

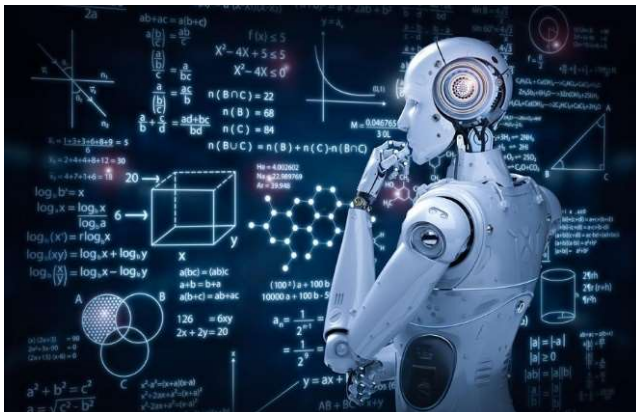
APPRENTISSAGE PROFOND ET IMAGES

LES BASES

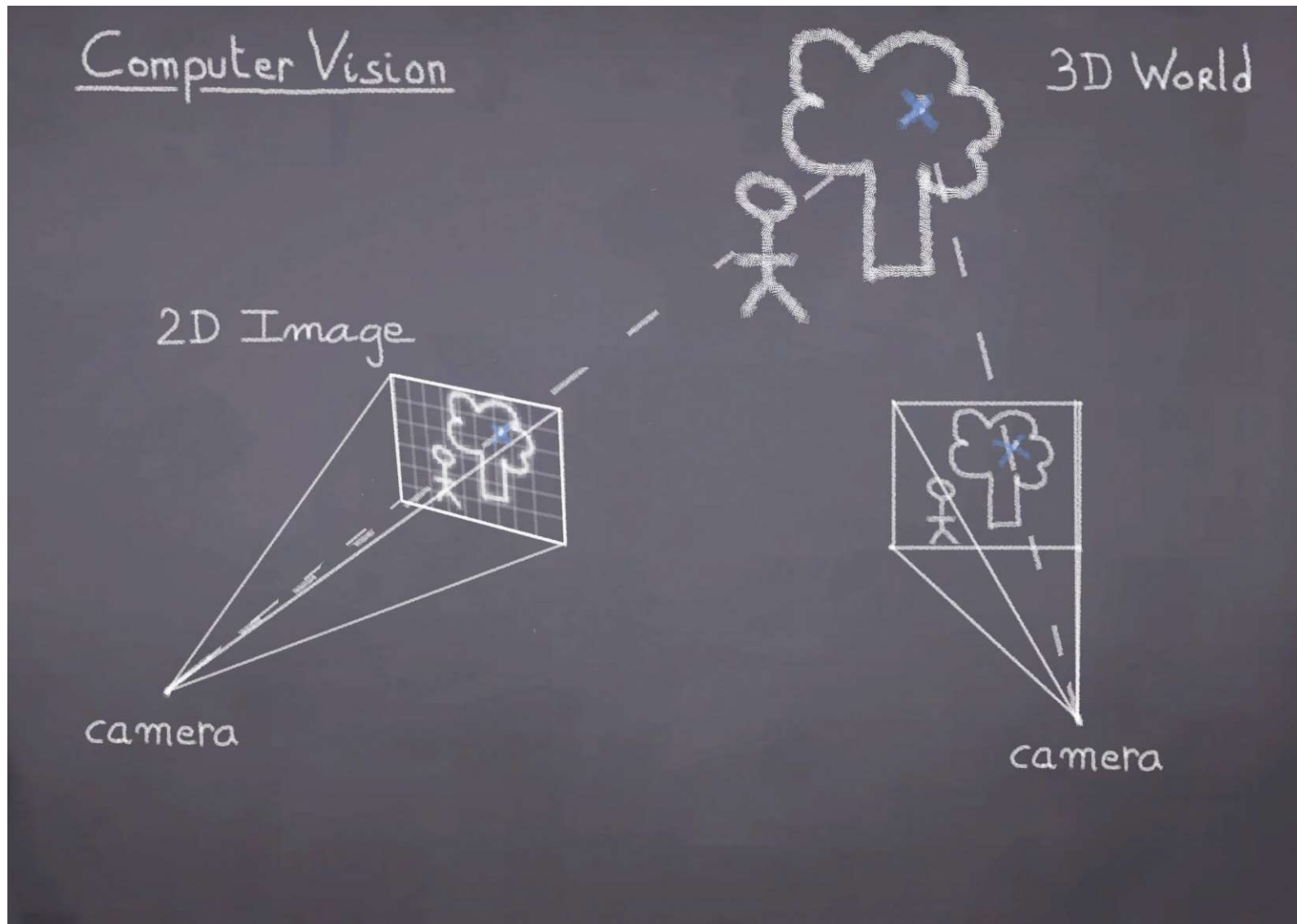
Alexandre Meyer¹

¹Equipe SAARA, laboratoire LIRIS

Master ID3D et IA



Vision par ordinateur



Vision par ordinateur et vision humaine



