



Architectures de traitement complexes – ATC

19/09/2024

YOUNESS EL MARHRAOUI

elmar.youness@gmail.com

Programme du cours 1

Rappels

-  Introduction
- Types de problèmes
- Applications IA
- Types d'apprentissage
- Types de données

Principes du DL

-  Graphe d'apprentissage
- Neurone et couches de neurones artificiels
- Fonctions d'activation
- Processus d'entraînement
- Processus d'optimisation



Introduction au DL

- Fonctionnement des réseaux de neurones
- Pourquoi le deep learning?
- Librairies & frameworks



Projet de groupe

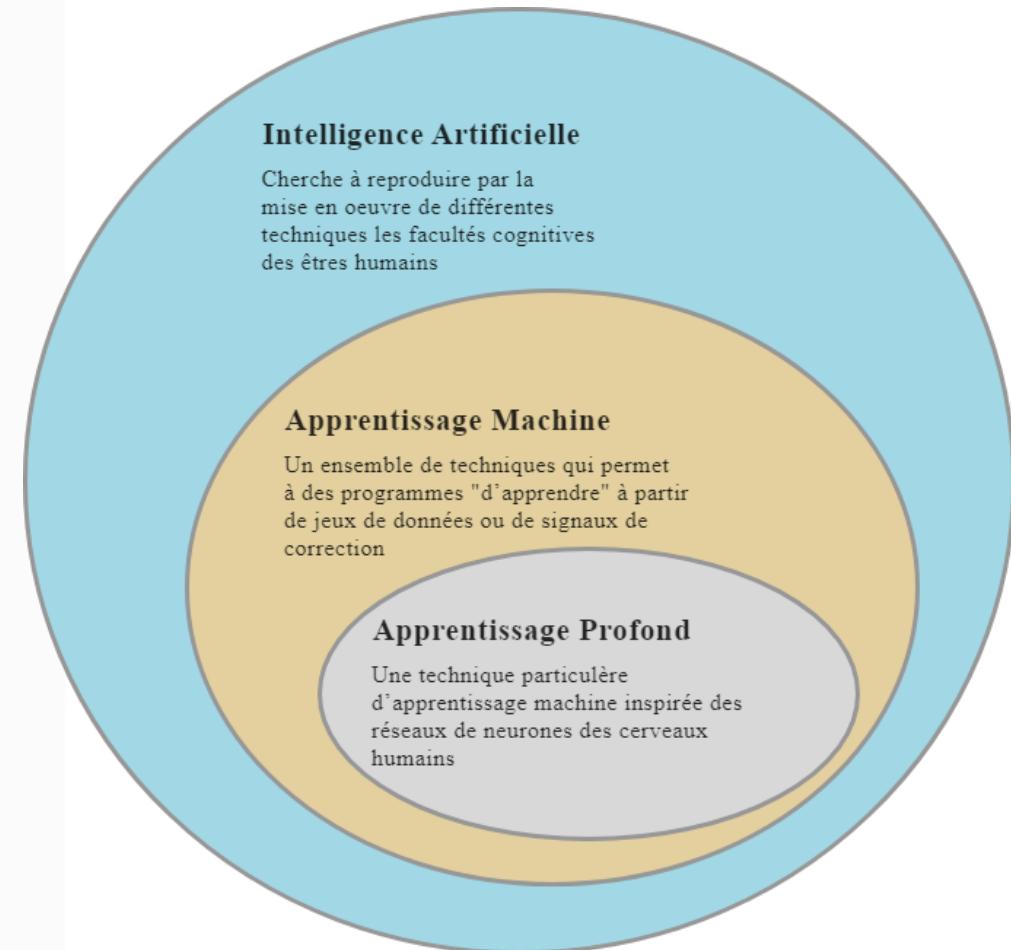
Rappels



- **Introduction**
- Types de problèmes
- Applications IA
- Types d'apprentissage
- Types de données

Définition

Machine Learning ou Apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine (au sens large) d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques.



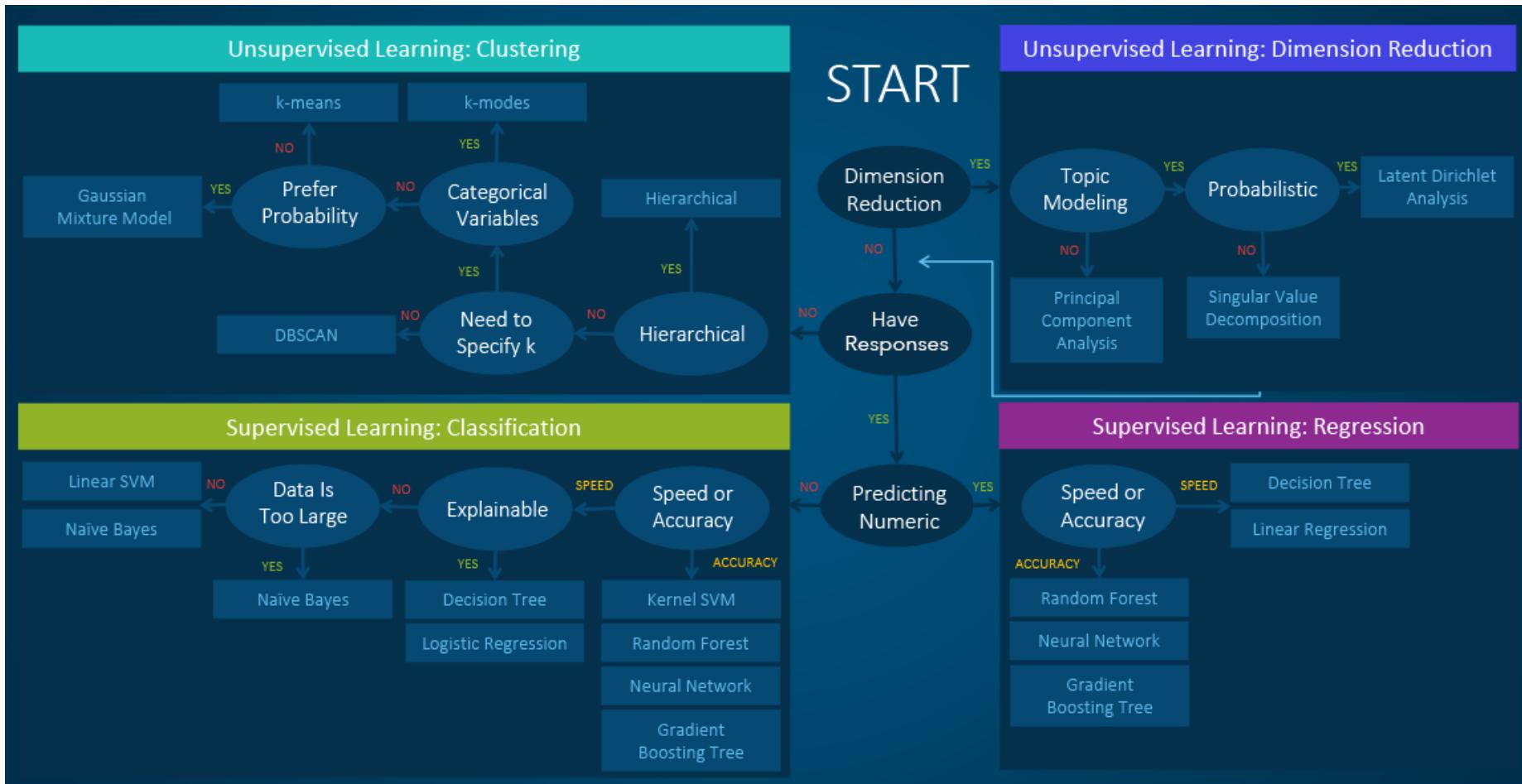
Rappels



- Introduction
- **Types de problèmes**
- Applications IA
- Types d'apprentissage
- Types de données

Types de problèmes

Récapitulatif



Rappels



- Introduction
- Types de problèmes
- **Applications IA**
- Types d'apprentissage
- Types de données

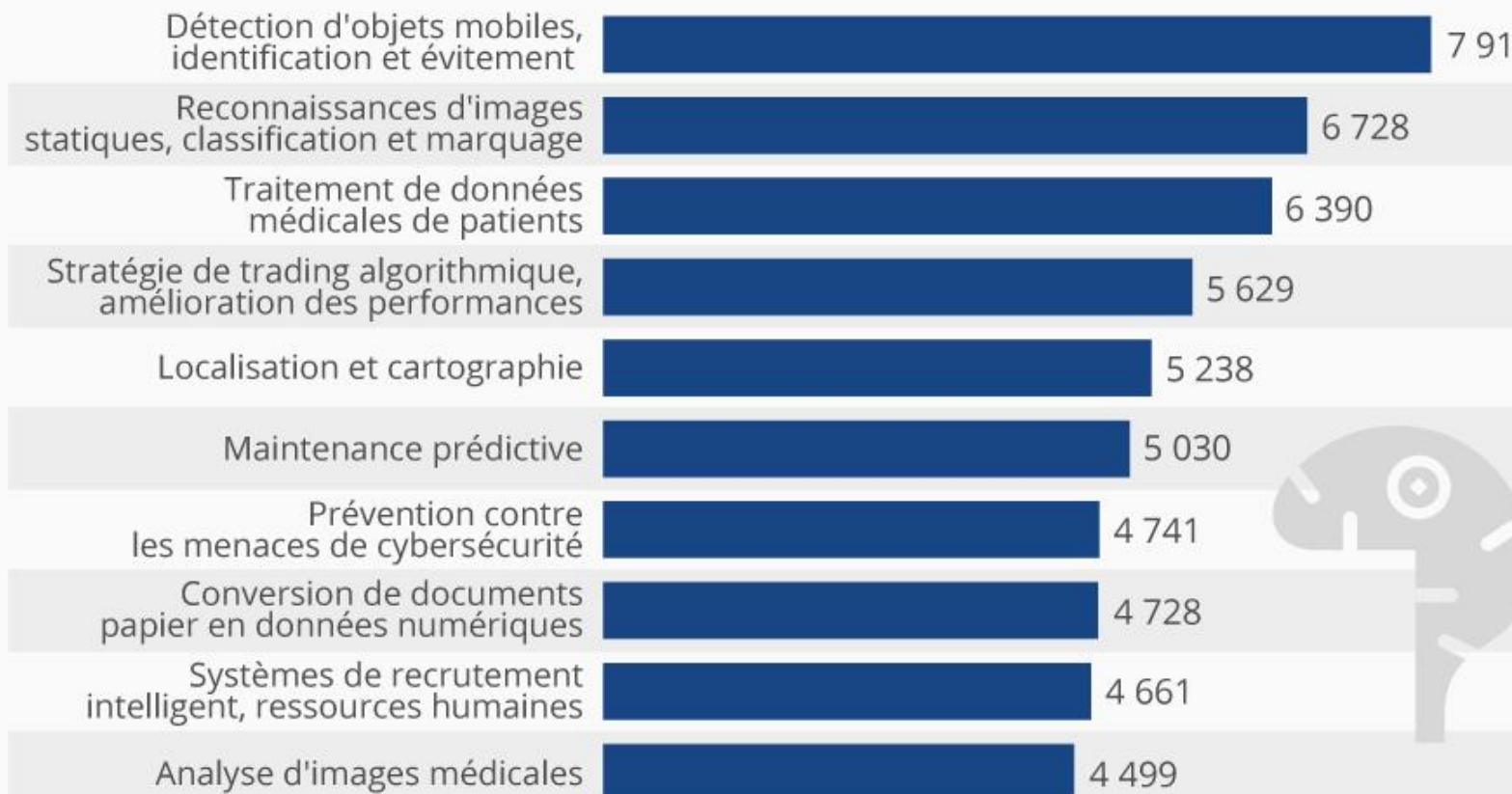
Applications



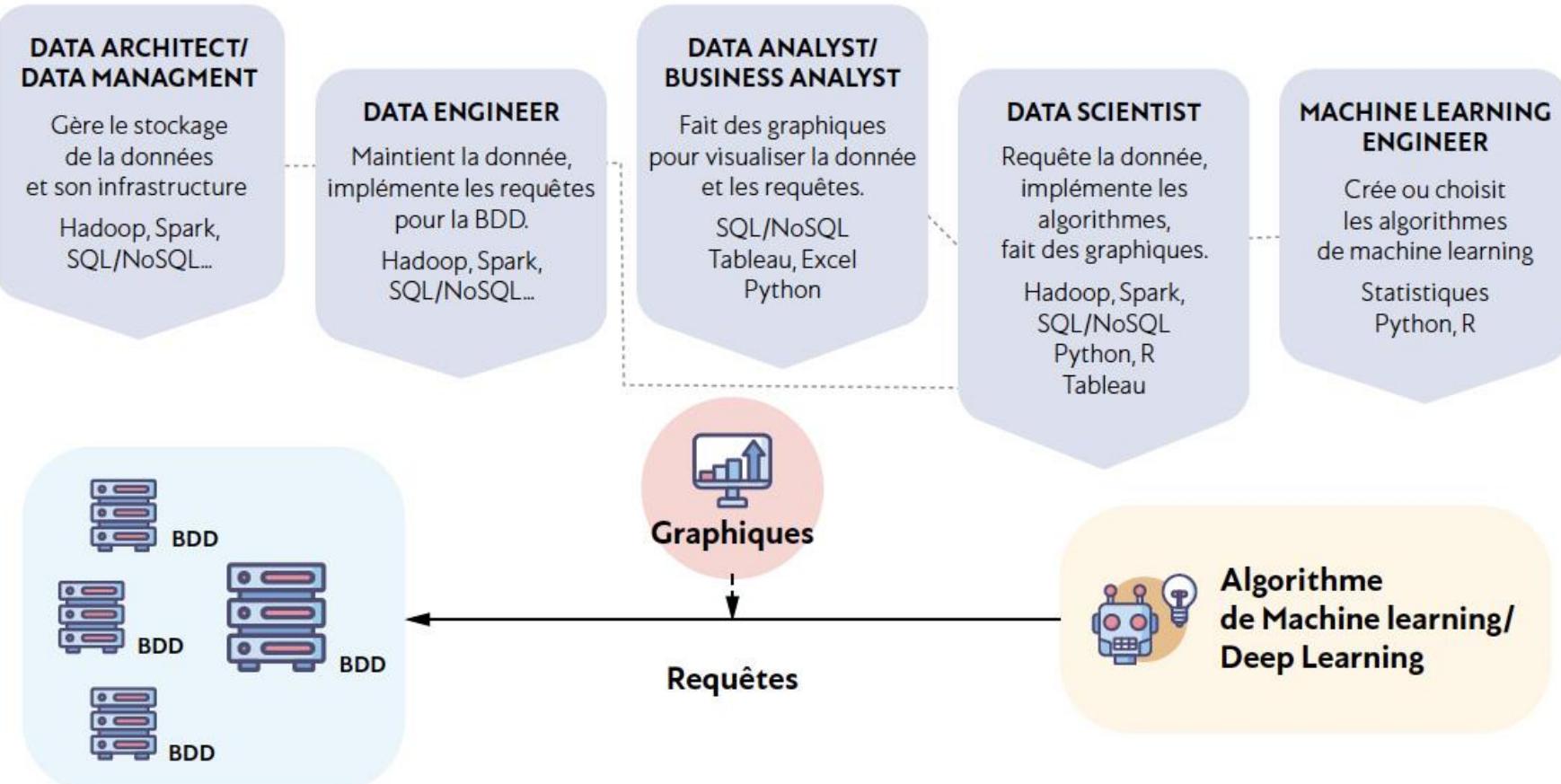
Applications

Le futur de l'intelligence artificielle

Estimation du chiffre d'affaires mondial cumulé de 2016 à 2025, en millions d'euros



Métiers

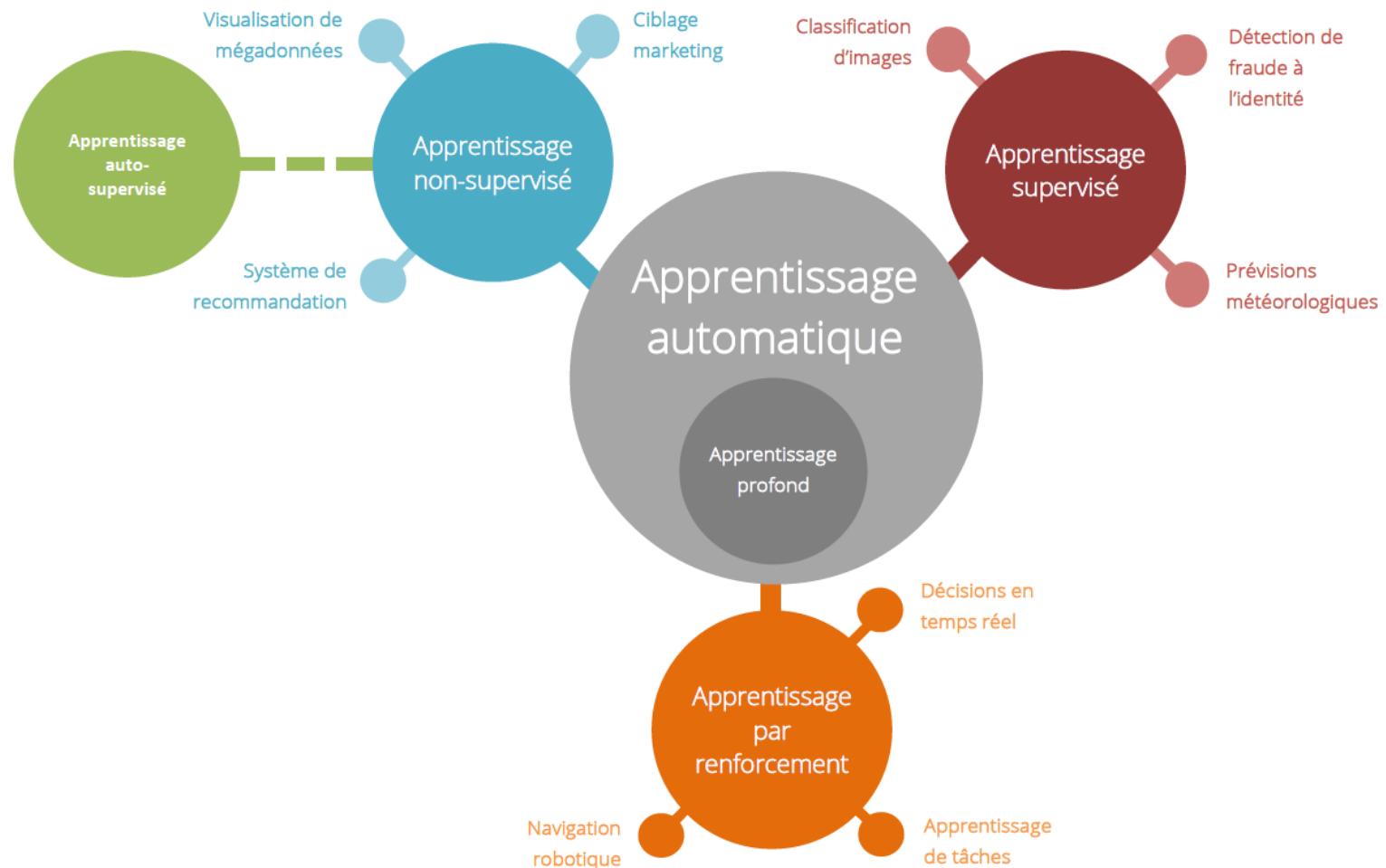


Rappels



- Introduction
- Types de problèmes
- Applications IA
- **Types d'apprentissage**
- Types de données

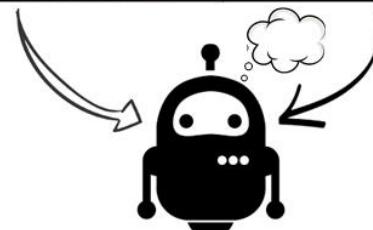
Types d'apprentissage



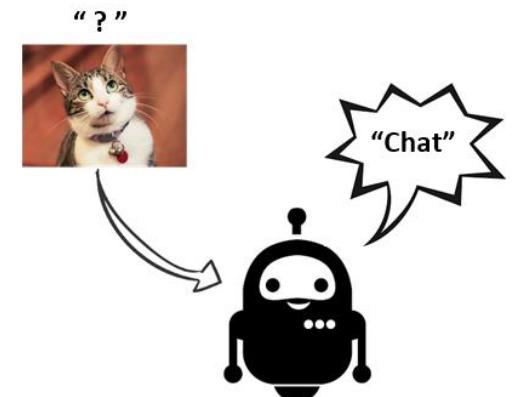
Apprentissage supervisé

L'apprentissage supervisé (en anglais : Supervised Learning) consiste à superviser l'apprentissage de la machine en lui montrant des **exemples** (des données) de la tâche qu'elle doit réalisée.

x	→	y
		“Chien”
		“Chien”
		“Chat”
		“Chien”

