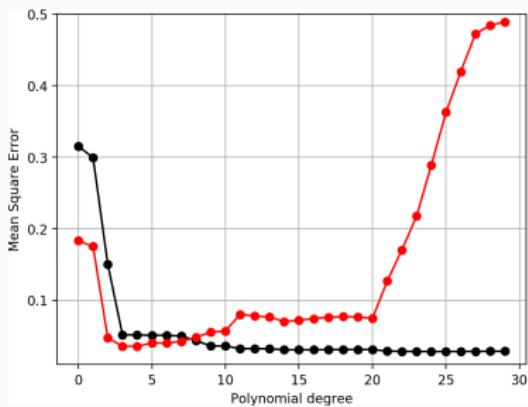
Deg.	Train Score	Val. Score
1	0.17	0.23
3	0.045	0.062
30	0.035	0.27

# 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

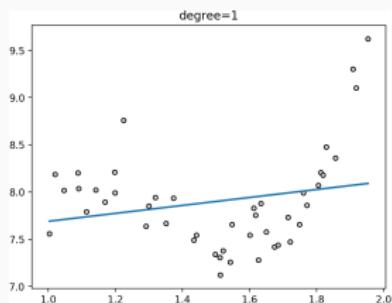


# Choice of the model

## Polynomial regression

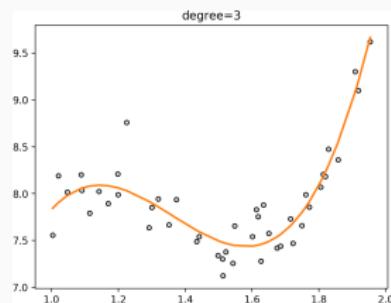
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^d \theta_i X^i$$

degree = 1 (linear)



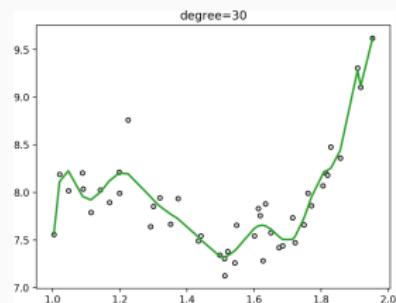
underfitting

degree = 3



good fit

degree = 30



overfitting

# Train/Validation split

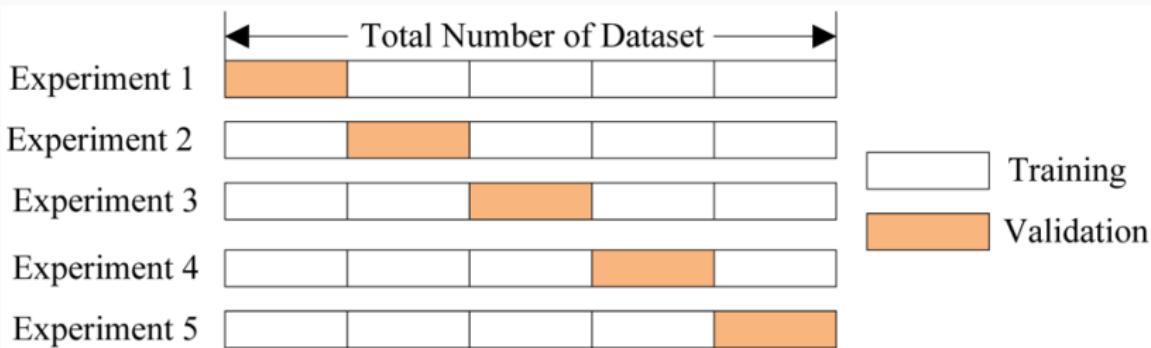
## Drawbacks

- drastically reduce the number of samples which 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

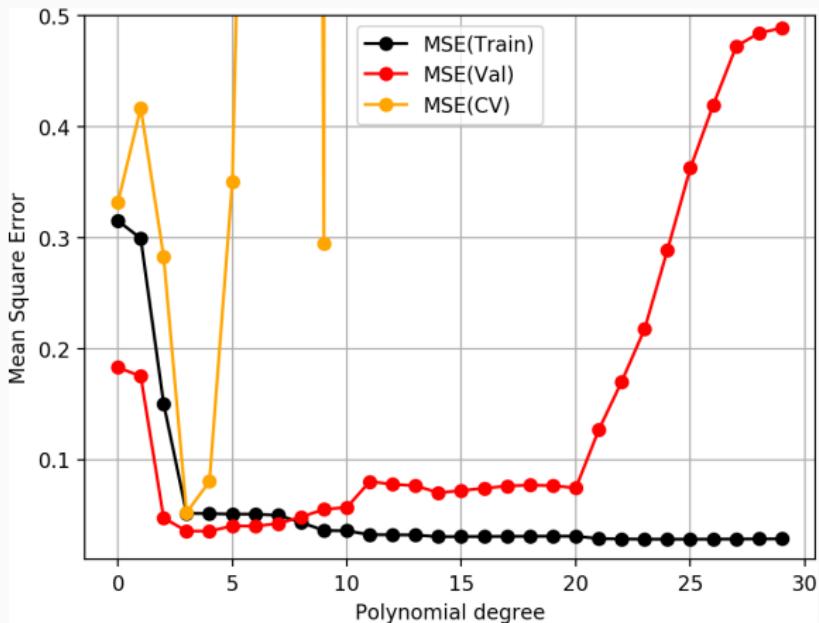
## The idea

- 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 independantly the performance of our model, we should compute the score on a **third independant dataset: The test dataset.**

