Hands-on Introduction to Deep Learning

Artificial Neural Networks

INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

Objectives of this section:

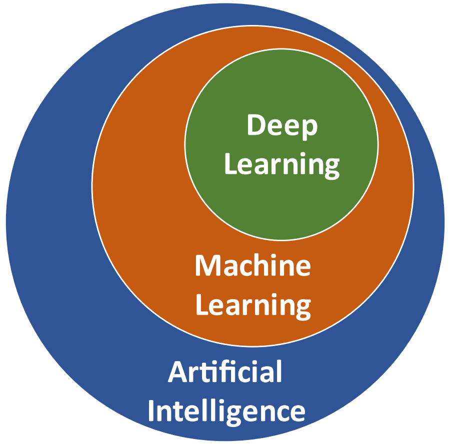
- Understand the origin and development of neural networks
- Master the fundamental functioning of neural networks

Duration : ½ day

Document annex : TP1 Instructions

Aspects addressed :

- Definitions
- Applications
- Machine Learning
- Histoiry
- Context
- Mathematics
- Essentials
- Neurons



Algorithms

Artificial Intelligence (AI) :

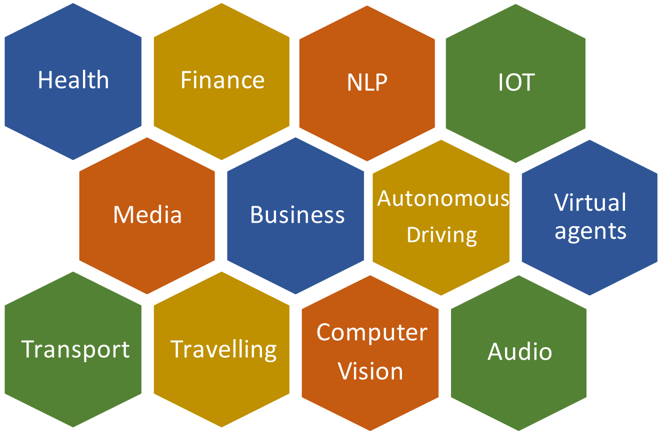
- Systems capable of reproducing human actions or decisions
- A very large field with many algorithm families
- Multiple levels of AI :
  - Narrow : Specialised for one task
  - General : Strong AI, replacing humans
  - Super : Beyond human capacity

Machine Learning :

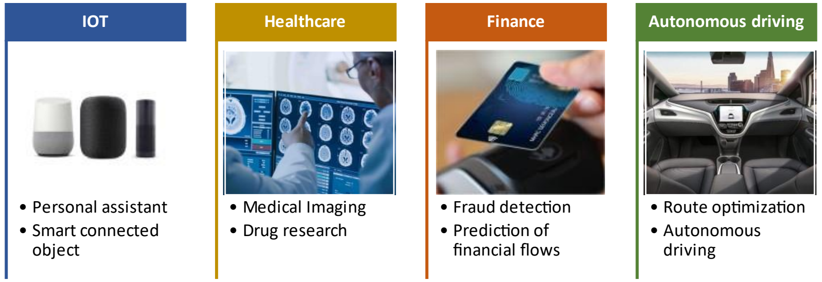
- Algorithms mainly based on statistical methods, often with an iterative aspect from which comes the term « Learning »
- Dependency of large quantities of data
- Modifiable coding of the solution

Deep Learning :

- A group of models based on logical units called neurons and distributed in layers
- The number of layers implies the « Deep » aspect of these models



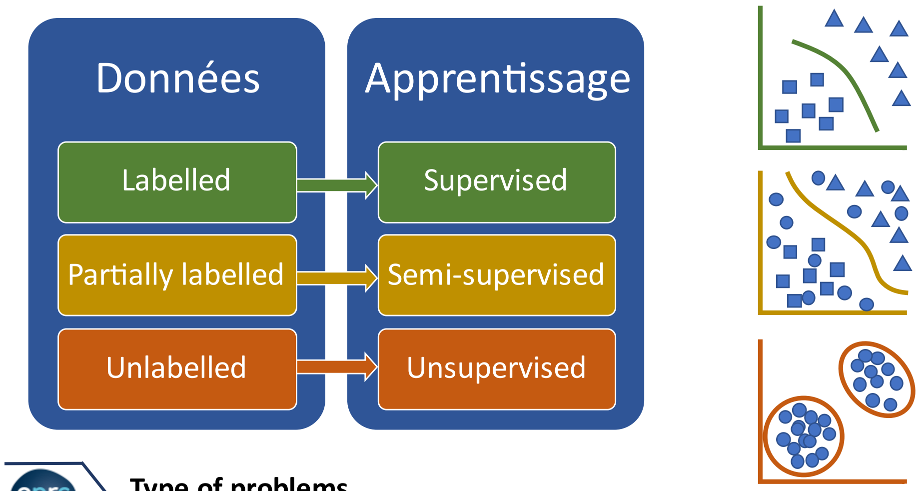
Application fields



Application fields

- Domain driven by successes in industry and research
- Wide variety of activity sectors
- Many different data types
- Datasets with different properties
- Different tasks to accomplish