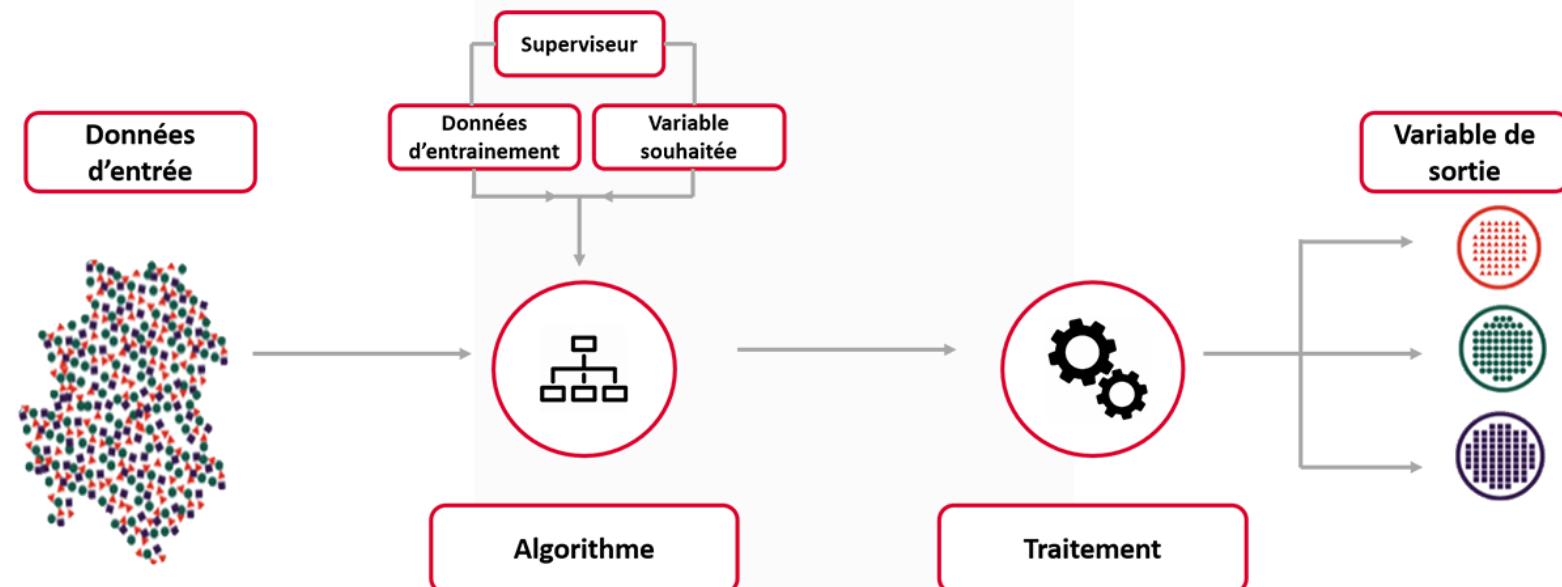


Apprentissage Supervisé



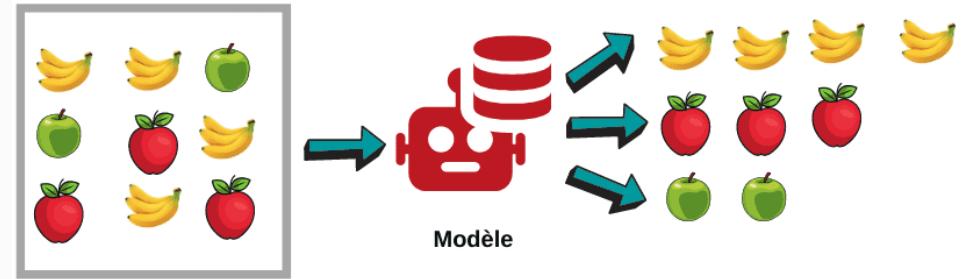
Apprentissage supervisé

Schéma d'entraînement



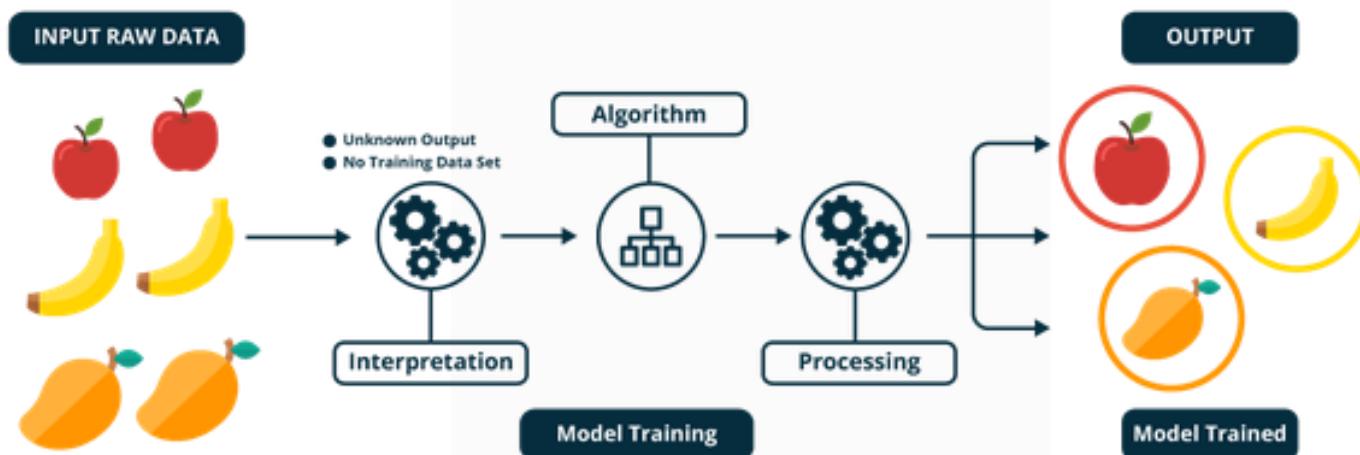
Apprentissage non-supervisé

L'apprentissage non-supervisé (en anglais : Unsupervised Learning) consiste à utiliser un jeu de données **non-étiqueté** afin que le modèle crée ses propres labels. Ces derniers sont construits à partir d'analyses et de **groupement de données**.



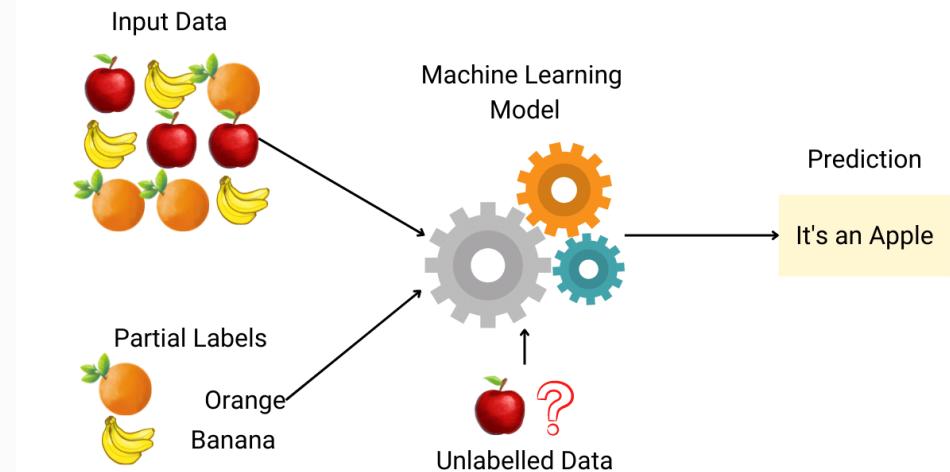
Apprentissage non-supervisé

Schéma d'entraînement



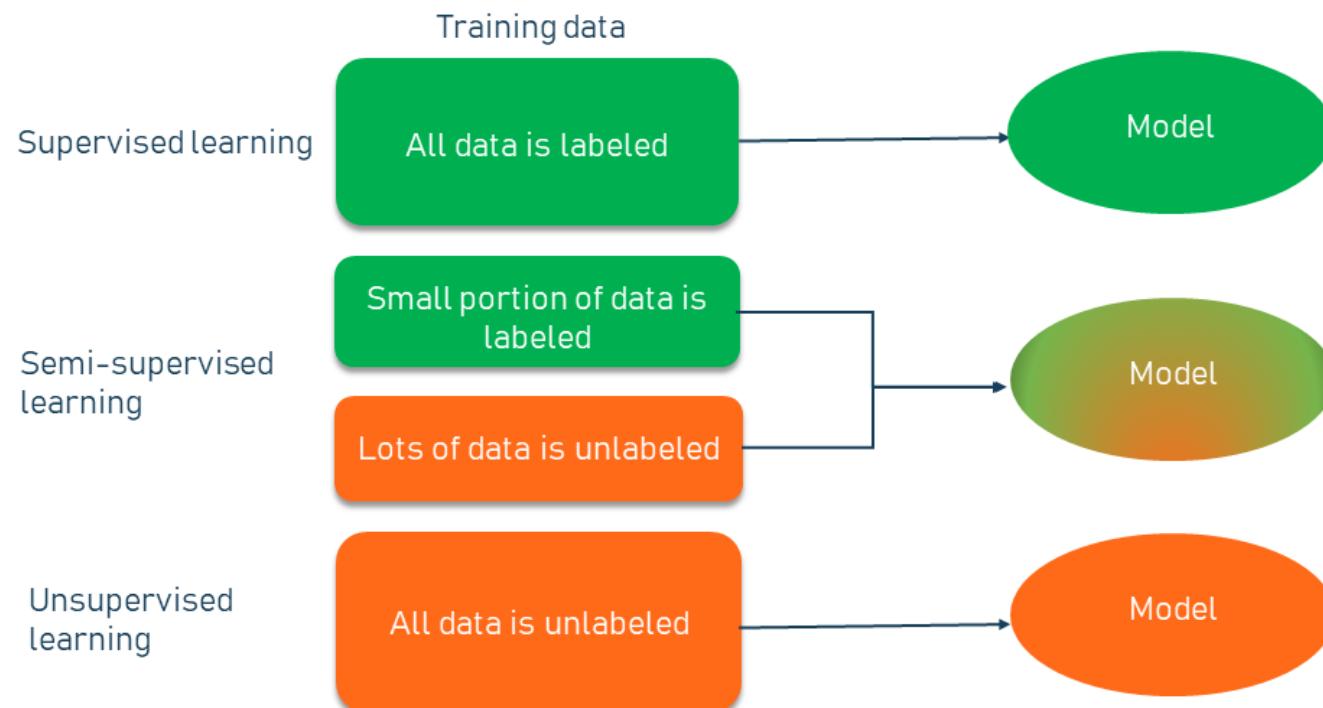
Apprentissage semi-supervisé

L'apprentissage non-supervisé (en anglais : Semi-supervised Learning) consiste à entraîner un modèle d'apprentissage sur un jeu de données **partiellement annoté** qui comporte quelques données annotées et beaucoup de données non annotées. L'idée est d'attribuer les annotations en utilisant la similarité entre les données annotées et les données non annotées.



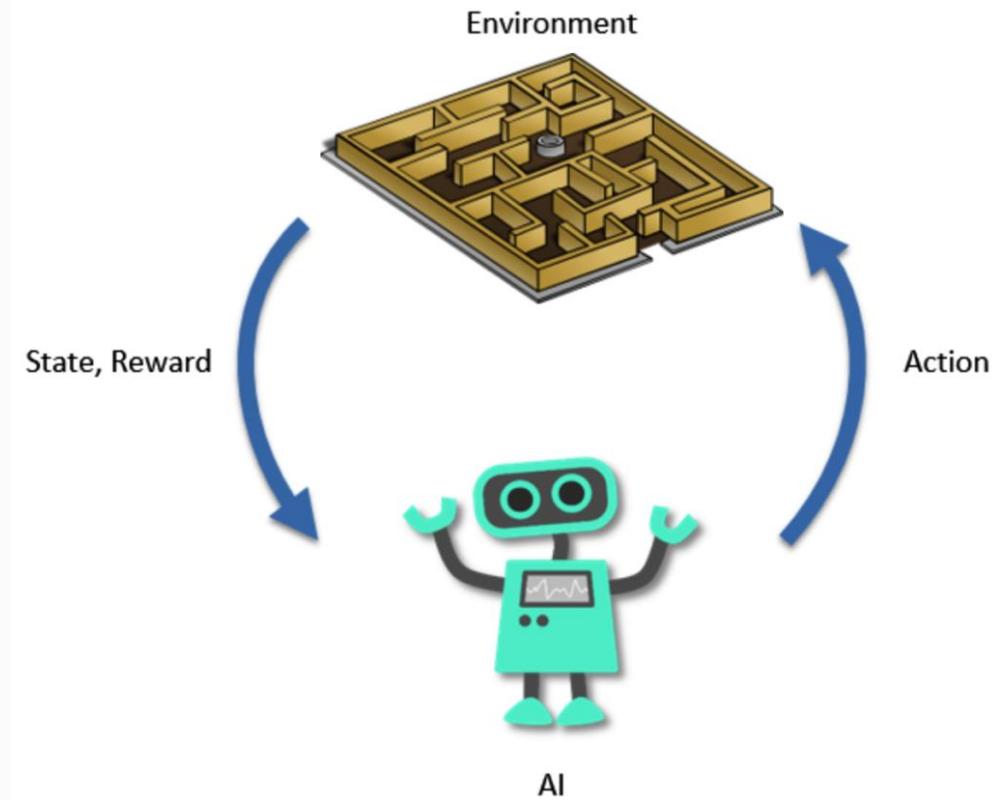
Récapitulatif

SUPERVISED LEARNING vs SEMI-SUPERVISED LEARNING vs UNSUPERVISED LEARNING

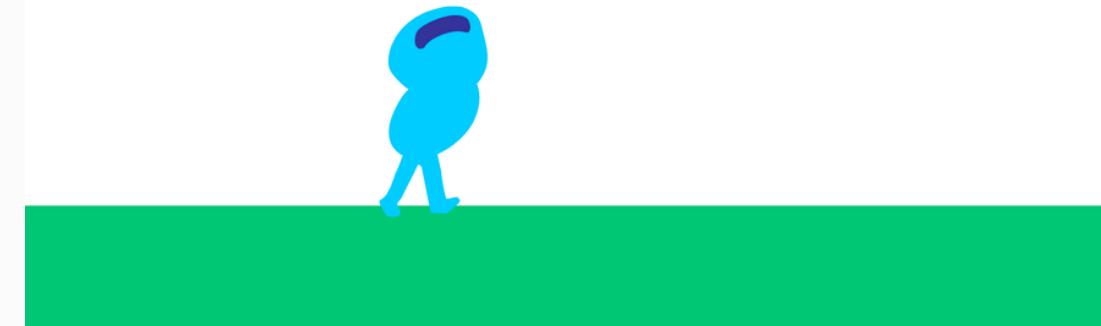
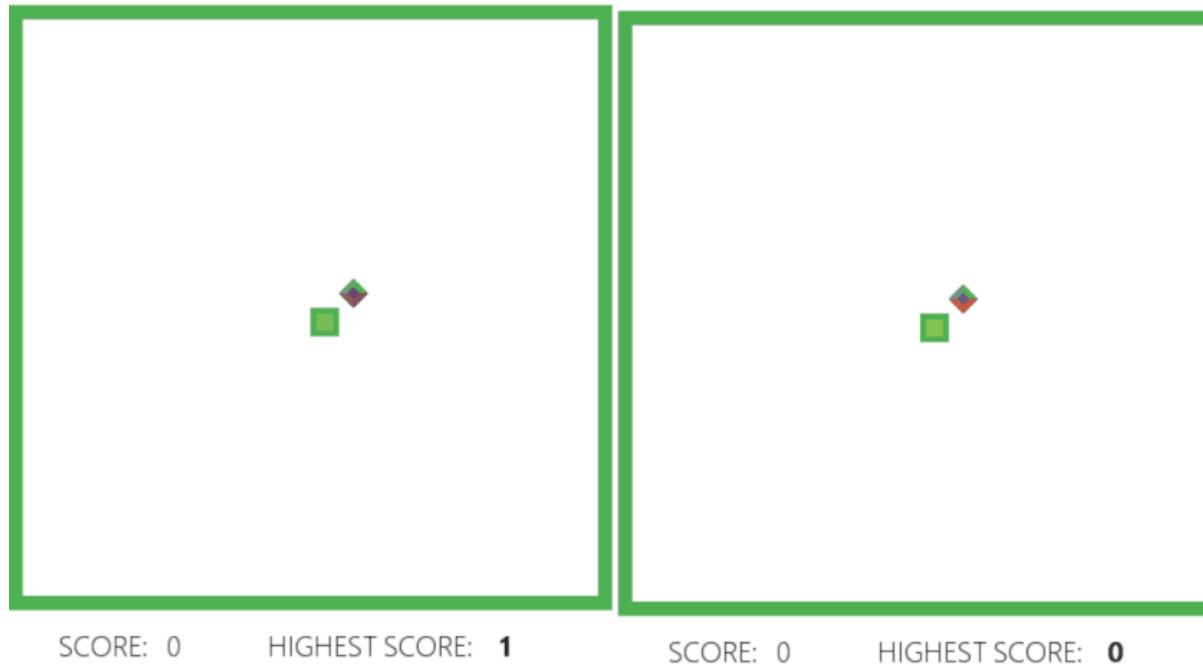


Apprentissage par renforcement

L'apprentissage par renforcement (RL pour Reinforcement Learning) fait référence à une classe de problèmes d'apprentissage automatique, dont le but est d'apprendre, à partir d'expériences successives, ce qu'il convient de faire de façon à trouver la meilleure solution.



Apprentissage par renforcement

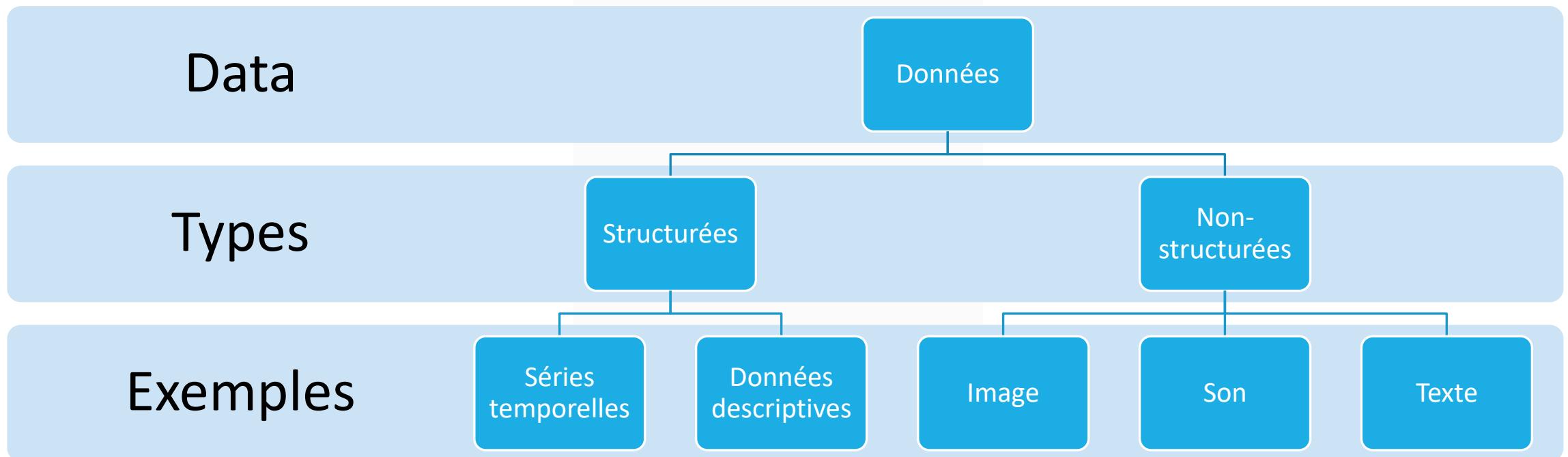


Rappels



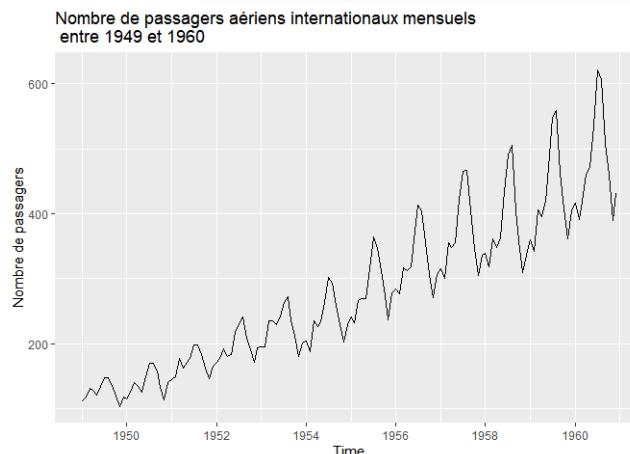
- Introduction
- Types de problèmes
- Applications IA
- Types d'apprentissage
- **Types de données**

