INTRODUCTION AU DEEP LEARNING

Notions de base et cas pratique

Gino Frazzoli

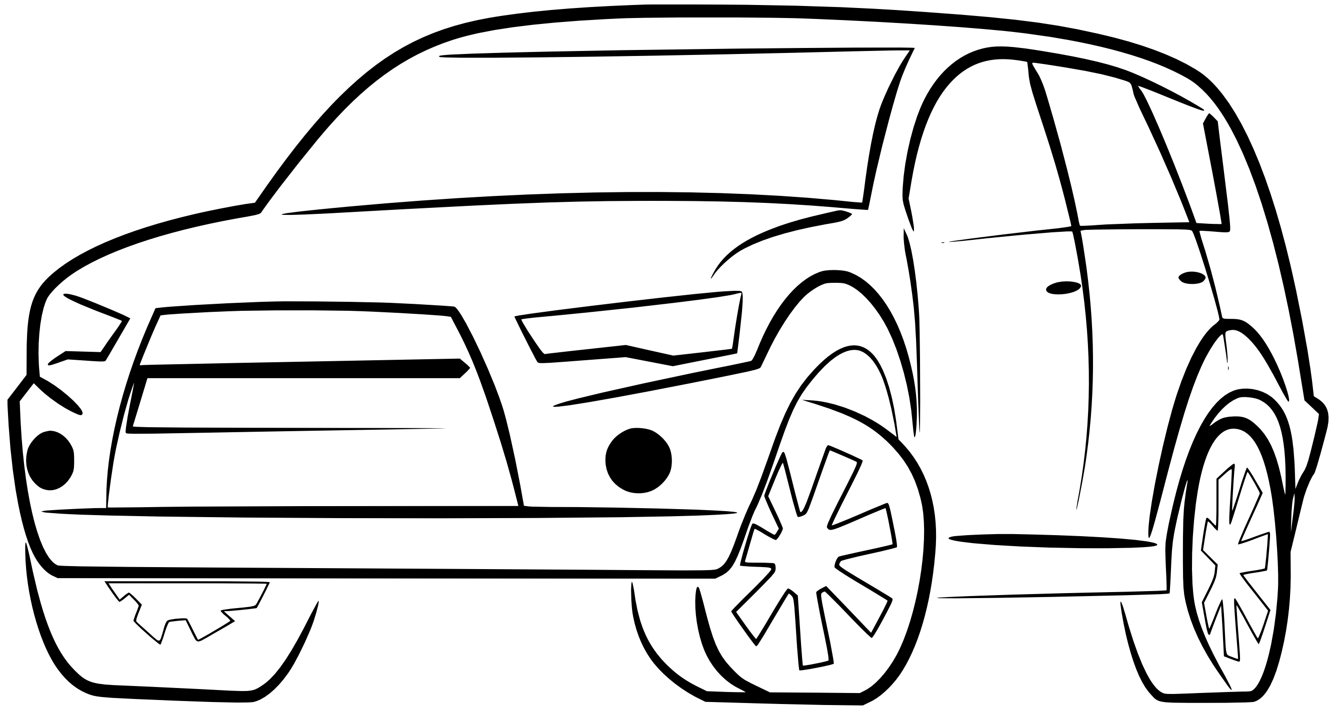
ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

UNIVERSITÉ DE MONTPELLIER

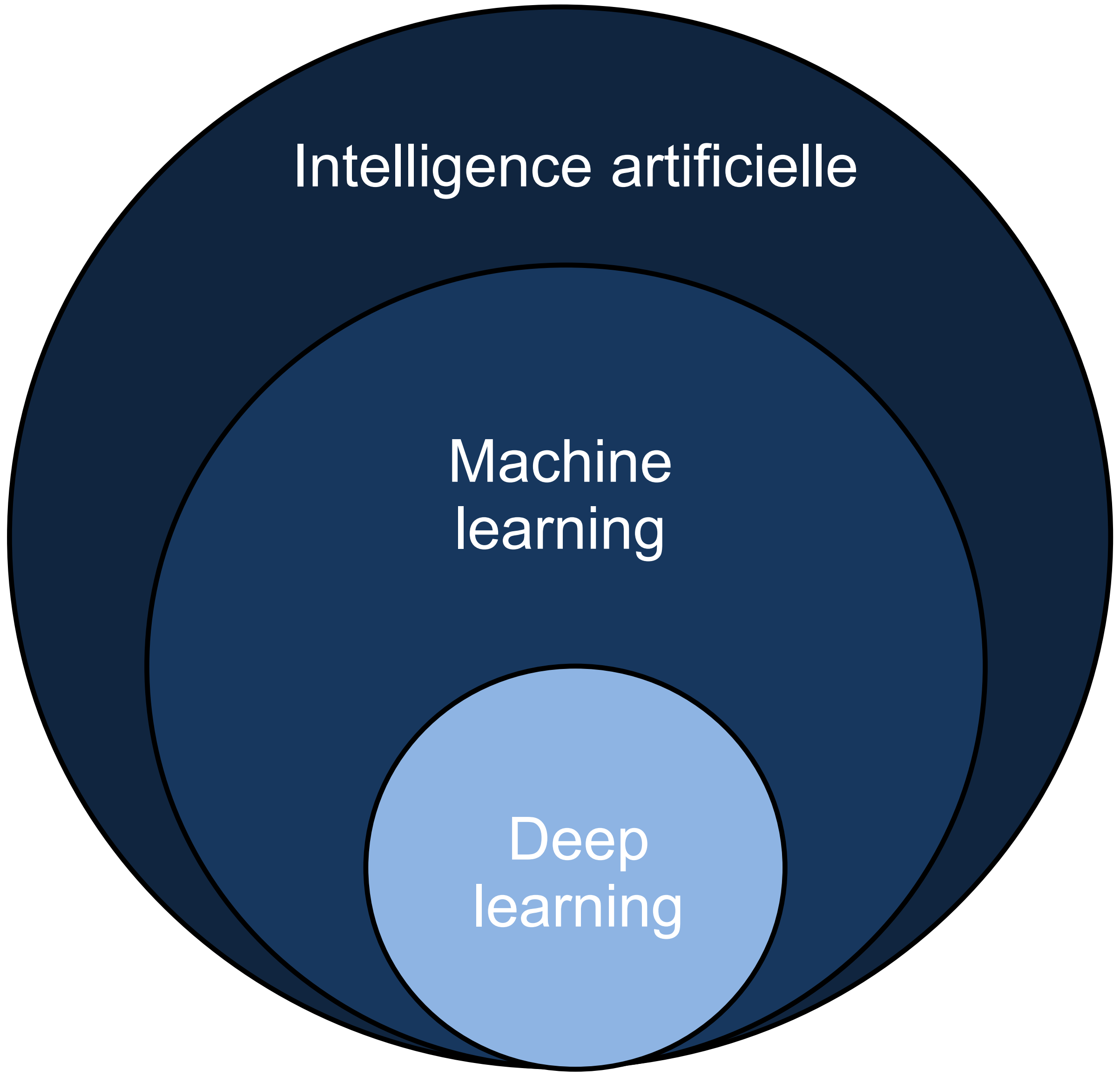
Inserm
La science pour la santé
From science to health

Introduction au deep learning

Le Deep Learning, qu'es aquò ?



Voiture autonome



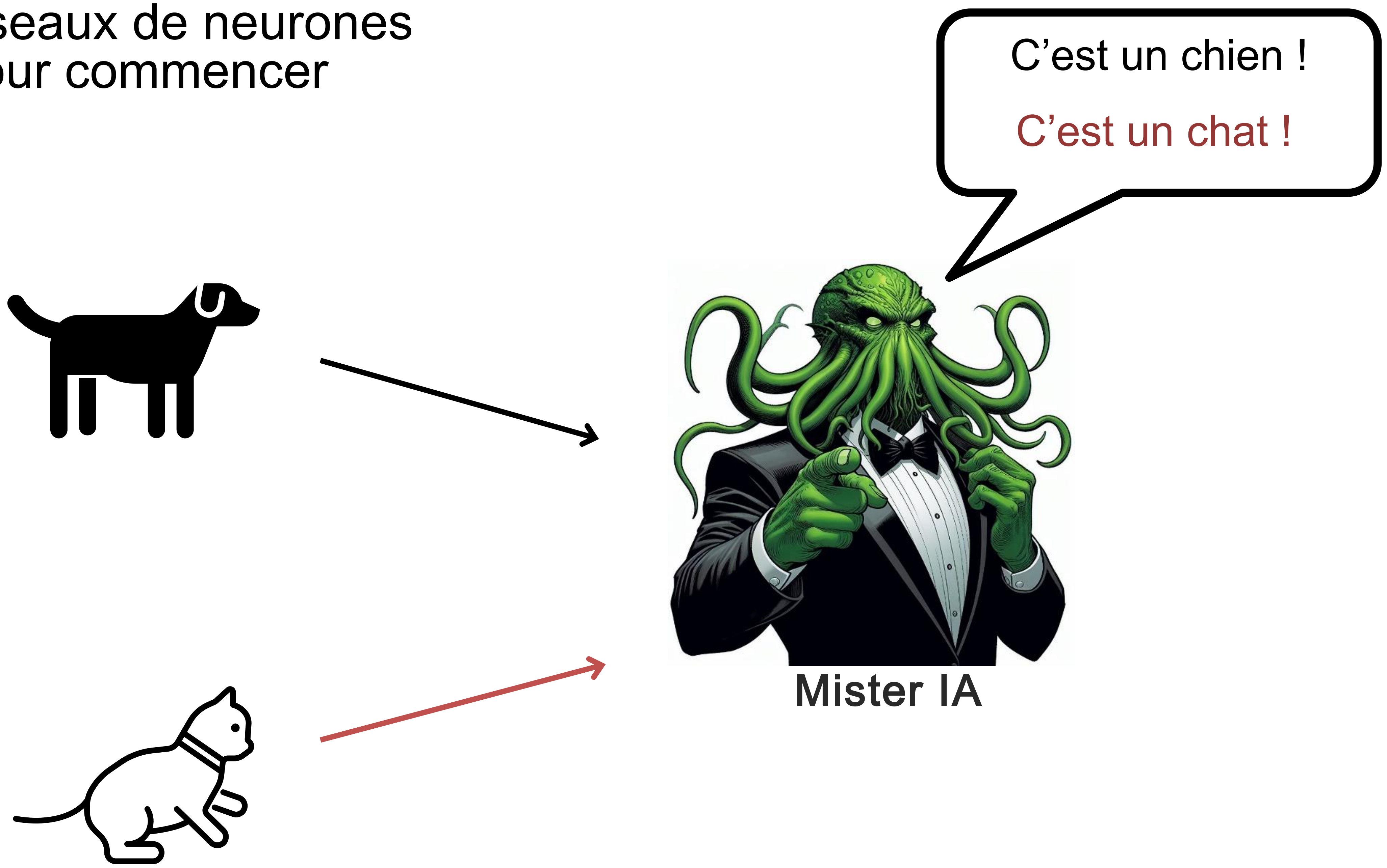
Introduction au deep learning

Objectifs :