- Reusable architectures/concepts between different domains due to:
  - Similarities in data and problems
  - Task similarities in different domains



Type of problems

Supervised :

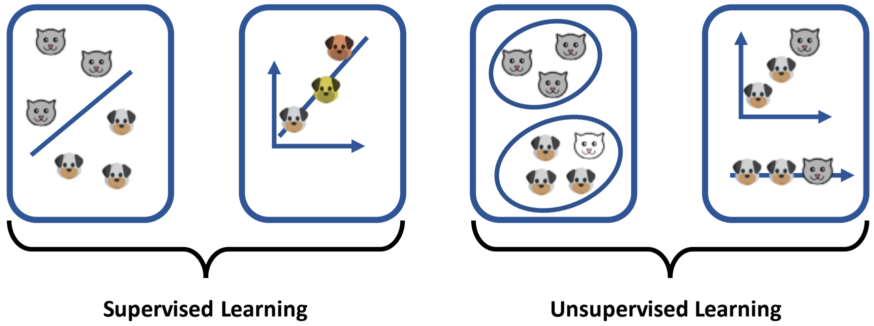
- Data labeling
- Make predictions in a pre-defined solution space
- Learn the characteristics which enable the prediction of labels
- The learned information must be useful for new unlabeled cases
- Difficulty : Creation of the dataset

Unsupervised :

- « Autonomous » learning which aims not to predict information but to extract it by maximising certain criteria (Compression rate, Changing the representation space, ...)
- Difficulty : Evaluating the model and determining which rule to use for optimising

Semi-supervised :

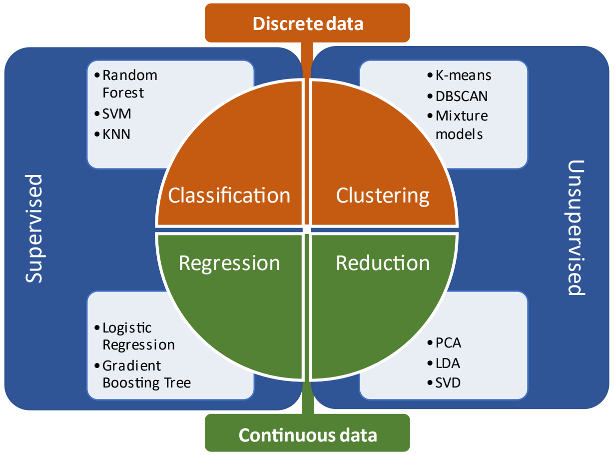
- Using labeled and unlabeled data
- Objective : Reduce the quantity of data to label,
  Improve performance
- Avoiding human bias by using more data and placing more importance on the data than on the labels



Algorithms

Examples :

- Supervised learning :
  - Classification of dog and cat images. The model is trained on a base of tagged images.
  - Prediction of a dog's age from its health data. The model is trained on the health data of dogs with unknown ages.
- Unsupervised learning :
  - Clustering of untagged images. The model is trained to maximise a data separability criterion.
  - Image compression



Algorithms

Types of data :

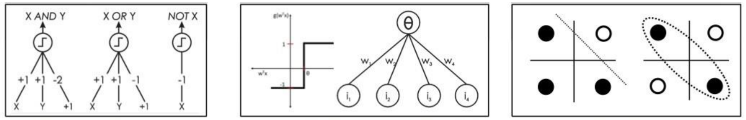
- Continuous
- Discrete

Learning tasks :

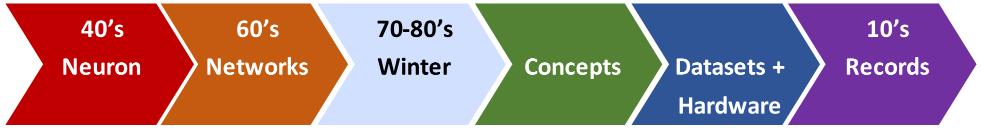
- Classification : Predict one or more discrete outputs (classes)
- Regression : Predict a continuous output in function of the input
- Clustering : Unsupervised non-parametric models, aiming to regroup similar data

  Dimensionality reduction: Reduce the number of data characteristics to compress the information. Overcome the curse of dimensionality (a learning risk).