Types de données

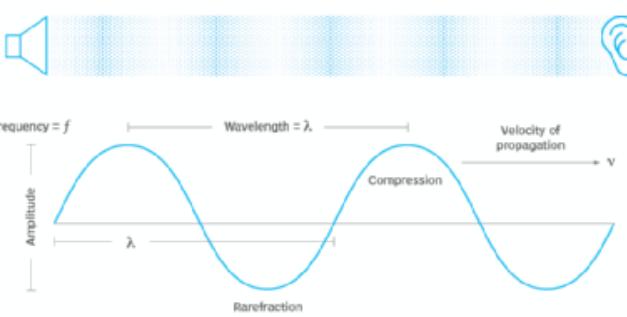


Types de données

Année	Catégorie	Produit	Ventes	Évaluation
2017	Composants	Chaînes	200€	75 %
2015	Vêtements	Chaussettes	37€	22 %
2017	Vêtements	Cuissards	40€	22 %
2015	Vêtements	Shorts	133€	56 %
2017	Vêtements	Collants	360€	100 %
2015	Composants	Guidons	23€	35 %
2016	Vêtements	Chaussettes	23€	28 %
2016	Pièces détachées	Freins	34€	36 %
2016	Vélos	Vélos tout-terrain	63€	40 %
2017	Composants	Freins	54€	38 %
2016	Accessories	Casques	170€	90 %
2016	Accessories	Feux	216€	90 %
2016	Accessories	Antivols	298€	90 %
2016	Composants	Axes de pédales	10€	23 %
2015	Vêtements	Maillots	67€	5 %
2017	Pièces détachées	Axes de pédales	6€	27 %
2015	Vélos	Vélos de route	35€	50 %
2017	Vêtements	Maillots	75€	40 %
2017	Accessories	Pneus et chambres	637€	90 %



Characteristics of a sound wave



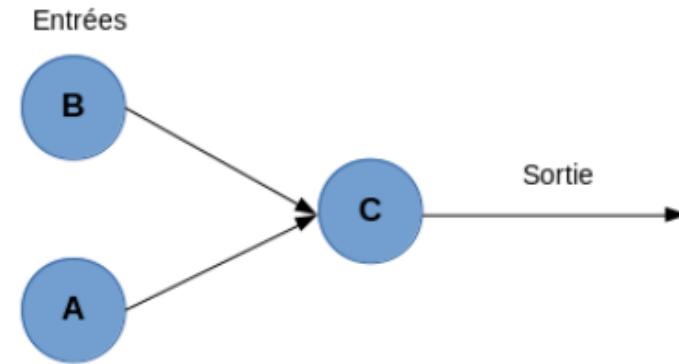
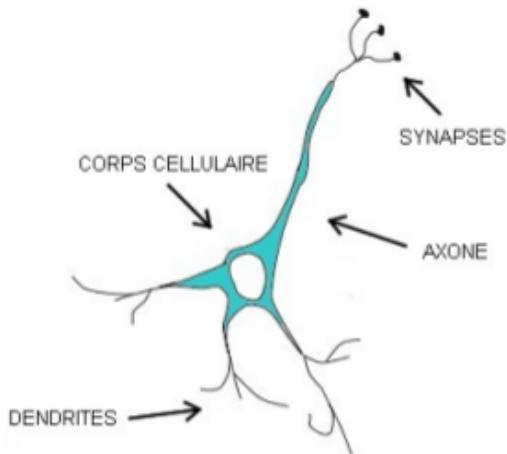
News Cryptocurrency News Today June 12 DATE Bitcoin GPE Dogecoin Shiba Inu PERSON and other top coins prices and all latest updates cryptocurrency Latest News ORG Today June 12 DATE Bitcoin GPE and all major top cryptocurrencies were trading in red at 345 pm TIME on Saturday June 12 DATE In line with its recent trends overall global crypto market was down by over 15 per cent on the weekend DATE View in App GPE Bitcoin GPE was down by 6 CARDINAL and was trading at Rs 2728815 DATE after hitting days high of Rs 2900208 Source Reuters ORG Reported By ZeeBiz NORP WebTeam Written By Ravi Kant Kumar PERSON Updated Sat Jun PERSON 12 20210646 pm TIME Patna ORG ZeeBiz WebDesk PERSON RELATED NEWS Cryptocurrency Latest News Today June 14 DATE Bitcoin GPE leads crypto rally up over 12 CARDINAL after ELON MUSK TWEET Check Ethereum Polka ORG Dot Dogecoin Shiba Inu PERSON and other top coins INR ORG price World India ORG updates Bitcoin GPE law is only latest headturner by El Salvadors MILLENNIAL ORG PRESIDENT Chinas cryptocurrency mining crackdown spreads to Yunnan GPE in southwest media Cryptocurrency latest news ALERT Rs

Introduction au DL



- **Fonctionnement des réseaux de neurones**
- Pourquoi le deep learning?
- Librairies & frameworks

Fonctionnement des réseaux de neurones

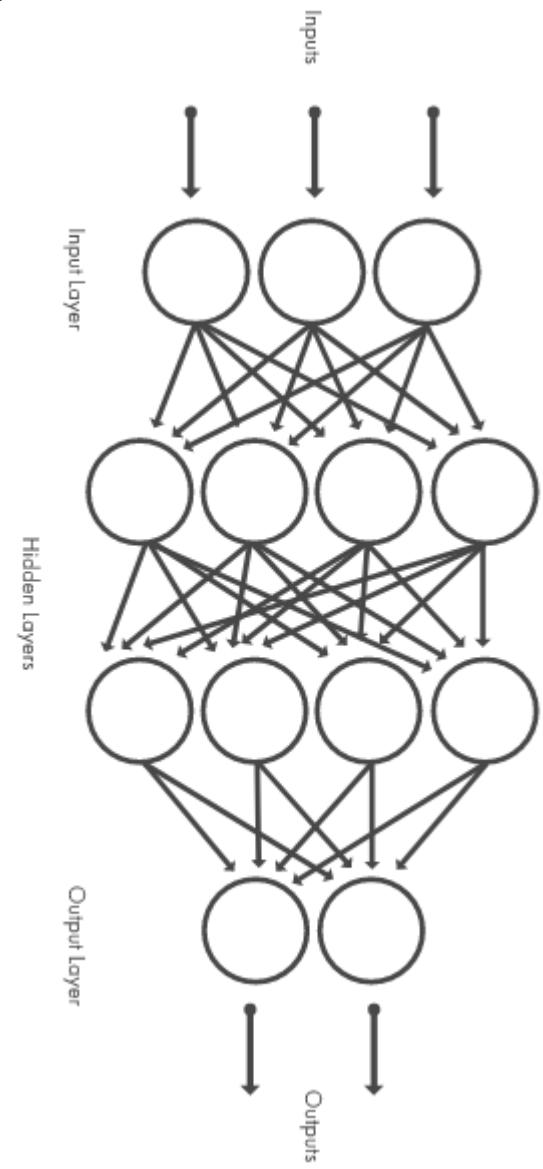


Un neurone biologique se compose d'un **corps cellulaire** qui comprend le noyau du neurone et la plupart des éléments complexes de la cellule. De nombreux **prolongements appelés dendrites** et un très long prolongement appelé axone. À son extrémité, l'axone se décompose en plusieurs ramifications que l'on appelle télodendrons qui se terminent par de minuscules structures appelés **synapses** et qui sont directement reliés à des dendrites ou directement au corps cellulaire d'autres neurones.

Un neurone reçoit des signaux électriques par le biais des dendrites et lorsque que le neurone reçoit suffisamment de signaux en un temps donné (quelques millisecondes) alors il déclenche ses propres signaux.

Fonctionnement des réseaux de neurones

- Des modèles **modifiables** en termes de **profondeur** et quantité de paramètres (couches / modules)
- Permet d'obtenir une sortie **non-linéaire, hiérarchique** et une **représentation abstraite** des données
- Flexibilité par rapport au **type et la taille des entrées et sorties**
- **Programmation fonctionnelle différentiable**



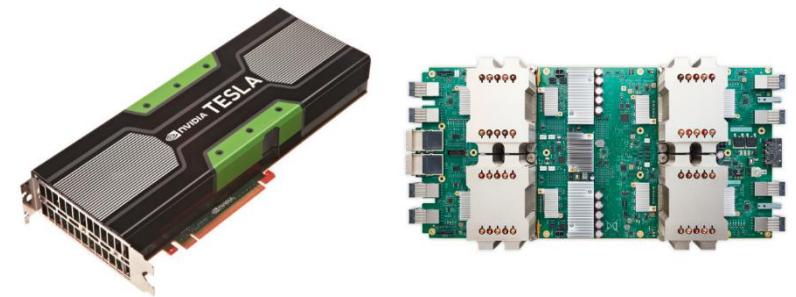
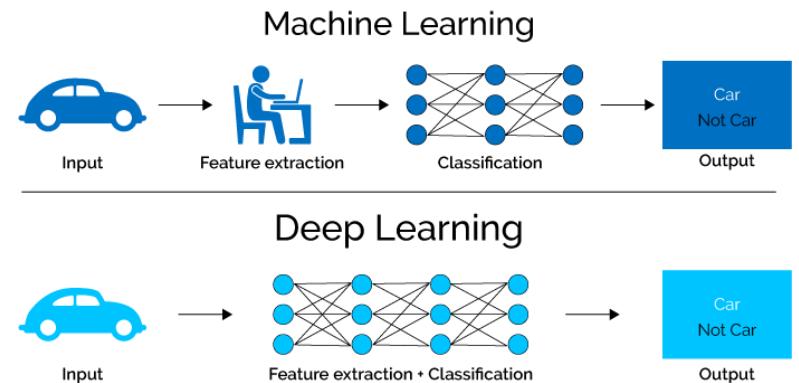
Introduction au DL



- Fonctionnement des réseaux de neurones
- **Pourquoi le deep learning?**
- Librairies & frameworks

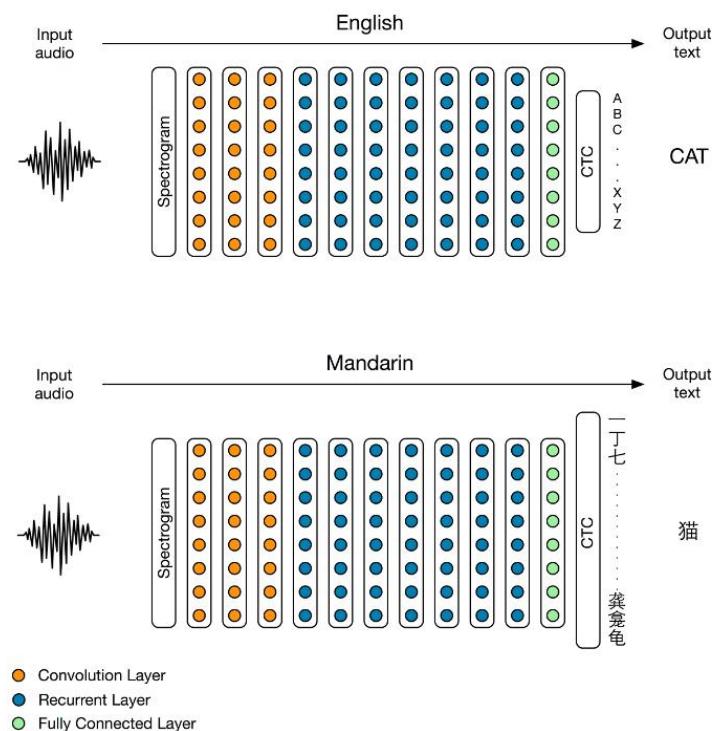
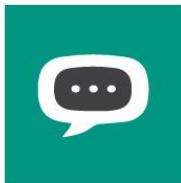
Pourquoi le deep learning?

- Permet de travailler sur **tout type de données**
- Meilleurs algorithmes et apprentissage
- Peut profiter des **puissances de calcul** (GPUs, TPUs, ...)
- Peut facilement **s'adapter** à n'importe quelle problématique d'apprentissage (supervisé, non-supervisé, ...)
- Outils et modèles entraînés disponibles et en **open source**



Pourquoi le deep learning?

Speech-to-Text



[Baidu 2014]

Pourquoi le deep learning?

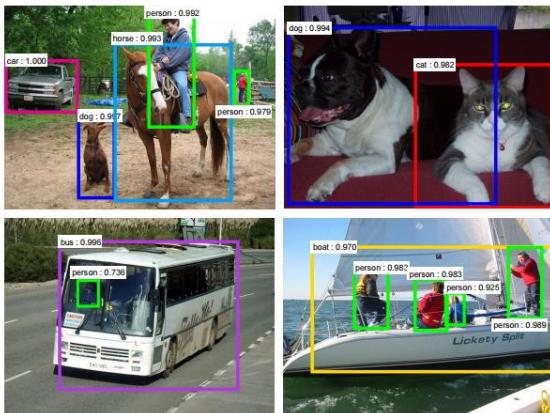
Vision



[Krizhevsky 2012]



[Ciresan et al. 2013]



[Faster R-CNN - Ren 2015]



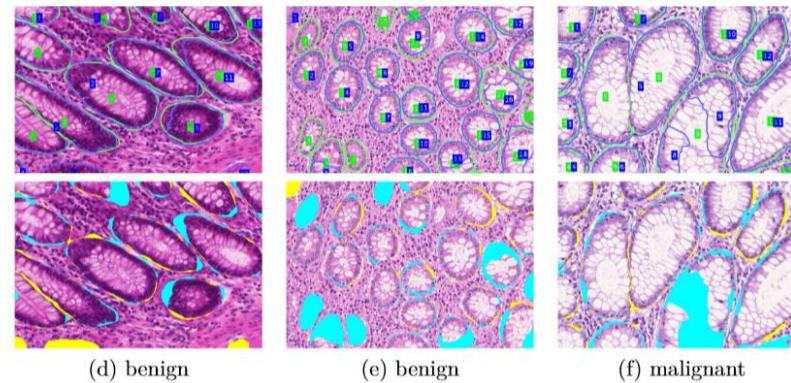
[NVIDIA dev blog]

Pourquoi le deep learning?

Vision



[Stanford 2017]



[Nvidia Dev Blog 2017]

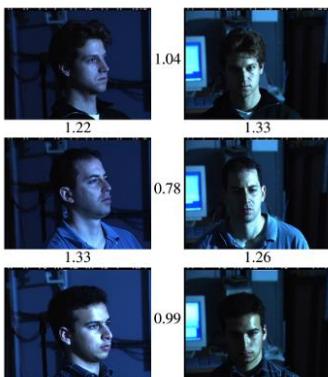
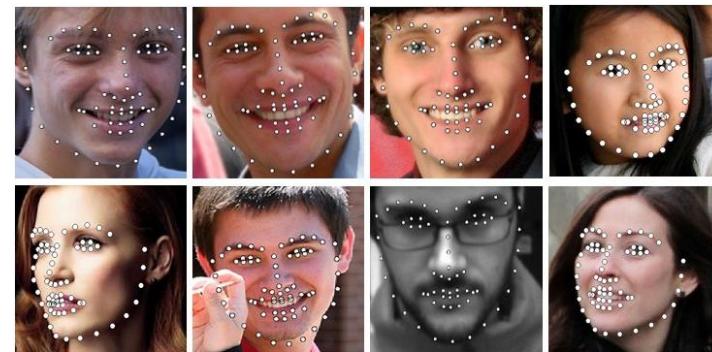


Figure 1. Illumination and Pose invariance.

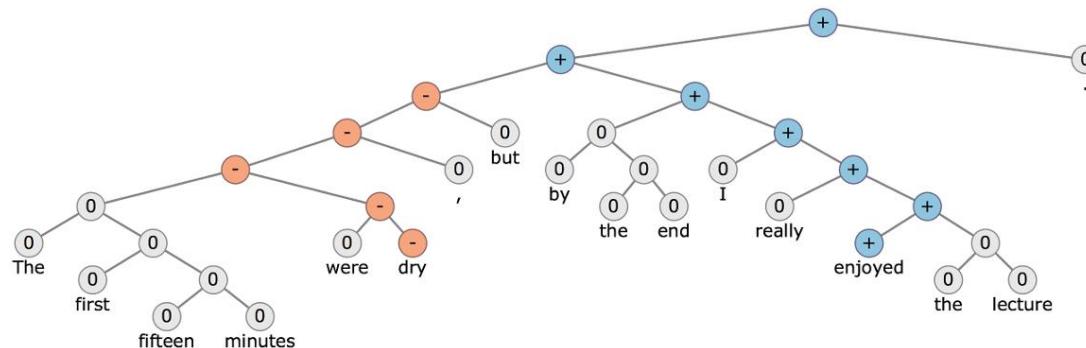
[FaceNet - Google 2015]



[Facial landmark detection CUHK 2014]

Pourquoi le deep learning?

NLP



[Socher 2015]

Pourquoi le deep learning?

NLP



Salit Kulla

to me

11:29 AM · · ·

Hey, Wynton Marsalis is playing this weekend. Do you have a preference between Saturday and Sunday?

-S

I'm down for either.

Let's do Saturday.

I'm fine with whatever.



Reply



Forward

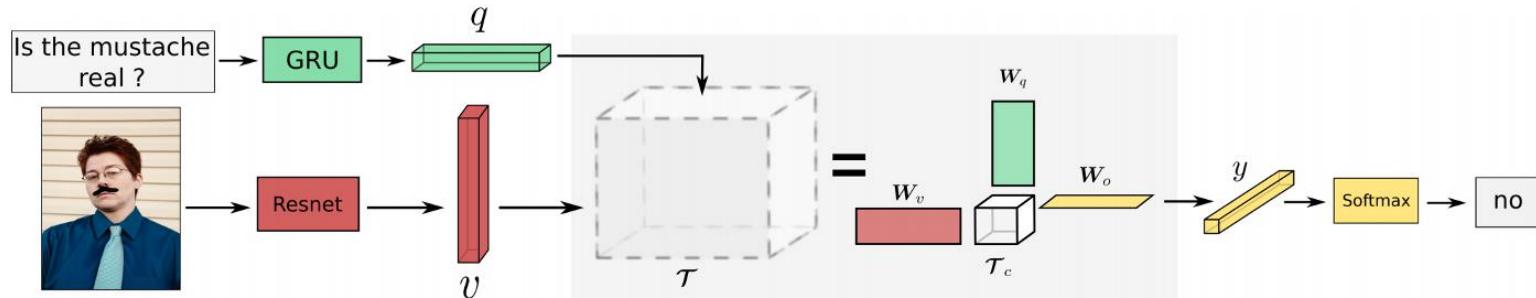


[Google Inbox Smart Reply]

[Amazon Echo / Alexa]

Pourquoi le deep learning?

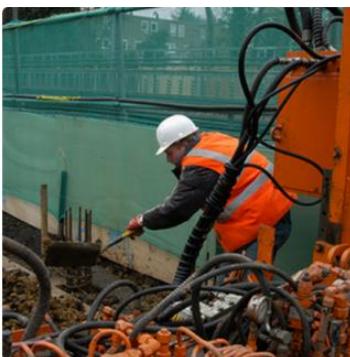
Vision + NLP



[VQA - Mutan 2017]



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."

[Karpathy 2015]

Pourquoi le deep learning?

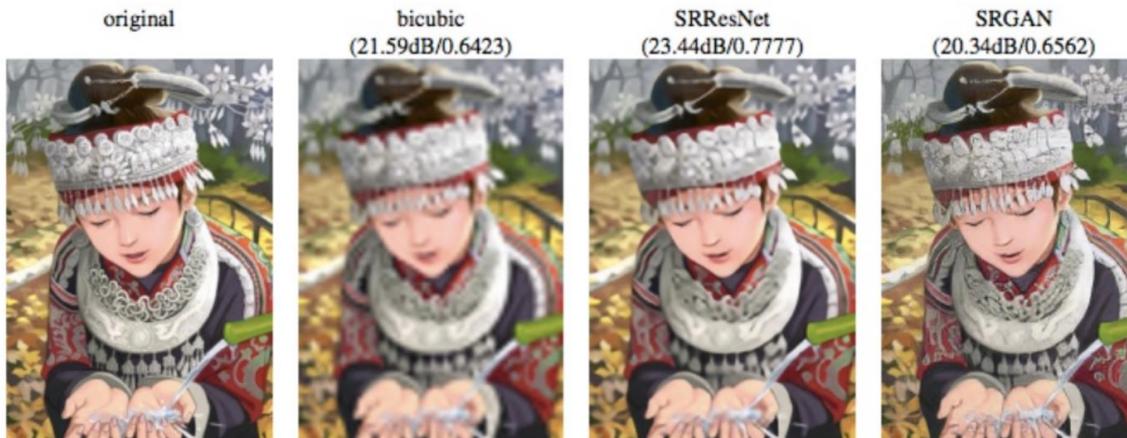
Image translation



[DeepDream 2015]



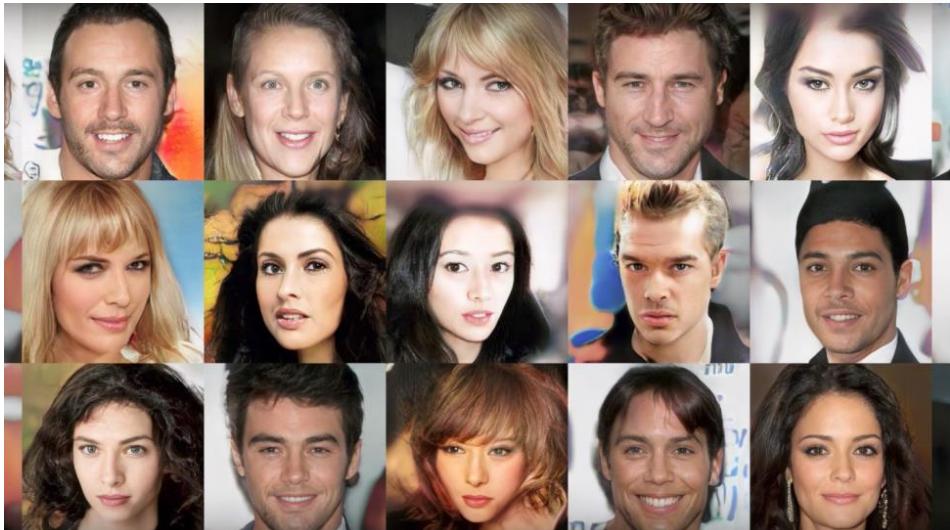
[Gatys 2015]



[Ledig 2016]

Pourquoi le deep learning?