## Steps of a machine learning process

---

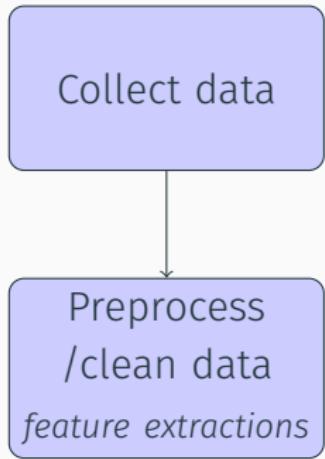
# Steps

---

Collect data

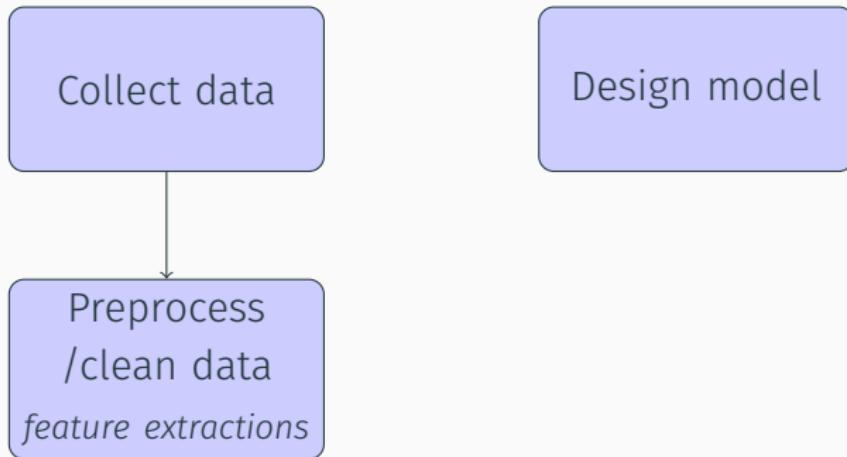
# Steps

---

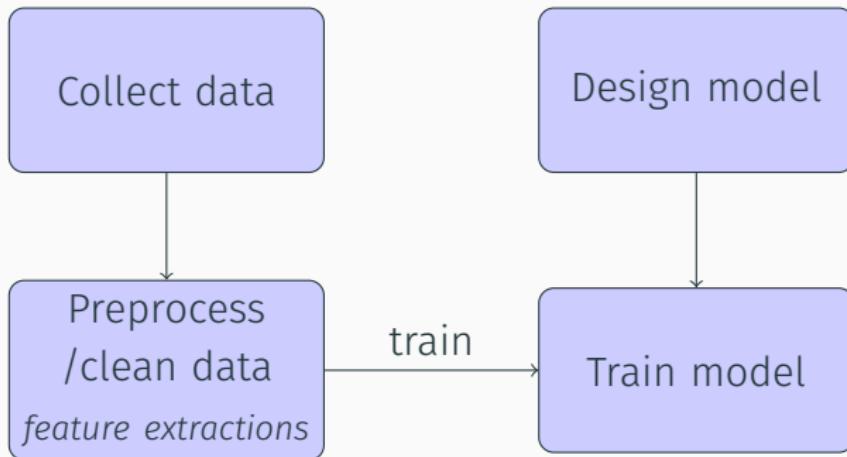


# Steps

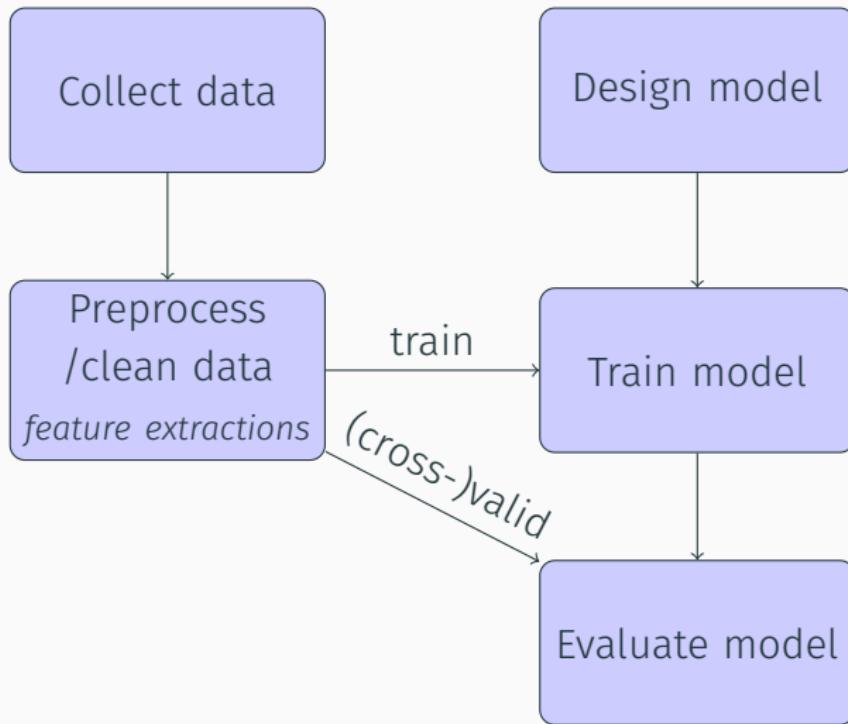
---



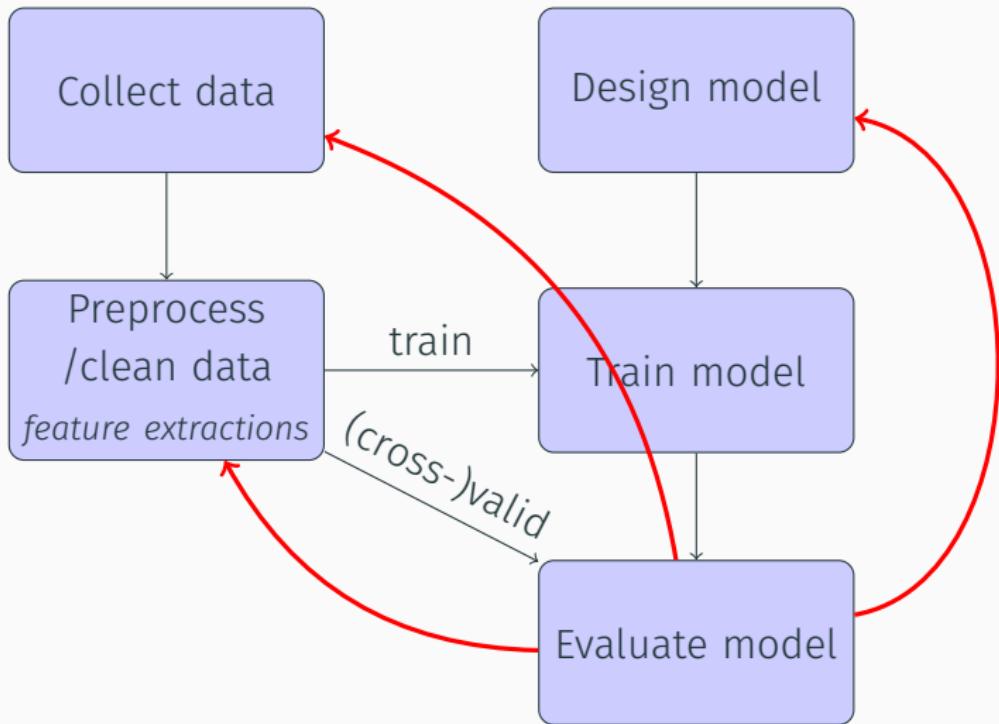
# Steps



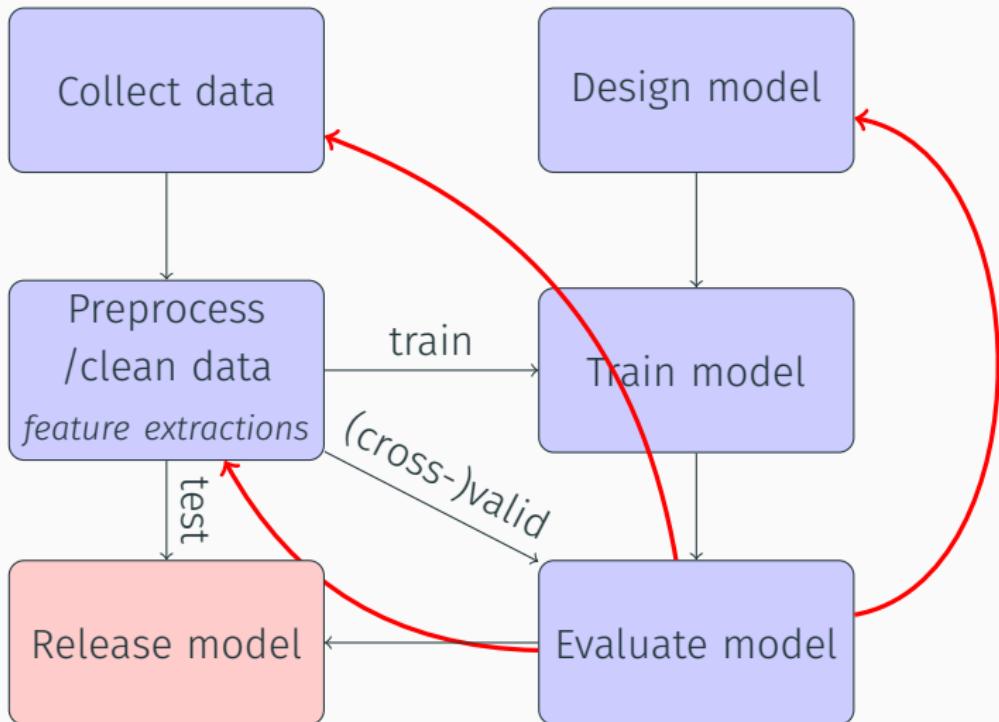
# Steps



# Steps



# Steps



## 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.

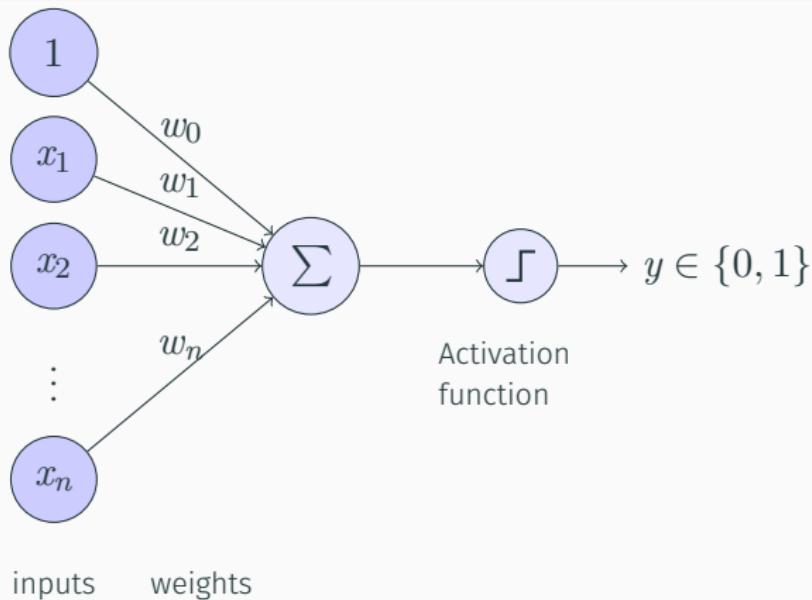
Some parameters of the model (e.g. polynomial order in a polynomial regression) were determined from the validation dataset.

