scai

SORBONNE CLUSTER FOR
ARTIFICIAL INTELLIGENCE

AI Introduction: From Linear Regression to Large Language Generative Models (part 1/2)

Raphael Cousin
Data Scientist
raphael.cousin@sorbonne-universite.fr

raphaelcousin.com

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Plan

AI - Introduction:
- Definition
- Application
- History
- Context

Duration : ~10 min

Machine Learning:
- Data characteristics
- Parametric Models
- Loss function
- Training
- Not Parametric Model

Duration : ~30 min

Deep Learning:
- From Neuron to MLP
- CPU vs GPU
- The NN Layers

Duration : ~5 min

Data Science:
- Data Preprocessing:
- Hyperparameters
- Model evaluation methodology

Duration : ~10 min

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

AI - Landing Overview

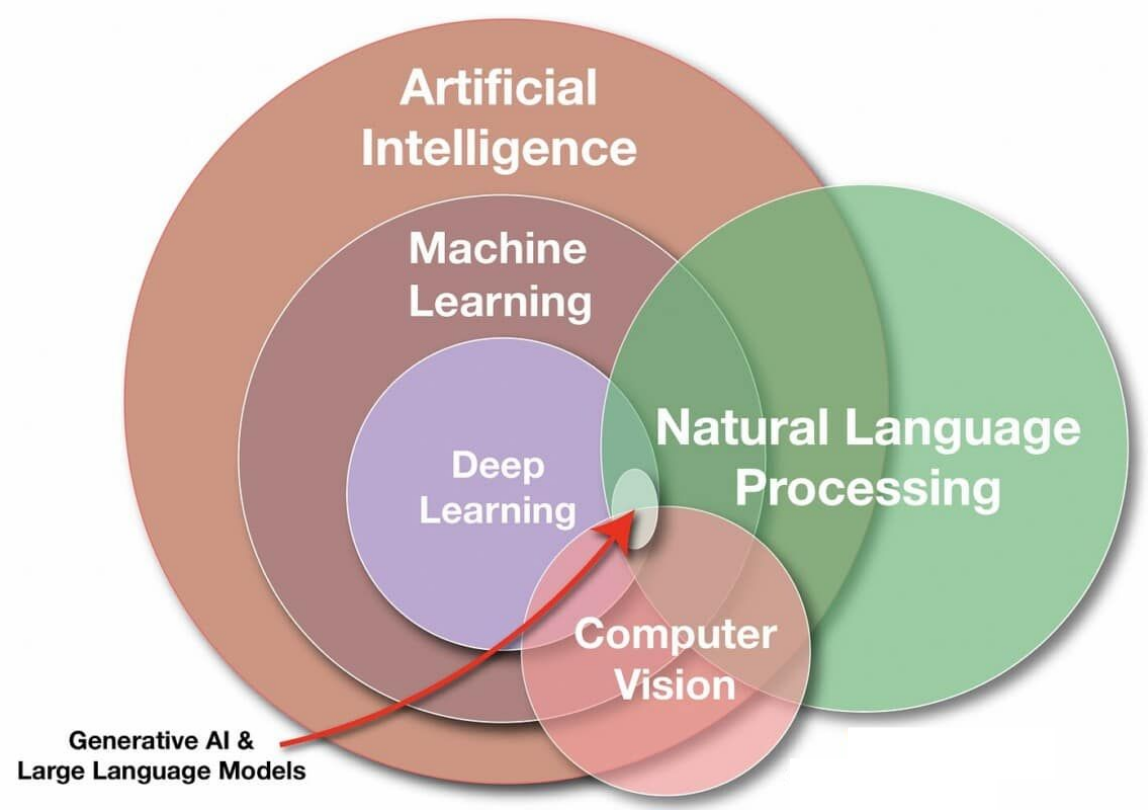
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

AI : Definitions

AI: Automation involving algorithms to reproduce cognitive capabilities such as reasoning, perception, and decision-making.

ML: Statistical algorithms that automatically learn patterns from data without being explicitly programmed for each task.

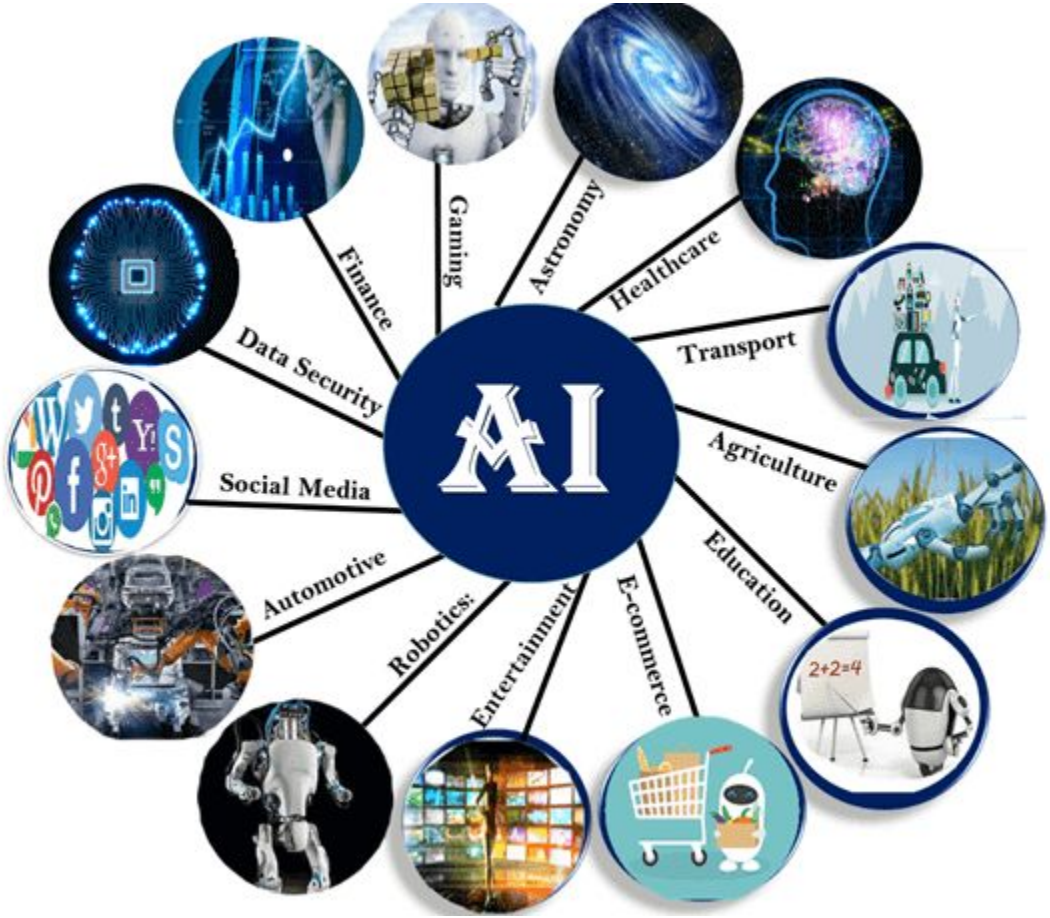
DL: Neural networks with multiple layers (millions of parameters) that can learn complex representations from raw data.



source: https://medium.com/geekculture/ai-revolution-your-fast-paced-introduction-to-machine-learning-914ce9b6ddf

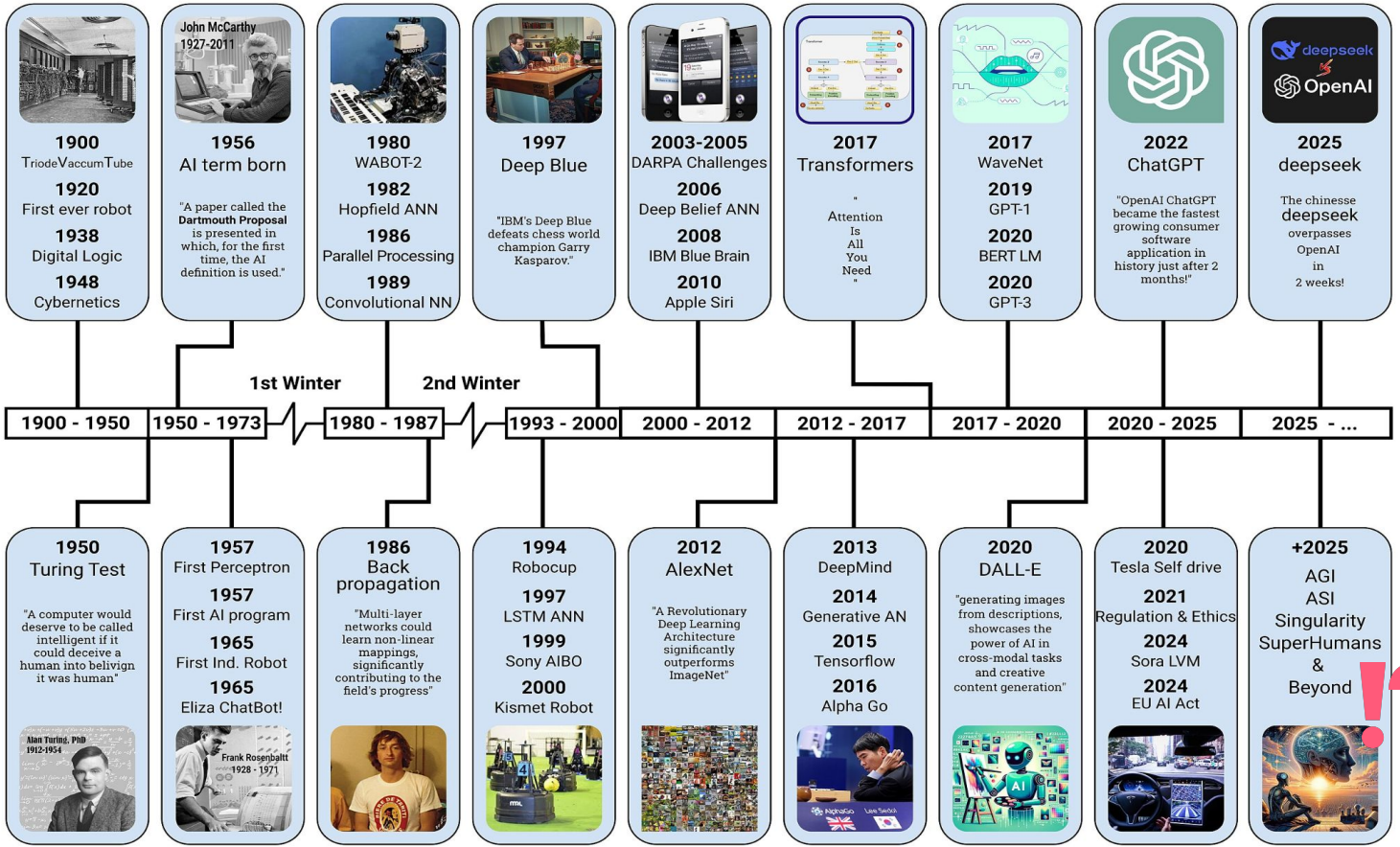
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