Modèles génératifs



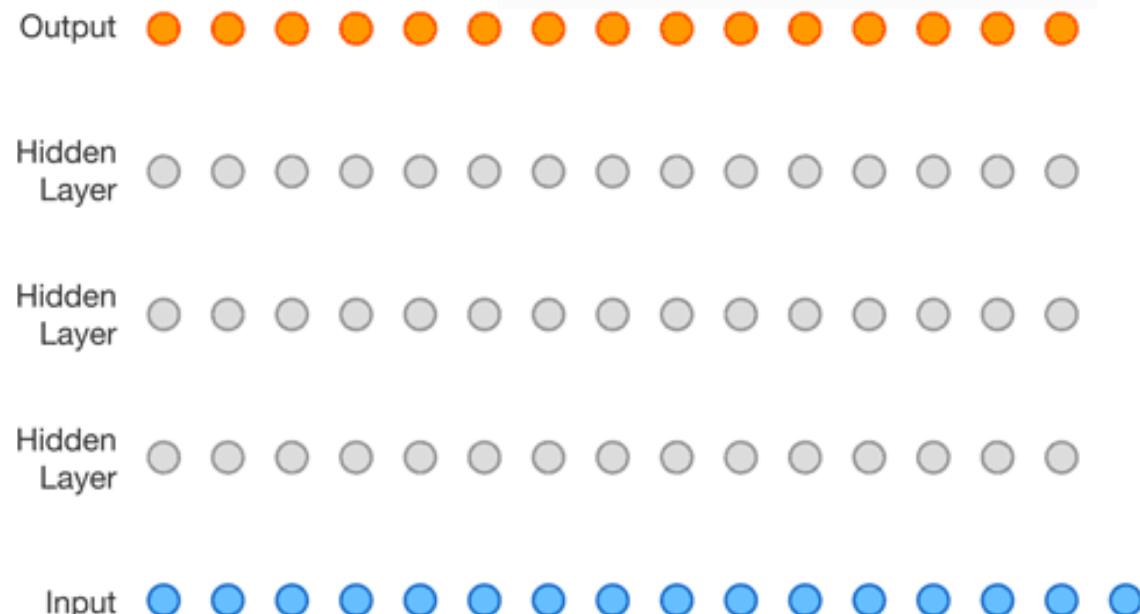
Sampled celebrities [Nvidia 2017]



StackGAN v2 [Zhang 2017]

Pourquoi le deep learning?

Modèles génératifs



Sound generation with WaveNet [DeepMind 2017]

Pourquoi le deep learning?

Modèles génératifs



Laquelle est une voix générée?

Pourquoi le deep learning?

Language / image models

TEXT PROMPT

an armchair in the shape of an avocado [...]

AI-GENERATED IMAGES



[View more or edit prompt ↓](#)

TEXT PROMPT

a store front that has the word 'openai' written on it [...]

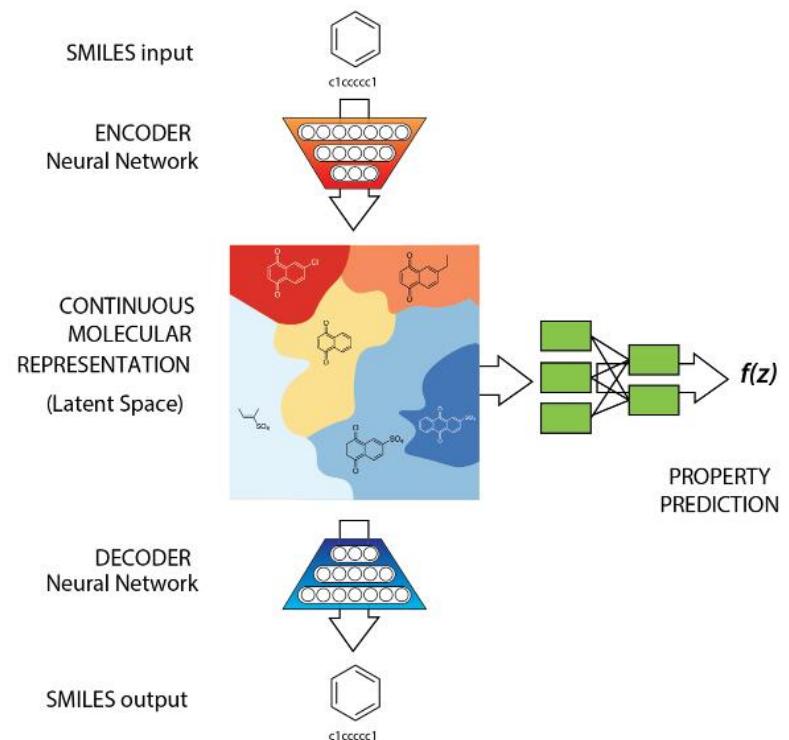
AI-GENERATED IMAGES



[View more or edit prompt ↓](#)

Pourquoi le deep learning?

Chimie / Physique



[Gómez-Bombarelli 2016]



[Tompson 2016]

Pourquoi le deep learning?

Chimie / Physique

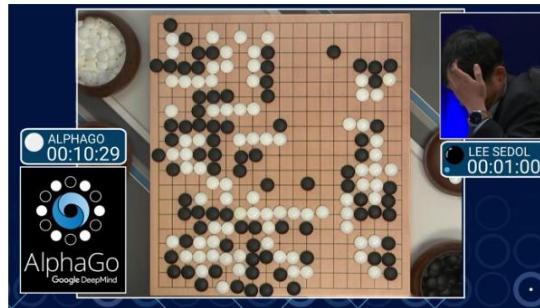


MakeAGIF.com

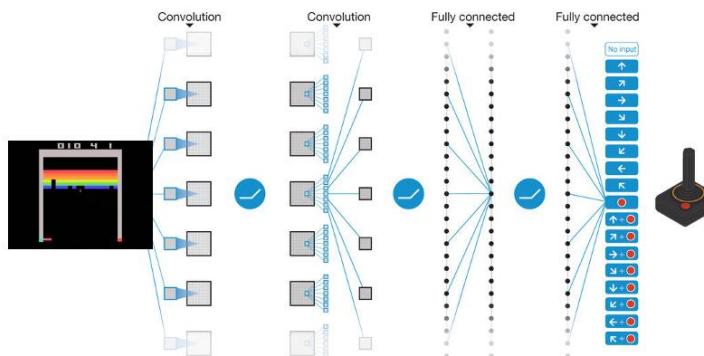
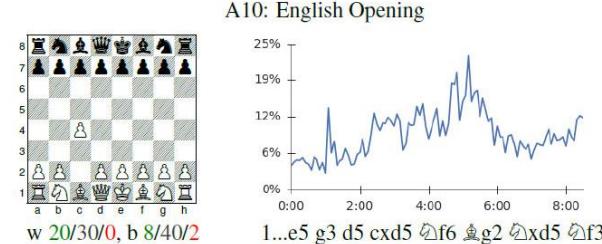
- Finite element simulator accelerated (~100 fold) by a 3D convolutional network

Pourquoi le deep learning?

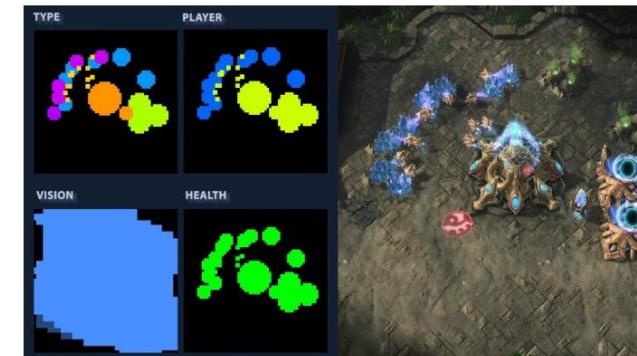
Jeux vidéo



[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]



[Starcraft 2 for AI research]

Introduction au DL



- Fonctionnement des réseaux de neurones
- Pourquoi le deep learning?
- **Librairies & frameworks**

Librairies & frameworks



PYTORCH



Microsoft
CNTK

☕ Caffe2

dmlc
mxnet

gensim **spaCy**

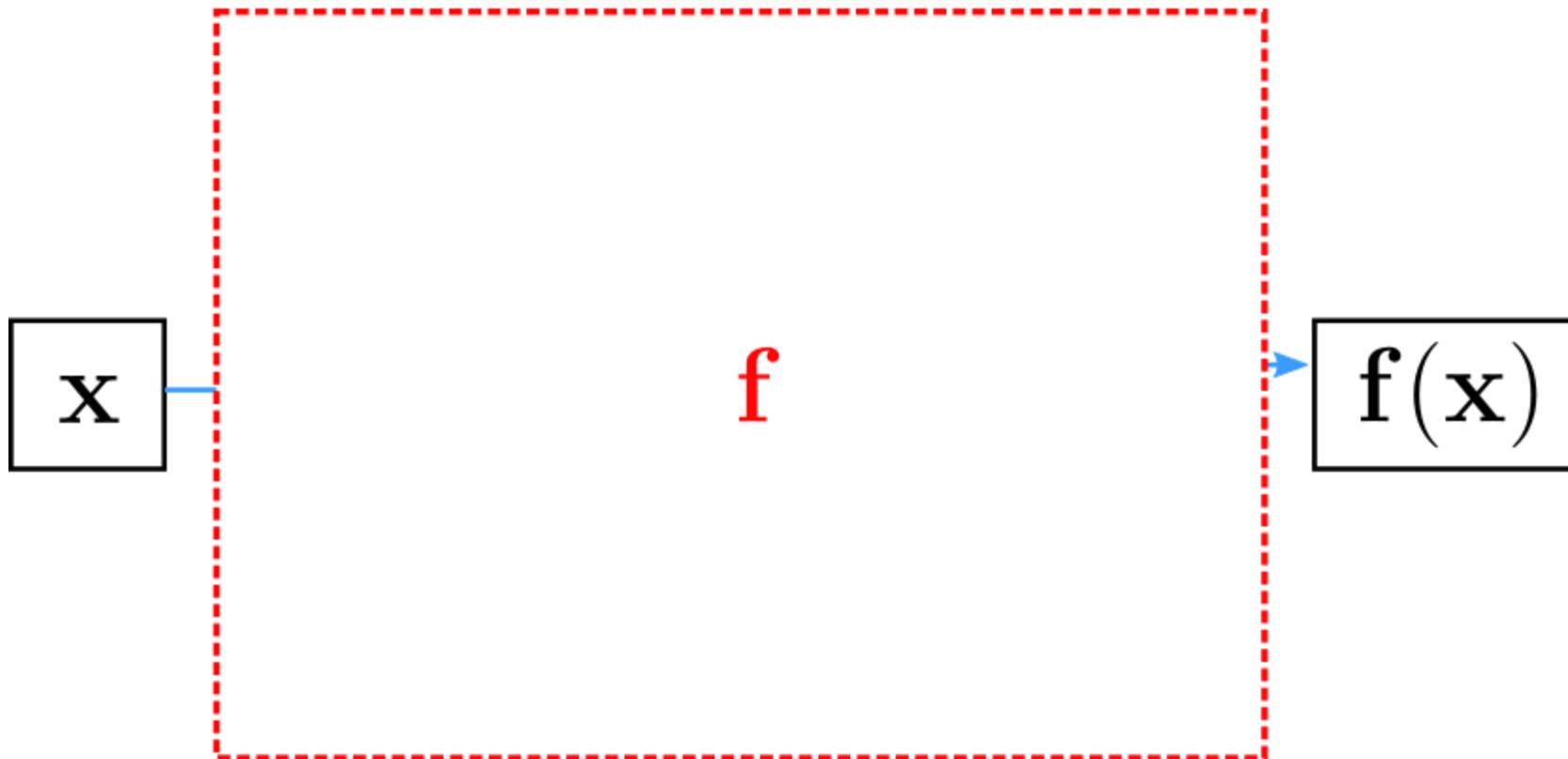
theano

Principes du DL



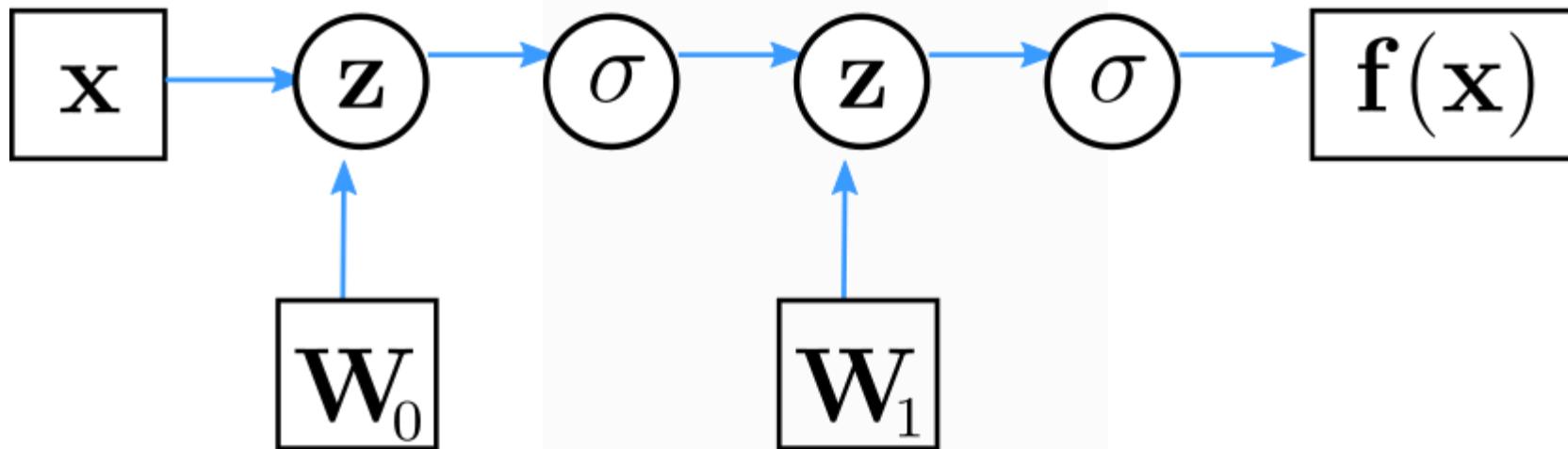
- **Graphe d'apprentissage**
- Neurone et couches de neurones artificiels
- Fonctions d'activation
- Processus d'entraînement
- Processus d'optimisation

Graphe d'apprentissage



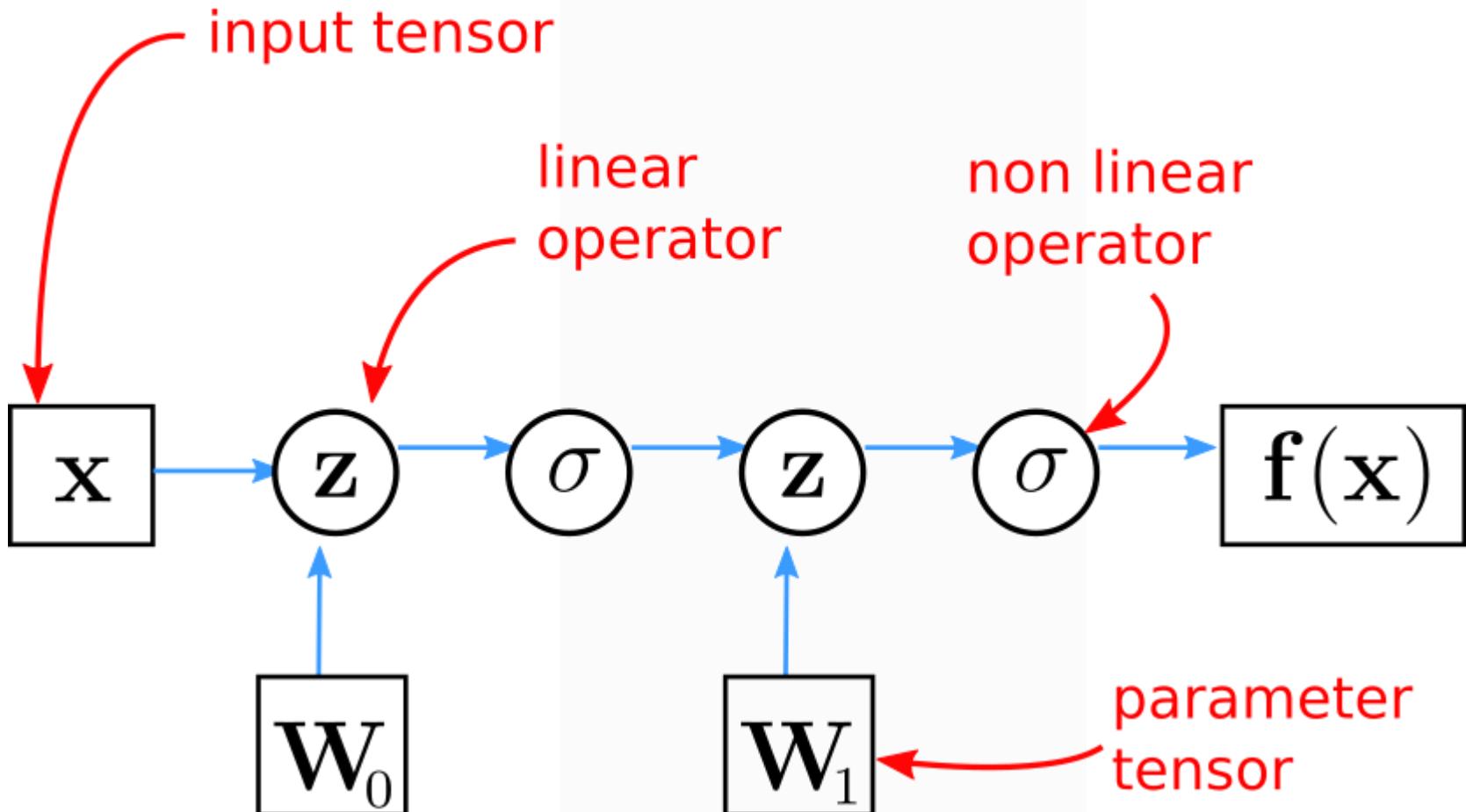
Neural network = parametrized, non-linear function