In certain domains, the specialised tasks combine multiple elementary tasks.



- Adjustable Weights
- Weights are not Learned
- Learnable Weights and Threshold
- XOR Problem



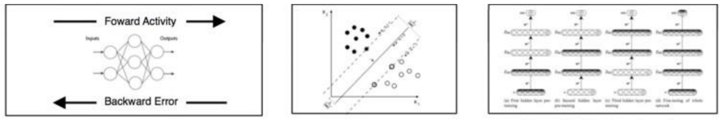
History of Deep Learning

The beginning : 1940

- Computer learning : The Turing test
- Artificial neuron : W.Pitts and S. McCuloch
- Perceptron : Frank Rosenblatt
- ADALINE : Summed single layer neural network
  - Iterative learning
  - Error calculated before activation which serves as the classifier.

The winter of AI : 1974 – 1980

- Inadequte when faced with non-linear problems as simple as XOR
- Lack of accomplishments and progress



- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting
- Limitations of learning prior knowledge
- Kernel function: Human Intervention
- Hierarchical feature Learning

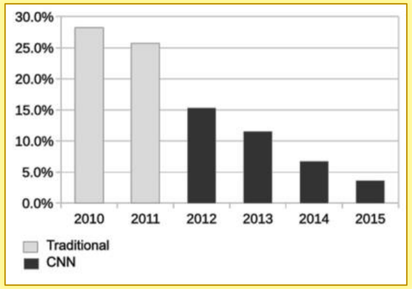
40's Neuron | 60's Networks | 70-80's Winter | Concepts | Datasets + Hardware | 10's Records

History of Deep Learning

Transition to multilayers, new activation functions, optimisers, … each part is incrementally improved.
Models increase in complexity and training becomes difficult.

Second winter of AI : 1987- 1993

- Simultaneously: The SVMs are efficient, effective, mathematical.



40's Neuron | 60's Networks | 70-80's Winter | Concepts | Datasets + Hardware | 10's Records

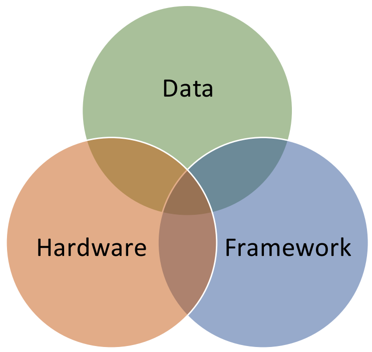
History of Deep Learning

Revolutions :

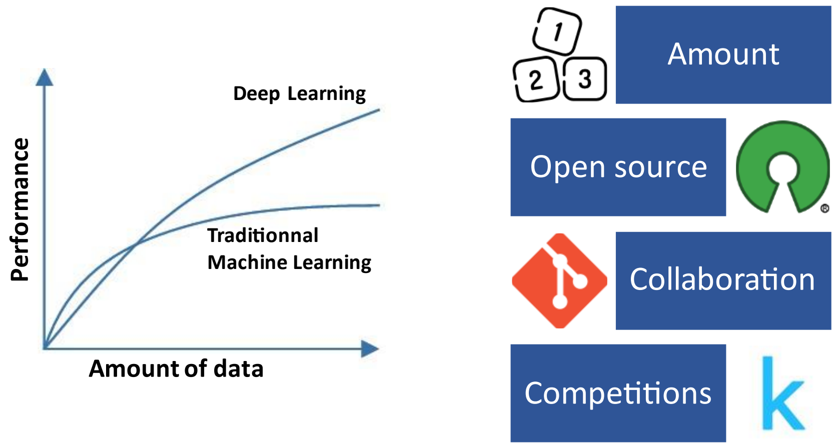
- Transition to shared load (convolutional networks)
- Hardware
- Data

Numerous achievements in Deep Learning :

- Scientific benchmarks
- Industrial successes



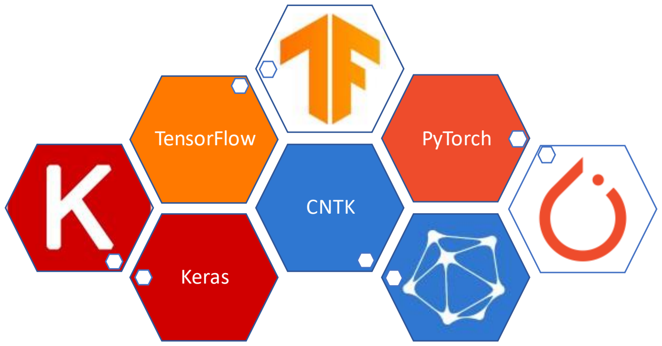
3 decisive factors



Data

Success factors :