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

# Neural Networks

---

# The perceptron : an artificial neuron



Rosenblatt, 1957

## Computation

$$y = f(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n) = f(w_0 + \sum_{i=1}^n w_i \cdot x_i)$$

## Some remarks

---

- Inputs  $x_i$  are the different features of the data

## Some remarks

---

- Inputs  $x_i$  are the different features of the data
- Weight  $w_i$  are the parameters of the model to optimize

## Some remarks

---

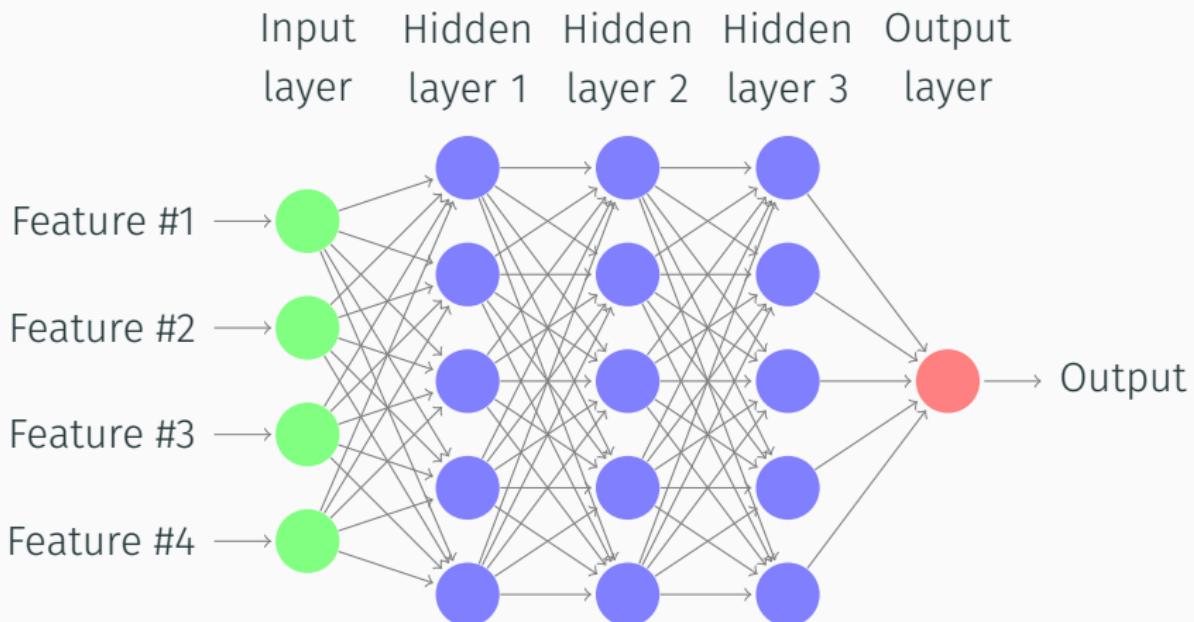
- Inputs  $x_i$  are the different features of the data
- Weight  $w_i$  are the parameters of the model to optimize
- If the activation function is identity, it is equivalent to a linear regression

## Some remarks

- Inputs  $x_i$  are the different features of the data
- Weight  $w_i$  are the parameters of the model to optimize
- If the activation function is identity, it is equivalent to a linear regression

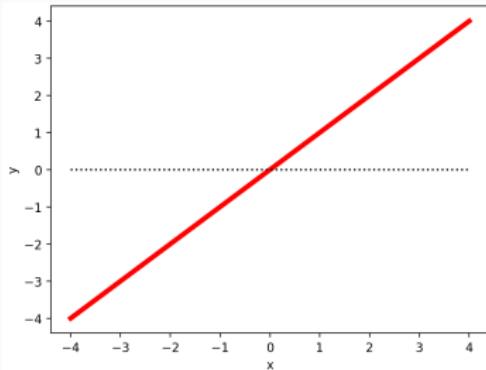
More complexe models are build by combining several perceptrons