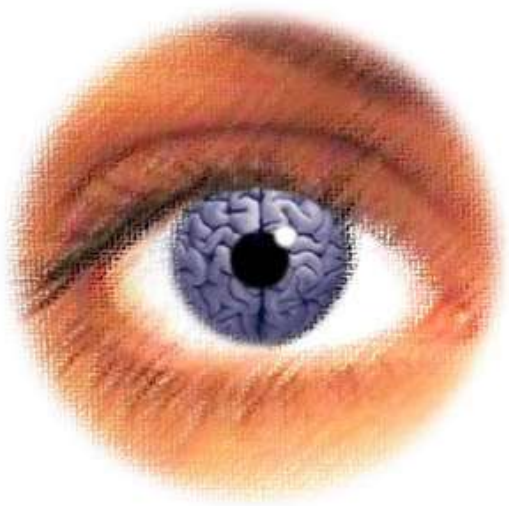
Nature

≠

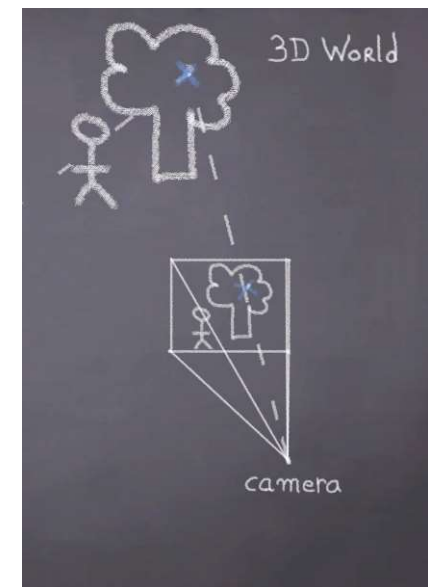
inspire l'



Artificiel



≠



Vision par ordinateur et vision humaine

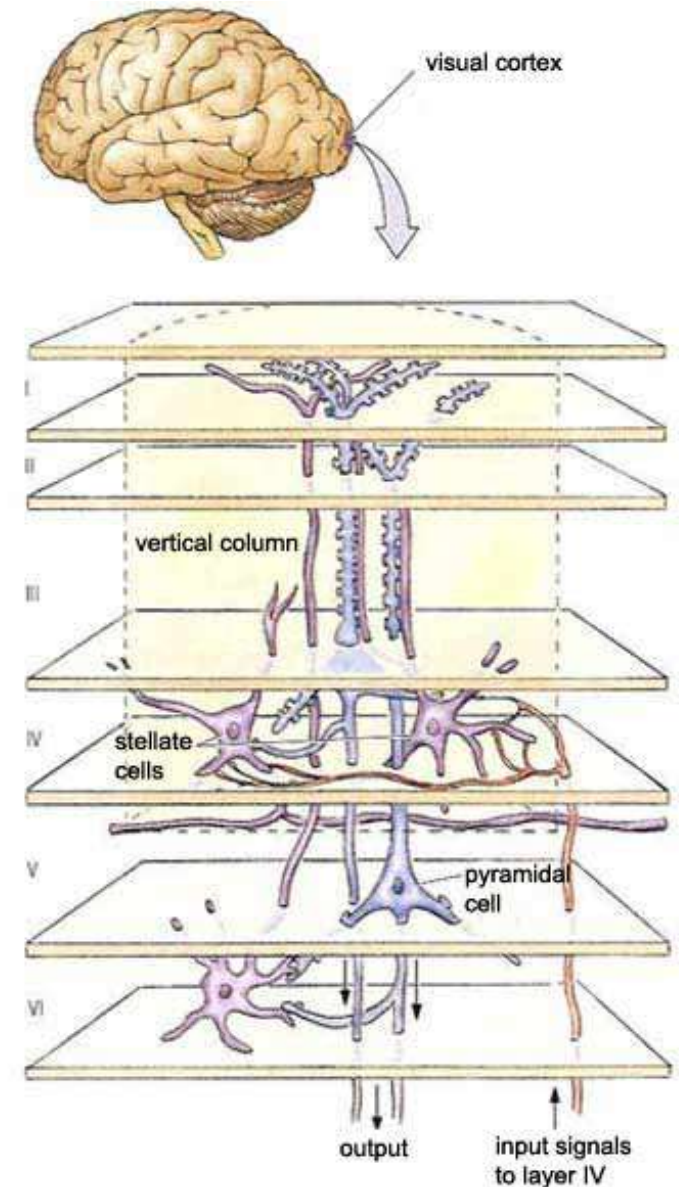
- Vision par ordinateur reste limitée même si d'énormes progrès ont été réalisés ...