- Définir le deep learning
- Saisir la logique derrière les réseaux de neurones
- Aborder un cas (très simple) pour commencer

Cas pratique :

- Classifier des animaux

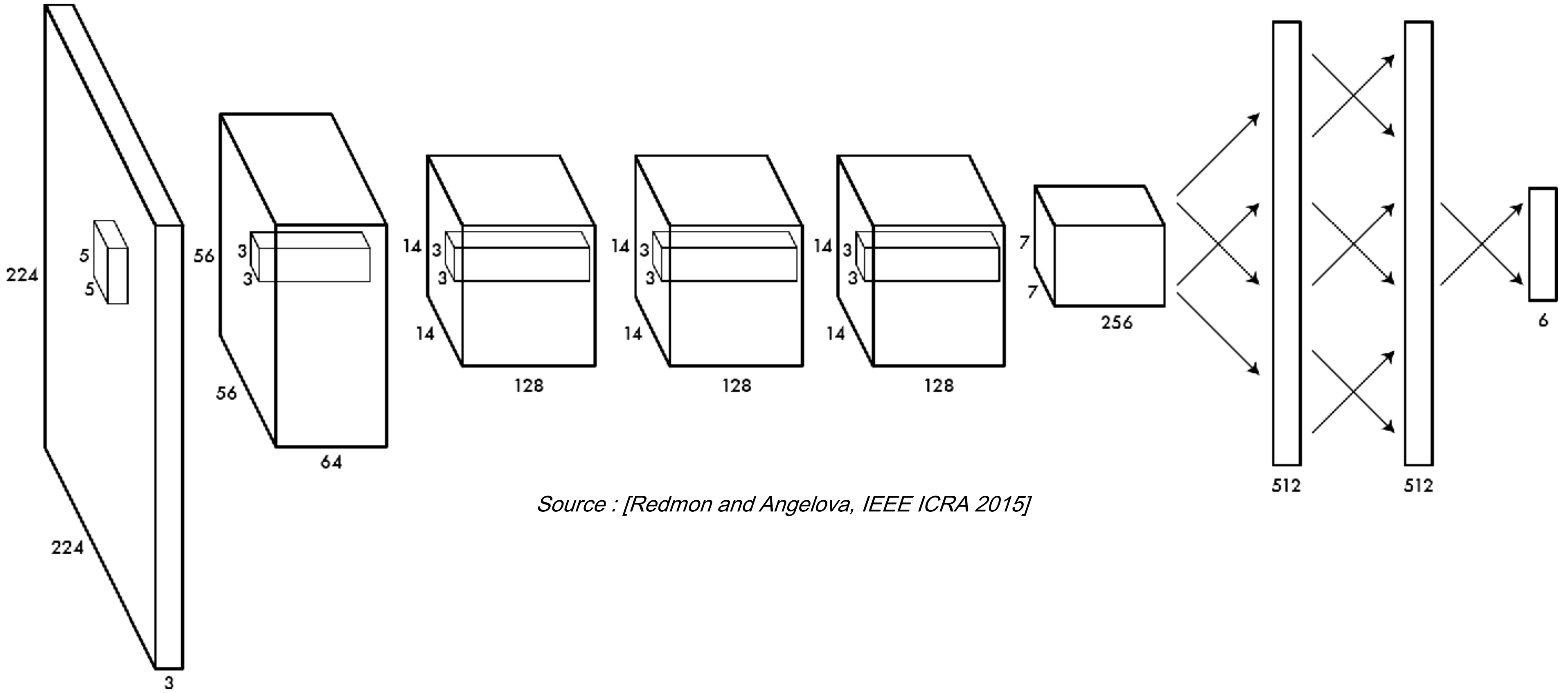


SOMMAIRE

Transformer la representation des données

Un réseau de neurones, c'est quoi ?

Image RVB :
224 x 224



Source : [Redmon and Angelova, IEEE ICRA 2015]

Classes à prédire

Chien	Corbeau	Tortue	Hibou	Aigle	Loutre
0.11	0.05	0.19	0.36	0.12	0.17

Probabilités associées

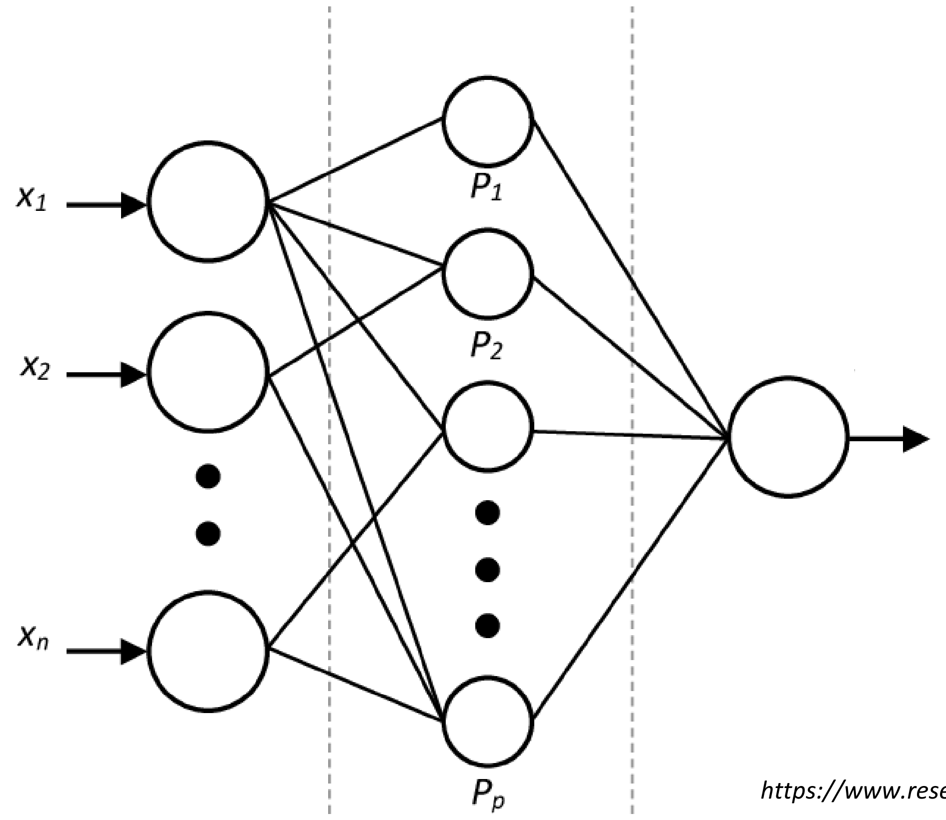
Transformer la representation des données

Brique de base : couche dense

Principe : chaque neurone effectue la somme pondérée de son entrée x_i avec un biais b_i :

$$y_i = \sum^j w_i \; x_i^j + b_i$$

Sortie *Poids* *Entrée* *Biais*



https://www.researchgate.net/profile/Hadley-Brooks

Poids et biais sont des paramètres propres à chaque neurone d'une même couche dense.

Couche dense composée de p neurones :

- neurone P_1 : biais b_1, poids w_1
- neurone P_2 : biais b_2, poids w_2
- …
- neurone P_p : biais b_p, poids w_p

Transformer la representation des données

Brique de base : couche dense

Exemple : notes de classes

élèves	Mathématiques	Sport	S.V.T	Physique	Chimie	LV1	LV2
Gino	12	2	13	18	12	10	9
Marin	2	15	9	10	14	18	19
Arsène	10	20	14	13	11	13	14

Cas 1 : moyenne en matières scientifiques de Gino

- Poids : w_{scient} = ($\frac{1}{4}$, 0 , $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{4}$, 0 , 0)
- Entrée : x_{Gino} = (12 , 2 , 13 , 18 , 12 , 10 , 9)
- Biais : $b_{Gino\ m.scient}$ = 0

$y_{Gino\ scient} = \mathrm{w} * \mathrm{x} + \mathrm{b}$

= (0,25 * 12) + (0,25 * 13) + (0,25 * 18) + (0,25 * 12)
= 13,75

Gino a une moyenne scientifique de 13,75

$$y_{Gino\ scient} = \sum_1^7 w_{scient}^i \, x_{Gino}^i + b_{Gino\ m.scient}$$

Comment sélectionner les informations qui nous intéressent uniquement ?