Graphe d'apprentissage



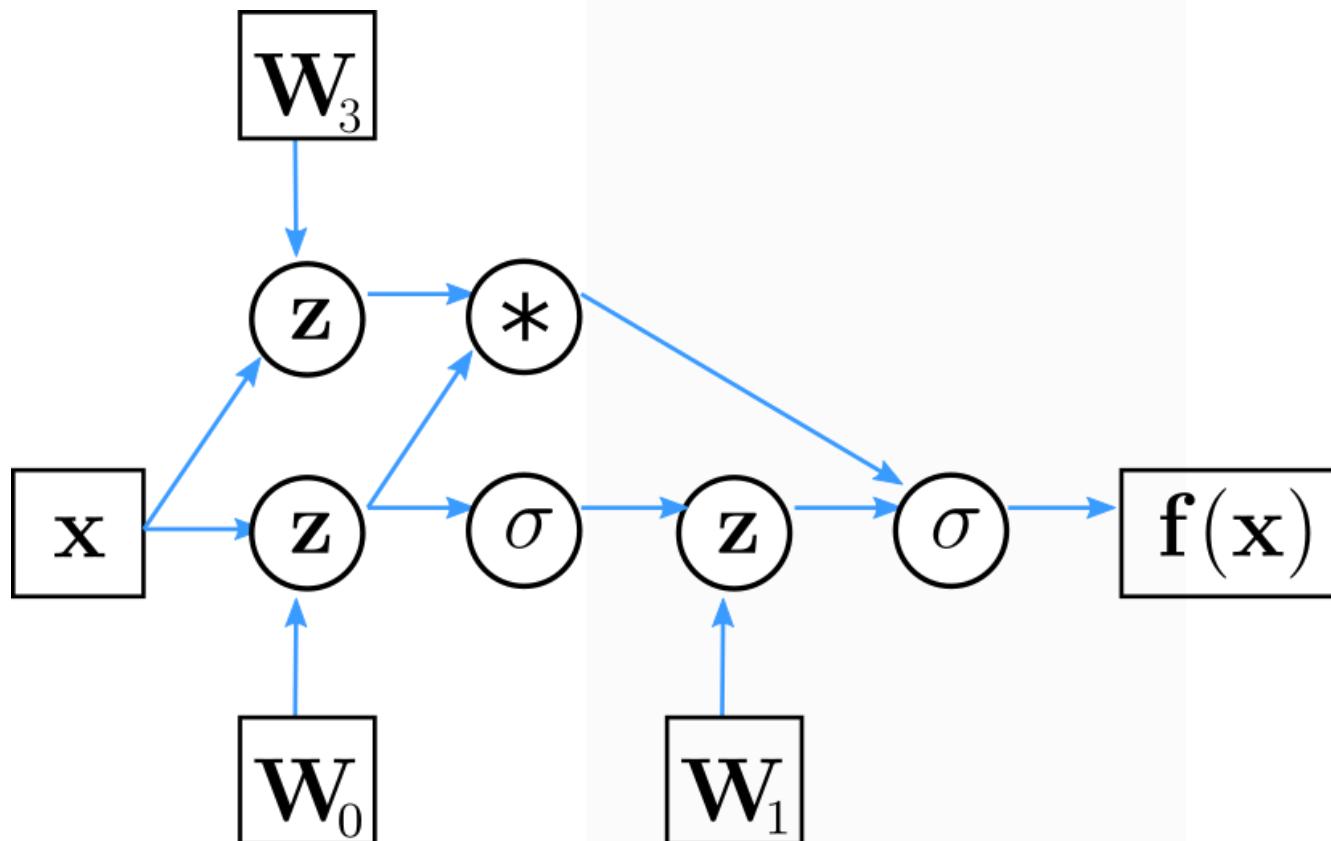
Computation graph: Directed graph of functions, depending on parameters (neuron weights)

Graphe d'apprentissage



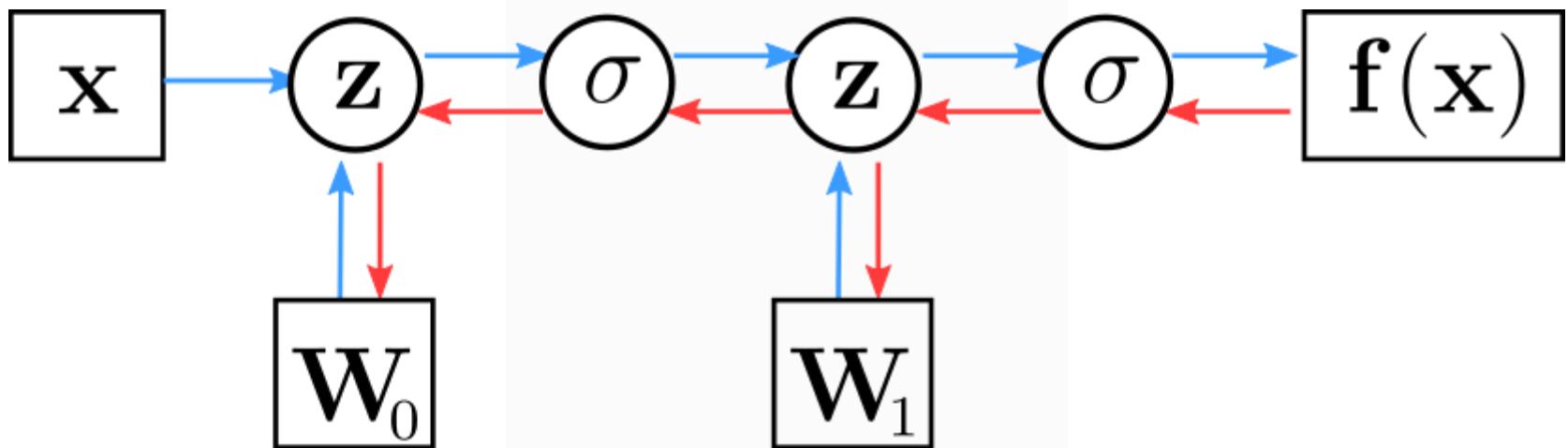
Combination of linear (parametrized) and non-linear functions

Graphe d'apprentissage



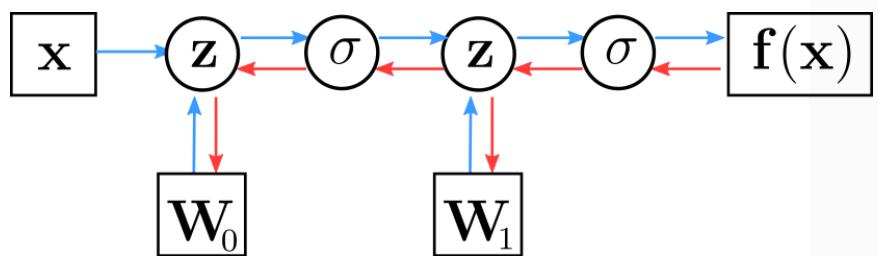
Not only sequential application of functions

Graphe d'apprentissage



Automatic computation of gradients: all modules are **differentiable!**

Graphe d'apprentissage



```
model = Sequential()
model.add(Dense(H, input_dim=N)) # defines W0
model.add(Activation("tanh"))
model.add(Dense(K)) # defines W1
model.add(Activation("softmax"))
```

Principes du DL



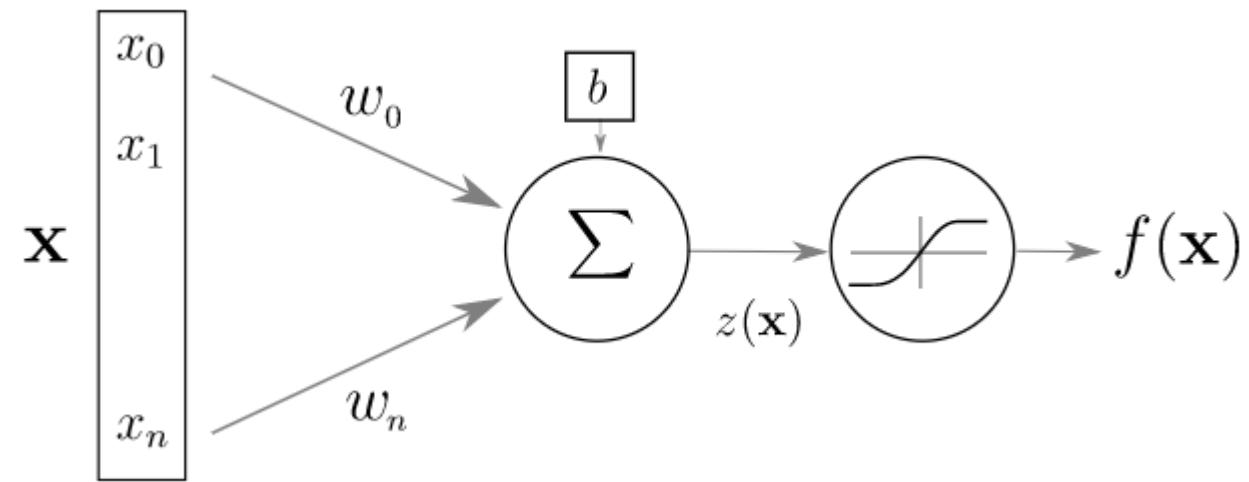
- Graphe d'apprentissage
- **Neurone et couches de neurones artificiels**
- Fonctions d'activation
- Processus d'entraînement
- Processus d'optimisation

Neurone et couches de neurones artificiels

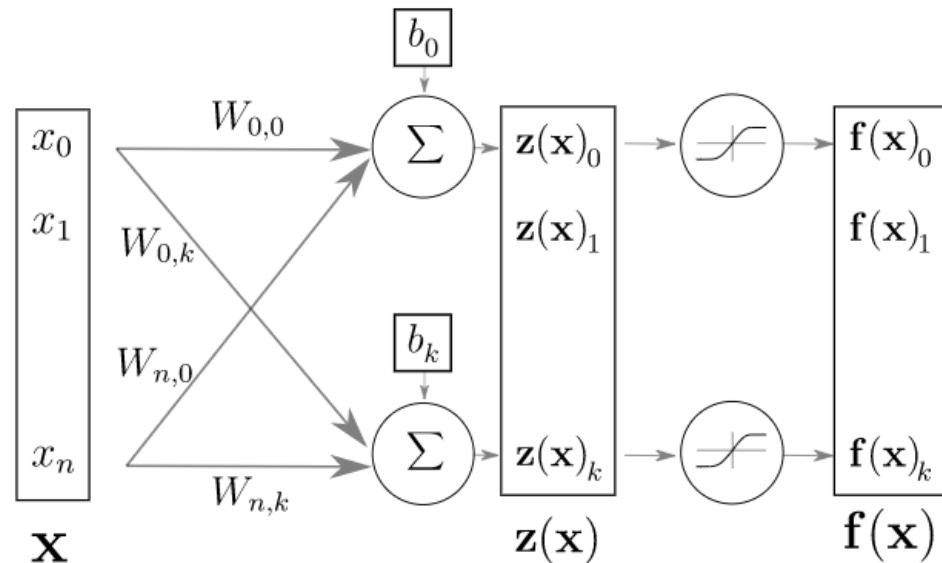
$$z(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

$$f(\mathbf{x}) = g(z(\mathbf{x}))$$

- $\mathbf{x}, f(\mathbf{x})$ input and output
- $z(\mathbf{x})$ pre-activation
- \mathbf{w}, b weights and bias
- g activation function



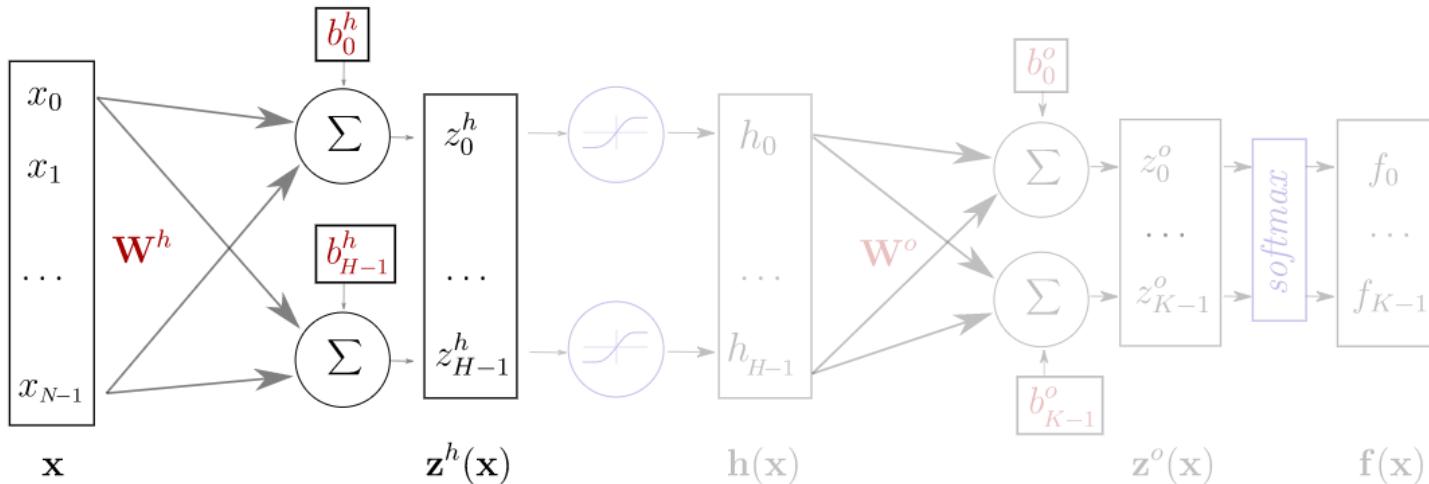
Neurone et couches de neurones artificiels



$$\mathbf{f}(\mathbf{x}) = g(\mathbf{z}(\mathbf{x})) = g(\mathbf{W}\mathbf{x} + \mathbf{b})$$

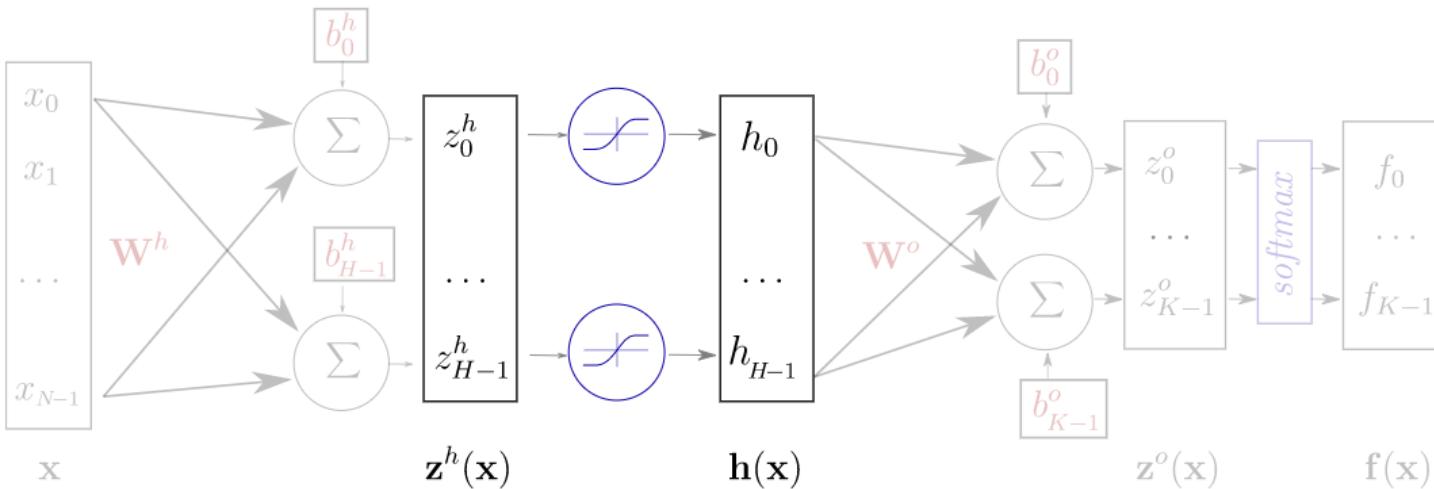
- \mathbf{W}, \mathbf{b} now matrix and vector

Neurone et couches de neurones artificiels



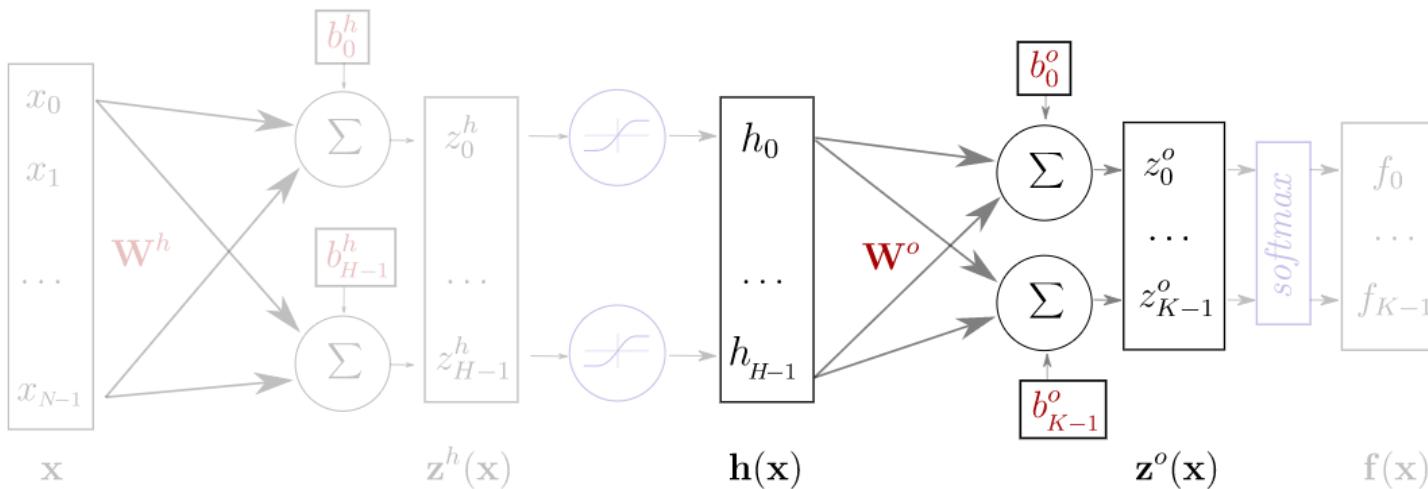
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h \mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h \mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

Neurone et couches de neurones artificiels



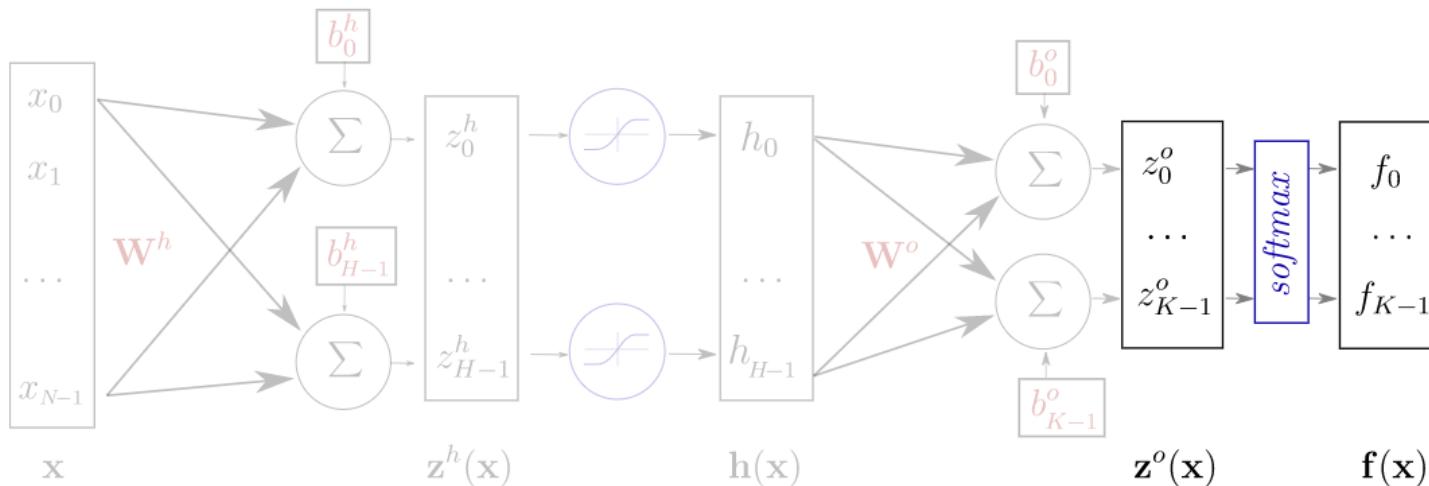
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h \mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h \mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

Neurone et couches de neurones artificiels



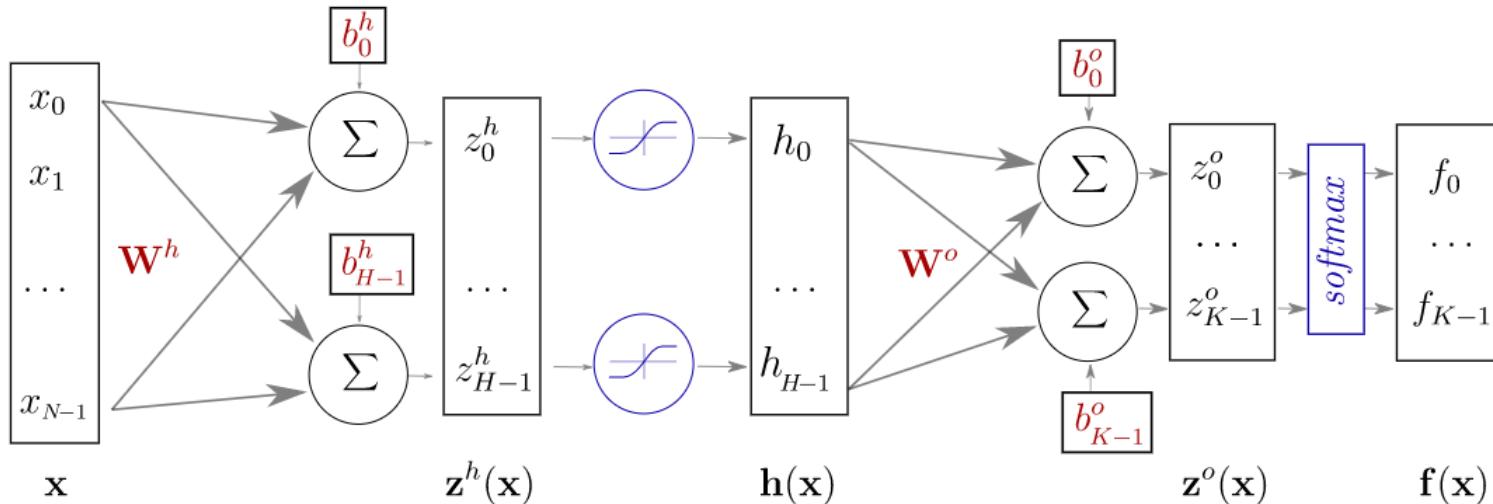
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h \mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h \mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