- Architectures and models developed : More and more complex due to research, facilitated by libraries
- Enhanced performance due to GPUs after the end of Lisp machines
- Data in abundance to train models and new techniques to label them



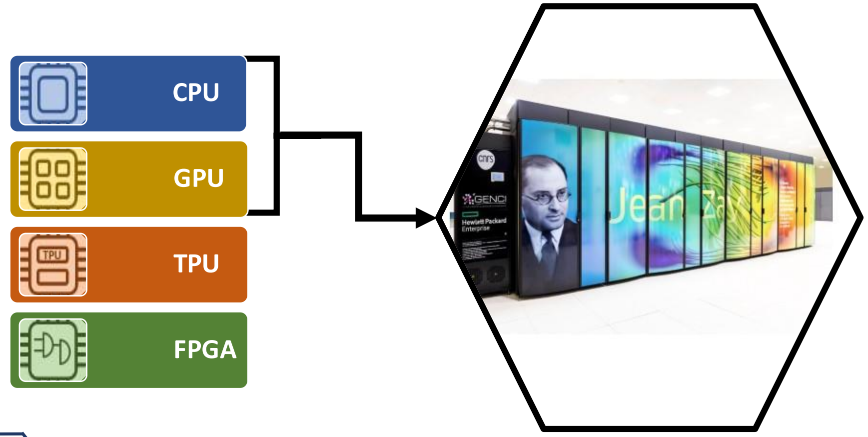
Frameworks and libraries

Various actors :

- Facebook
- Google
- Amazon
- Microsoft
- Academic sector (universities, researchers, …)
- NVIDIA
- On-line communities
- Start-ups
- …

Software layers at different levels :

- GPU integration (Cuda, OpenCL)
- Optimised APIs : Torch, Keras
- Libraries : Pytorch, Tensorflow
- Wrappers : Pytorch Lightning
- Visualisation tools and profiling



Hardware

CPU : Simple development for usage on CPUs

GPU : Specialised for processing images/videos

- Strong parallelisation
- Requires code adaptation / CUDA |OpenCL compatible libraries

TPU : Very effective for vectorial computing

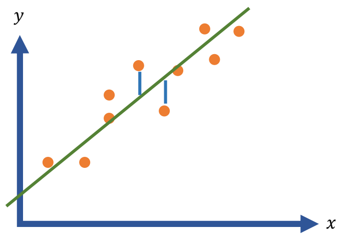
- Optimised for TensorFlow and its operations, so low flexibility
- Effective for large batches

FPGA : Increasingly efficient and usable

- Previously low flexibility : Configured for an application
- Now pre-configured FPGA architectures and optimised for a type of application and compatibility with popular frameworks
- Availability in the cloud

Jean Zay : Supercomputer

- Accelerated nodes (GPU)
- High bandwidth
- Preprocessing nodes



$$Y = X \cdot \Theta + N \qquad \hat{Y} = X \cdot \widehat{\Theta}$$

With :

$\Theta = (a, b)$

N , noise

$$\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Cost function

Convex problem : Direct solution

- $\widehat{\Theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$

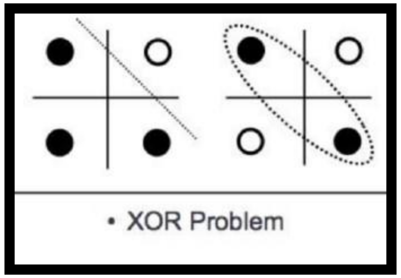
Linear regression

Data can be modelled by a linear expression
Characterise it : Find the weight and bias
Enables making linear predictions

Cost function : Measures the quality of the estimation and indicates to the optimiser how to improve the model

Optimiser : Modifies the model with the objective of minimising the cost function and improving the estimation

Convex problems : There is a direct solution
But there is also an alternative solution : Gradient descent



$$\boldsymbol{f} = \boldsymbol{W}\,\boldsymbol{x}$$

$$\widehat{\boldsymbol{W}} = \boldsymbol{W_2} \cdot \boldsymbol{W}$$
$$\hat{\boldsymbol{f}} = \widehat{\boldsymbol{W}}\,\boldsymbol{x}$$

Non-linear problem?