AI : applications



scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

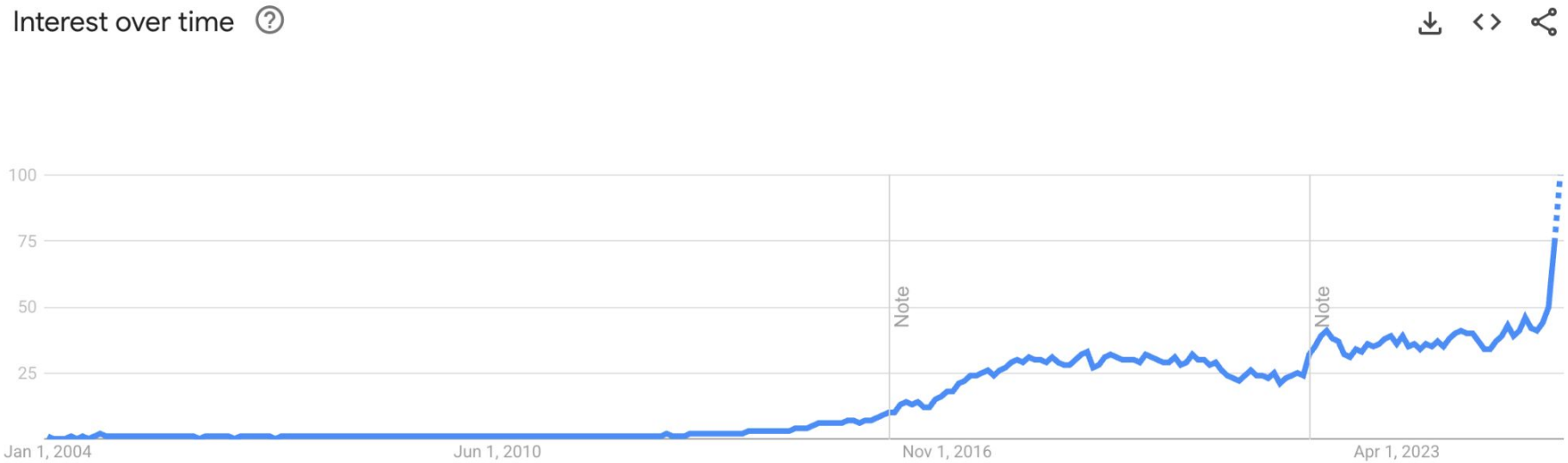
AI History



source: https://commons.wikimedia.org/wiki/File:AI-History-Timeline-300dpi.jpg

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

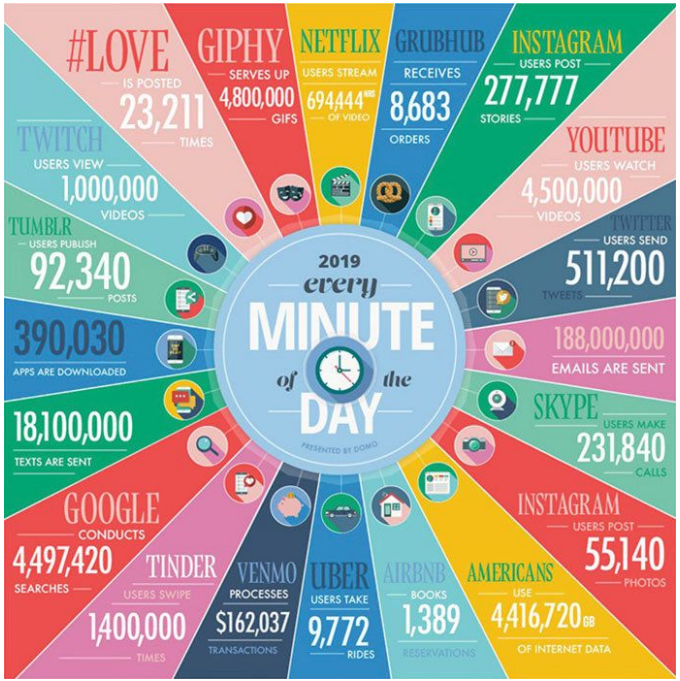
The Deep Learning Revolution



The convergence of three critical factors enabled the deep learning revolution:

Source: https://trends.google.fr/trends/explore?date=all&q=deep%20learning&hl=en

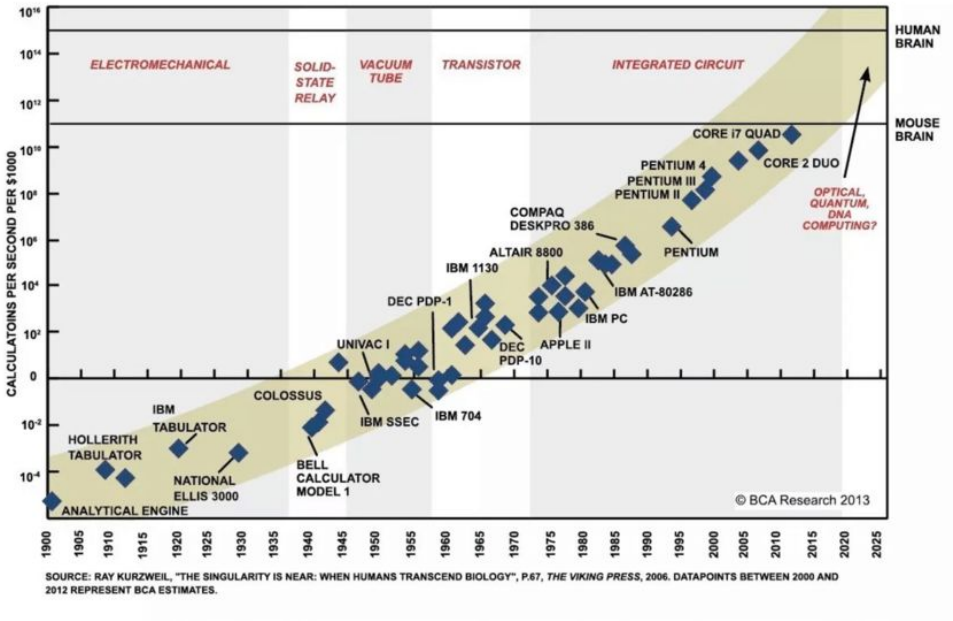
1. Big Data



The internet era generated massive datasets: billions of images, text documents, videos, and user interactions. This data explosion provided the fuel needed to train complex models that require millions of examples to learn robust patterns.

Source: https://www.domo.com/learn/datanever-sleeps-7

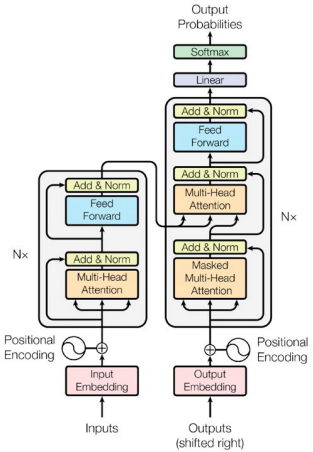
2. Computational Power



GPUs originally designed for gaming proved perfect for neural network computations. A single modern GPU can perform trillions of operations per second, enabling training of models with billions of parameters in days rather than years.

Source: https://www.researchgate.net/figure/The-exponential-progress-of-computing-power-from-1900-to-2013-with-projections-into_fig1_335422453

3. Algorithmic Innovation



PyTorch

Breakthrough techniques like ReLU activation, batch normalization, dropout, and attention mechanisms solved critical training challenges. These innovations made it practical to train networks with dozens or even hundreds of layers. Open source code with big communities.

Source: https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture) Source: https://pytorch.org/

Machine Learning

The Machine Learning Objective

Machine Learning definition:

Systems that automatically learn patterns from data.

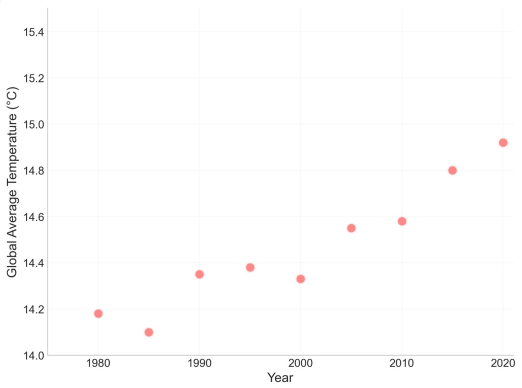
Example

Using historical data to predict the future

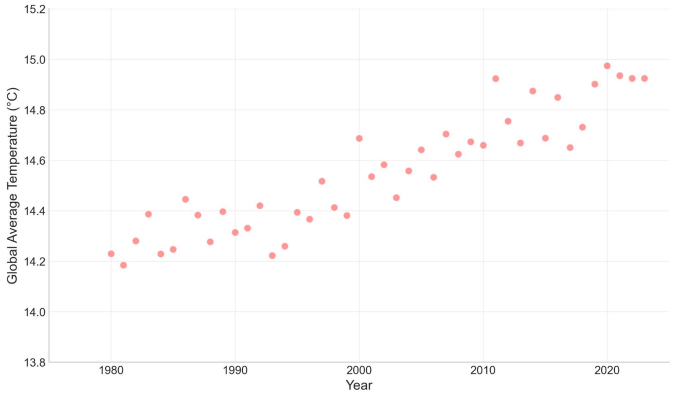
Data Characteristics

Quantity : Sample size: number of observations

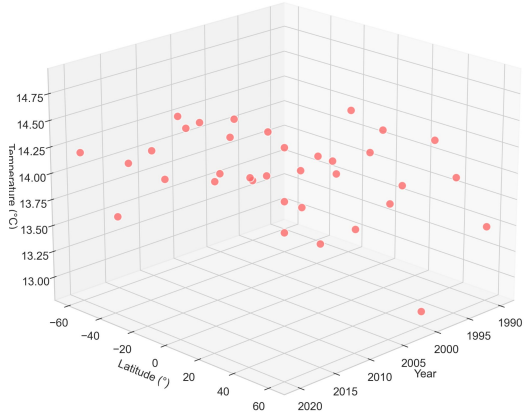
Dimension: number of variables per observation (features)



9 observations / 2 variables



44 observations / 2 variables



9 observations / 3 variables

Année	Latitude	Altitude	CO_2	Humidité	°C
2000	45.0	200	370	65	13.8
2010	30.0	800	390	45	12.5
2020	60.0	50	415	80	14.2

3 observations / 6 variables

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

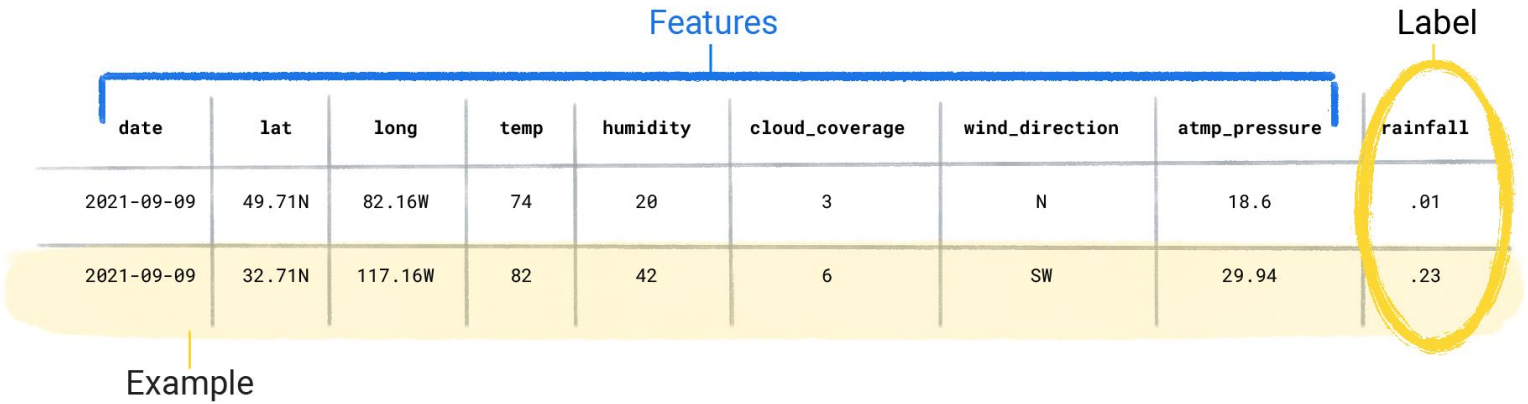
ML : Features and Target/Label

X : Explanatory variables (features/characteristics)