Transformer la representation des données

Brique de base : couche dense

élèves	Mathématiques	Sport	S.V.T	Physique	Chimie	LV1	LV2
Gino	12	2	13	18	12	10	9
Marin	2	15	9	10	14	18	19
Arsène	10	20	14	13	11	13	14

Cas 2 : sélection des moyennes en sport supérieures à 10,
à l'aide du biais

- Poids : w_0 = (0 , 1 , 0 , 0 , 0 , 0 , 0)
- Entrée : x_{Gino} = (12 , 2 , 13 , 18 , 12 , 10 , 9)
  x_{Marin} = (2 , 15 , 9 , 10 , 14 , 18 , 19)
  $x_{Arsène}$ = (10 , 20 , 14 , 13 , 11 , 13 , 14)

- Biais : b = -10

$s_i = \sum^j w_j \; x_i^j + b$ →

Gino	-8
Marin	5
Arsène	10

$y_i = \max(\, 0 \,, s_i \,)$ →

Gino	0
Marin	5
Arsène	10

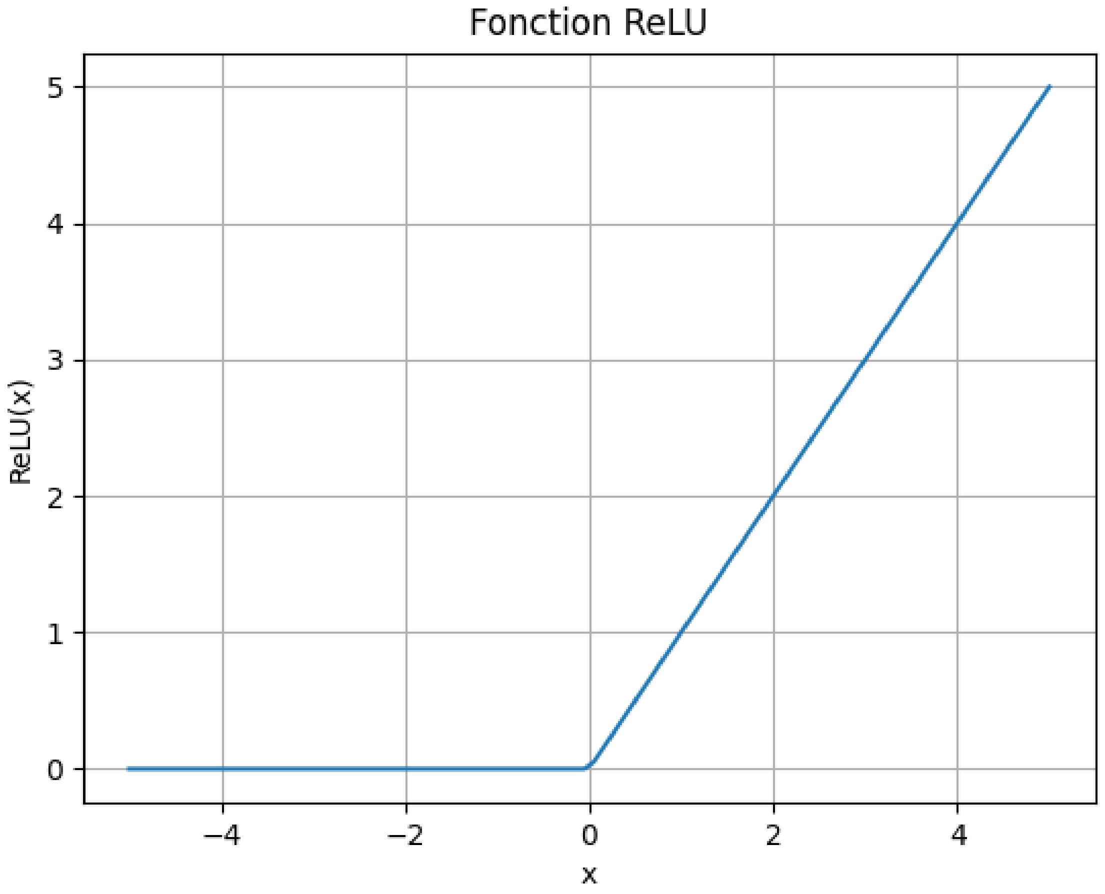
La fonction $\boldsymbol{y_i}$ a pour rôle de mettre à zéro l'information inutile.
Dans notre cas, on ne veut pas garder les moyennes négatives.

On nomme cette fonction ReLU.

Transformer la representation des données

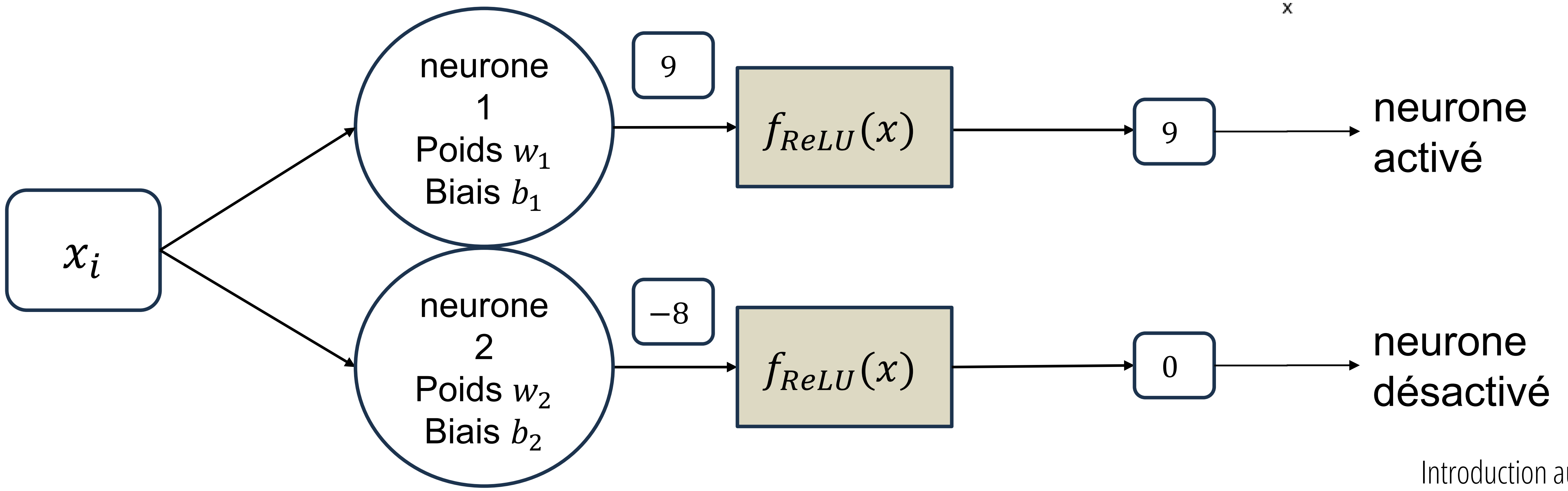
Brique de base : fonction d'activation

- Sélectionner les données, en déterminant si un neurone artificiel doit être activé ou non , et si oui à quel degré ?



Fonction d'activation vue précédemment :

- $f_{ReLU}(x) = \begin{cases} 0 \; si \; x \leq 0 \\ x \; si \; x > 0 \end{cases}$