The exclusive-or (XOR) problem:

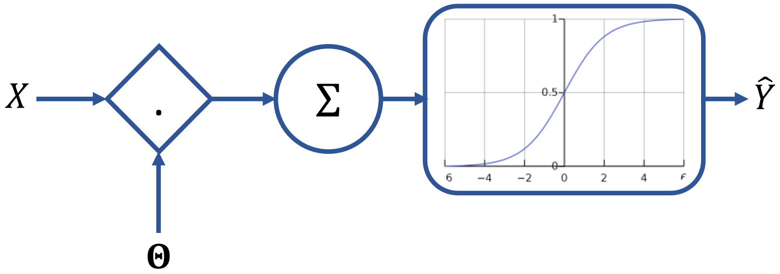
- Non-linear
- But simple

Linear classifier combination = one linear classifier

Requires breaking the neuron linearity :

- Activation function

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)$$ **Cross-entropy loss**

Logistic regression

Break the model linearity : Apply a non-linear function

- Activation function

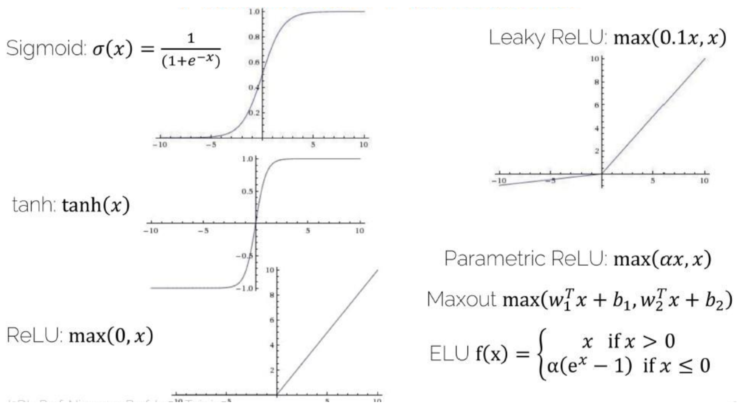
Dual purpose :

- Break the linearity
- Obtain predictions restricted in a sub-space

Example :

- Sigmoid to generate a probability

Loss function : Cross-entropy for the classification



Activation functions

Problems to avoid :

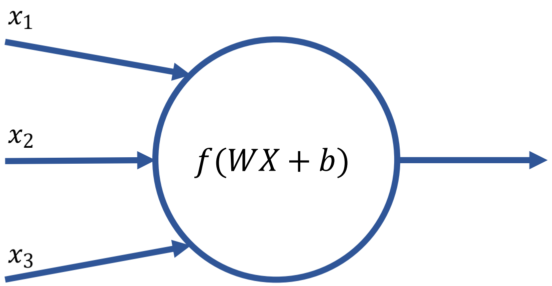
- Vanishing gradients
- Exploding gradients
- Neuron deaths

Mathematical characteristics :

- Range
- Smoothness
- Monotone
- Monotone derivative
- Identity in 0
- Some are more complex and slower to calculate

Various activation functions :

- Linear : Use for a simple regression
- ReLU : Popular, effective, rapid. For very efficient CNNs. Specialises the neurons. Can make some of them useless.
- Softmax : Popular for multi-class classification.



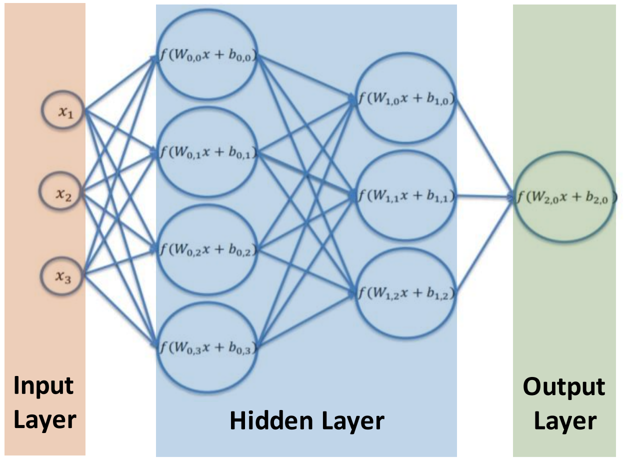
1 neuron

The basic neuron is finally rather simple :

- x : Input
- f : Activation function
- W : Weights (poids)
- b : bias

What is complex :

- The neuron architecture
- The choice of hyperparameters
- The optimiser
- The selection of an appropriate cost function
- Training the model
- Obtaining the data necessary for the training



Neural networks

By assembling the neurons, we obtain a neural network :

- Input layer : The data determine the input dimension
- Hidden layer : To be defined according to the complexity of the problem
  - Depth = Number of layers
  - Width = Number of neurons per layer
- Output layer : The task determines the output dimension

Several questions arise :

- How to size the network?
  - The computer
  - Preliminary study of the data
  - Comparison to similar problems
- How to initialise the neuron weights ?
  - Randomly
  - From a distribution optimising the training
  - From the weights previously calculated for a given problem
- How to choose the cost function ?
- How to optimise the weights ?