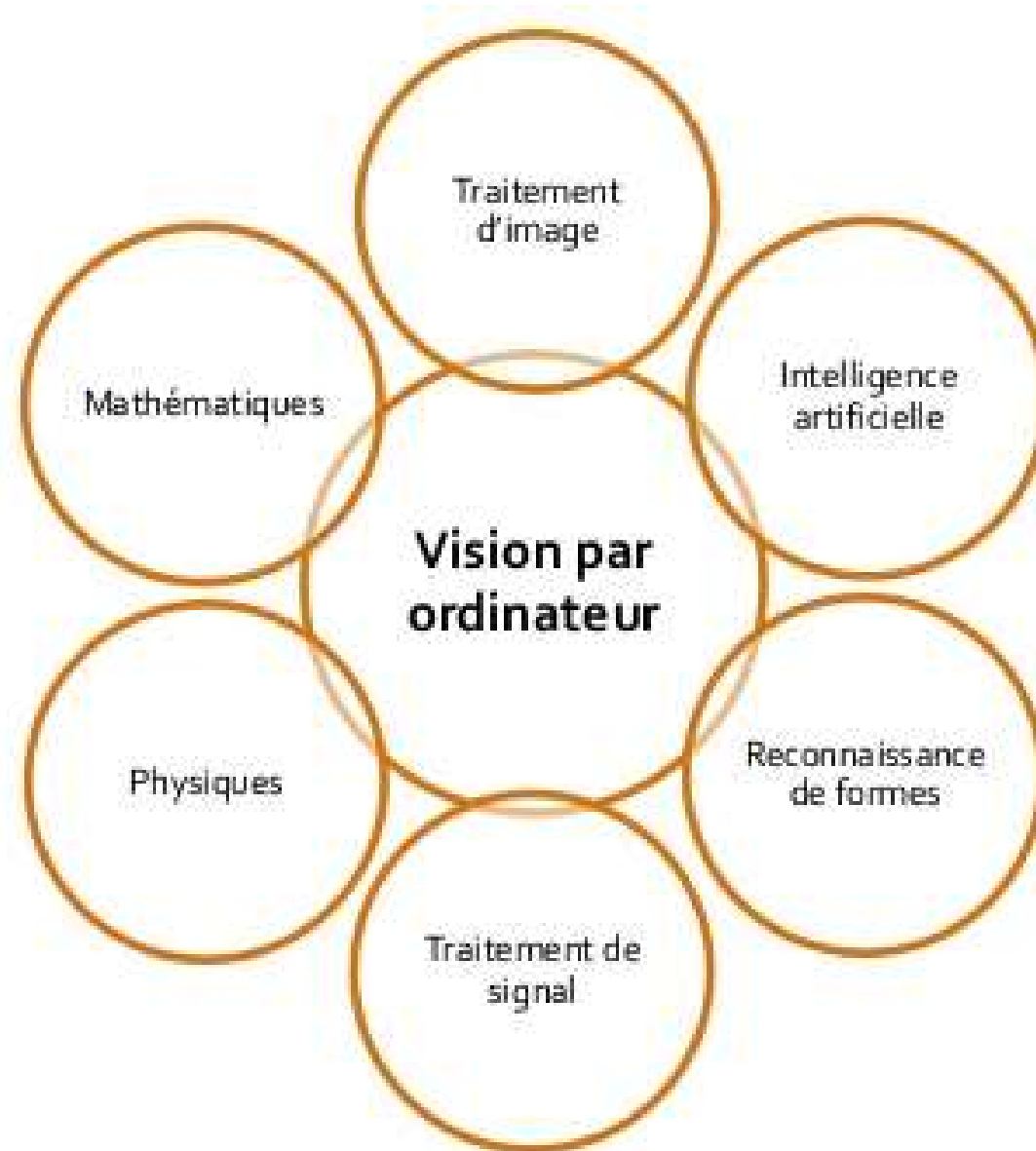
26x27 pixels

Vision par ordinateur et vision humaine

- Vision par ordinateur
 - Traitement d'images
 - Changer la luminosité
 - Mettre en évidence certains aspects
 - ...
 - Reconnaissance des formes
 - Retrouver les lignes, les cercles, etc.
 - ...
 - Retrouver des visages
- Vision par ordinateur
 - Identifier les motifs
 - « On voit un visage humain »
 - « Il s'agit de Paul »
 - Identifier des actions
 - « La personne vis un boulon »
 - Analyser une action