Transformer la representation des données

Brique de base : fonction d'activation

Introduire de la non-linéarité entre nos couches de neurones

➔ Modifier la représentation spatiale des données

On compare deux nombres

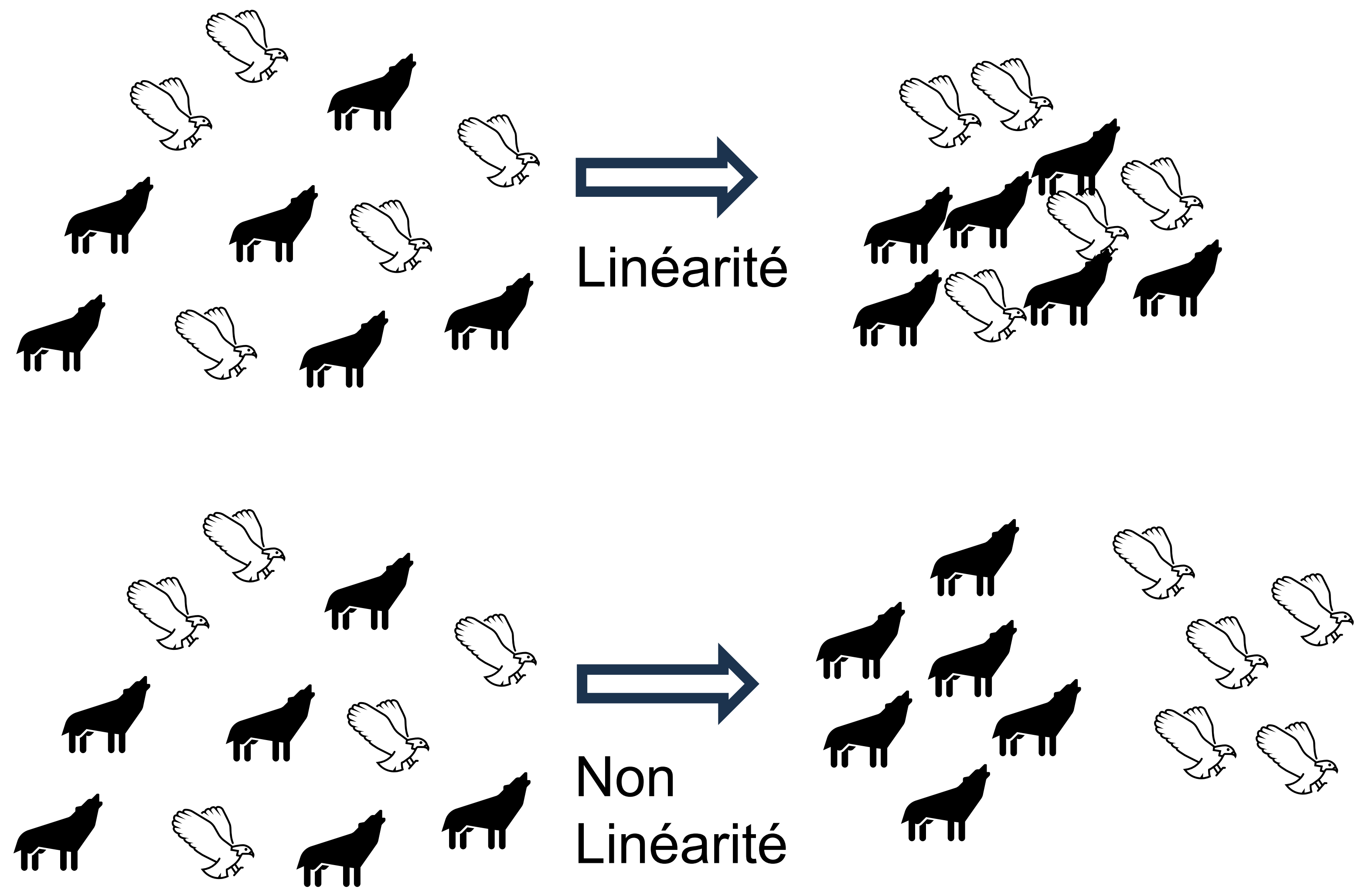
60 < 90

$f(x) = 2 * x$

120 < 180

$f(x) = \cos(x)$

0,5 > 0

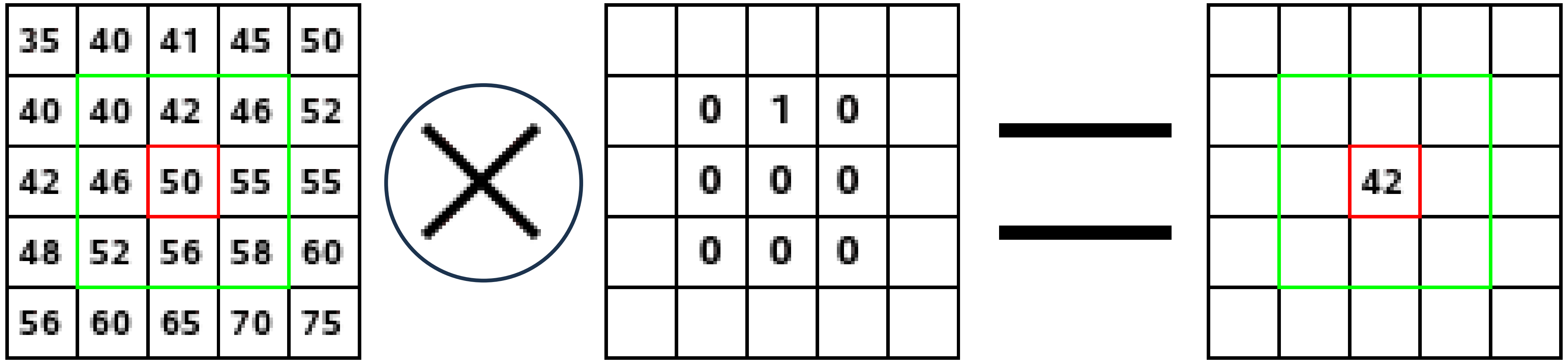


- Perspective différente
- Problème non-linéaire ↔ Solution complexe

Transformer la representation des données

<u>Brique de base</u> : couche de convolution

Filtres de convolution en traitement d'images

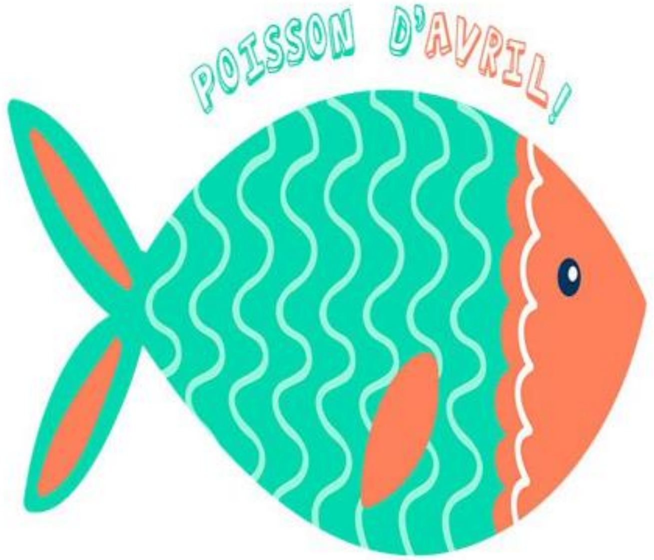


⊗ : produit de Hadamard, multiplication terme à terme : on applique le filtre à chaque pixel (et ses voisins)

Transformer la representation des données

Brique de base : couche de convolution

Exemples de filtres :

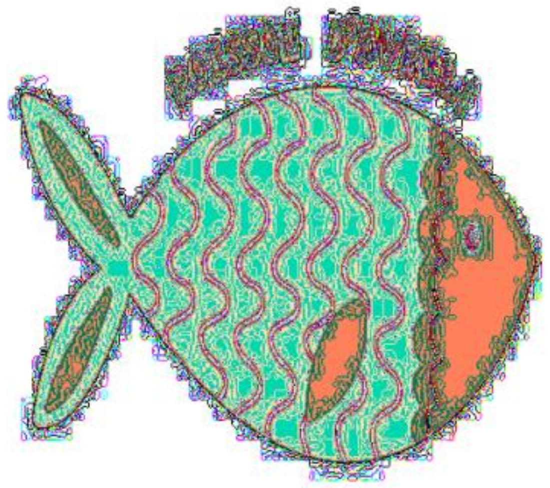


Source : https://www.familiscope.fr/

Augmentation contraste → $\otimes$

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

→



Filtre flou → $\otimes$

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

→

$\otimes$: produit de Hadamard, multiplication terme à terme : on applique le filtre à chaque pixel (et ses voisins)

Transformer la representation des données

<u>Brique de base</u> : couche de convolution

Exemples de filtres :

Repoussage $\otimes$ $\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$

Renforcement des bords $\otimes$ $\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$\otimes$: produit de Hadamard, multiplication terme à terme : on applique le filtre à chaque pixel (et ses voisins)

Transformer la representation des données

Brique de base : couche de convolution

Repoussage

$\otimes$

-2	-1	0
-1	1	1
0	1	2

ReLU

Transformer la representation des données

Brique de base : couche de convolution

Rajout du biais et de la ReLU

Repoussage

⊗

-2	-1	0
-1	1	1
0	1	2

Biais = 0
+ ReLU

Sélection de l'information sur les pixels de l'image

Biais = -0,2
+ ReLU

Biais = - 0,5
+ ReLU

Transformer la representation des données

Brique de base : couche de convolution

Principes d’une couche de convolution :

- Extraire l’information
- Identifier des paternes dans les données
- Propriété d’invariance par translation : le réseau reconnait un paterne quelle que soit sa position dans l’image

Image d’entrée

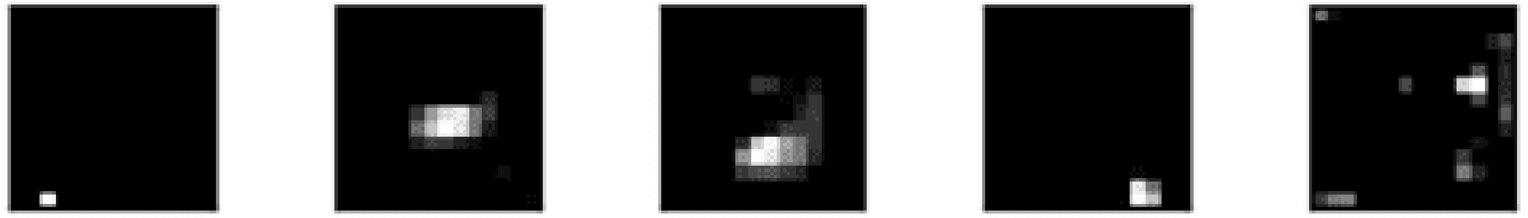
Source : Robin, by Chris Heald

Couche de convolution

Sortie 1^{e} couche

Sortie 2^{e} couche

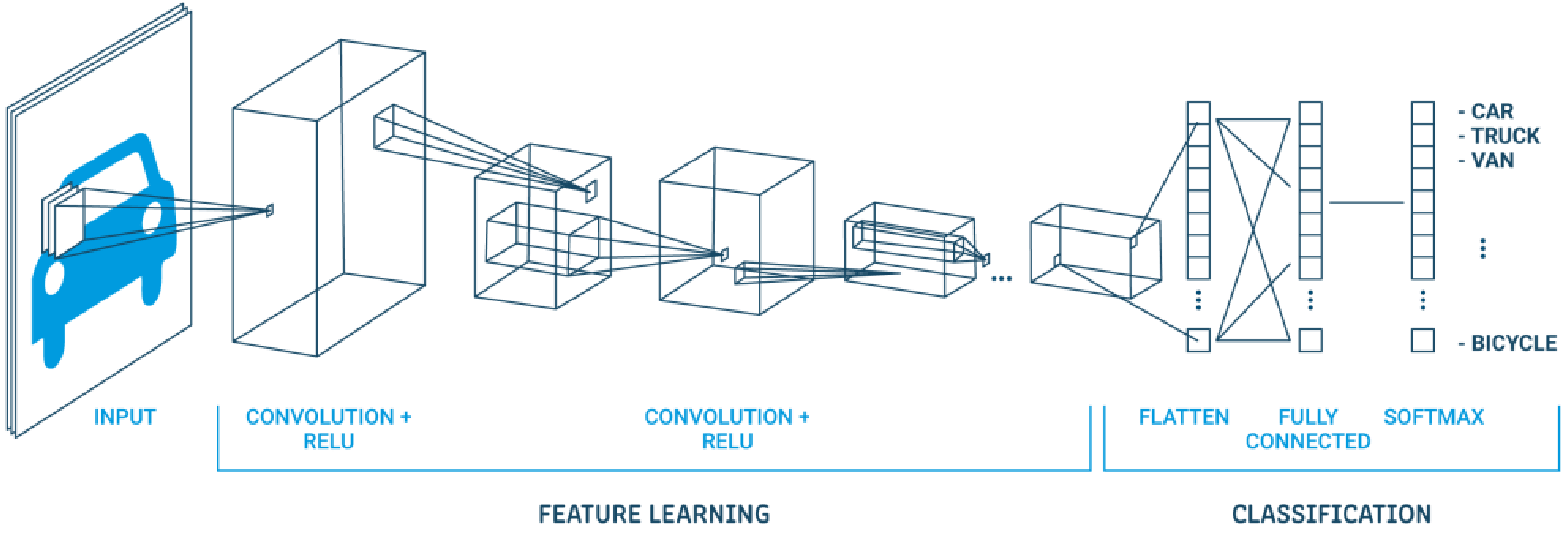
Sortie $n^{ième}$ couche



➔ *Features map*

Transformer la representation des données

Assemblage de nos briques



Source : https://towardsdatascience.com/

Transformer la representation des données

Mais comment notre réseau connait ces fameux poids, filtres, biais ?