- **Linear systems**
  - LU, QR, Cholesky, Jacobi, Gauss-Seidel, CG, PCG, …
- **Non-linear systems**
  - First order : Gradient Descent, SGD
  - Second order : Newton, Gauss-Newton, LM, (L)BFGS
- **Autres**
  - Genetic algorithms, Metropolis-Hastings, …
  - Complex and constrained solver : ADMM, Primal -Dual, …

Optimization

There are several other optimisation methods.

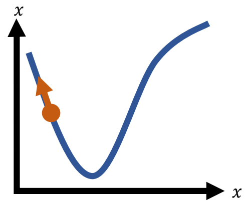
We do not systematically choose gradient descent when the problem is simple.

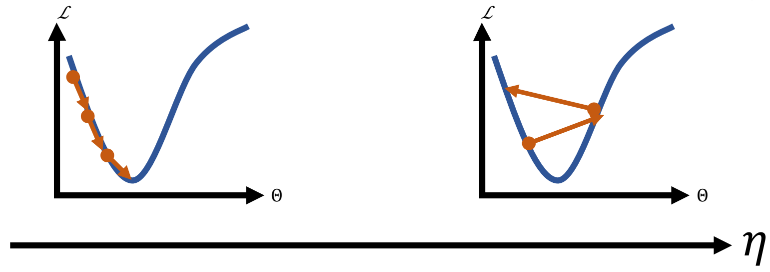
The choice of solver is determined by :

- The type of problem
- The available computing hardware
- Curiosity and scientific experimentation

Iterative solution

- Gradient descent
- $\hat{\Theta}_{t+1} = \hat{\Theta}_t - \eta \nabla_{\theta} \mathcal{L}(\hat{y}_i, y_i)$
- η Learning rate





Gradient descent

Gradient :

- The direction of the largest function increase
- Generalisation for functions with multiple variables from a function derivative of a single variable

Why use the gradient descent ?

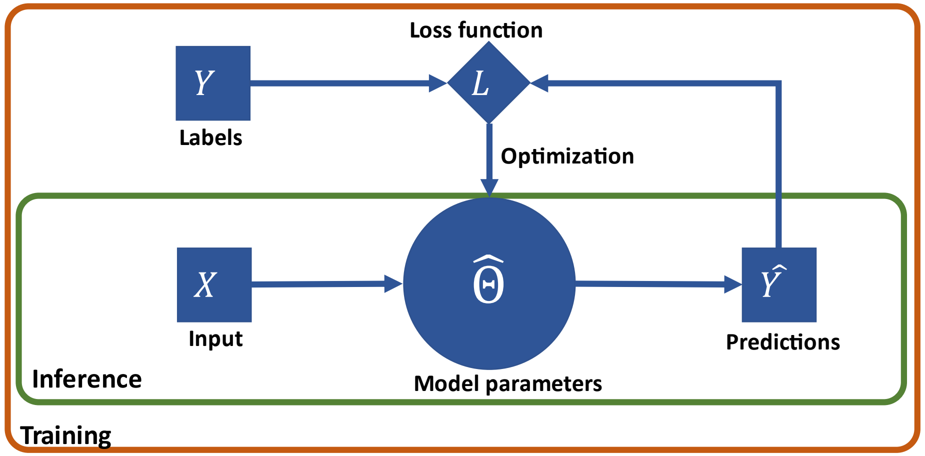
- Some problems do not have a direct solution
- The direct solution can be difficult to calculate

Low learning rate :

- Many iterations to achieve a local minimum
- No guarantee of achieving the optimal minimum

High learning rate :

- Instability and possibility of no convergence



Training a model

Models optimised by gradient descent are used during two different processes :

- Inference
  - Mode of normal functioning while using the model
- Training
  - Mode of updating the model parameters at each iteration
  - Slower
  - Consumes more memory

Generic terms :

- Optimiser : Algorithm which updates parameters
- Loss function : Distance between the label and the prediction
- Cost function : Loss function on multiple data

- **Regression loss**
  - Average absolute deviation : $L(y,\hat{y},\theta) = \frac{1}{n}\sum_i^n |y_i - \hat{y}_i|$
  - Least squares method : $L(y,\hat{y},\theta) = \frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2$
- **Classification loss**
  - Cross-Entropy : $\mathrm{E}(y,\hat{y},\theta) = -\frac{1}{n}\sum_i^n \sum_j^m y_{ij} \log \widehat{y_{ij}}$

Loss function

There is no perfect function for all situations.