Y: Variable to explain (target)

Concretely, we want to find a relationship Y = f(X)

Example: f(year, latitude, altitude) = predicted temperature



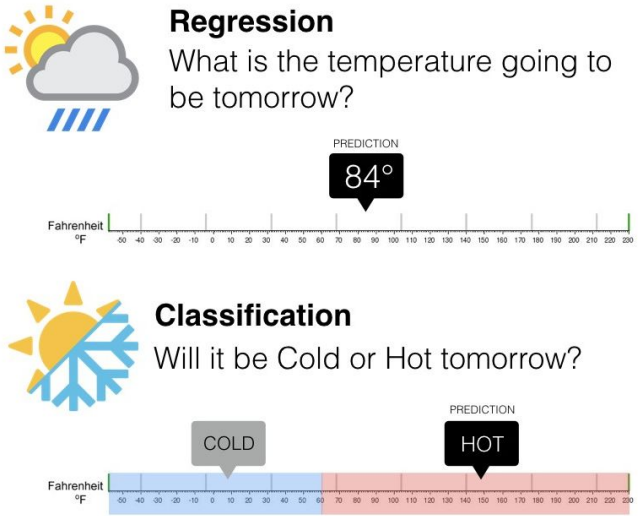
date	lat	long	temp	humidity	cloud_coverage	wind_direction	atmp_pressure	rainfall
2021-09-09	49.71N	82.16W	74	20	3	N	18.6	.01
2021-09-09	32.71N	117.16W	82	42	6	SW	29.94	.23

Regression, Classification

Regression: Predict continuous values

$$y \in \mathbb{R}^k$$

Examples: House price



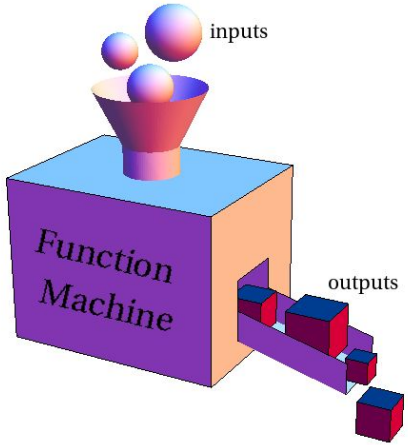
Classification: Predict discrete categories

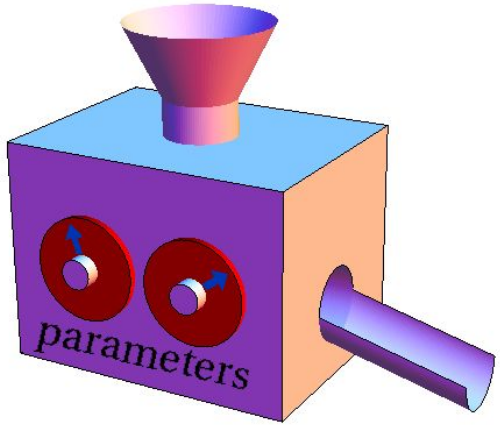
$$y \in \{1, 2, \ldots, K\}$$

Examples: Spam detection

In practice we will predict continuous value "probability" in [0,1]

Parametric Models characteristics





Parametric Models are functions defined by

Parameters: Number of values the model must learn

Parameter vector: $\theta \in \mathbb{R}^p$

p is the number of parameters

Input/Output Dimension:

$$f_\theta : \mathbb{R}^d \to \mathbb{R}^k$$

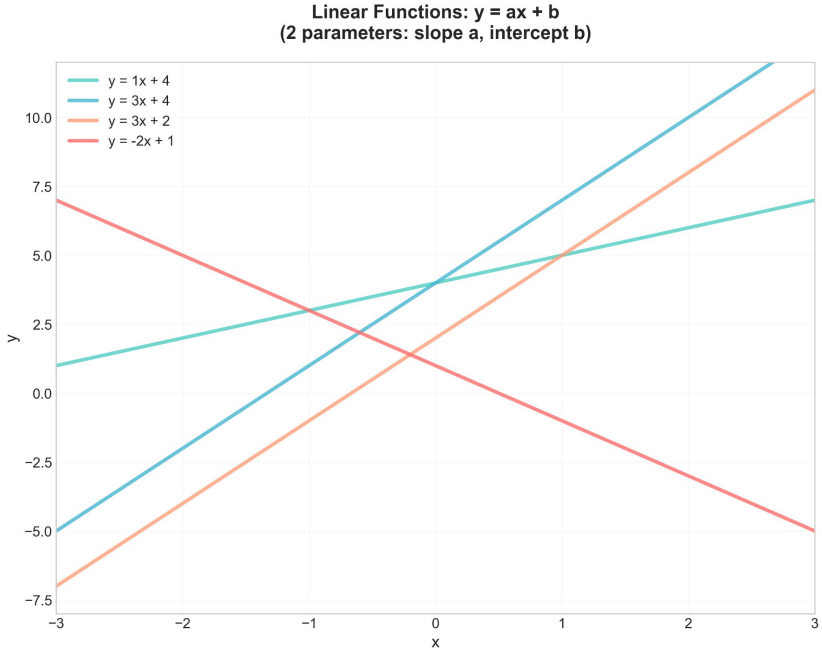
Input dimension: Dimension of features

Output dimension: Dimension of features targets

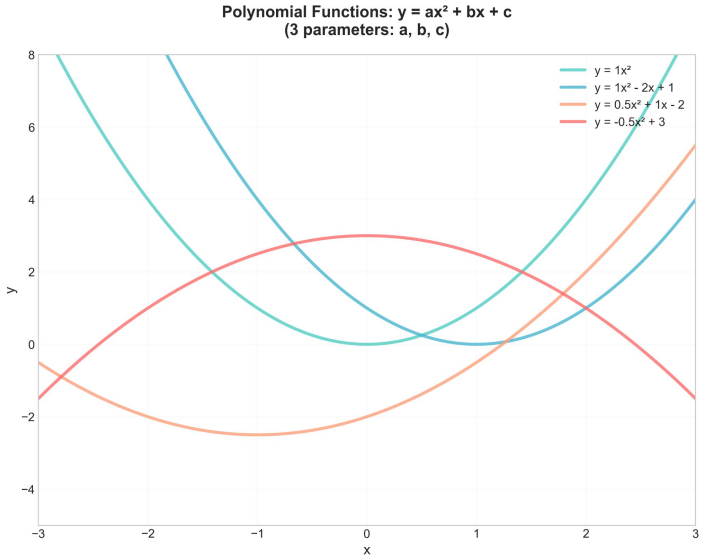
Operations: Between parameters and input data

Operations: addition, multiplication, composition

Parametric Models



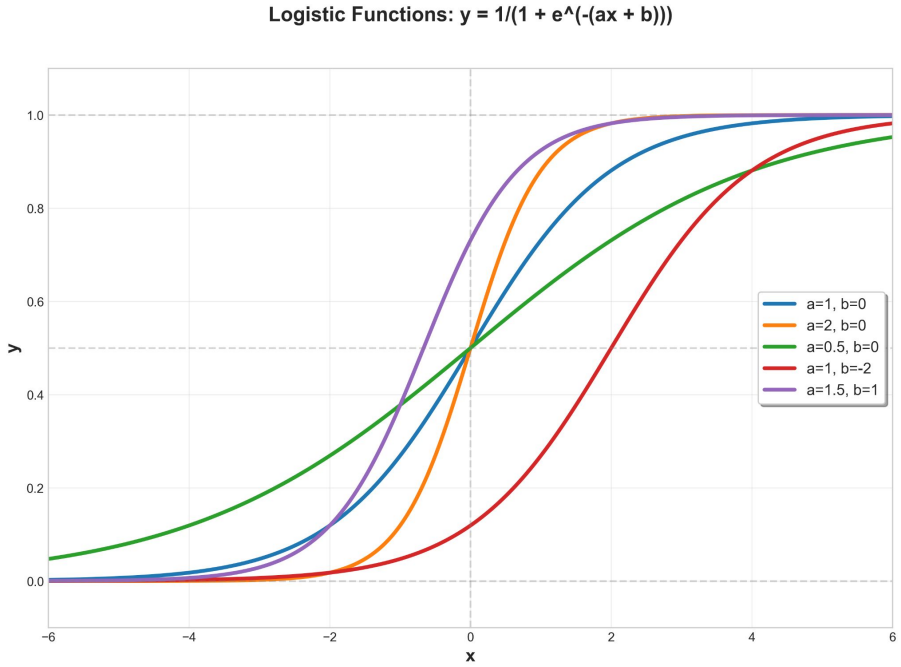
$\mathbb{R} \to \mathbb{R},\ 2$ params



$\mathbb{R} \to \mathbb{R},\ 3$ params

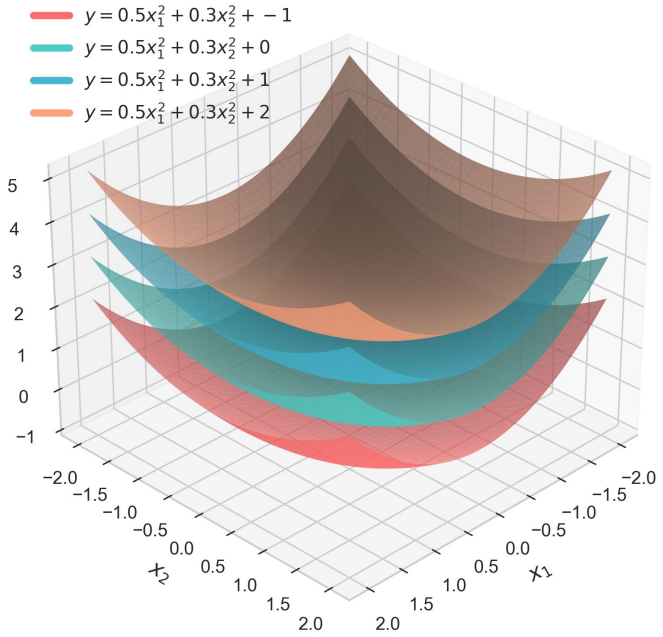
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Parametric Models



Logistic $\mathbb{R} \to [0, 1]$, 2 params

Quadratic Functions: $y = ax_1^2 + bx_2^2 + cx_1 + dx_2 + e$
(5 parameters)



Quadratic function $\mathbb{R}^2 \to \mathbb{R}$, 5 params

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Loss Functions characteristics

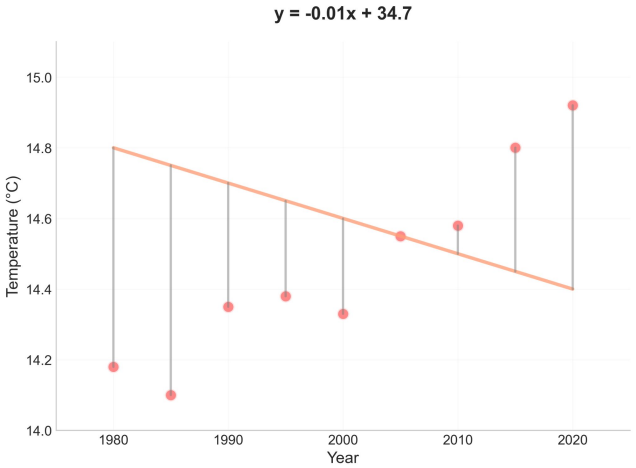
The Role of Loss Functions Loss functions quantify the error/distance between real targets $\hat{y} = f_\theta(x)$

$$l : \mathbb{R}^{k \times n} \times \mathbb{R}^{k \times n} \to \mathbb{R}$$
$$(Y, \hat{Y}) \mapsto l(Y, \hat{Y})$$