Réseau incompétent

?

Réseau entraîné

SOMMAIRE

02

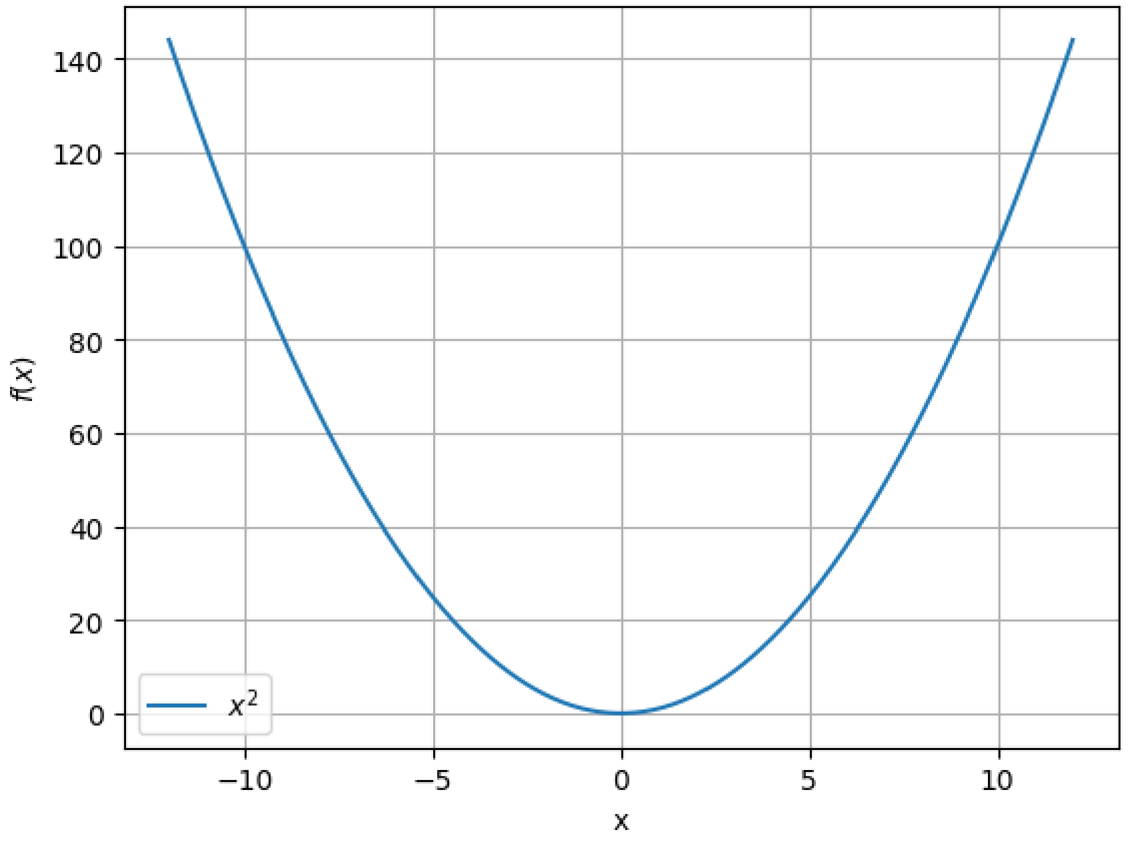
ENTRAÎNEMENT : DESCENTE DE GRADIENT

Entraînement

Descente de gradient

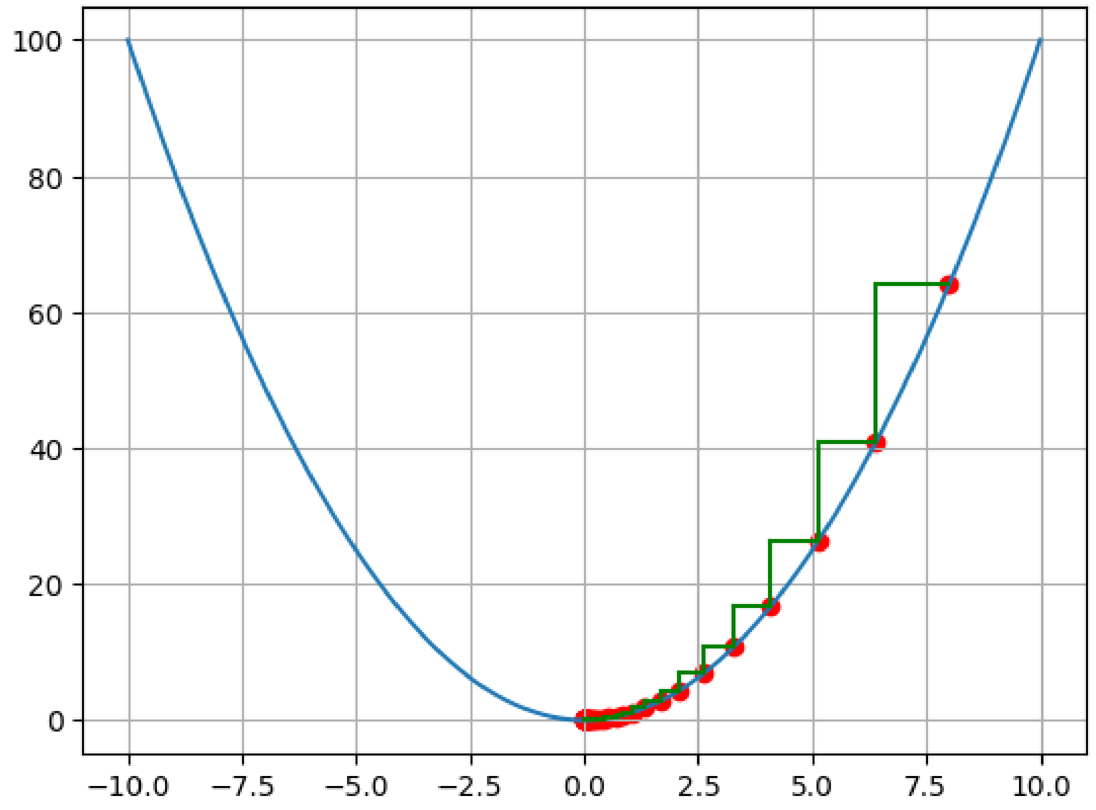
Principe : trouver le minimum d'une fonction

Exemple ci-contre avec la fonction $f_{coût}(x) = x^2$



Algorithme :

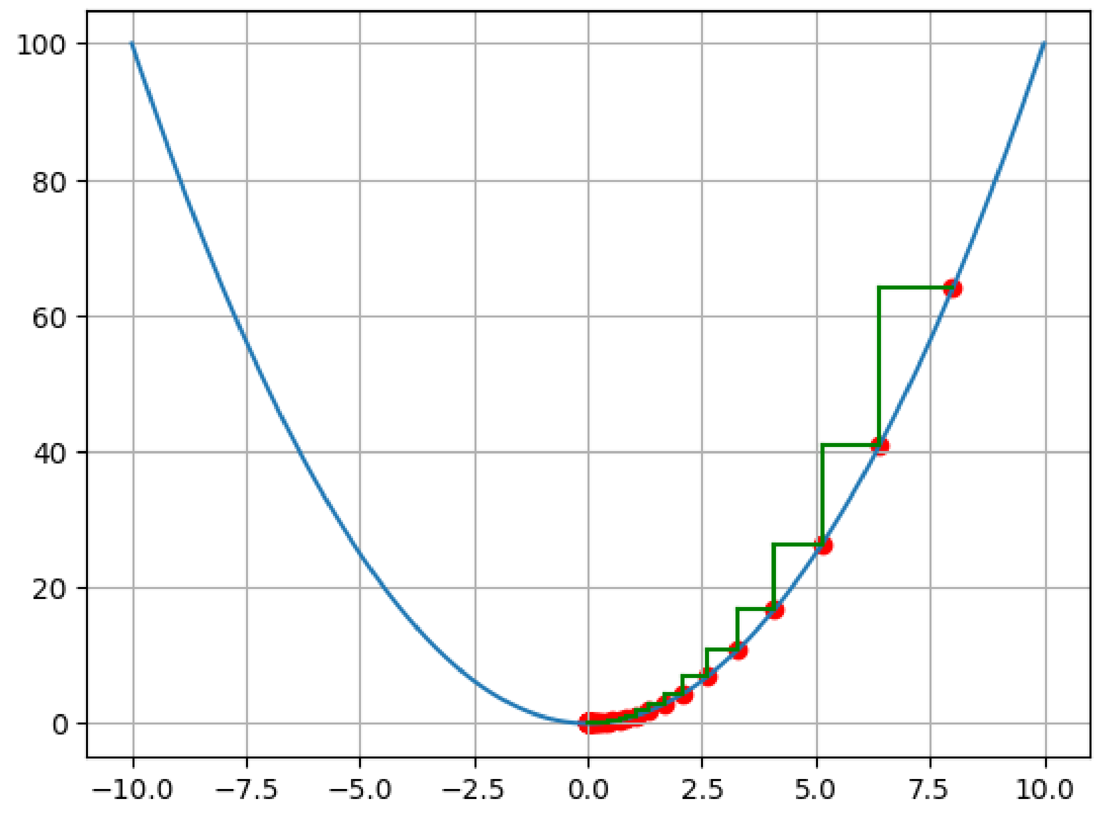
- Calcul du gradient en un point x_i (pente $\nabla f(x_i)$)
- Test d'arrêt : ($\|\nabla f(x_i)\| < \varepsilon$)
- Calcul du pas : $\alpha_i = \alpha * |\nabla f(x_i|$
- Nouvelle position : $x_{i+1} = x_i - \alpha_i$