Neurone et couches de neurones artificiels

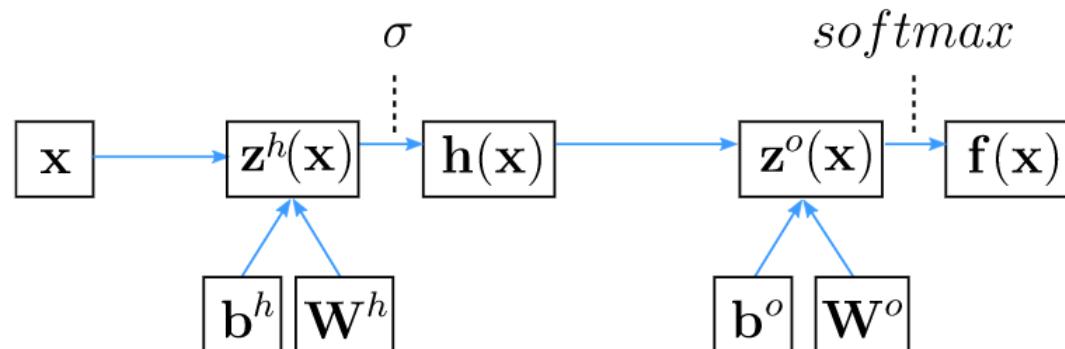


- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h \mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h \mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o \mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

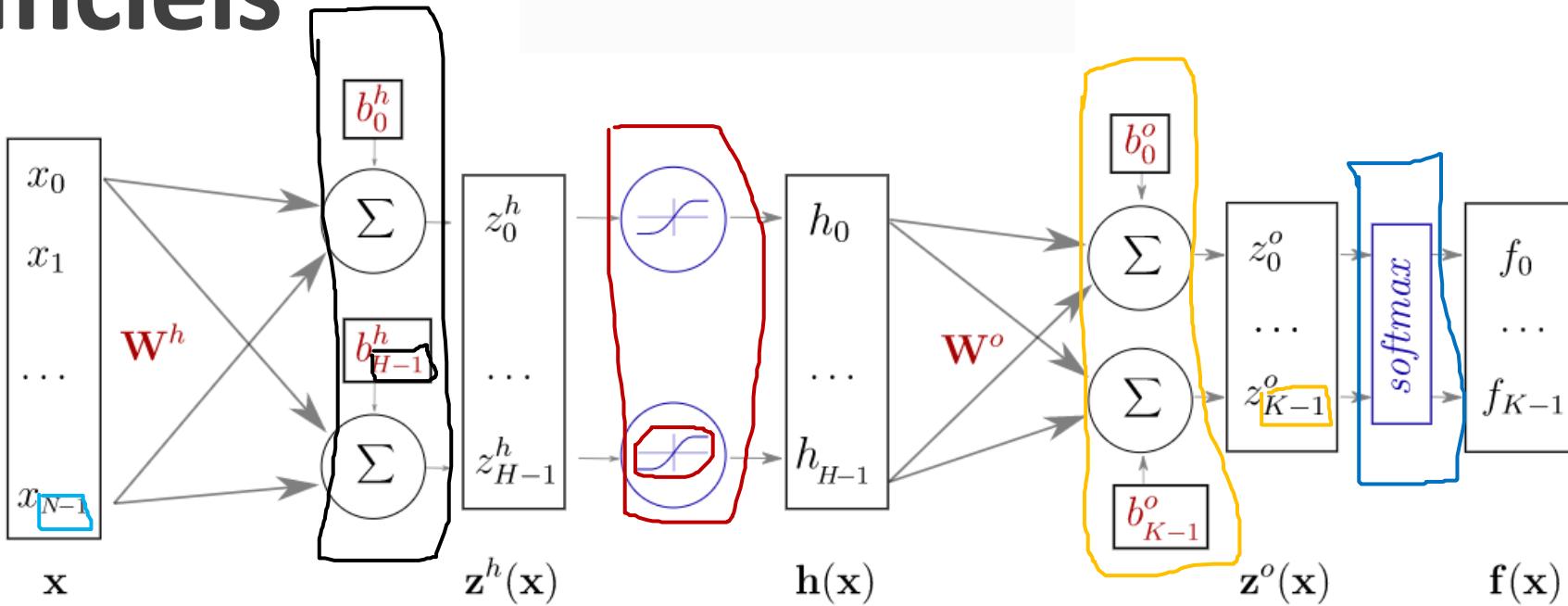
Neurone et couches de neurones artificiels



Alternate representation



Neurone et couches de neurones artificiels



Keras implementation

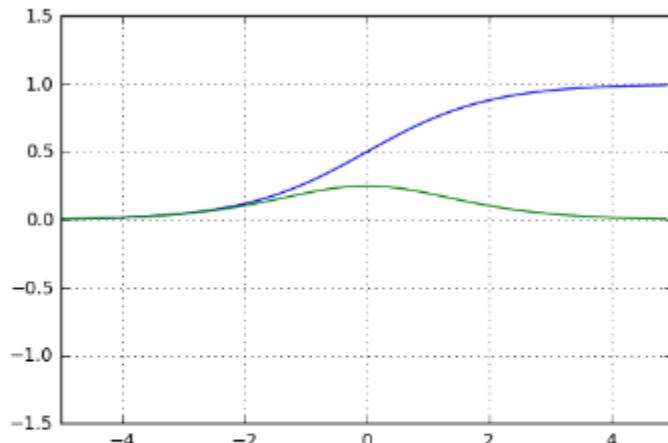
```
model = Sequential()
model.add(Dense(H, input_dim=N)) # weight matrix dim [N * H]
model.add(Activation("tanh"))
model.add(Dense(K)) # weight matrix dim [H x K]
model.add(Activation("softmax"))
```

Principes du DL



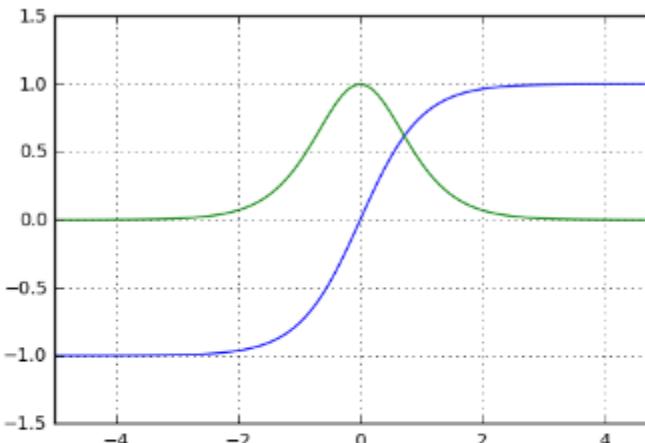
- Graphe d'apprentissage
- Neurone et couches de neurones artificiels
- **Fonctions d'activation**
- Processus d'entraînement
- Processus d'optimisation

Fonctions d'activation



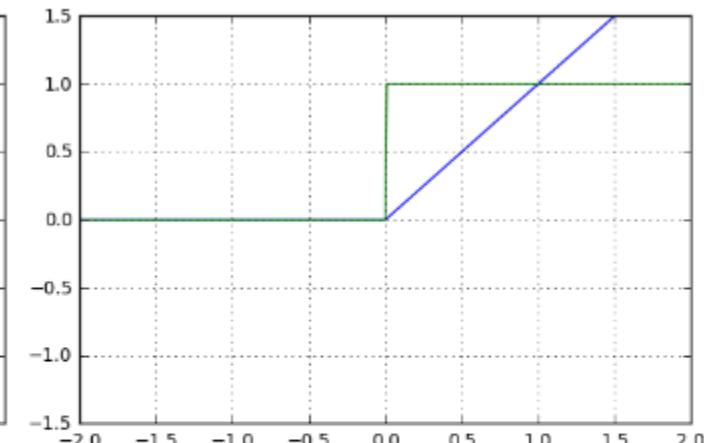
$$\text{sigm}(x) = \frac{1}{1 + e^{-x}}$$

$$\text{sigm}'(x) = \text{sigm}(x)(1 - \text{sigm}(x))$$



$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$\tanh'(x) = 1 - \tanh(x)^2$$



$$\text{relu}(x) = \max(0, x)$$

$$\text{relu}'(x) = 1_{x>0}$$

Fonctions d'activation

Softmax function

$$\text{softmax}(\mathbf{x}) = \frac{1}{\sum_{i=1}^n e^{x_i}} \cdot \begin{bmatrix} e^{x_1} \\ e^{x_2} \\ \vdots \\ e^{x_n} \end{bmatrix}$$

$$\frac{\partial \text{softmax}(\mathbf{x})_i}{\partial x_j} = \begin{cases} \text{softmax}(\mathbf{x})_i \cdot (1 - \text{softmax}(\mathbf{x})_i) & i = j \\ -\text{softmax}(\mathbf{x})_i \cdot \text{softmax}(\mathbf{x})_j & i \neq j \end{cases}$$

- Vecteur de valeurs entre 0 et 1
- La somme des valeurs donne 1
- Il s'agit d'une probabilité conditionnelle de chaque classe en fonction des autres classes
- Les sorties avant la couche softmax sont appelés des « logits »

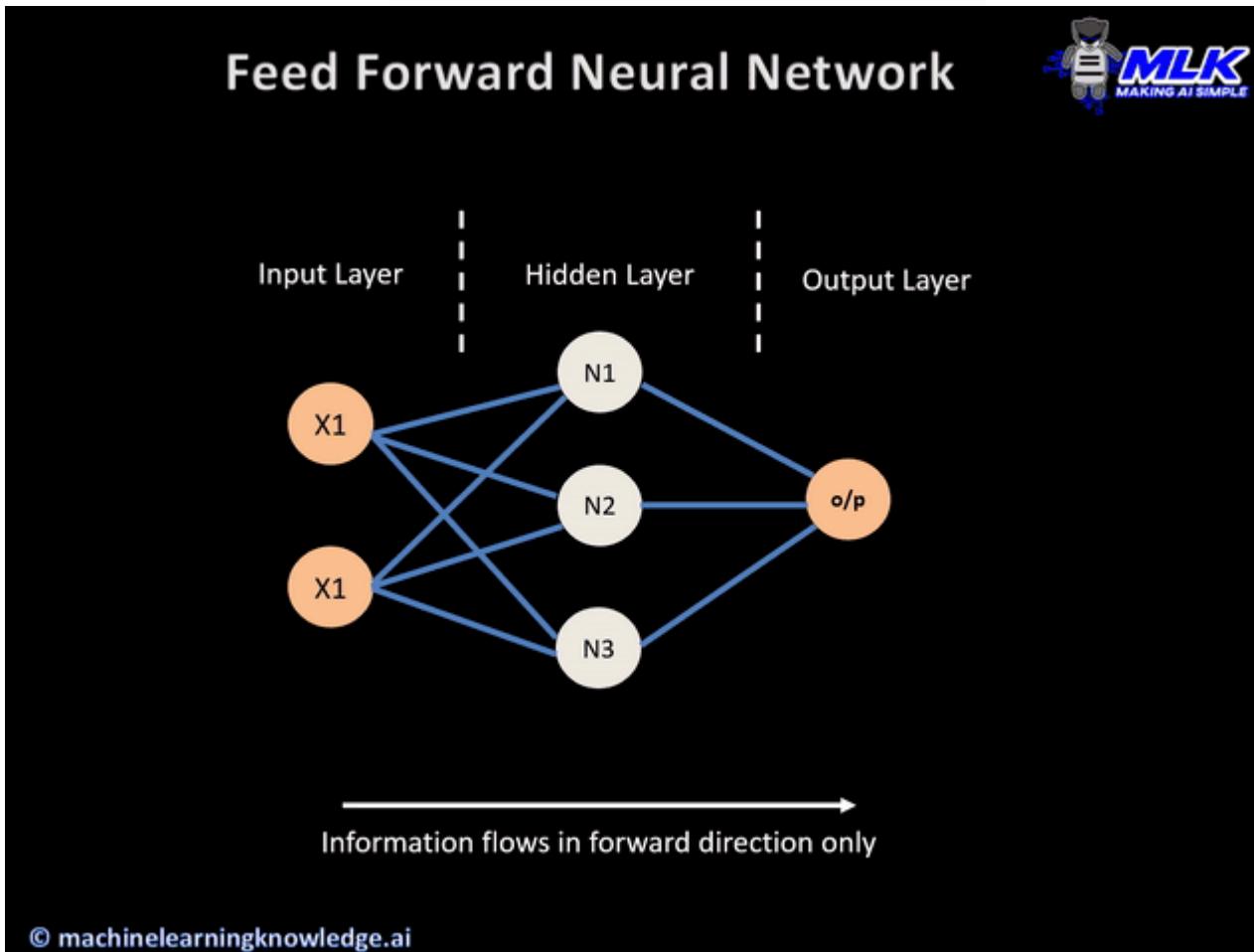
Principes du DL



- Graphe d'apprentissage
- Neurone et couches de neurones artificiels
- Fonctions d'activation
- **Processus d'entraînement**
- Processus d'optimisation

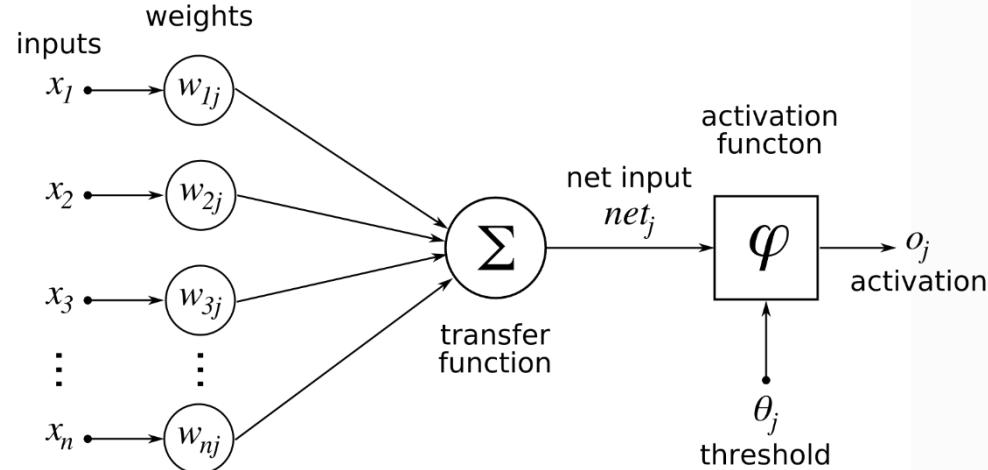
Processus d'entraînement

Forward propagation



Processus d'entraînement

Forward propagation



$$Z = W \cdot X + b$$

$$A = \frac{1}{1 + e^{-Z}}$$

$$\mathcal{L} = \frac{-1}{m} \sum y \cdot \log(A) + (1 - y) \cdot \log(1 - A)$$

- X correspond à l'entrée du neurone (image, signal, texte, etc.)
- W est la matrice des poids (ce qui permet l'entraînement du modèle)
- b est le vecteur qui reflète le biais
- Z est la sortie d'une neurone (assimilé à une régression linéaire)
- A correspond à la fonction d'activation à réaliser en sortie du neurone (assimilé à une régression logistique en cas de sigmoid)
- \mathcal{L} est une valeur qui correspond à l'erreur entre y prédicté et y réel, (y prédicté est la valeur après l'activation)

Processus d'entraînement

Forward propagation – Détails sur le calcul de chaque sortie

$$Z = W \cdot X + b \quad X = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \in \mathbb{R}^{n \times 1} \text{ et } b \in \mathbb{R}$$

Processus d'entraînement

Forward propagation – Détails sur le calcul de chaque sortie

$$\begin{aligned} Z &= \begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(m)} \end{bmatrix} = \begin{bmatrix} w_1 x_1^{(1)} + w_2 x_2^{(1)} + \cdots + w_{10} x_{10}^{(1)} + b \\ w_1 x_1^{(2)} + w_2 x_2^{(2)} + \cdots + w_{10} x_{10}^{(2)} + b \\ \vdots \\ w_1 x_1^{(m)} + w_2 x_2^{(m)} + \cdots + w_{10} x_{10}^{(m)} + b \end{bmatrix} \\ &= \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_{10}^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_{10}^{(2)} \\ \vdots & \dots & & \vdots \\ x_1^{(m)} & x_2^{(m)} & \dots & x_{10}^{(m)} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} + \begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix} = X \cdot W + b \end{aligned}$$

Processus d'entraînement

Forward propagation – Détails sur la fonction de coût

$$\mathcal{L} = \frac{-1}{m} \sum_{i=1}^m \left[y^{(1)} \times \log(a^{(1)}) + \left(1 - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \right) \times \log \left(1 - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \right) \right]$$

Processus d'entraînement

Forward propagation – Détails sur la fonction de coût

$$\mathcal{L} = \frac{-1}{m} \sum_{i=1}^m \begin{bmatrix} y^{(1)} \times \log(a^{(1)}) \\ y^{(2)} \times \log(a^{(2)}) \\ \vdots \\ y^{(m)} \times \log(a^{(m)}) \end{bmatrix} + \left(1 - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \right) \times \log \left(1 - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \right)$$

$$\mathcal{L} = \frac{-1}{m} \sum_{i=1}^m \begin{bmatrix} y^{(1)} \times \log(a^{(1)}) + (1 - y^{(1)}) \times \log(1 - a^{(1)}) \\ y^{(2)} \times \log(a^{(2)}) + (1 - y^{(2)}) \times \log(1 - a^{(2)}) \\ \vdots \\ y^{(m)} \times \log(a^{(m)}) + (1 - y^{(m)}) \times \log(1 - a^{(m)}) \end{bmatrix}$$

Processus d'entraînement

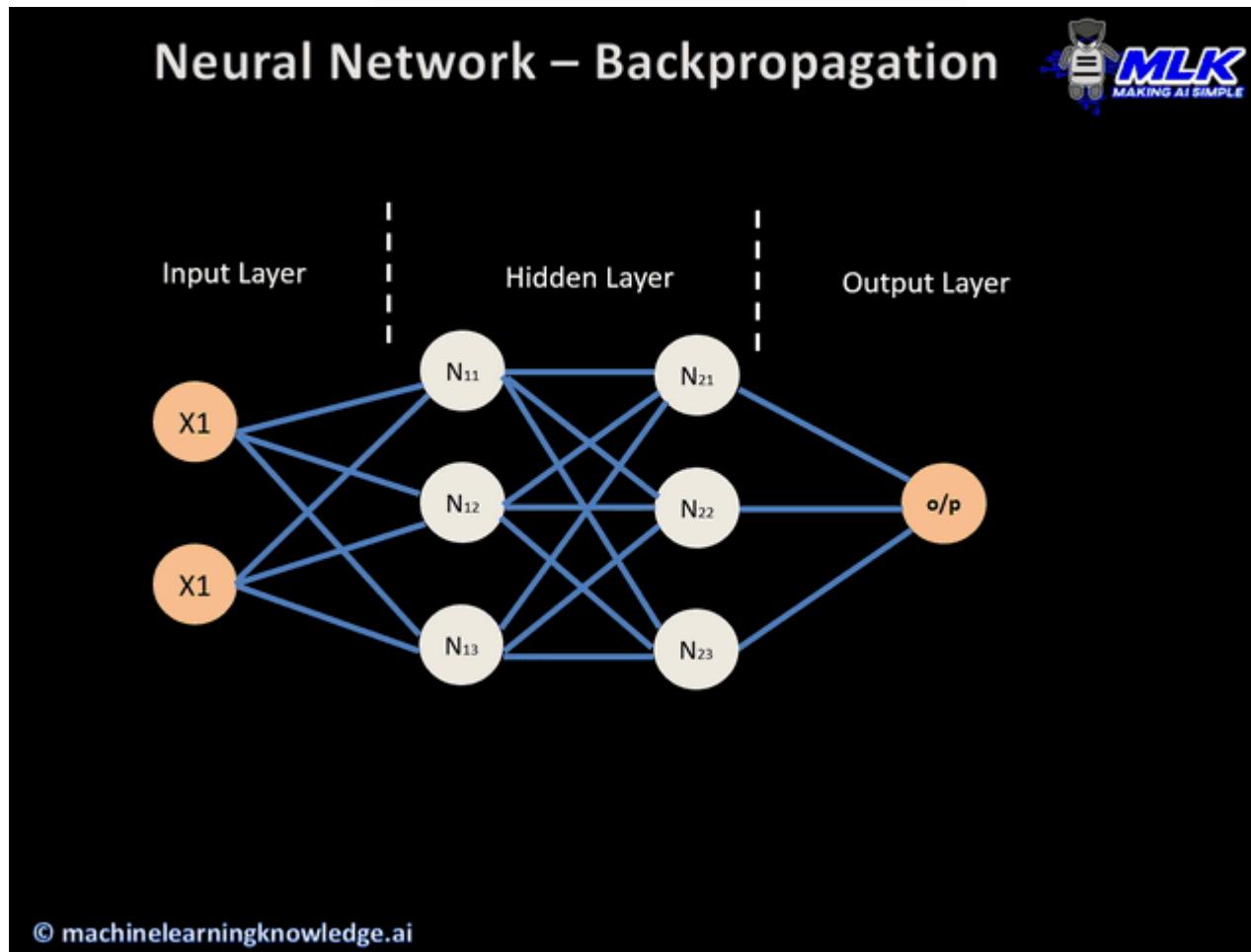
Forward propagation – Détails sur la fonction de coût

$$\mathcal{L} = \frac{-1}{m} \sum_{i=1}^m y^{(1)} \times \log(a^{(1)}) + (1 - y^{(1)}) \times \log(1 - a^{(1)}) + y^{(2)} \times \log(a^{(2)}) + (1 - y^{(2)}) \times \log(1 - a^{(2)}) + \dots + y^{(m)} \times \log(a^{(m)}) + (1 - y^{(m)}) \times \log(1 - a^{(m)})$$