Criteria :

- Statistics linked to the database
- Quantity of values outside the norm (outliers)
- Results we are looking for :
  - Regression
  - Classification
  - Number of outputs
  - …
- Several strategies are possible and combinable

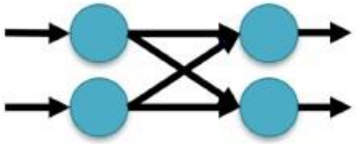
$$\widehat{\Theta}_{t+1} = \widehat{\Theta}_t - \eta \nabla_\Theta \left[\mathcal{L}(\hat{y}_i, y_i) + \lambda R(\widehat{\Theta}_t)\right]$$

L1 : LASSO	L2 : Ridge
$\lvert\Theta\rvert$	Θ^2

Regularization

Possibility of adding restrictions to the updating function to achieve various effects :

- L1 (LASSO) : Focuses neuron attention on certain characteristics
- L2 (Ridge) (weight decay) : Forces the use of all the information
- ElasticNet : Combination of LASSO and Ridge



$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial x}$$

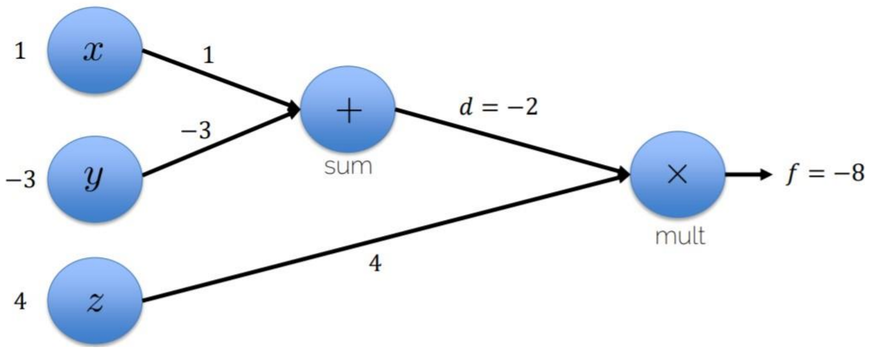
$$\frac{\partial L}{\partial w_{i,k}} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_{i,k}}$$

Calculation graph and chain rule

Calculation graph:

- Includes the nodes
- Includes the edges
- Oriented or not
- Organised in layers, in our case

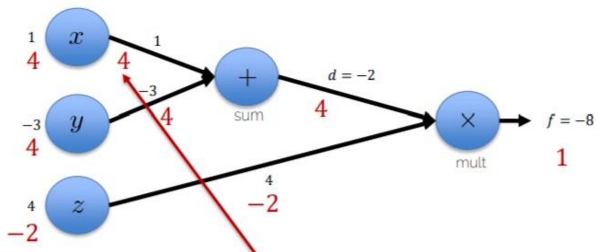
- $f(x,y,z) = (x+y)\cdot z$ Initialization $x = 1, y = -3, z = 4$



Forward pass

$f(x,y,z) = (x+y)\cdot z$

with $x = 1, y = -3, z = 4$



$d = x + y$ $\quad \frac{\partial d}{\partial x} = 1 \quad \frac{\partial d}{\partial y} = 1$