Entraînement

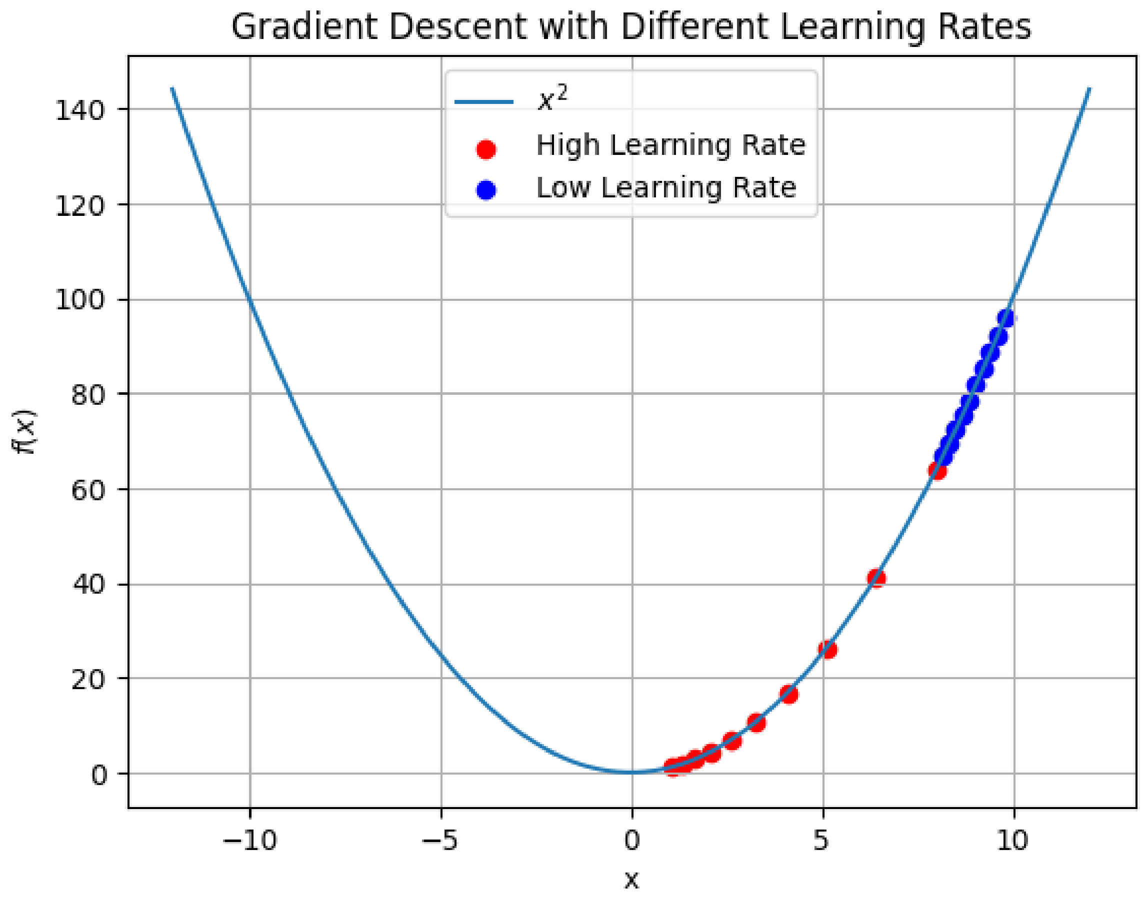
Descente de gradient

Principe : trouver le minimum d’une fonction



On note plusieurs paramètres importants :

- position initiale (point de départ)
- le pas : taux d’apprentissage ou *learning rate*
- l’algorithme de descente de gradient



Entraînement

Descente de gradient

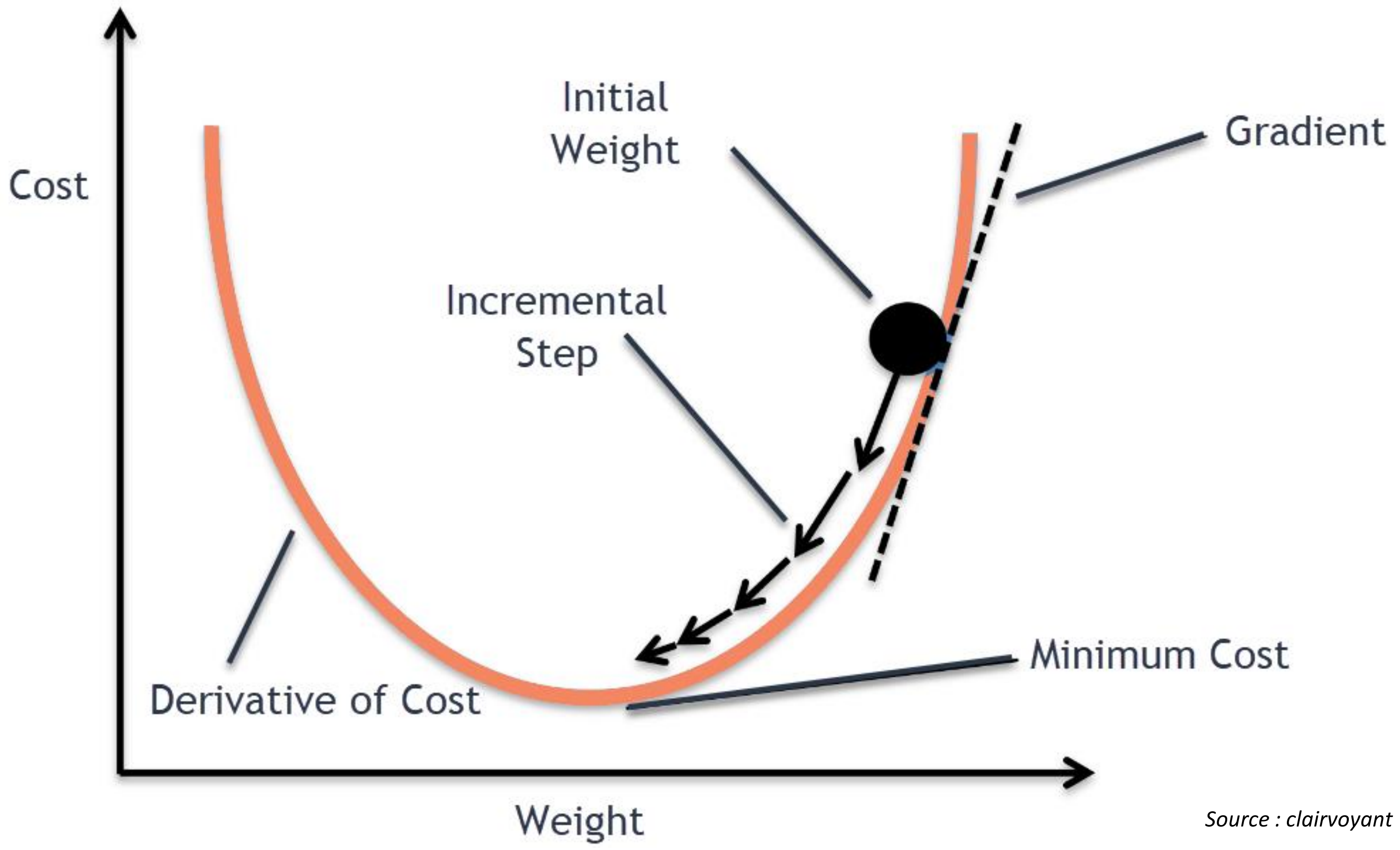
Optimiser ok, mais optimiser quoi ?

➔ Notre fonction de coût $f_{coût}$ qui prend en paramètres les poids de nos couches

Minimiser la fonction de coût ⬄ réduire l'erreur

Erreur = différence entre $y_{i,prédiction}$ (associée à l'entrée x_i) et $y_{i,réalité}$

Optimiser la fonction de coût = modifier les poids du réseau, afin que l'erreur soit minimisée, i.e dans la direction où la fonction diminue le plus vite.