Il s'agit d'une valeur réelle

Processus d'entraînement

Backward propagation



Processus d'entraînement

Backward propagation

1- Calcul de la prédiction:

$$Z = W \cdot X + b$$

$$A = \frac{1}{1 + e^{-Z}}$$

2- Calcul de l'erreur:

$$\mathcal{L} = \frac{-1}{m} \sum y \cdot \log(A) + (1 - y) \cdot \log(1 - A)$$

3- Calcul des dérivées: (donne une information sur la variation de la fonction de perte / erreur)

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial A} \frac{\partial A}{\partial Z} \frac{\partial Z}{\partial W} = \frac{1}{m} \sum (A - y) \cdot X$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial A} \frac{\partial A}{\partial Z} \frac{\partial Z}{\partial b} = \frac{1}{m} \sum (A - y)$$

4- Mise à jour des paramètres du modèle:

$$W = W - \alpha \frac{\partial \mathcal{L}}{\partial W}$$

$$b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

Avec α le taux d'apprentissage
(permet de dire si on fait des grands ou des petits « pas » d'optimisation)

Processus d'entraînement

Backward propagation – Règle de chaîne

Règle de la chaîne en calcul différentiel:

Cette règle permet de simplifier les calculs car la formule de la fonction de perte ne dépend pas directement de la matrice des poids mais dans les formules précédentes, on trouve que:

- \mathcal{L} dépend de A
- A dépend de Z
- Z dépend de W

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial A} \frac{\partial A}{\partial Z} \frac{\partial Z}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial A} \frac{\partial A}{\partial Z} \frac{\partial Z}{\partial b}$$

Processus d'entraînement

Backward propagation – Mise à jour des poids

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \frac{\partial \mathcal{L}}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_{10}} \end{bmatrix}$$

Processus d'entraînement

Backward propagation – Généralisation pour les couches

$$Z^{[1]} = W^{[1]}.X + b^{[1]}$$

$$A^{[1]} = \frac{1}{1 + e^{-Z^{[1]}}}$$

Ici, l'exposant indique la couche concernée

$$Z^{[2]} = W^{[2]}.A^{[1]} + b^{[2]}$$

$$A^{[2]} = \frac{1}{1 + e^{-Z^{[2]}}}$$

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \frac{\partial \mathcal{L}}{\partial A^{[2]}} \frac{\partial A^{[2]}}{\partial Z^{[2]}} \frac{\partial Z^{[2]}}{\partial W^{[2]}}$$

$$\mathcal{L} = \frac{-1}{m} \sum y \cdot \log(A^{[2]}) + (1 - y) \cdot \log(1 - A^{[2]})$$

$$\frac{\partial \mathcal{L}}{\partial b^{[2]}} = \frac{\partial \mathcal{L}}{\partial A^{[2]}} \frac{\partial A^{[2]}}{\partial Z^{[2]}} \frac{\partial Z^{[2]}}{\partial b^{[2]}}$$

Processus d'entraînement

Backward propagation – Généralisation pour les couches

$$Z^{[1]} = W^{[1]}.X + b^{[1]}$$

$$A^{[1]} = \frac{1}{1 + e^{-Z^{[1]}}}$$

$$Z^{[2]} = W^{[2]}.A^{[1]} + b^{[2]}$$

$$A^{[2]} = \frac{1}{1 + e^{-Z^{[2]}}}$$

$$\mathcal{L} = \frac{-1}{m} \sum y \cdot \log(A^{[2]}) + (1 - y) \cdot \log(1 - A^{[2]})$$

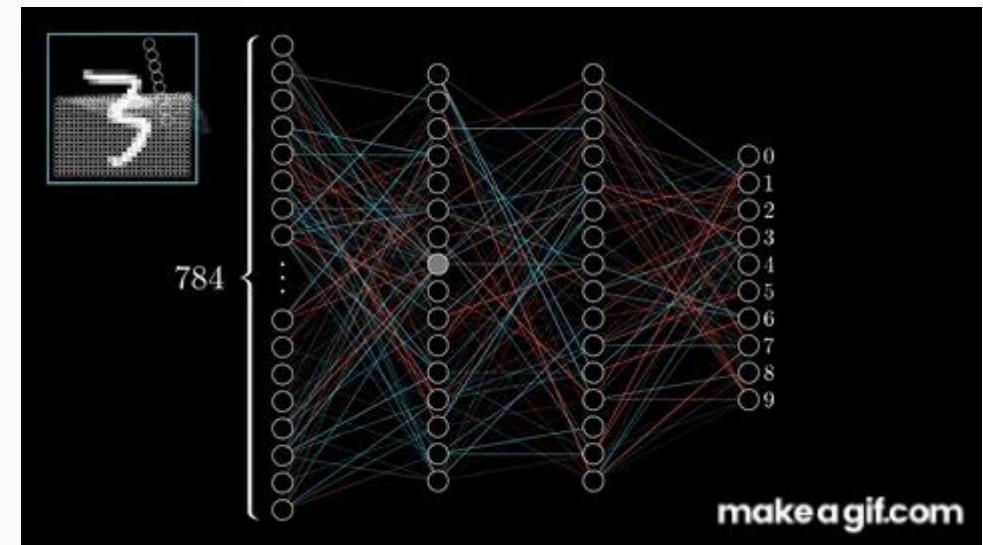
$$\frac{\partial \mathcal{L}}{\partial W^{[1]}} = \frac{\partial \mathcal{L}}{\partial A^{[2]}} \frac{\partial A^{[2]}}{\partial Z^{[2]}} \frac{\partial Z^{[2]}}{\partial A^{[1]}} \frac{\partial A^{[1]}}{\partial Z^{[1]}} \frac{\partial Z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial \mathcal{L}}{\partial b^{[2]}} = \frac{\partial \mathcal{L}}{\partial A^{[2]}} \frac{\partial A^{[2]}}{\partial Z^{[2]}} \frac{\partial Z^{[2]}}{\partial A^{[1]}} \frac{\partial A^{[1]}}{\partial Z^{[1]}} \frac{\partial Z^{[1]}}{\partial b^{[1]}}$$

Processus d'entraînement

Calcul des sorties

- Binary classification: $y \in [0, 1]$
 - $Y|X = \mathbf{x} \sim Bernoulli(b = f(\mathbf{x}; \theta))$
 - output function: $logistic(x) = \frac{1}{1+e^{-x}}$
 - loss function: binary cross-entropy
- Multiclass classification: $y \in [0, K - 1]$
 - $Y|X = \mathbf{x} \sim Multinoulli(\mathbf{p} = \mathbf{f}(\mathbf{x}; \theta))$
 - output function: softmax
 - loss function: categorical cross-entropy
- Continuous output: $\mathbf{y} \in \mathbb{R}^n$
 - $Y|X = \mathbf{x} \sim \mathcal{N}(\mu = \mathbf{f}(\mathbf{x}; \theta), \sigma^2 \mathbf{I})$
 - output function: Identity
 - loss function: square loss



Principes du DL

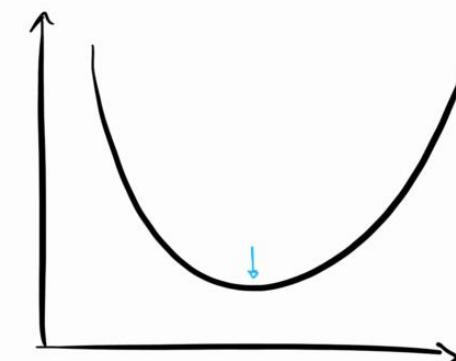


- Graphe d'apprentissage
- Neurone et couches de neurones artificiels
- Fonctions d'activation
- Processus d'entraînement
- **Processus d'optimisation**

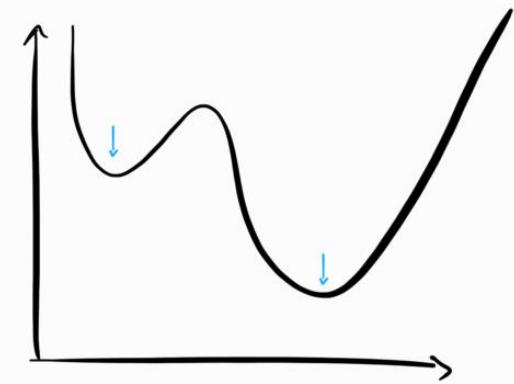
Processus d'optimisation

Descente du gradient

La **Descente de Gradient** est un algorithme d'optimisation qui permet de trouver le **minimum** de n'importe quelle **fonction convexe** en convergeant progressivement vers celui-ci



Fonction Convexe



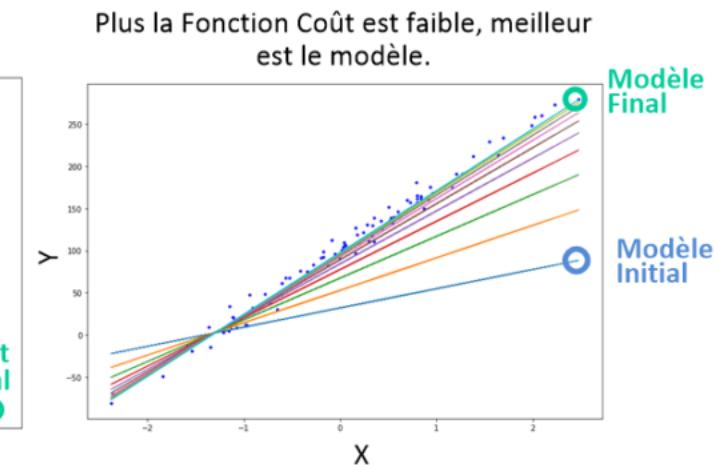
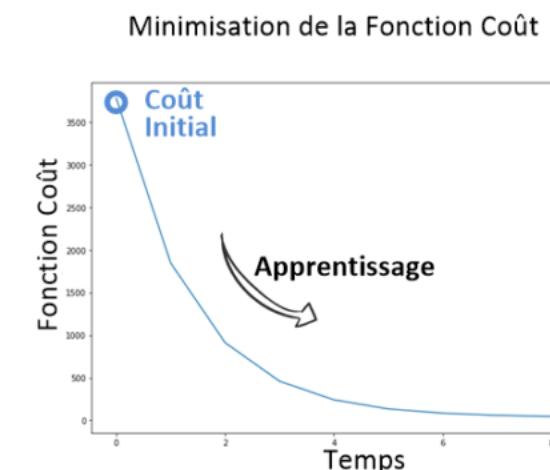
Fonction Non-Convexe

Processus d'optimisation

Descente du gradient

En Machine Learning, on va utiliser l'algorithme de la Descente de Gradient dans les problèmes d'apprentissage supervisé pour minimiser la **fonction coût**, qui justement est une fonction convexe (par exemple l'erreur quadratique moyenne).

C'est grâce à cet algorithme que la machine **apprend**, c'est-à-dire trouve le **meilleur modèle**. En effet, rappelez-vous que minimiser la fonction coût revient à trouver les paramètres a , b , c , etc. qui donnent les plus petites erreurs entre notre modèle et les points y du Dataset. Une fois la fonction coût minimisée.



Processus d'optimisation

Descente du gradient - Etapes

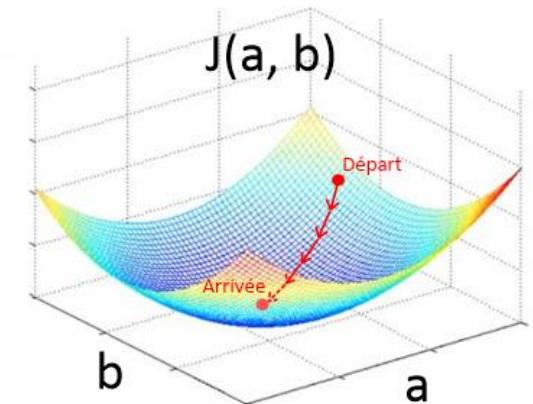
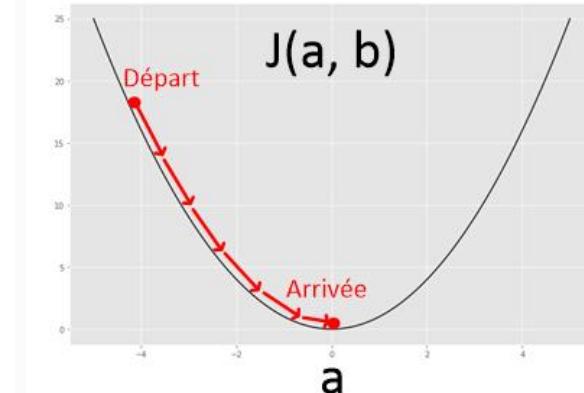
Initialize θ randomly

For E epochs perform:

- Randomly select a small batch of samples ($B \subset S$)
 - Compute gradients: $\Delta = \nabla_{\theta} L_B(\theta)$
 - Update parameters: $\theta \leftarrow \theta - \eta \Delta$
 - $\eta > 0$ is called the learning rate
- Repeat until the epoch is completed (all of S is covered)

Stop when reaching criterion:

- null stops decreasing when computed on validation set



Processus d'optimisation

Descente du gradient - Etapes

- **Dataset :** (x, y) avec m exemples

- **Modèle :** $f(x) = ax + b$

- **Fonction Coût :** $J(a, b) = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$

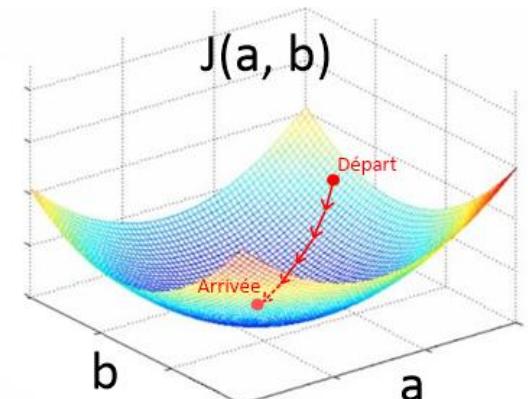
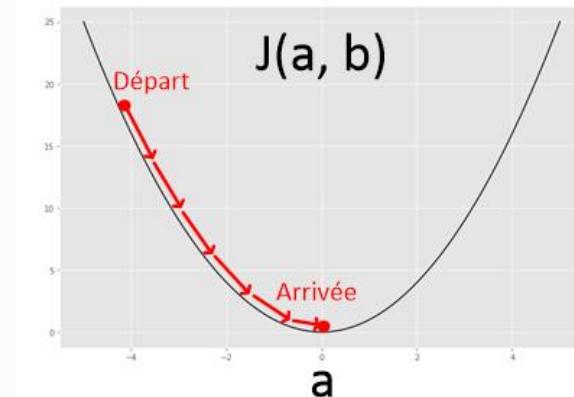
- **Gradients :**

$$\frac{\partial J(a, b)}{\partial a} = \frac{1}{m} \sum_{i=1}^m x^{(i)}(ax^{(i)} + b - y^{(i)})$$

$$\frac{\partial J(a, b)}{\partial b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$

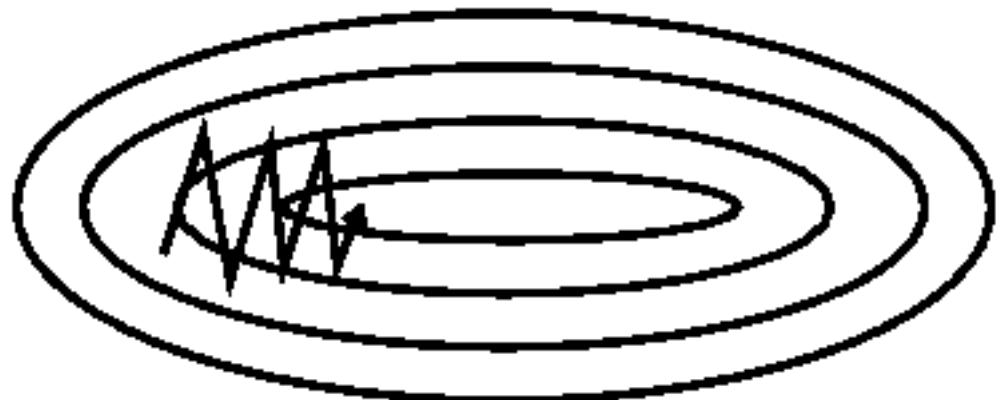
- **Algorithme de Gradient Descent :**


$$\begin{cases} a := a - \alpha \frac{\partial J(a, b)}{\partial a} \\ b := b - \alpha \frac{\partial J(a, b)}{\partial b} \end{cases}$$