# Multi-layer perceptron (Densely connected layers)

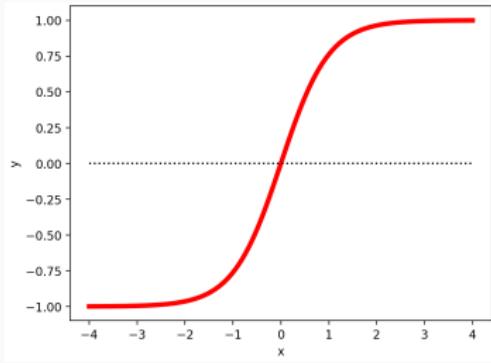


# More usual activation functions

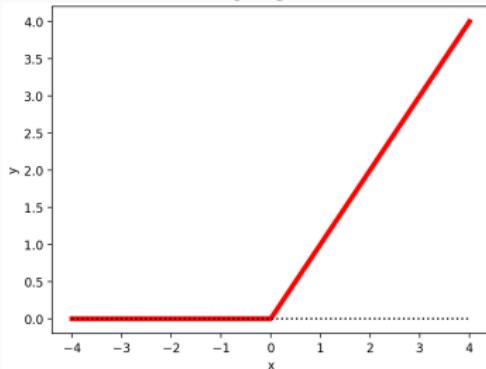
Linear



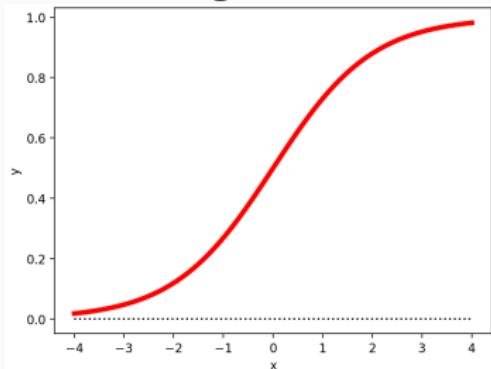
Hyperbolic tangent



ReLU

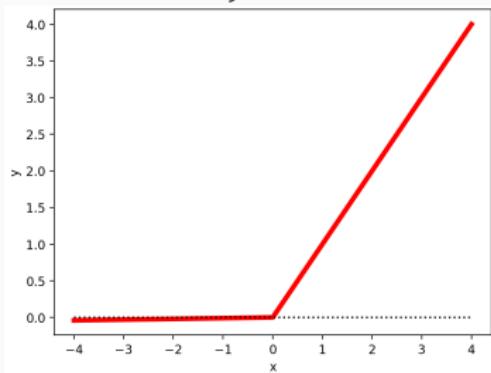


Sigmoid

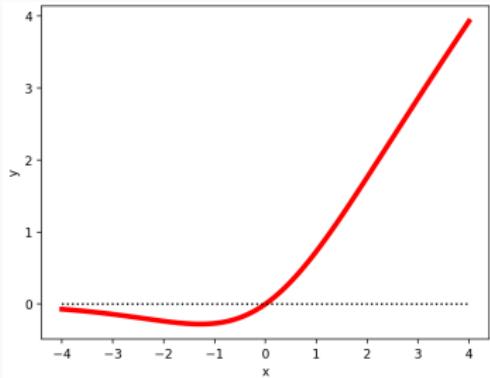


# Other "fancy" activation functions

Leaky-ReLU



Swish



## Regression

- Last layer:  
linear or hyperbolic  
tangent
- Loss function:

$$L(\hat{y}, y) = \sum_i (\hat{y}_i - y_i)^2$$

# Classification and regression loss

## Regression

- Last layer:  
linear or hyperbolic  
tangent
- Loss function:

$$L(\hat{y}, y) = \sum_i (\hat{y}_i - y_i)^2$$

## Classification

- Last layer:  
Soft-max

$$p_j = f_j(\mathbf{h}) = \frac{e^{h_j}}{\sum_k e^{h_k}}$$

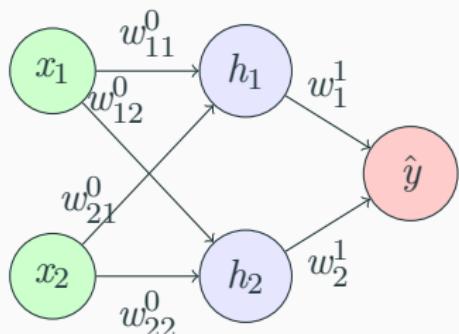
- Loss function:  
Negative crossentropy

$$L(p, y) = - \sum_i \sum_j y_{i,j} \log p_{i,j}$$

## Gradient backpropagation

---

# Training a neural-net: gradient backpropagation



- Given a couple  $(x, y)$

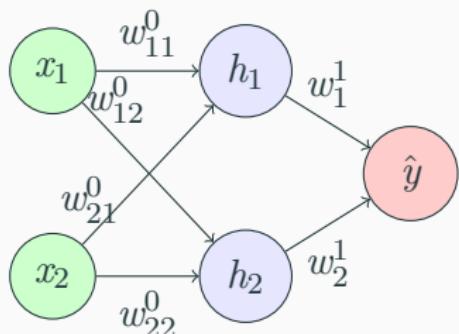
## Objective

Determination of the best set of weights  $\mathbf{w}$  to minimize the Loss function