Source : clairvoyant

Entraînement

Descente de gradient

Étapes :

➔Prédiction de notre réseau

➔Calcul de l'erreur

➔Calcul des gradients partiels pour chaque poids

➔Multiplication de chaque gradient par le *learning rate*, soustraire cette valeur pour obtenir les nouveaux poids et biais

➔Répéter pour chaque entrée à prédire

Entraînement

Rétropropagation du gradient

Une fois l'erreur $f_{coût}(y_{i,pred}, y_{i,réel})$ connue, une propagation inverse est faite pour réajuster les poids

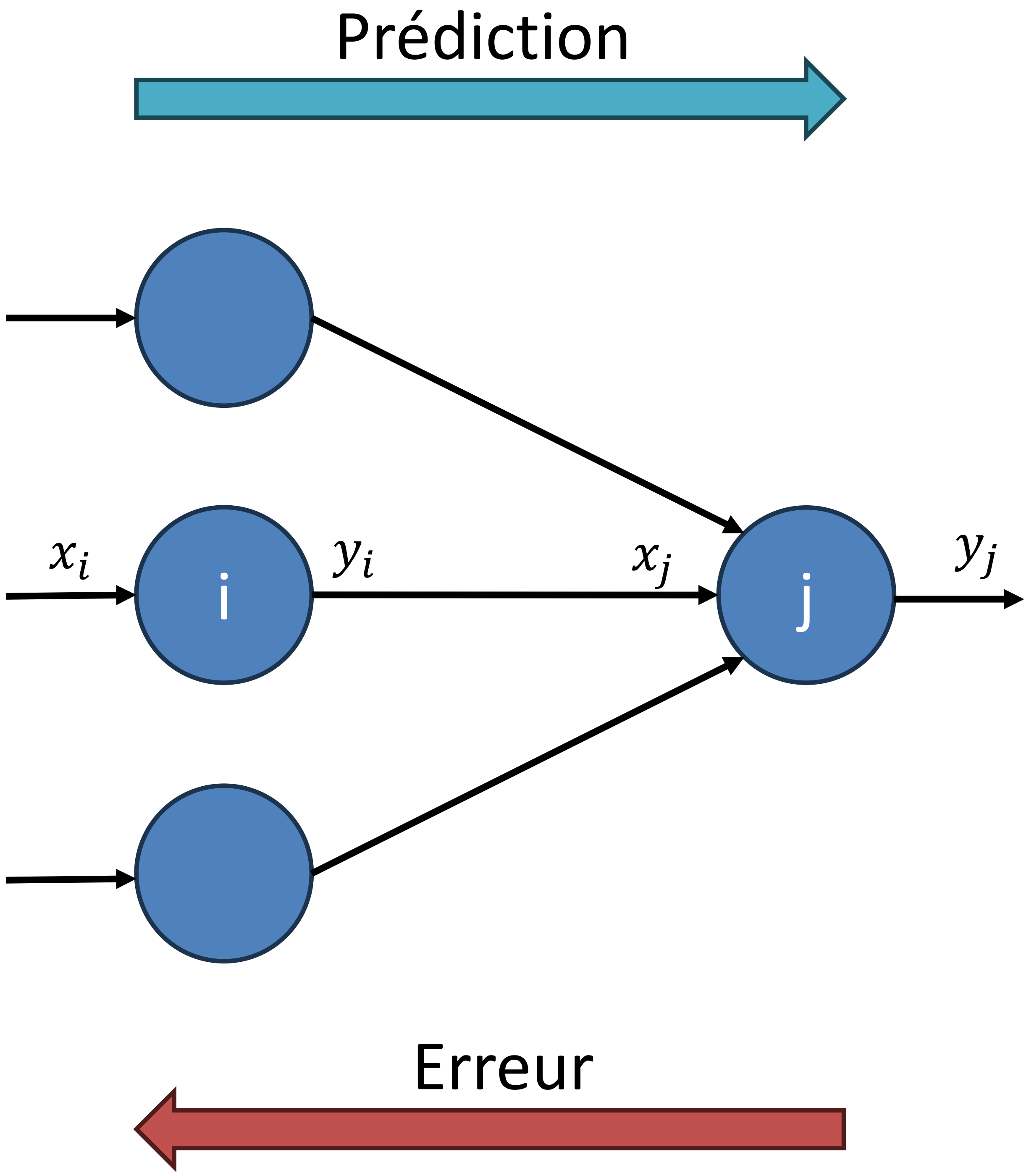
Mathématiquement : expression du gradient de notre erreur en fonction des poids synaptiques des couches du réseau.

Leibnitz Chain Rule :

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_j}$$

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_j}$$

Rétropropagation du gradient ≠ descente de gradient



Entraînement

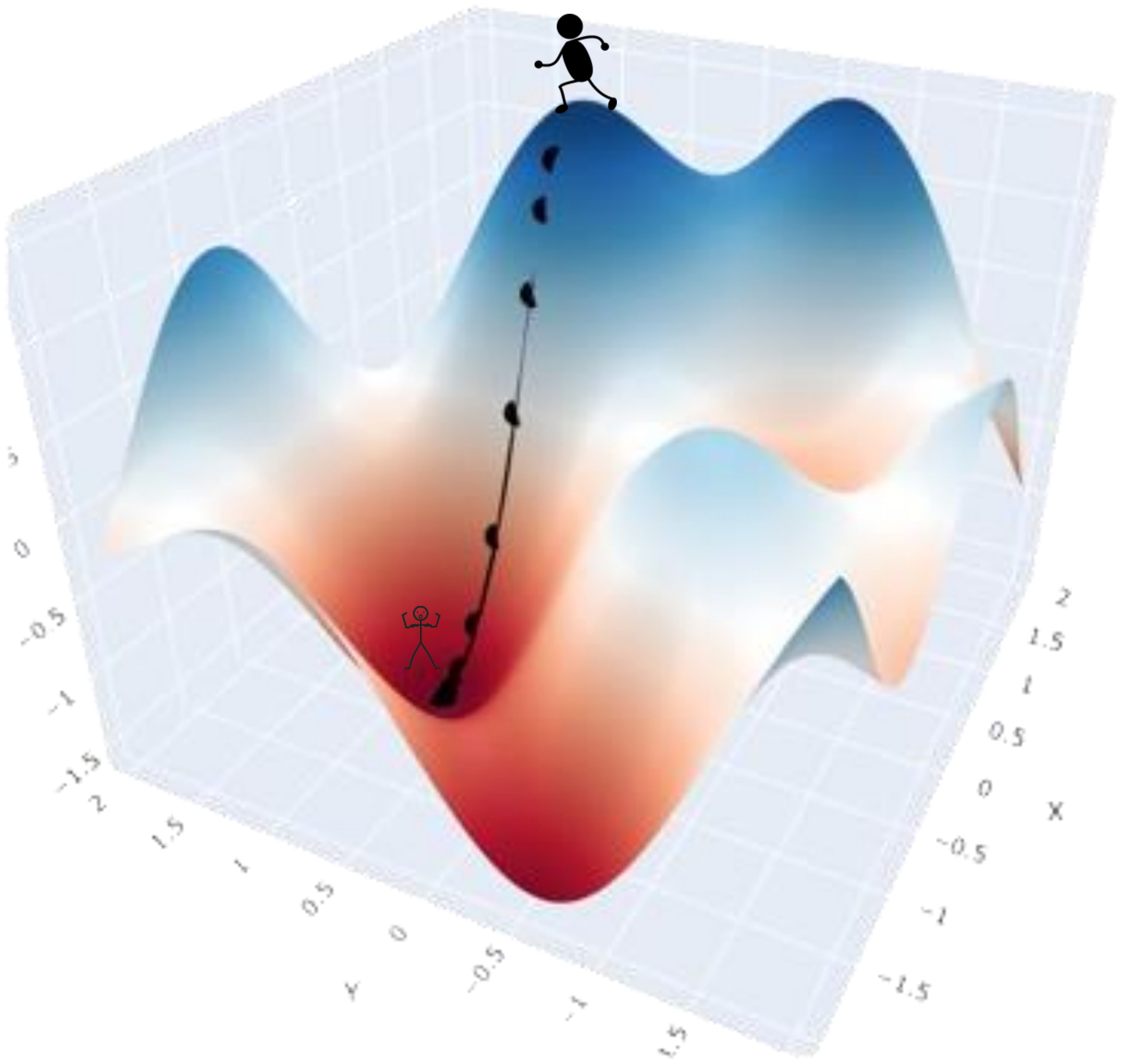
Analogie de la montagne

Objectif : Descendre au plus bas, **sans information**, étant **mal-voyant**

➔Selon la pente sous nos pieds, on va là où ça descend le plus
et on balise le chemin

➔Une fois en bas, reprendre le chemin inverse pour
comprendre comment l'impact du chemin pris sur le point d'arrivée

Aller encore plus bas la prochaine fois



SOMMAIRE

03
LES DONNÉES

LES DONNÉES

Les réseaux de neurones ne prennent que des nombres, et notamment un format : les <u>tenseurs</u>

Scalaire	Vecteur	Matrice	Tenseur
3	$\begin{bmatrix} 3 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 3 & 5 \\ 2 & 0 \end{bmatrix}$	$\begin{bmatrix} (3,2) & (1,5) \\ (1,2) & (0,2) \end{bmatrix}$

Audio ⟶ Séries temporelles

Texte ⟶ Tokenization

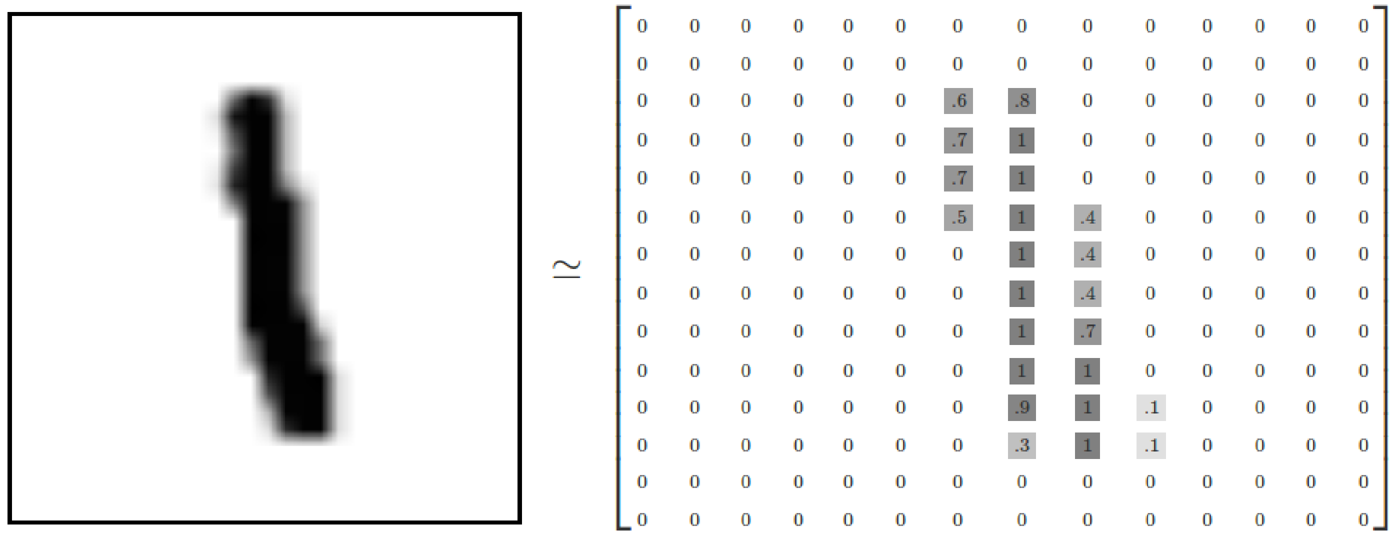
Je mange un avocat ⟶

Je
mange
un
avocat

⟶

0	**0**	**1**	**0**
0	0	0	1
0	0	0	1
0	1	0	0

Image ⟶ Image convertie en tenseur



Dataset MNIST

LES DONNÉES

Correspondance données / but

Exemple : Classification chat/chien

Données : chats, chiens (neige)

Entraînement

Modèle entraîné

Test

Classification

CHIEN !