Example Loss Functions

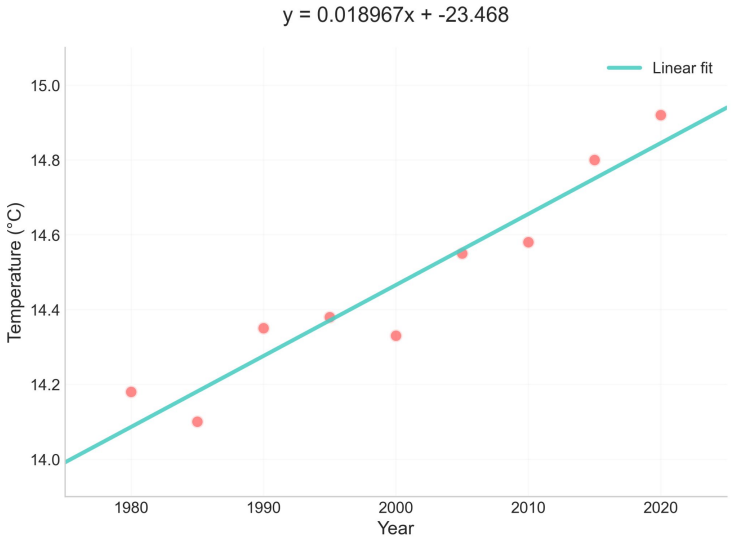
Mean Squared Error

$$\ell_{MSE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



The Fundamental Learning Problem

For a given parametric mode machine learning seeks to find the parameters that minimize the loss between real and predicted value :



$$\theta^* = \arg\min_{\theta \in \mathbb{R}^p} \ell(\theta) = \arg\min_{\theta \in \mathbb{R}^p} l(Y, f_\theta(X))$$

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

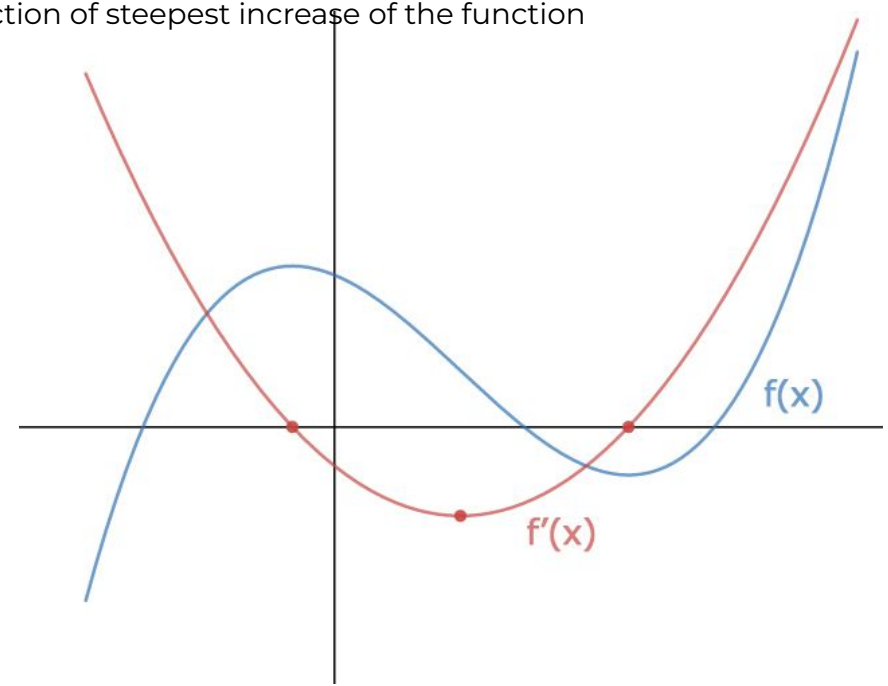
Derivatives and Gradients

Derivative (1D):

- Measures how much a function changes when its input changes

Gradient (Multi-dimensional):

- Extension of derivative to functions with multiple inputs $\nabla f = [\partial f/\partial x_1, \partial f/\partial x_2, ..., \partial f/\partial x_n]$
- Points in the direction of steepest increase of the function



scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Gradient Descent and Optimization

0. Initialization

Initialize θ_0

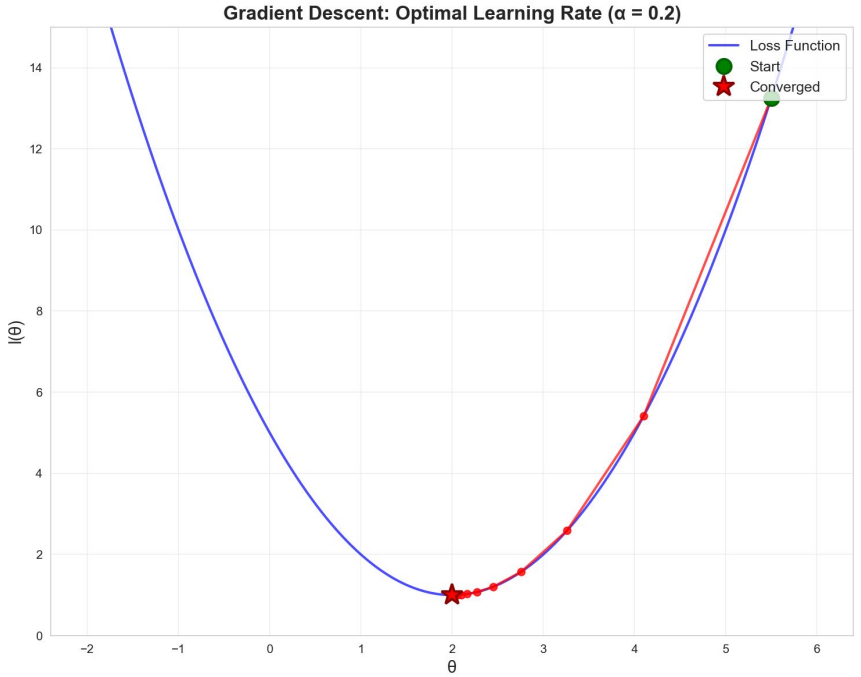
Initialize parameters randomly

1. Update Rule

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta l(\theta_t)$$

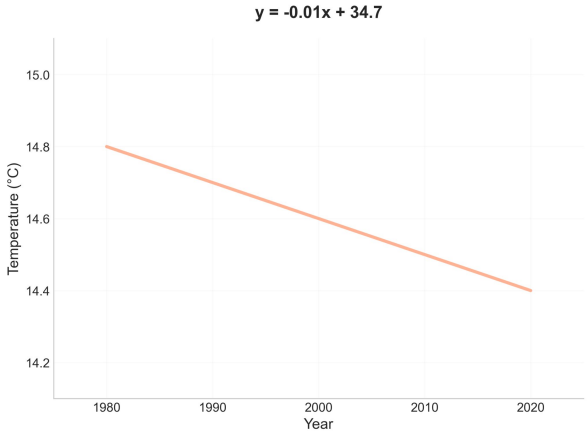
1. Iterate

Repeat 1. while $l(\theta_{t+1}) < l(\theta_t)$

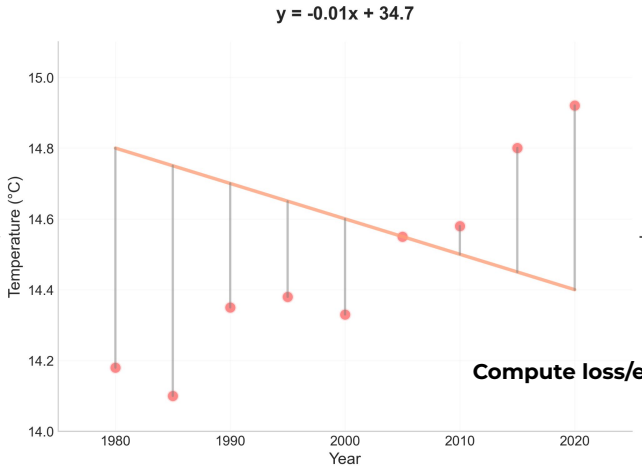


$$\theta^* = \arg\min_{\theta \in \mathbb{R}^p} \ell(\theta) = \arg\min_{\theta \in \mathbb{R}^p} l(Y, f_\theta(X))$$

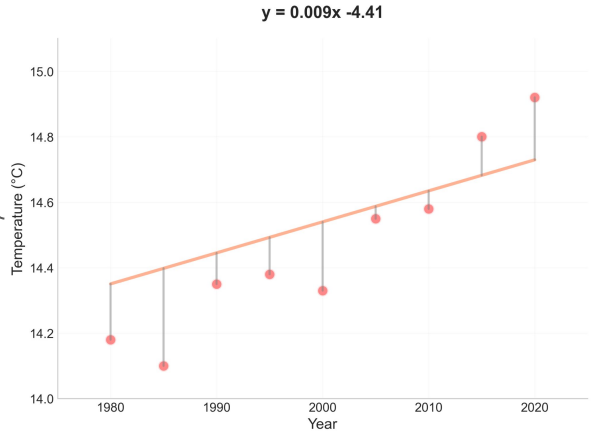
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE



Initialisation



Compute loss/error gradient

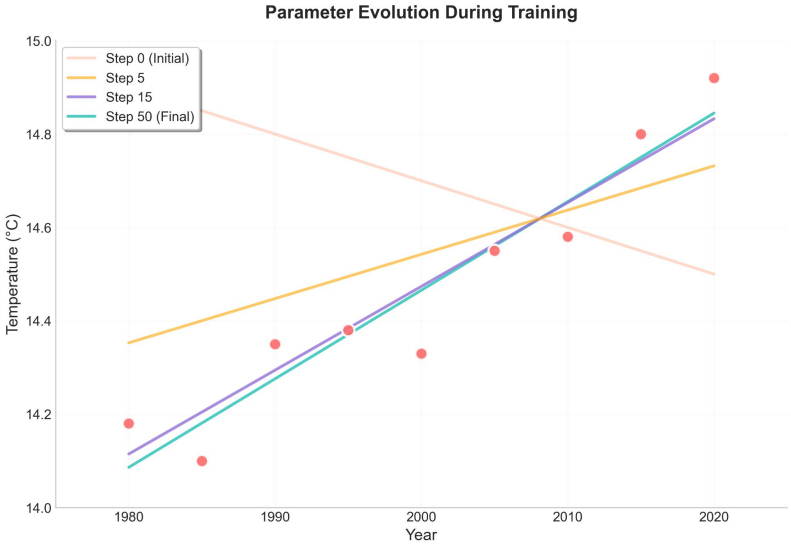


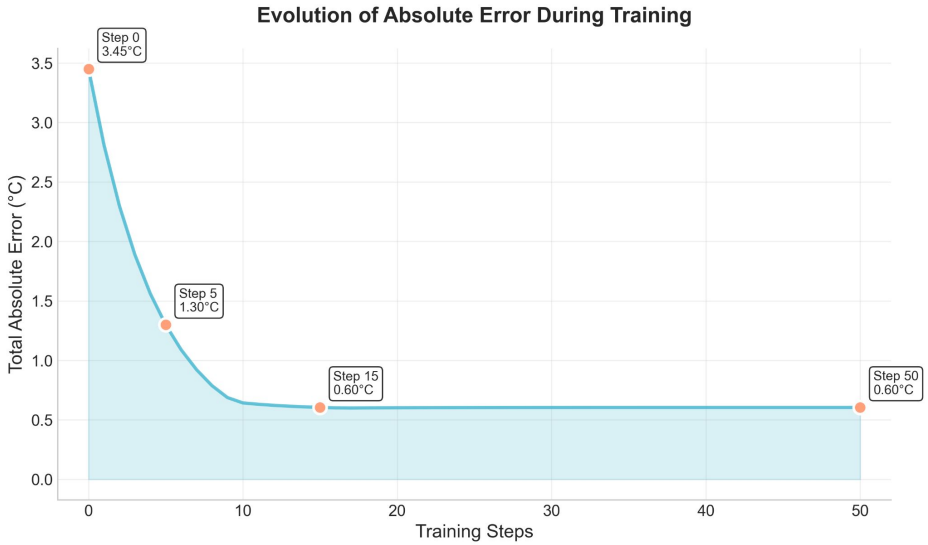
Update params (gradient descent)