$$L(\mathbf{w}) = \|\hat{y}(\mathbf{w}) - y\|^2.$$

Calculation of  $\partial L / \partial w$

# Training a neural-net: gradient backpropagation



- Given a couple  $(x, y)$
- Forward computation:

$$h_j = f_0(\sum_{i=1}^2 w_{ij}^0 \cdot x_i)$$

$$\hat{y} = f_1(\sum_{j=1}^2 w_j^1 \cdot h_j)$$

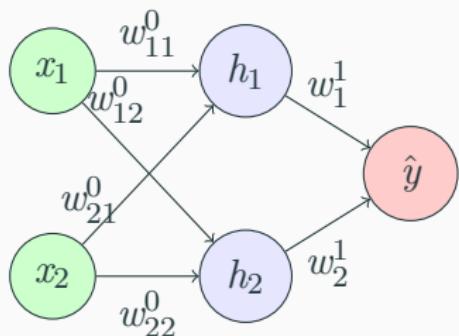
## Objective

Determination of the best set of weights  $\mathbf{w}$  to minimize the Loss function

$$L(\mathbf{w}) = \|\hat{y}(\mathbf{w}) - y\|^2.$$

Calculation of  $\partial L / \partial w$

# Training a neural-net: gradient backpropagation



- Given a couple  $(x, y)$
- Forward computation:  
$$h_j = f_0(\sum_{i=1}^2 w_{ij}^0 \cdot x_i)$$
$$\hat{y} = f_1(\sum_{j=1}^2 w_j^1 \cdot h_j)$$
- Compute the gradient of the loss:

$$\partial L / \partial \hat{y}$$

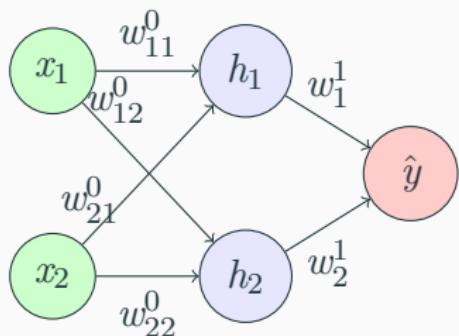
## Objective

Determination of the best set of weights  $\mathbf{w}$  to minimize the Loss function

$$L(\mathbf{w}) = \|\hat{y}(\mathbf{w}) - y\|^2.$$

Calculation of  $\partial L / \partial w$

# Training a neural-net: gradient backpropagation



## Objective

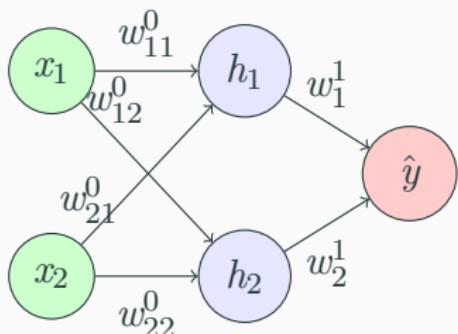
Determination of the best set of weights  $\mathbf{w}$  to minimize the Loss function

$$L(\mathbf{w}) = \|\hat{y}(\mathbf{w}) - y\|^2.$$

Calculation of  $\partial L / \partial w$

- Given a couple  $(x, y)$
- Forward computation:  
$$h_j = f_0(\sum_{i=1}^2 w_{ij}^0 \cdot x_i)$$
$$\hat{y} = f_1(\sum_{j=1}^2 w_j^1 \cdot h_j)$$
- Compute the gradient of the loss:  
$$\boxed{\partial L / \partial \hat{y}}$$
- Gradient Backpropagation:

# Training a neural-net: gradient backpropagation



## Objective

Determination of the best set of weights  $\mathbf{w}$  to minimize the Loss function

$$L(\mathbf{w}) = \|\hat{y}(\mathbf{w}) - y\|^2.$$

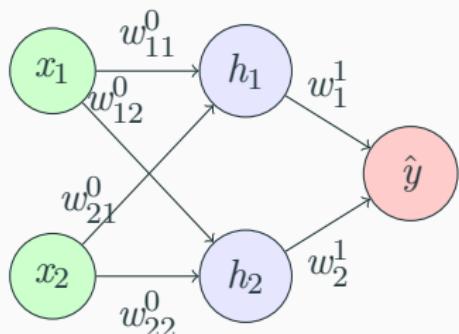
Calculation of  $\partial L / \partial w$

- Given a couple  $(x, y)$
- Forward computation:  
$$h_j = f_0(\sum_{i=1}^2 w_{ij}^0 \cdot x_i)$$
$$\hat{y} = f_1(\sum_{j=1}^2 w_j^1 \cdot h_j)$$
- Compute the gradient of the loss:  
$$\boxed{\partial L / \partial \hat{y}}$$
- Gradient Backpropagation:

- Layer 1

$$\partial L / \partial w_j^1 = \boxed{\partial L / \partial \hat{y}} \cdot \partial f_1 / \partial w_j^1$$
$$\boxed{\partial L / \partial h_j} = \boxed{\partial L / \partial \hat{y}} \cdot \partial f_1 / \partial h_j$$