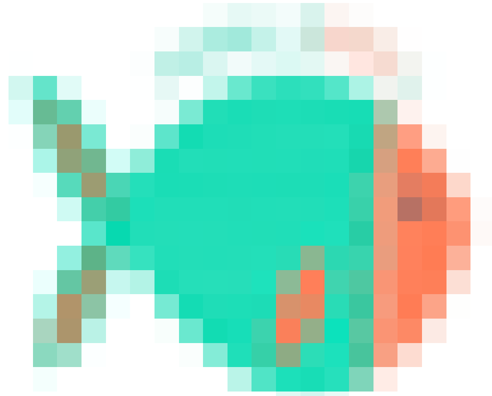
Les données doivent couvrir le panorama souhaité

LES DONNÉES

Trois points principaux :

➡ Données qualitatives

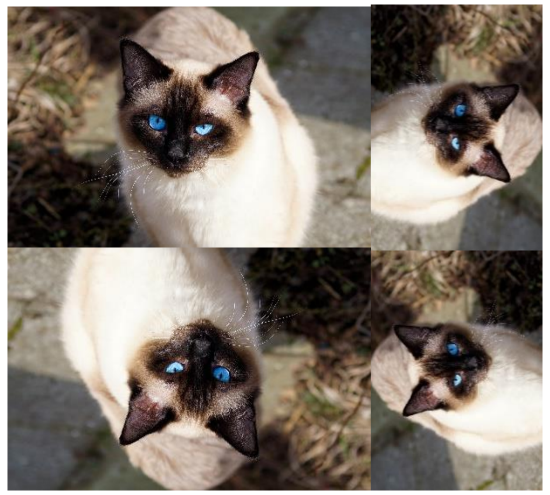
450x450



20x20

➡ Données quantitatives

➡ Données diversifiées

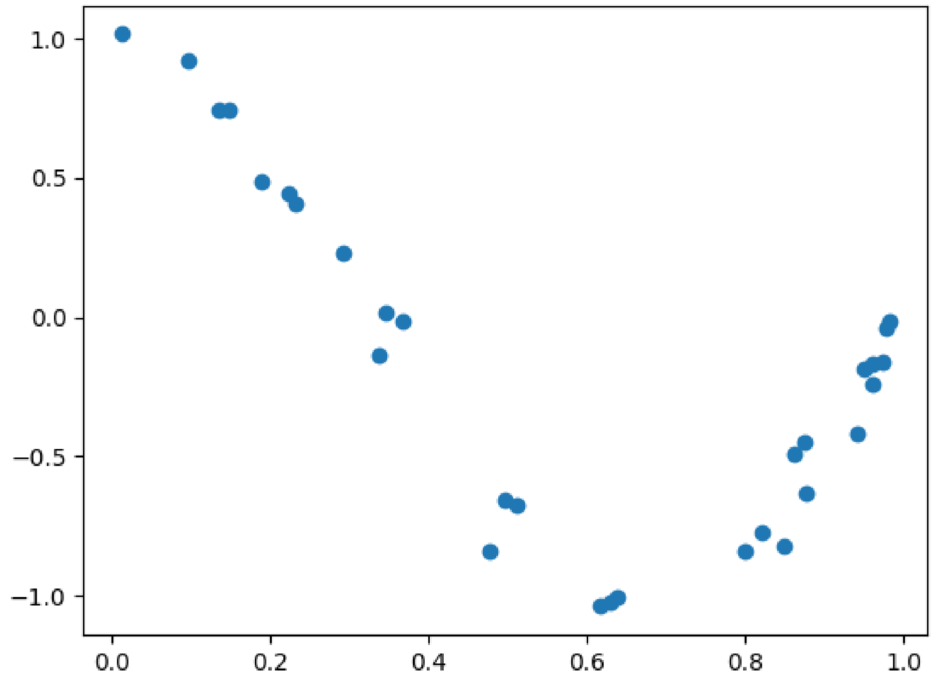


LES DONNÉES

Danger des données

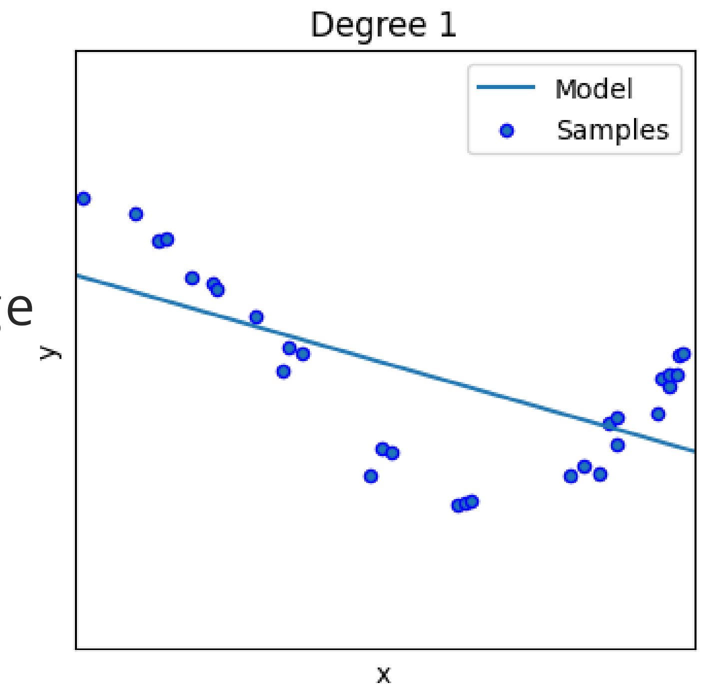
(ou trouver le bon compromis)

Distribution bruitée sur cos(x)



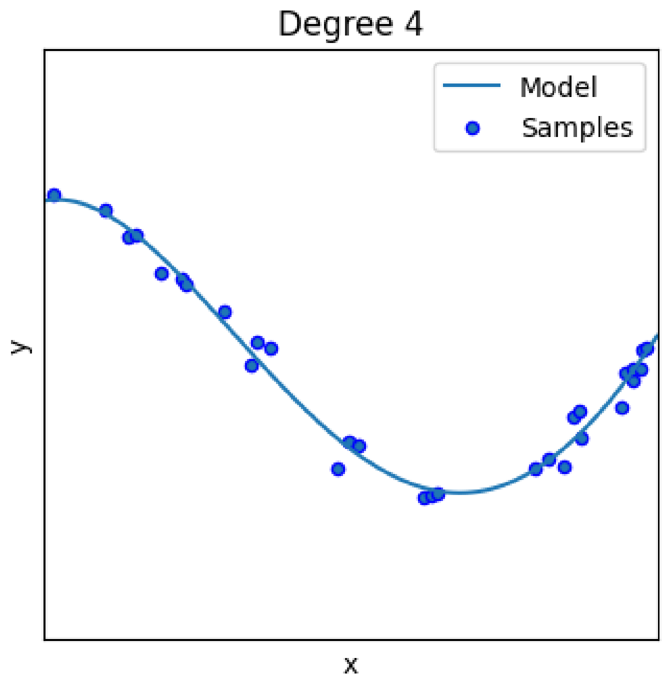
Sous-apprentissage

(underfitting)



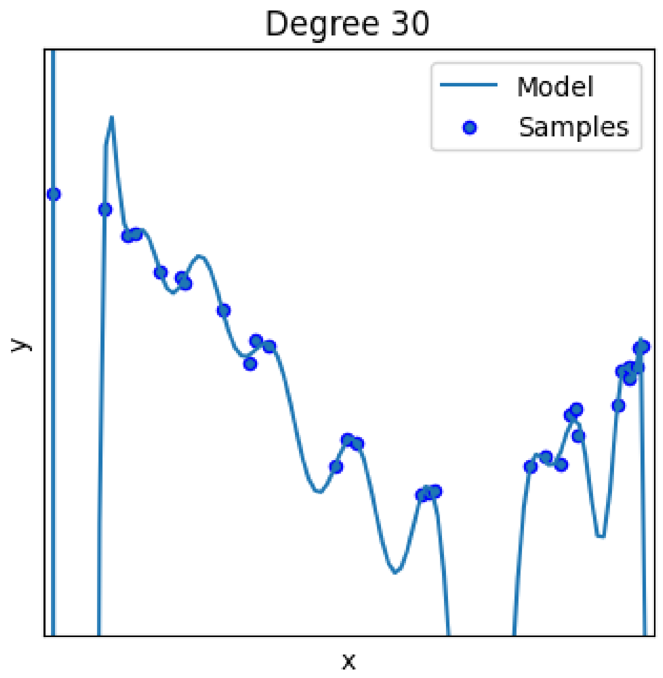
Bon

compromis



Sur-apprentissage

(overfitting)



Source : Dev.to

Passage au cas pratique

➔ https://gitlab.in2p3.fr/isdm_formation/introduction-deep-learning