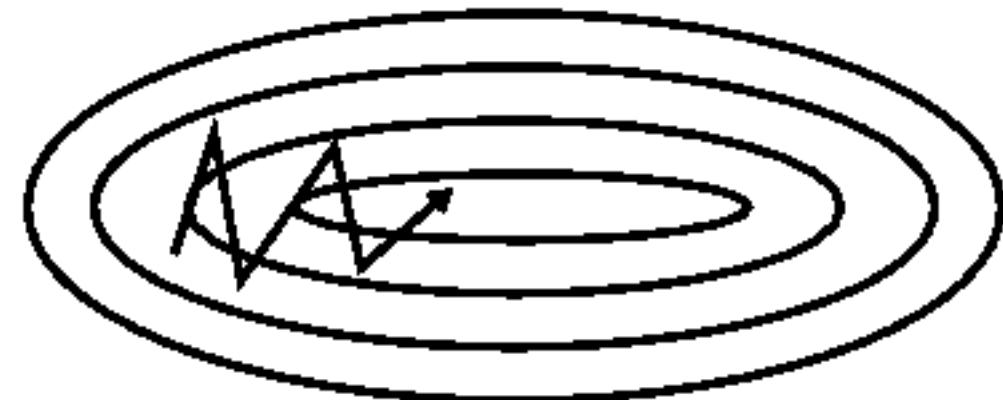


Processus d'optimisation

Descente du gradient - Momentum



SGD sans momentum

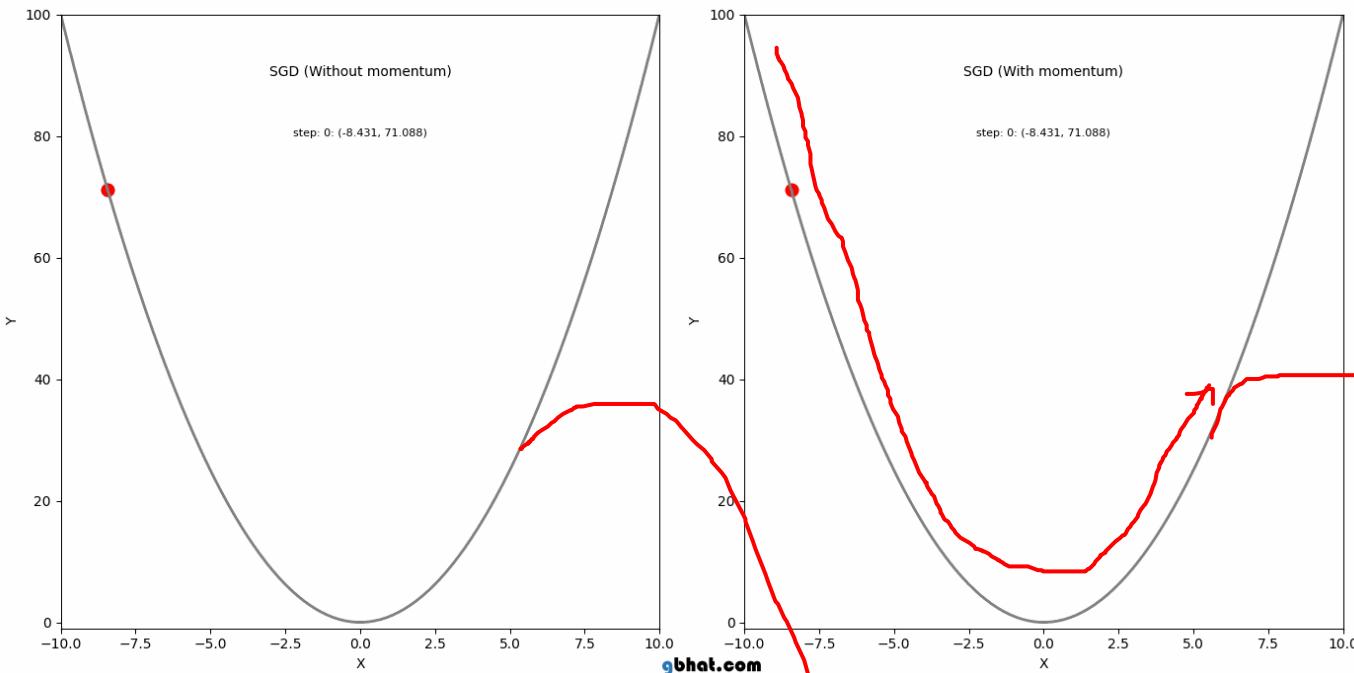


SGD avec momentum

On peut contrôler la vitesse de descente du gradient

Processus d'optimisation

Descente du gradient - Momentum

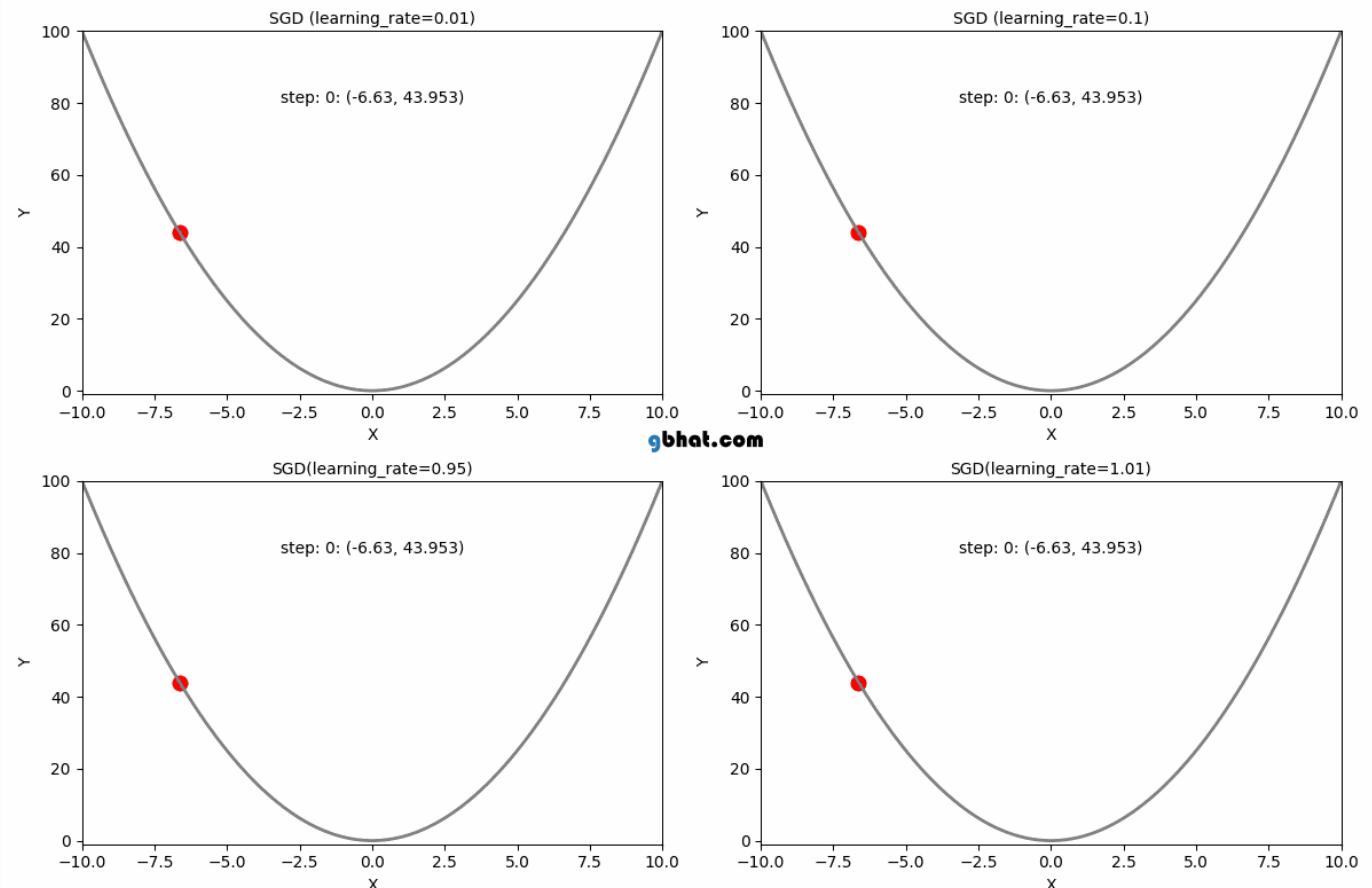


Momentum est une extension de l'algorithme d'optimisation de descente de gradient qui permet à la recherche de créer de l'inertie dans une direction dans l'espace de recherche et de surmonter les oscillations de gradients bruyants et de traverser des points plats de l'espace de recherche. Ce principe permet d'accumuler les gradients au long des différentes mise à jours liées au gradient.

Processus d'optimisation

Descente du gradient – Taux d'apprentissage

Le **taux d'apprentissage** est un **hyperparamètre** qui joue sur la **rapidité** de la descente de gradient : un nombre d'itérations plus ou moins important est nécessaire avant que l'algorithme ne **converge**, c'est-à-dire qu'un apprentissage optimal du réseau soit réalisé.



Processus d'optimisation

Descente du gradient – Taux d'apprentissage

- Très sensible:
 - **Très grande valeur:** Capable de sauter le minimum et peut même diverger
 - **Très petite valeur:** Convergence très lente

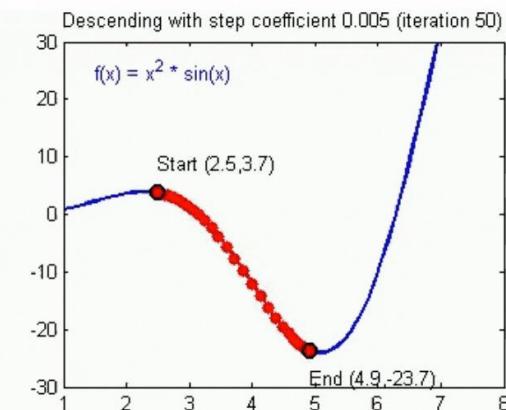
Astuce 1:

- Commencer par une grande valeur 0.1 ou 1
- Diviser par 10 à chaque étape
- Réessayer en cas de divergence

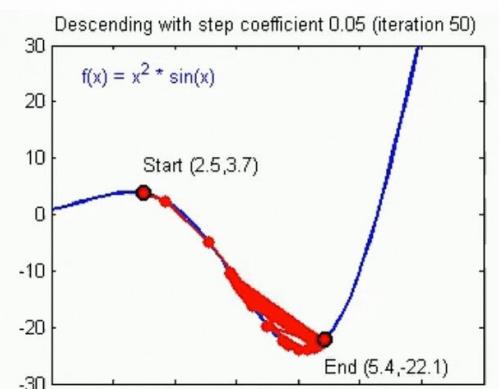
Astuce 2:

- Commencer par une grande valeur 0.1 ou 1
- Multiplier par $\beta < 1$ à chaque étape
- (Implémentation: fonction ReduceLROnPlateau)

Convergence



Divergence

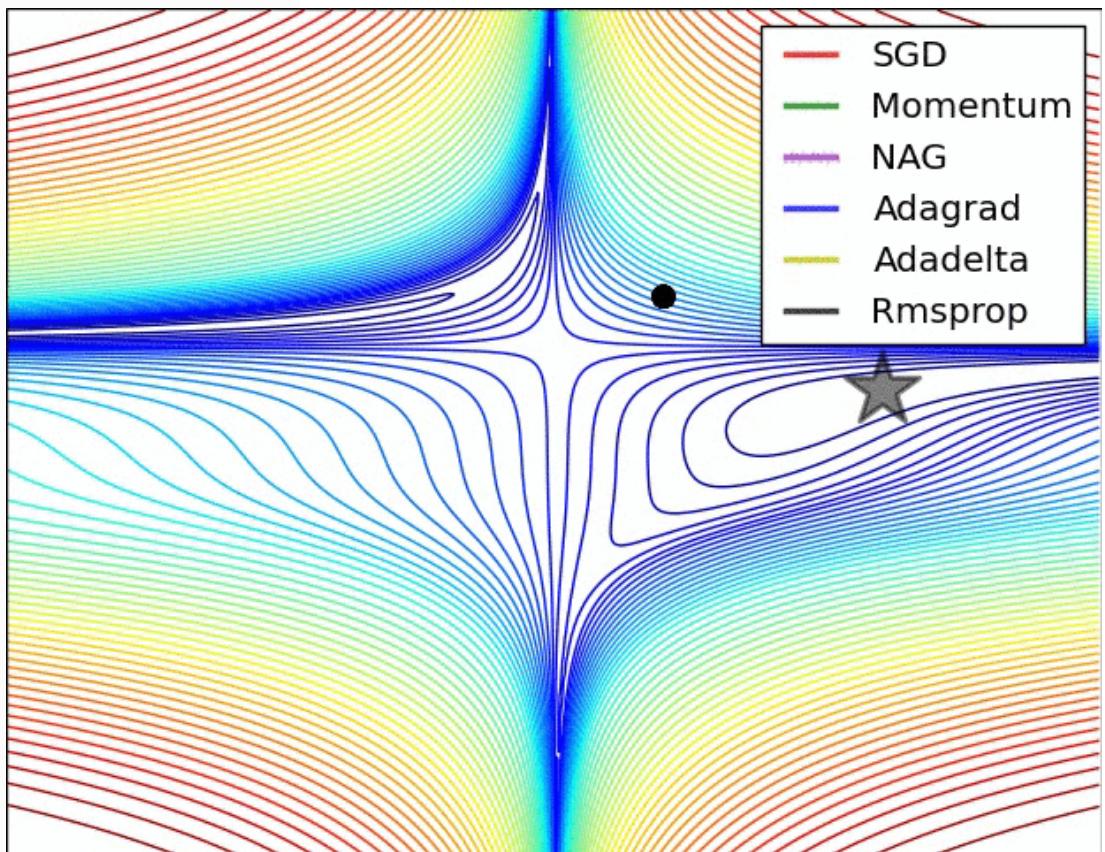


Processus d'optimisation

Alternatives

Algorithmes:

- SGD
- Adagrad
- Adadelta
- Rmsprop
- Adam
- AdamW
- Adamax
- ...



Processus d'optimisation

Récapitulatif des hyperparamètres d'entraînement

Hyperparamètres:

- Train-test split ratio
- Learning rate in optimization algorithms (e.g. gradient descent)
- Choice of optimization algorithm (e.g., gradient descent, stochastic gradient descent, or Adam optimizer)
- Choice of activation function in a neural network (nn) layer (e.g. Sigmoid, ReLU, Tanh)
- The choice of cost or loss function the model will use
- Number of hidden layers in a nn
- Number of activation units in each layer
- The drop-out rate in nn (dropout probability)
- Number of iterations (epochs) in training a nn
- Number of clusters in a clustering task
- Kernel or filter size in convolutional layers
- Pooling size
- Batch size

Paramètres:

- The coefficients (or weights) of linear and logistic regression models.
- Weights and biases of a nn
- The cluster centroids in clustering

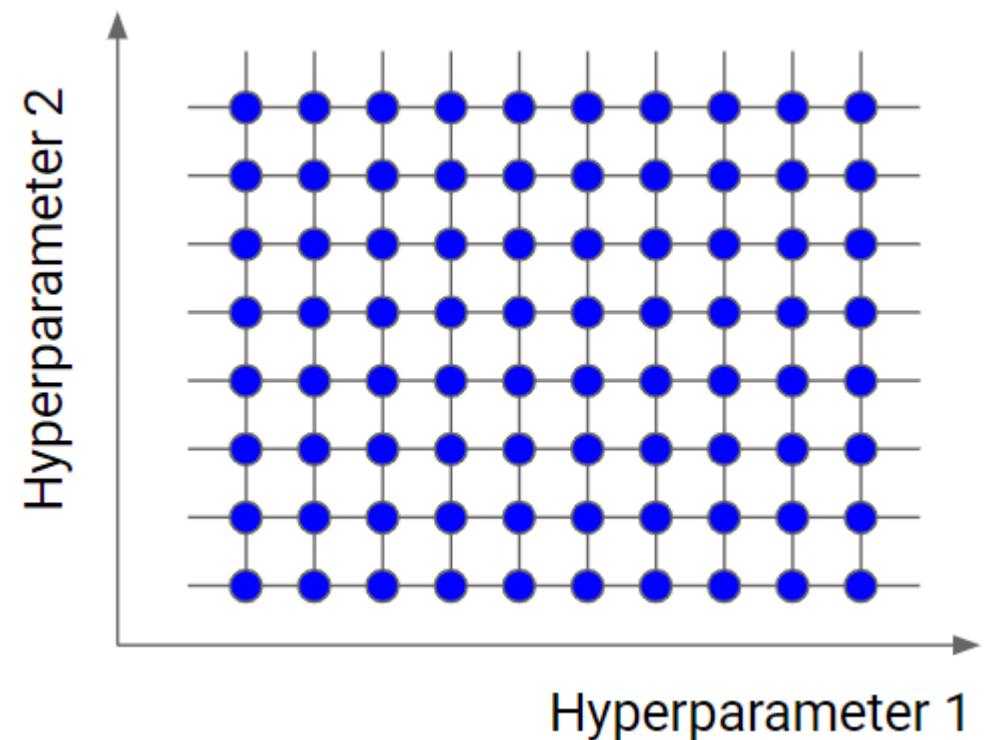
Processus d'optimisation

Recherche d'hyperparamètres – Grid-search

La méthode la plus **naïve** pour la **recherche d'hyperparamètres** est la recherche systématique, ou grid search. Elle consiste à explorer **toutes les combinaisons** de valeurs possibles dans un ensemble donné.

```
tuned_parameters = {'hidden_layer_sizes':[(5,), (20,), (50,), (100,), (150,), (200,)],  
                    'alpha': [0.001, 0.01, 1, 2]}
```

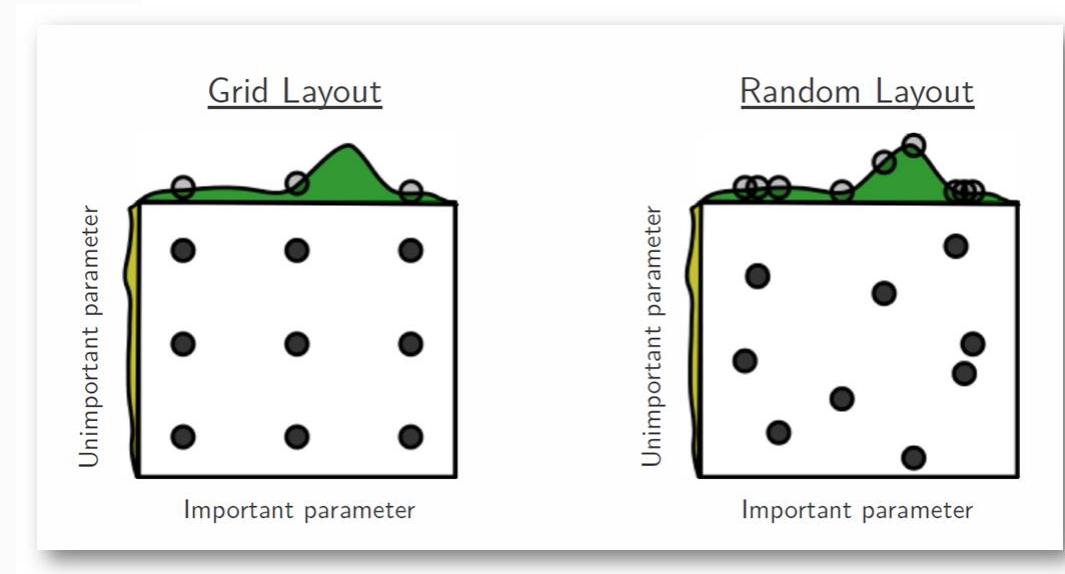
```
clf = GridSearchCV(MLPClassifier(solver='lbfgs', tol=5e-3), tuned_parameters, cv=5)  
  
# exécution de grid search  
clf.fit(X_train, y_train)
```



Processus d'optimisation

Recherche d'hyperparamètres – random-search

Cette fois, plutôt que de fixer un ensemble de valeurs prises par un hyperparamètre donné, on va fixer un intervalle de valeurs et on va associer à chaque hyperparamètre une distribution de probabilités. Ensuite, à chaque itération, on entraîne le modèle et on garde les résultats de performances en mémoire.



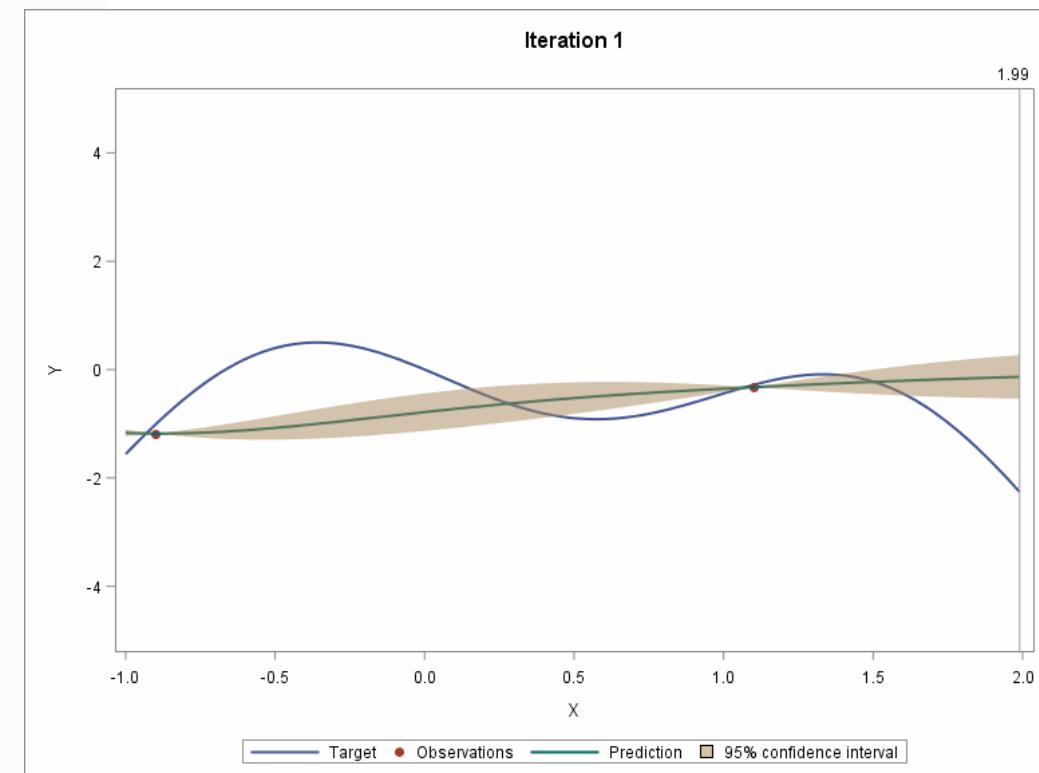
Processus d'optimisation

Recherche d'hyperparamètres – Approches bayésiennes

Comme en statistiques bayésiennes, on va fournir une **information a priori** à l'algorithme pour qu'il puisse faire ses recherches en laissant de côté les zones avec des performances trop faibles.

Avantages:

- Consomme moins de temps
- Recherche intelligente (évite les zones où les performances sont faibles)
- Garde en mémoire les résultats des anciens tests

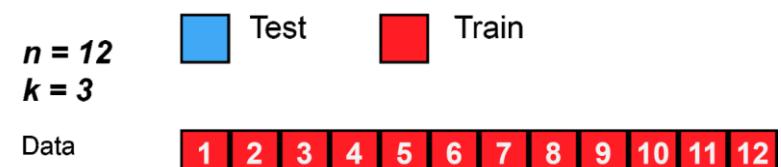


Processus d'optimisation

Cross validation

Afin d'évaluer les performances d'un modèle de Machine Learning, il est nécessaire de le tester sur de nouvelles données. En fonction des performances des modèles sur des données inconnues, on peut déterminer s'il est sous-ajusté, sur-ajusté, ou « bien généralisé ».

Cette méthode est aussi une procédure de « re-sampling » (ré-échantillonnage) permettant d'évaluer un modèle même avec des données limitées.



Processus d'optimisation

Cross validation