Year	Y_true	Y_pred	Error
1980	14.18	14.80	0.62
1985	14.10	14.75	0.65
1990	14.35	14.70	0.35
1995	14.38	14.65	0.27
2000	14.33	14.60	0.27
2005	14.55	14.55	0.00
2010	14.58	14.50	0.08
2015	14.80	14.45	0.35
2020	14.92	14.40	0.52

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

ML : Training example





Initial Model
a=-0.01, b=34.70

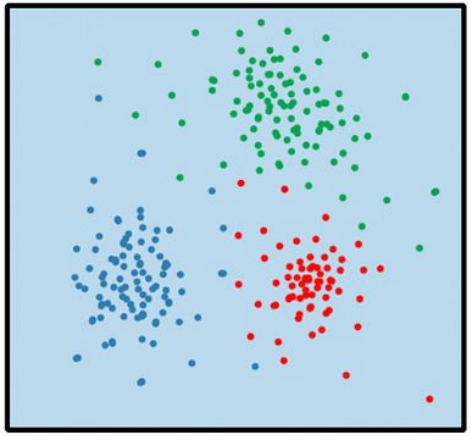
Initial model prediction for 2030
-0.01 * 2030 + 34.70 = 14.4

Trained Model
a=0.019, b=-23.47

Trained model prediction for 2030
0.019 * 2030 - 23.47 = 15.1

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

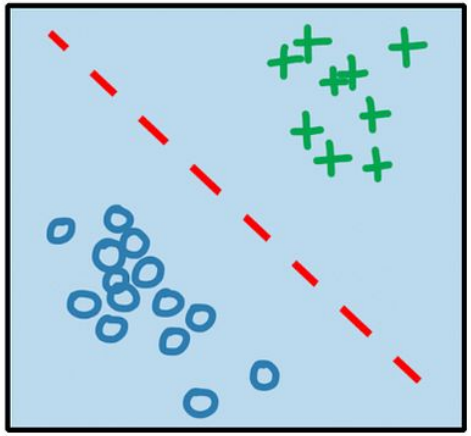
Types of Learning Problems



Unsupervised Learning:

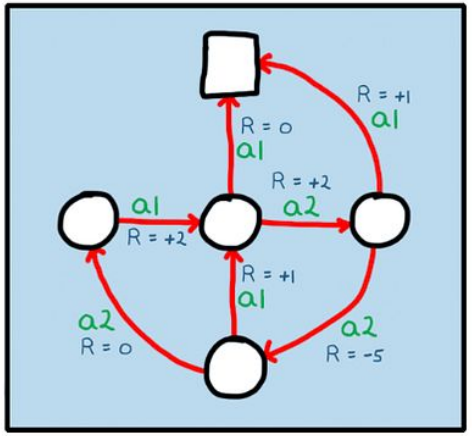
Learning structure from unlabeled data x

Clustering: Group similar data points



Supervised Learning:

Learning from labeled pairs features target (x,y)



Reinforcement Learning:

Learning through interaction and rewards in an environment

For most of the cases, we will reformulate the problem as an optimization problem

$$\arg\min_{f \in \mathcal{F}} \ell(f(X))$$

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Non Parametric Models

Parametric Models:
Fixed number of parameters regardless of dataset size

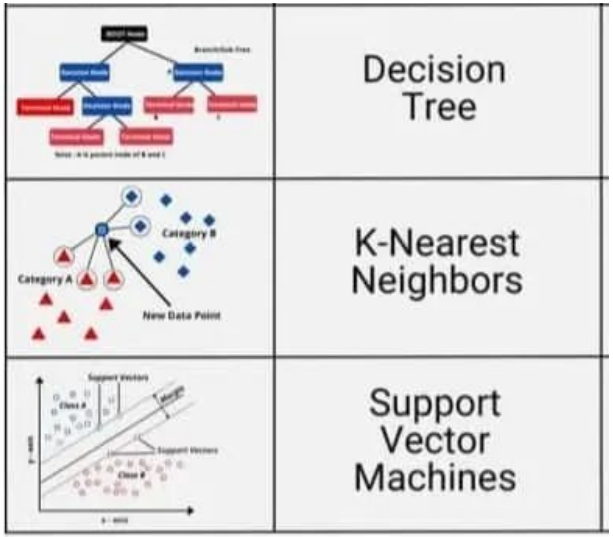
$$\arg\min_{\theta \in \mathbb{R}^p} \ell(Y, f_\theta(X))$$

Learning: gradient descent

Non-Parametric Models:
Number of parameters grows with data

$$\arg\min_{f \in \mathcal{F}} \ell(Y, f(X))$$

Learning: Various techniques (not gradient descent)



The objective remains the same, the way we minimize will differ

Deep Learning

Single Neuron unit

Input: $X = (x_1, \ldots, x_{r_0})$

Params: dimension of input + biais

$$w = (b, w_1, \ldots, w_{r_0})$$

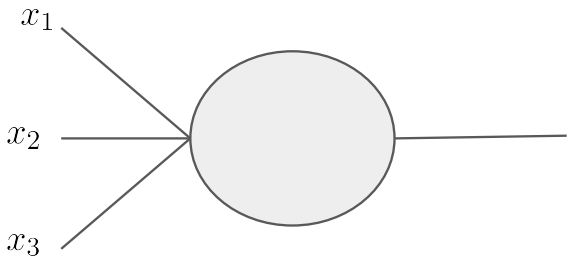
Operation:

Activation $o = \sigma(w \cdot X)$

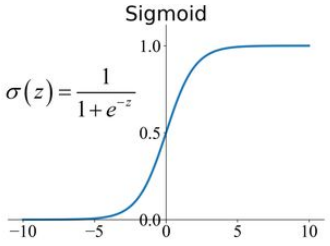
Output: dimension 1

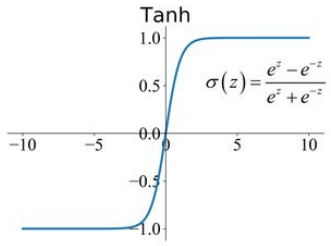
To notice:

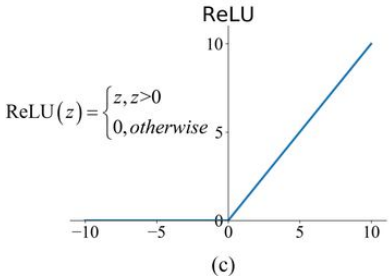
With activation identity, neuron is the same as linear regression

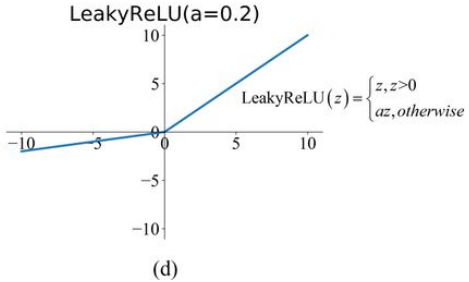


1 neurone









Multi-Layer Perceptron (MLP)

A model like any other

Input: $X = (x_1, \ldots, x_{r_0})$

Parameters:

$W^1, \ldots, W^l \qquad W^k \in \mathbb{R}^{r_{k-1} \times r_k}$

l Layers

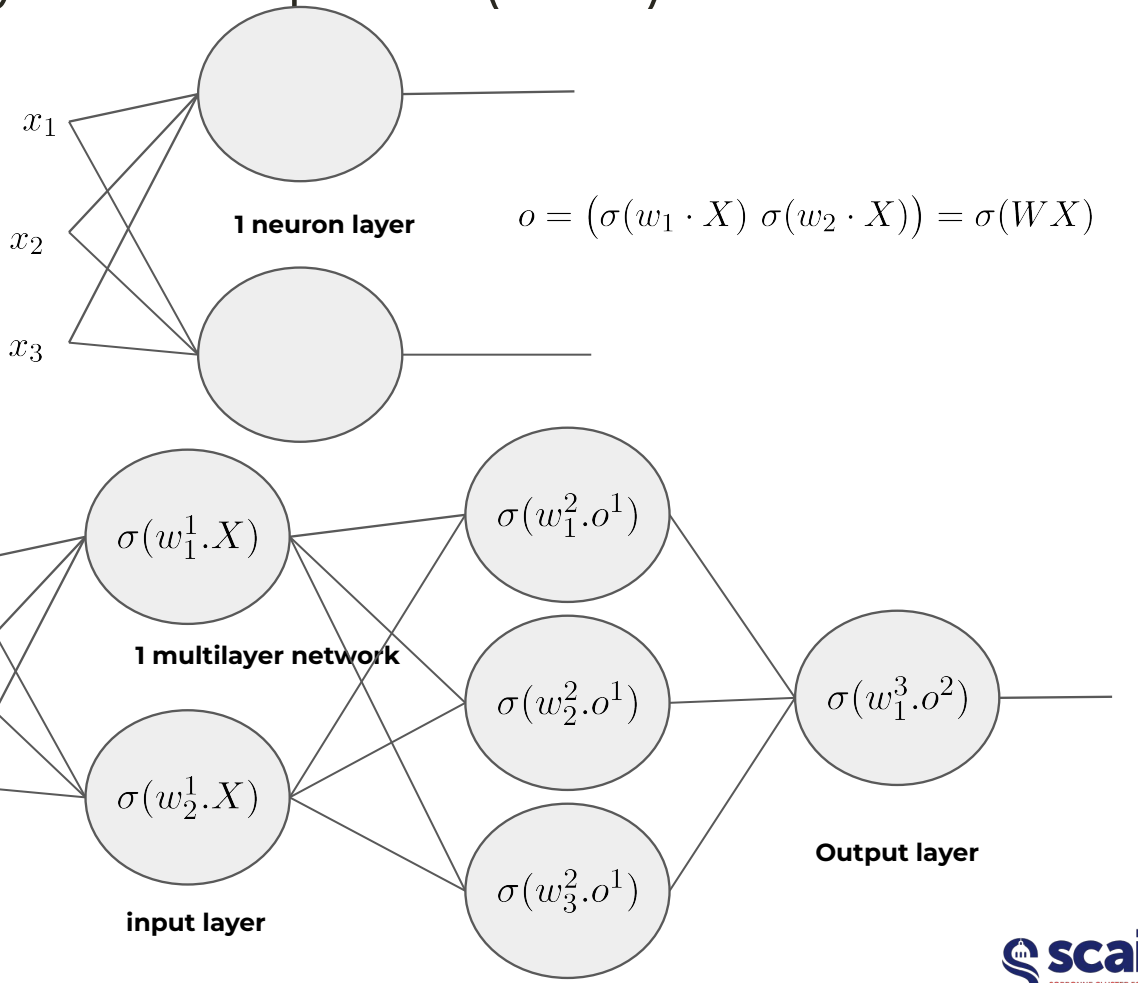
r_k neurons per layer

$\text{total} = \sum_{k=1}^{l} r_k \cdot (r_{k-1} + 1)$

Operation:

$f_\theta(X) = W^l \sigma(W^{l-1} \sigma(\ldots \sigma(W^1 X)))$

Output: number of neuron in last layer



Training Neural Networks

Same objective and training as other parametric model :

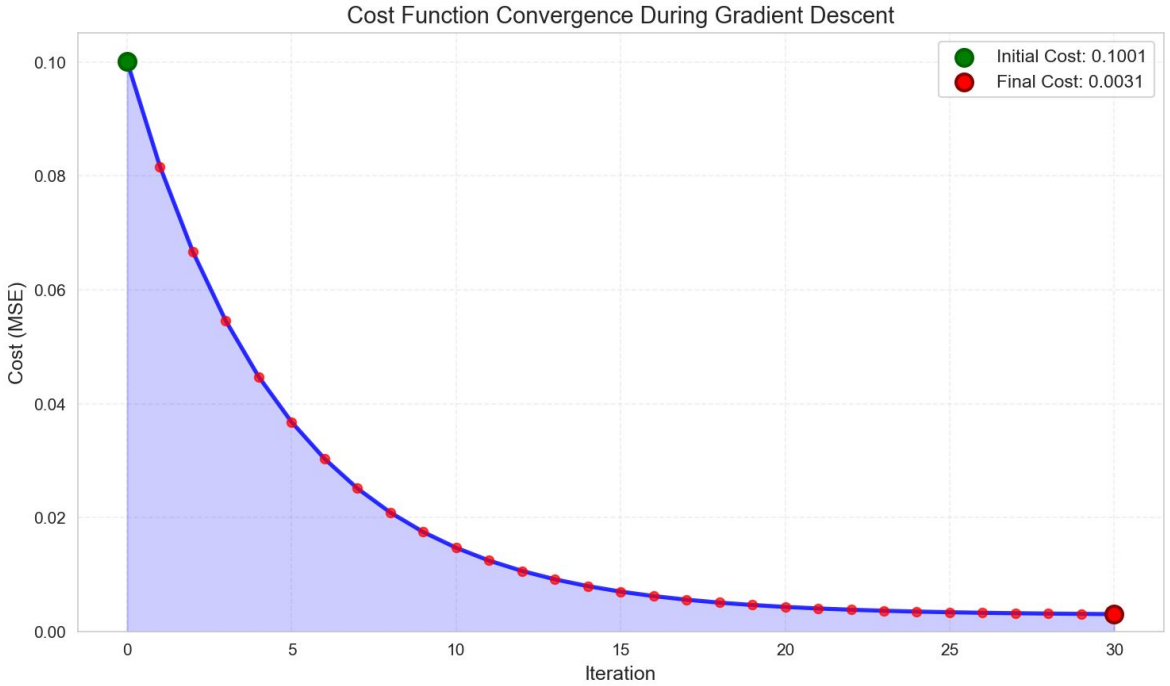
0. initialise the weights randomly

Train:

1. Compute the gradient of the loss
2. Update the parameters using gradient descent
3. Iterate

Predict:

Use the model to predict



$$\arg\min_{\theta \in \mathbb{R}^p} \ell(Y, f_\theta(X))$$

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Backpropagation

Compute directly the gradient of the loss with NN can be taught

$$f_\theta(X) = W^l \sigma(W^{l-1} \sigma(\dots \sigma(W^1 X)))$$

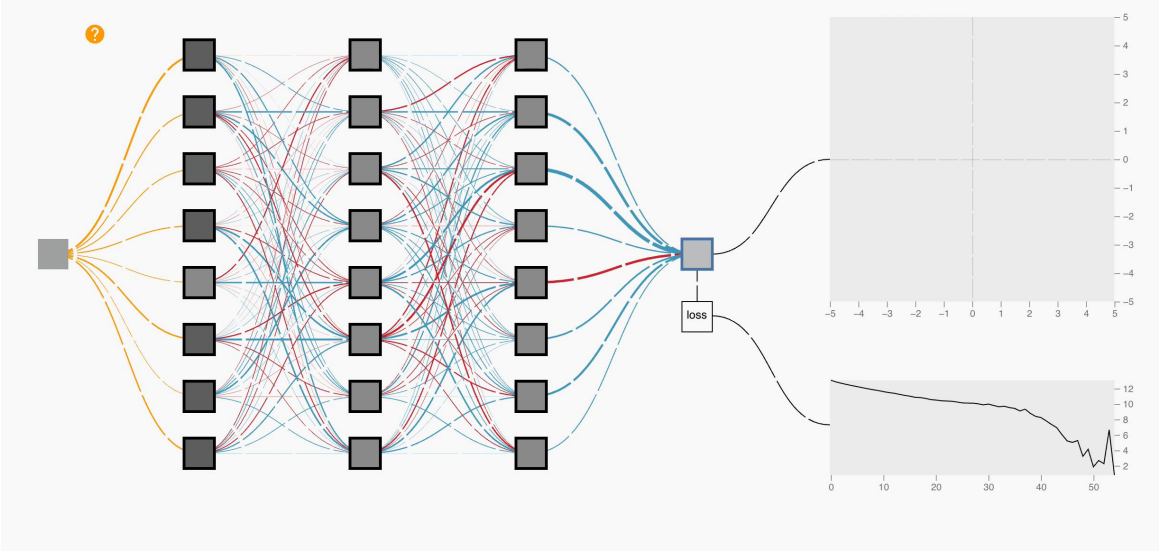
$$\nabla \ell(Y, f_\theta(X)) = \frac{\partial \ell(Y, f(X))}{\partial w_i^k} = ?$$

$$\forall k \in [0, l], \forall i \in [0, r_k]$$

Using chain rules

$$\frac{\partial z}{\partial x} = \sum_i \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

We are able to compute them efficiently



Step 1 : Forward Pass

For each Layer k, compute pre-activation and activation value

Step 2 : Backward Pass

Compute gradient using value from forward and next layer gradient value

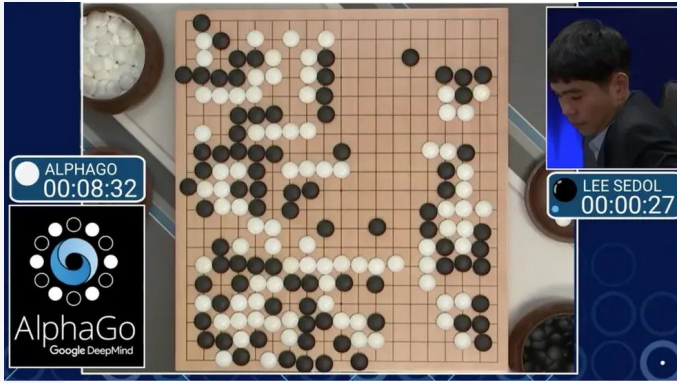
Step 3: Apply Gradient Descent

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Neural Networks became a standard

Key advantages:

- **Modularity:** very varied and adaptable architectures
- **Parallelization:** fast and efficient training
- **Performance:** modeling of complex information



Computer Vision

2012 — AlexNet wins ImageNet by a massive margin (15.3% vs 26.2% error)

Speech Recognition

2012 — Deep neural networks replace GMM-HMMs as the standard for acoustic modeling

Game Playing

2016 — AlphaGo defeats Lee Sedol

Machine Translation

2017 — Transformer architecture ("Attention Is All You Need") becomes the new paradigm

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Data Science

Data: The Fuel of Deep Learning

Volume: Deep networks typically require thousands to millions of training examples.

Quality: Clean, accurate data are crucial - error will reduce strongly model performances.

Bias: Systematic distortions in the data will be learned and reproduced by the model.

Diversity: Training data must capture the full spectrum of real-world scenarios.

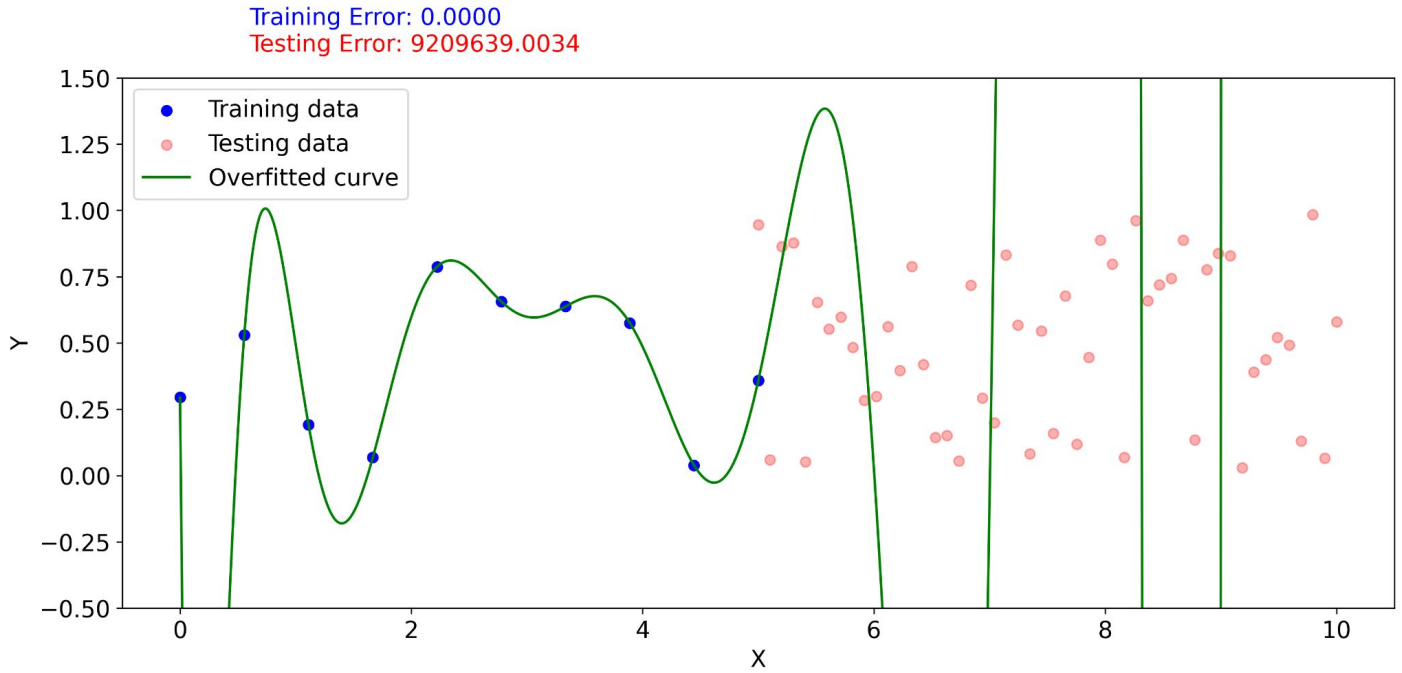
Data Preprocessing

Steps in Data Preprocessing :

- Handle Inconsistencies
- Handle Missing Values
- Handle Categorical Values
- Handle Duplicates
- Handle Outliers
- Feature Engineering
- Scaling and Normalization
- Feature Selection and Dimensionality Reduction