# Training a neural-net: gradient backpropagation



## Objective

Determination of the best set of weights  $\mathbf{w}$  to minimize the Loss function

$$L(\mathbf{w}) = \|\hat{y}(\mathbf{w}) - y\|^2.$$

Calculation of  $\partial L / \partial w$

- Given a couple  $(x, y)$
- Forward computation:  
$$h_j = f_0(\sum_{i=1}^2 w_{ij}^0 \cdot x_i)$$
  
$$\hat{y} = f_1(\sum_{j=1}^2 w_j^1 \cdot h_j)$$
- Compute the gradient of the loss:  
$$\boxed{\partial L / \partial \hat{y}}$$
- Gradient Backpropagation:

- Layer 1

$$\partial L / \partial w_j^1 = \boxed{\partial L / \partial \hat{y}} \cdot \partial f_1 / \partial w_j^1$$

$$\boxed{\partial L / \partial h_j} = \boxed{\partial L / \partial \hat{y}} \cdot \partial f_1 / \partial h_j$$

- Layer 0

$$\partial L / \partial w_{ij}^0 = \boxed{\partial L / \partial h_j} \cdot \partial f_1 / \partial w_{ij}^0$$

## Optimizing a machine learning (gradient method)

---

# Optimizing the loss

Several loss function (depending on the problem) can be defined.

For example, Mean Square Error:

## Method

Find a minimum of  $L$  by adjusting the parameters (weights)  $w$  given the gradient of the loss with respect to the weights

$$\nabla_w L.$$

# Batch Vs Stochastic training

Dataset:  $(X, Y)$  with  $N$  samples denoted  $(\mathbf{x}_i, y_i)$

## Batch gradient:

Require: Learning rate(s):  $\nu_k$

Require: Initial weights:  $\mathbf{w}$

$k \leftarrow 1$

while stopping criterion not met do

    Compute gradient:

$$\mathbf{g} \leftarrow \frac{1}{N} \sum_i^N \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i))$$

    Update weights:  $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$

$k \leftarrow k + 1$

end while

1 Update / N forwards

# Batch Vs Stochastic training

Dataset:  $(X, Y)$  with  $N$  samples denoted  $(\mathbf{x}_i, y_i)$

## Batch gradient:

Require: Learning rate(s):  $\nu_k$

Require: Initial weights:  $\mathbf{w}$

$k \leftarrow 1$

while stopping criterion not met do

    Compute gradient:

$$\mathbf{g} \leftarrow \frac{1}{N} \sum_i^N \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i))$$

    Update weights:  $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$

$k \leftarrow k + 1$

end while

1 Update / N forwards

## Stochastic gradient:

Require: Learning rate(s):  $\nu_k$

Require: Initial weights:  $\mathbf{w}$

$k \leftarrow 1$

while stopping criterion not met do

    Sample an example  $(\mathbf{x}, y)$  from  $(X, Y)$

    Compute gradient:  $\mathbf{g} \leftarrow \nabla_{\mathbf{w}} L(f(\mathbf{x}, y))$

    Update weights:  $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$

$k \leftarrow k + 1$

end while

1 Update / 1 forward

# Mini-Batch training

Dataset:  $(X, y)$  with  $N$  samples

## Mini-Batch gradient:

Require: Learning rate(s):  $\nu_k$

Require: Initial weights:  $\mathbf{w}$

$k \leftarrow 1$

while stopping criterion not met do

    Sample  $m$  examples  $(\mathbf{x}_i, y_i)$  from  $(X, y)$

    Compute gradient:  $\mathbf{g} \leftarrow \frac{1}{m} \sum_i^m \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i))$

    Update weights:  $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$

$k \leftarrow k + 1$

end while

# Mini-Batch training

Dataset:  $(X, y)$  with  $N$  samples

## Mini-Batch gradient:

Require: Learning rate(s):  $\nu_k$

Require: Initial weights:  $\mathbf{w}$

$k \leftarrow 1$

while stopping criterion not met do

    Sample  $m$  examples  $(\mathbf{x}_i, y_i)$  from  $(X, y)$

    Compute gradient:  $\mathbf{g} \leftarrow \frac{1}{m} \sum_i^m \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i))$

    Update weights:  $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$

$k \leftarrow k + 1$

end while

*m* Update / 1 forward

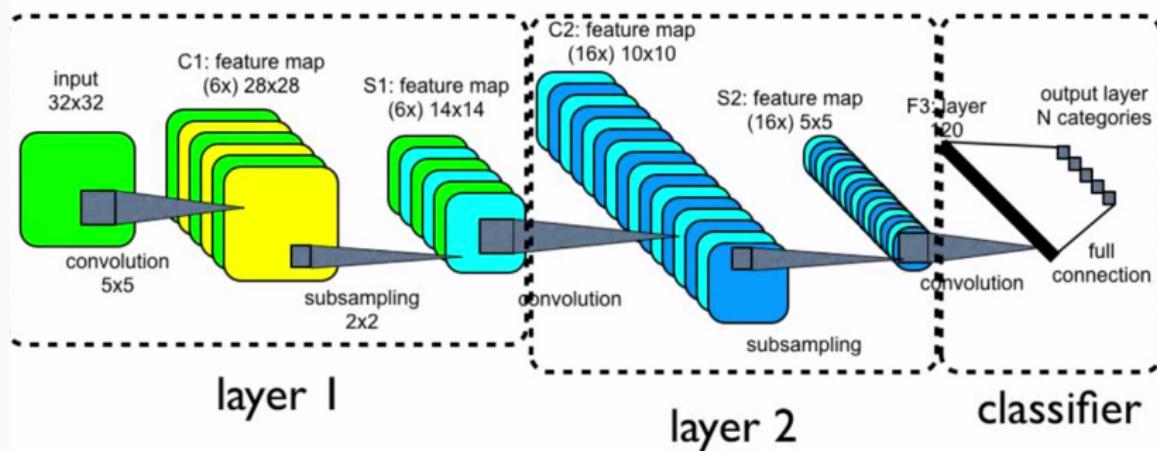
$m = 1$ : Pure stochastic gradient.