$f = d \cdot z$ $\quad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

Chain Rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial x}$$

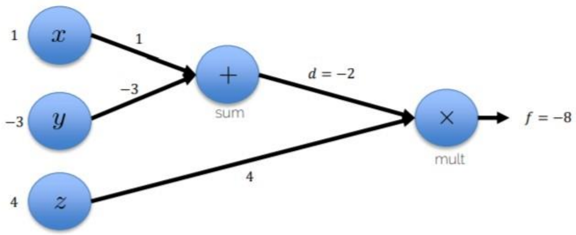
$$\frac{\partial f}{\partial x}$$

$\rightarrow \frac{\partial f}{\partial x} = 4 \cdot 1 = 4$

Backward pass

$f(x,y,z) = (x+y)\cdot z$

with $x = 1, y = -3, z = 4$

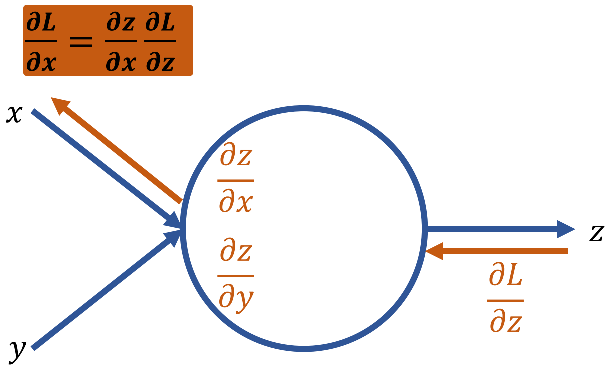


$d = x + y$ $\quad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$

$f = d \cdot z$ $\quad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

What is $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

Backward pass



Gradient Flow

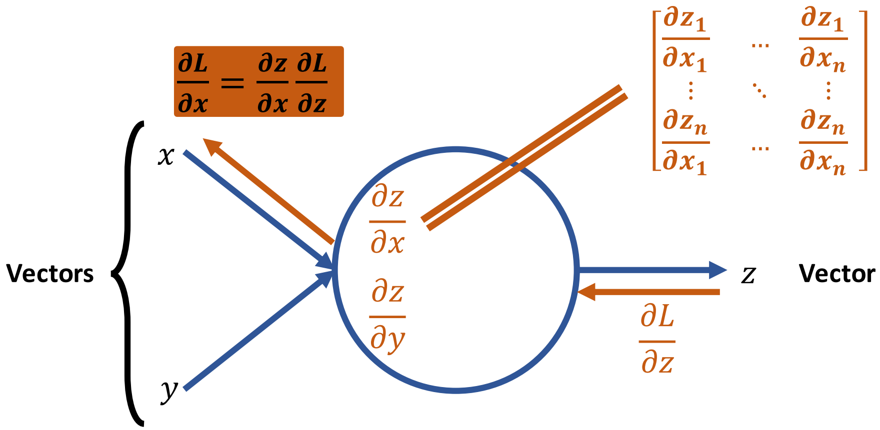
This rule can be applied in a neural network.

Limitations :

- Requires having the activations of each neuron in memory

There are two types of functions in our implementations :

- Forward for the output calculation for a given data
- Backward for back-propagation of the error for weights



Gradient Flow

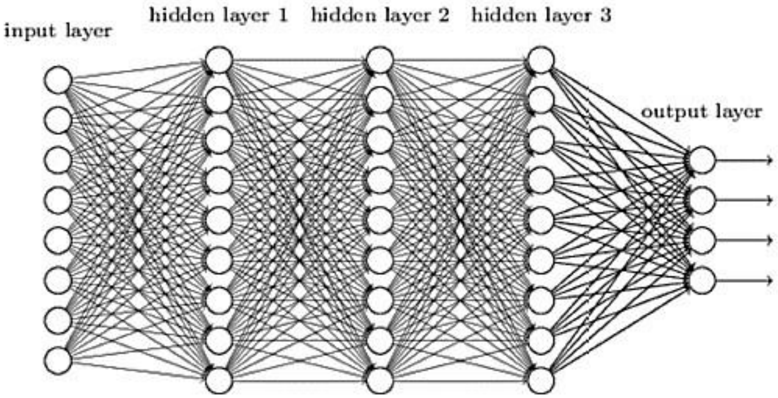
X, Y, Z can be vectors.

Need derivations of each element compared to each of the others :

- Jacobian matrix

What memory occupation ?

- Assuming X, Y, Z of size 4096
- dim(J) = 4096*4096 = 16.78 MiB
- If each variable is a float (4bytes) => 64 MB
- Often the trainings are done by batch. Assuming 16,
  - dim(J) = 16*4096 * 16 * 4096 = 4295MiB =>16GB



Network size

Complex data require complex models.

Complex models imply an explosion of memory occupation in addition to accentuated problems linked to the training (gradient problems).

- Chollet, Francois. *Deep learning with Python*. Simon and Schuster, 2021.
- *CS230 Deep Learning*. cs230.stanford.edu. Accessed 14 Mar. 2022.
- *I2DL*. niessner.github.io/I2DL.Accessed 14 Mar. 2022.
- Goodfellow, Ian,Yoshua Bengio, and Aaron Courville *Deep learning*. MIT press, 2016.

References