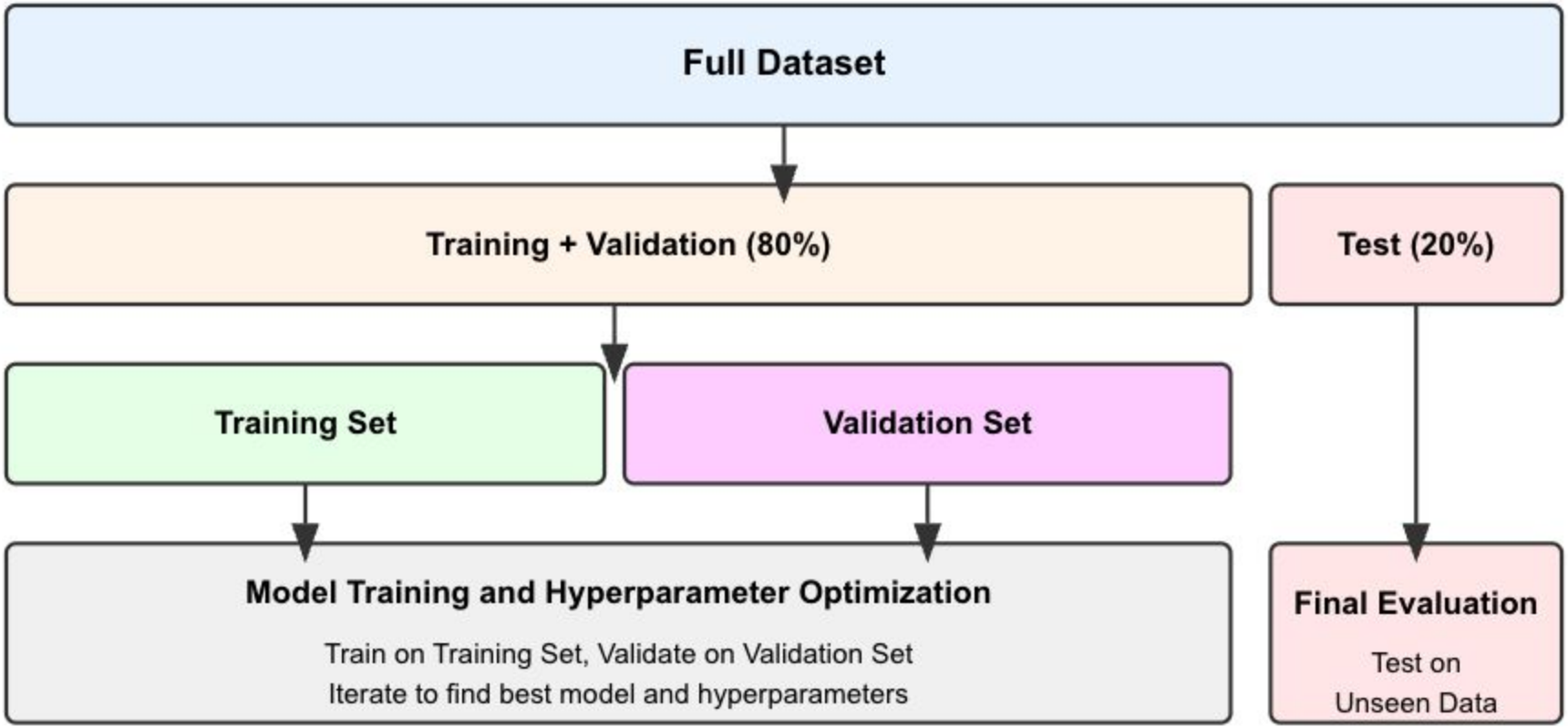
Is the model learning something?

$$\arg\min_{\theta \in \mathbb{R}^p} \ell(Y, f_\theta(X))$$



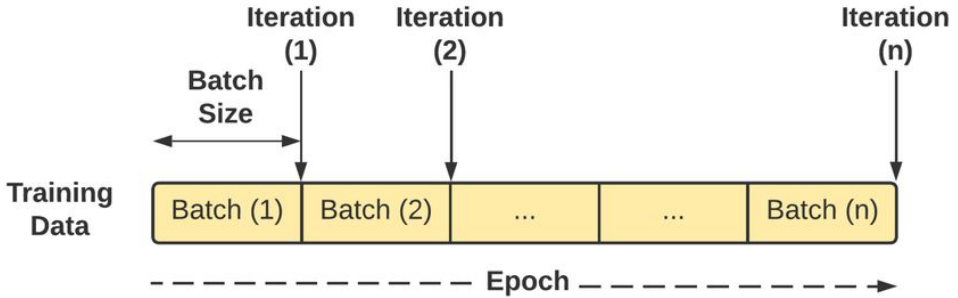
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Model Evaluation

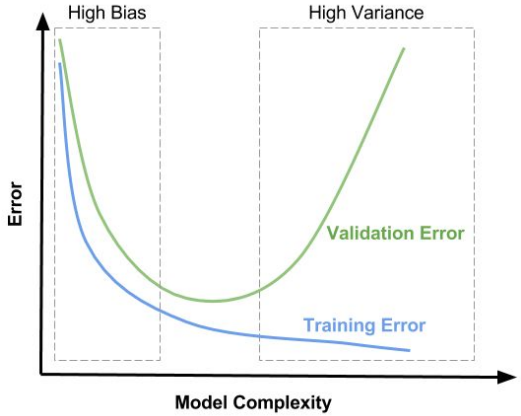


Science of hyperparameters

Hyperparameters: are settings chosen that control how a model learns



Batch size: Number of samples processed before updating model parameters.
Epoch: One complete pass through the entire training dataset.



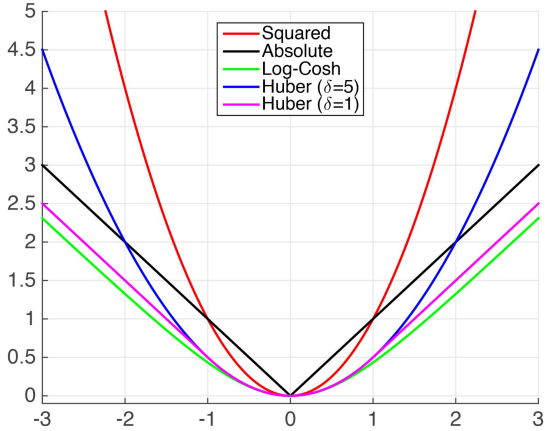
Overfitting: model learns the training data too well, including its noise and peculiarities, resulting in poor generalization to unseen data.

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Loss choice

The Role of Loss Functions

Loss functions quantify the error/distance between real y targets and our model outputs $\hat{y}$



Mean Squared Error (MSE):

$$\ell_{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Absolute Error (MAE):

$$\ell_{MAE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Binary Cross-Entropy:

$$\ell_{BCE}(y, \hat{y}) = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$$

Cross-Entropy Loss:

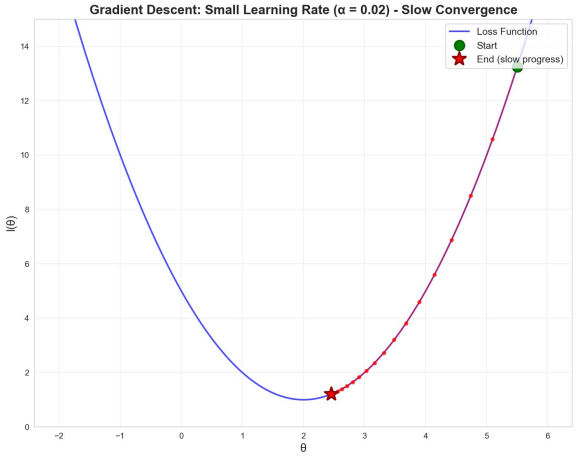
$$\ell_{CE}(y, \hat{y}) = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic})$$

Custom Loss (eg: Focal Loss for unbalanced classification)

$$\ell_{Focal}(y, \hat{y}) = -\alpha(1-\hat{y})^{\gamma} y\log(\hat{y}) - (1-\alpha)\hat{y}^{\gamma}(1-y)\log(1-\hat{y})$$

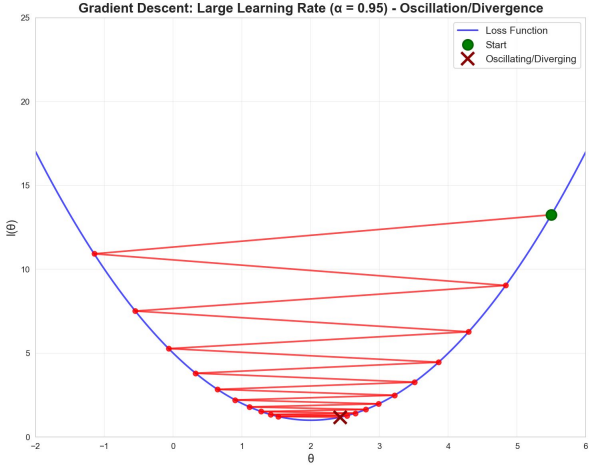
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Learning rate and batch size

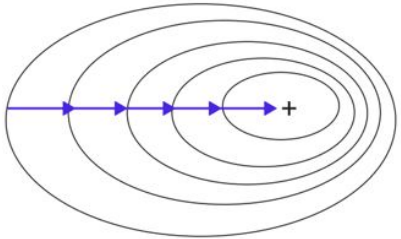


$$\theta^* = \arg\min_{\theta \in \mathbb{R}^p} \ell(\theta) = \arg\min_{\theta \in \mathbb{R}^p} l(Y, f_\theta(X))$$

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta l(\theta_t)$$

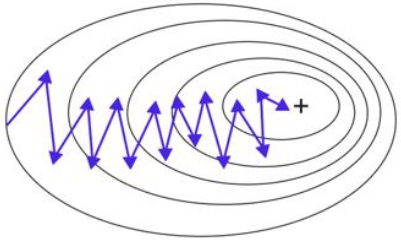


Batch Gradient Descent



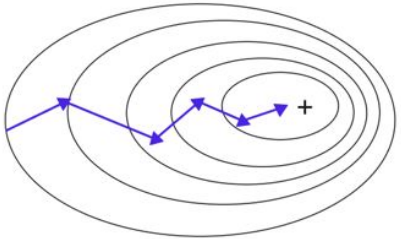
$$\nabla_\theta \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(f_\theta(x_i), y_i)$$

Stochastic Gradient Descent



$$\nabla_\theta \mathcal{L} \approx \nabla_\theta \ell(f_\theta(x_i), y_i)$$

Mini-Batch Gradient Descent



$$\nabla_\theta \mathcal{L} \approx \frac{1}{m} \sum_{i \in \mathcal{B}} \nabla_\theta \ell(f_\theta(x_i), y_i)$$

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Optimizer choice

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

Gradient Descent + Momentum:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L(w_t)$$

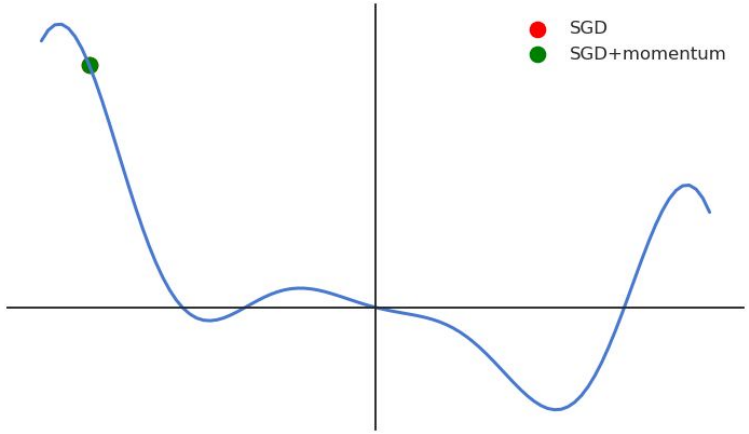
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L(w_t))^2$$