$m = N$ : Batch gradient

## A quick typology of few neural nets

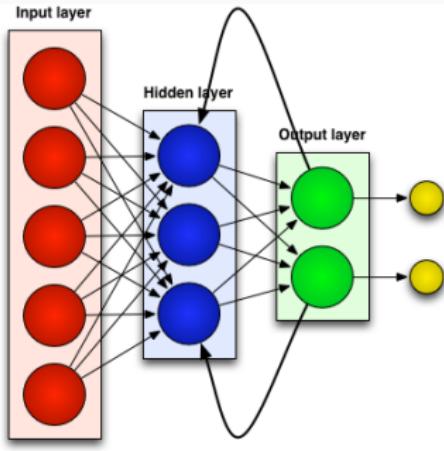
---

# Convolutional neural networks



- Very efficient for computer vision (processing images with spatial patterns)
- Act locally on the input feature space.

# Recurrent Neural Networks

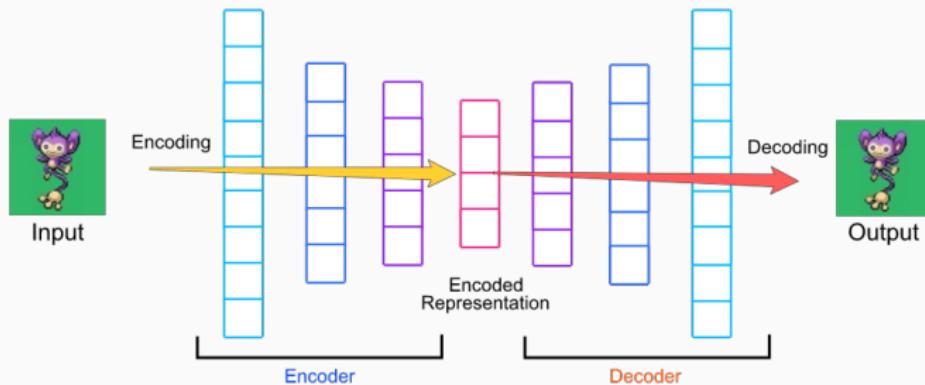


Some popular types of recurrent neural networks:

- Long short-term memory (LSTM)
- Gated Recurrent Unit (GRU)

Used in machine translation and text processing

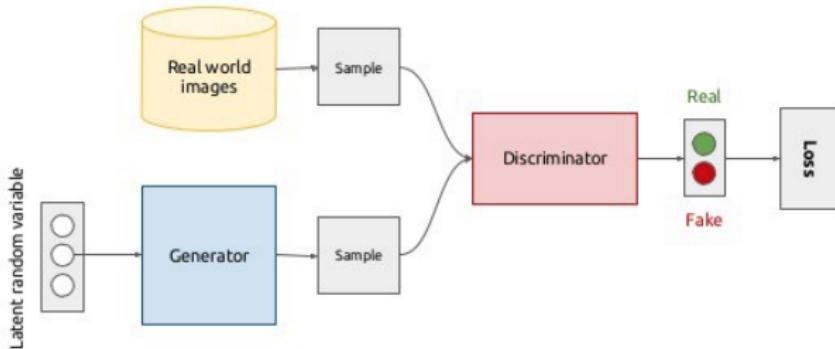
# Autoencoders



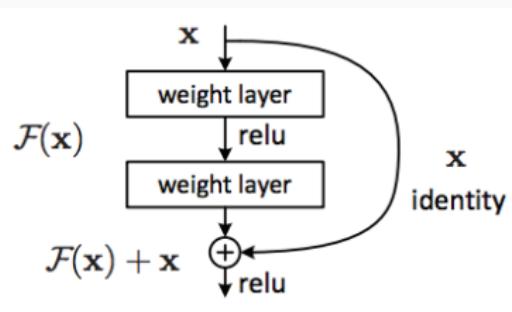
Used in image denoising, compressing, generation,...

# Generative adversarial networks

## Generative adversarial networks (conceptual)



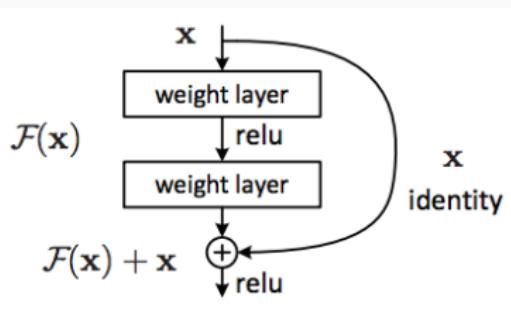
# Residual Networks



$x$ : input,  $y$ : output

$$y = x + \mathcal{F}(x)$$

# Residual Networks



$x$ : input,  $y$ : output

$$y = x + \mathcal{F}(x)$$