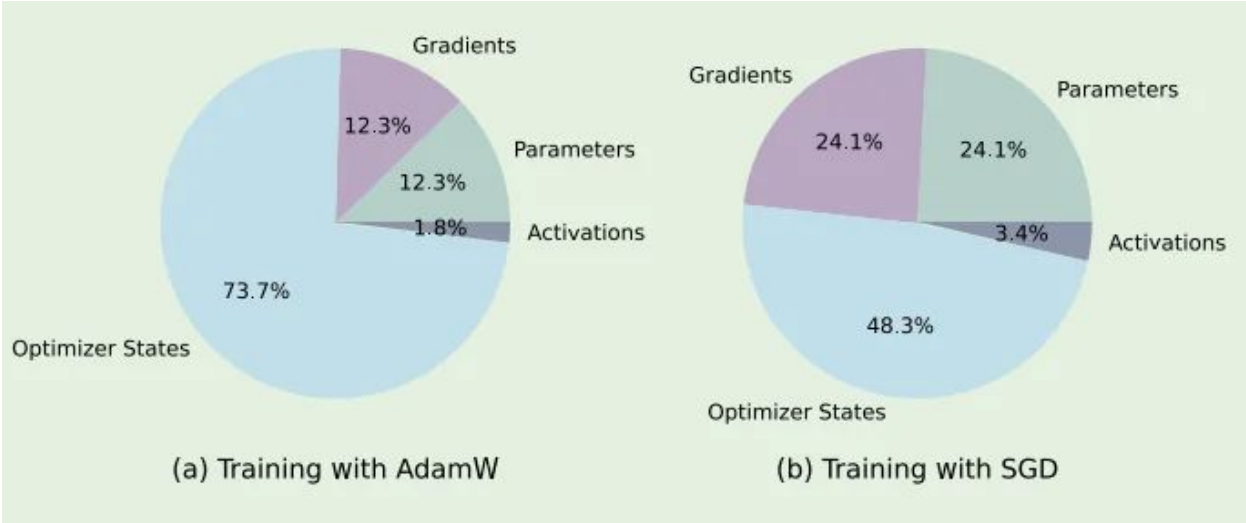
Adam/Adamw:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$w_{t+1} = w_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$





NN unit layers

Linear: $y = xW^T + b$

The fundamental building : Applies a weighted sum of inputs plus bias.

Embedding: $E(i) = W_i = [w_{i,1}, w_{i,2}, ..., w_{i,d}]$

Maps discrete value to dense continuous vectors

Convolution: $y_{i,j,c} = \sum_{m,n,k} x_{i \cdot s+m, j \cdot s+n, k} \cdot W_{m,n,k,c} + b_c$

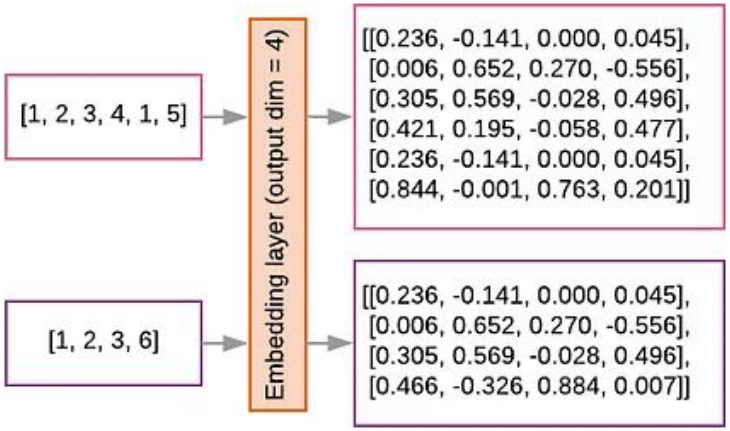
Detect spatial patterns like edges, textures, and shapes

Recurrent: $y^{(t)} = tanh(W_{xh}x^{(t)} + W_{hh}h^{(t-1)} + b)$

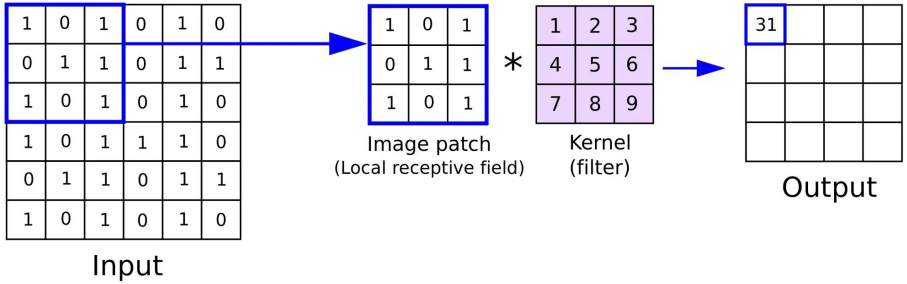
Processes sequences by combining current input with previous hidden state

Attention: $y = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

Computes weighted combinations of values based on query-key similarity



Embedding : categorical data



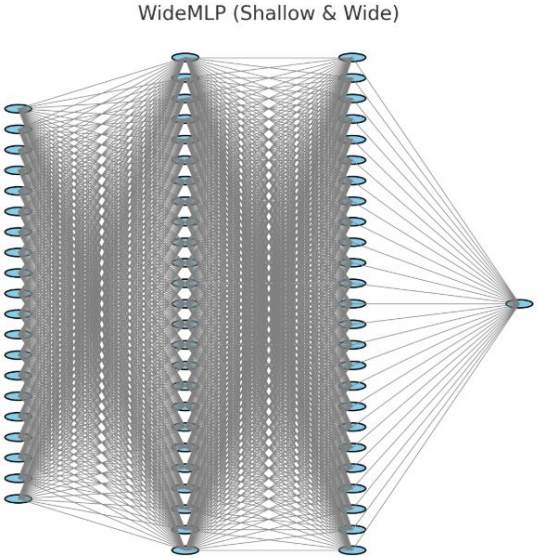
CNN : image data

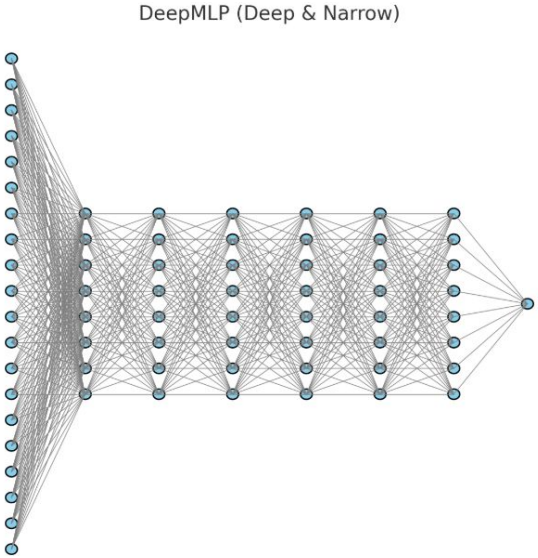
NN architectures

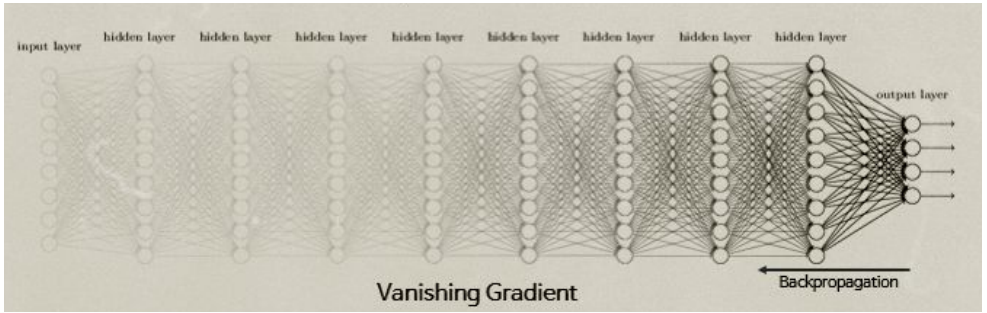
Number of Layers:
Depth of the network; more layers allow learning hierarchical representations but increase training difficulty.

Number of neurons per Layers:
Width of each layer; wider layers capture more features but increase computational cost and risk of overfitting.

Vanishing gradient:
During backpropagation, gradients shrink exponentially through layers, making early layers learn extremely slowly or not at all.





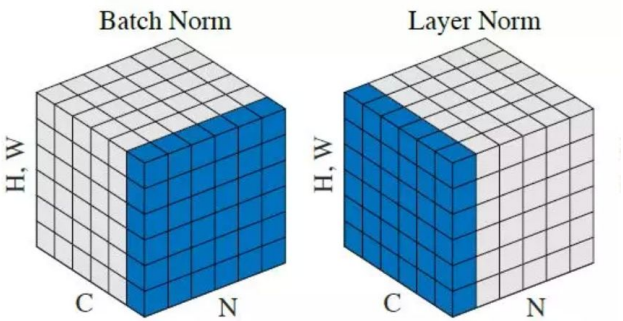


Architectural priors Components

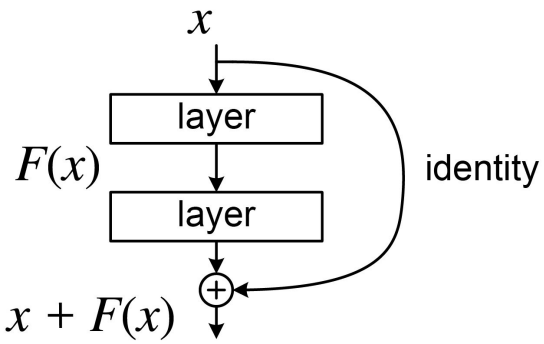
Dropout:
Randomly deactivates neurons during training, forcing the network to learn redundant representations and reducing overfitting.

Layer/Batch Normalization:
Normalizes activations across batches or features to stabilize training, allow higher learning rates, and reduce internal covariate shift.

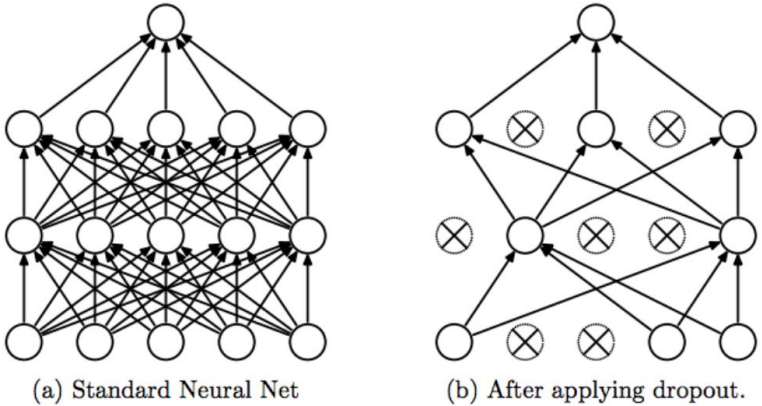
Skip Layer:
Adds the input directly to the layer's output (x + F(x)), enabling gradient flow through deep networks and making it easier to learn identity mappings.



Layer/batch norm



Skip Layer

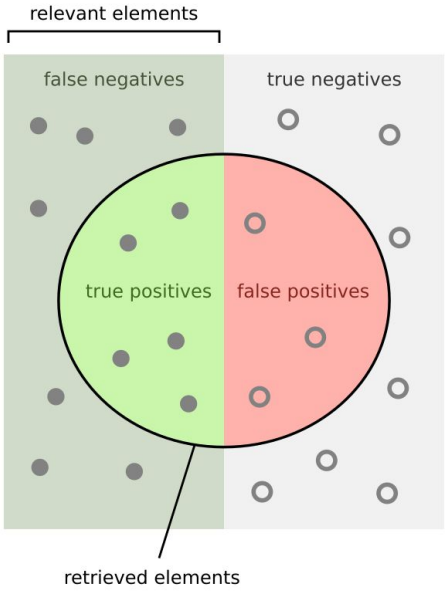


Dropout

Data Science Librairies

Thanks

Metrics



How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{retrieved elements}}$$

How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$