Vision par ordinateur

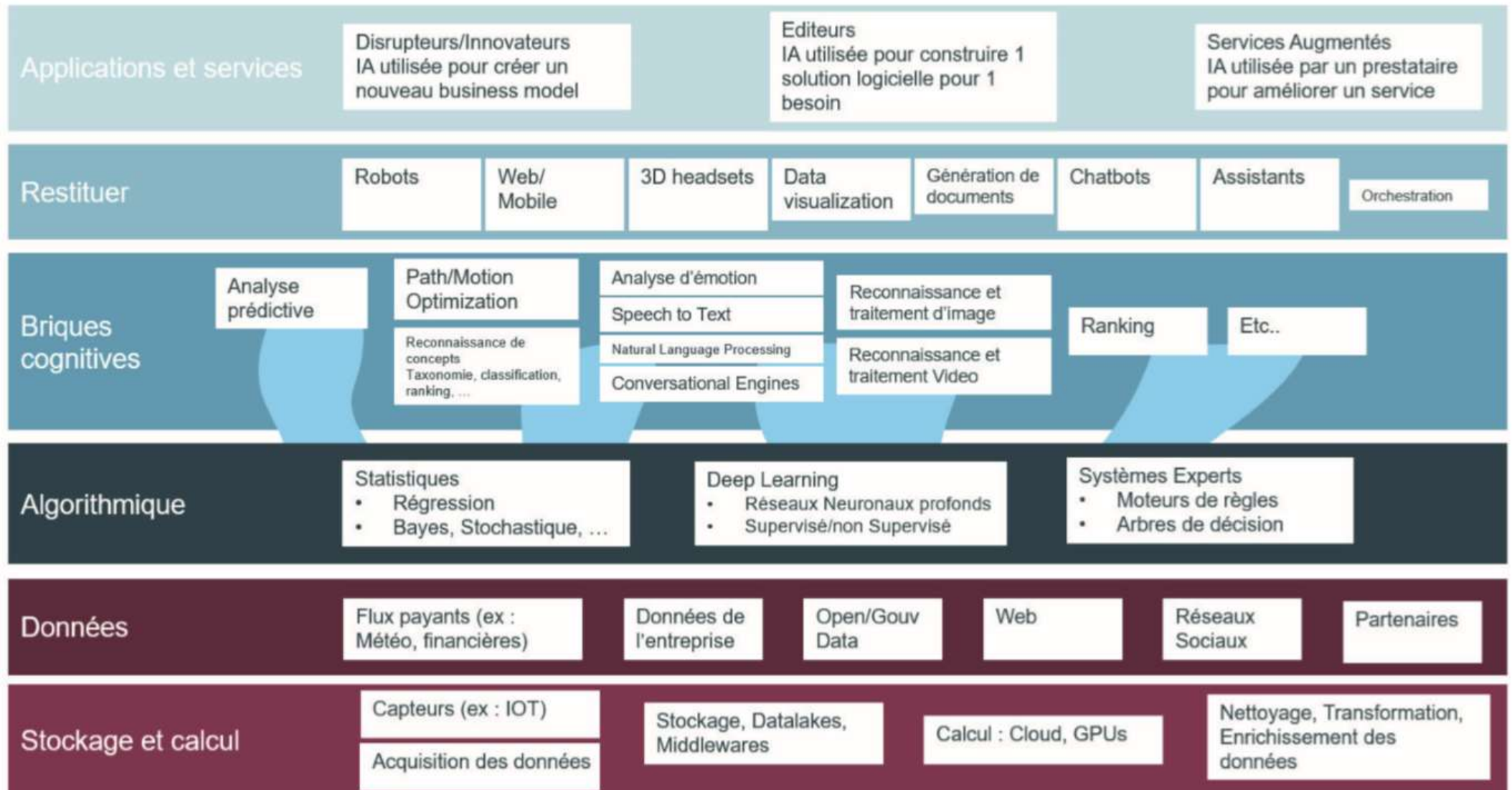


RESEAUX DE NEURONES (PROFONDS)

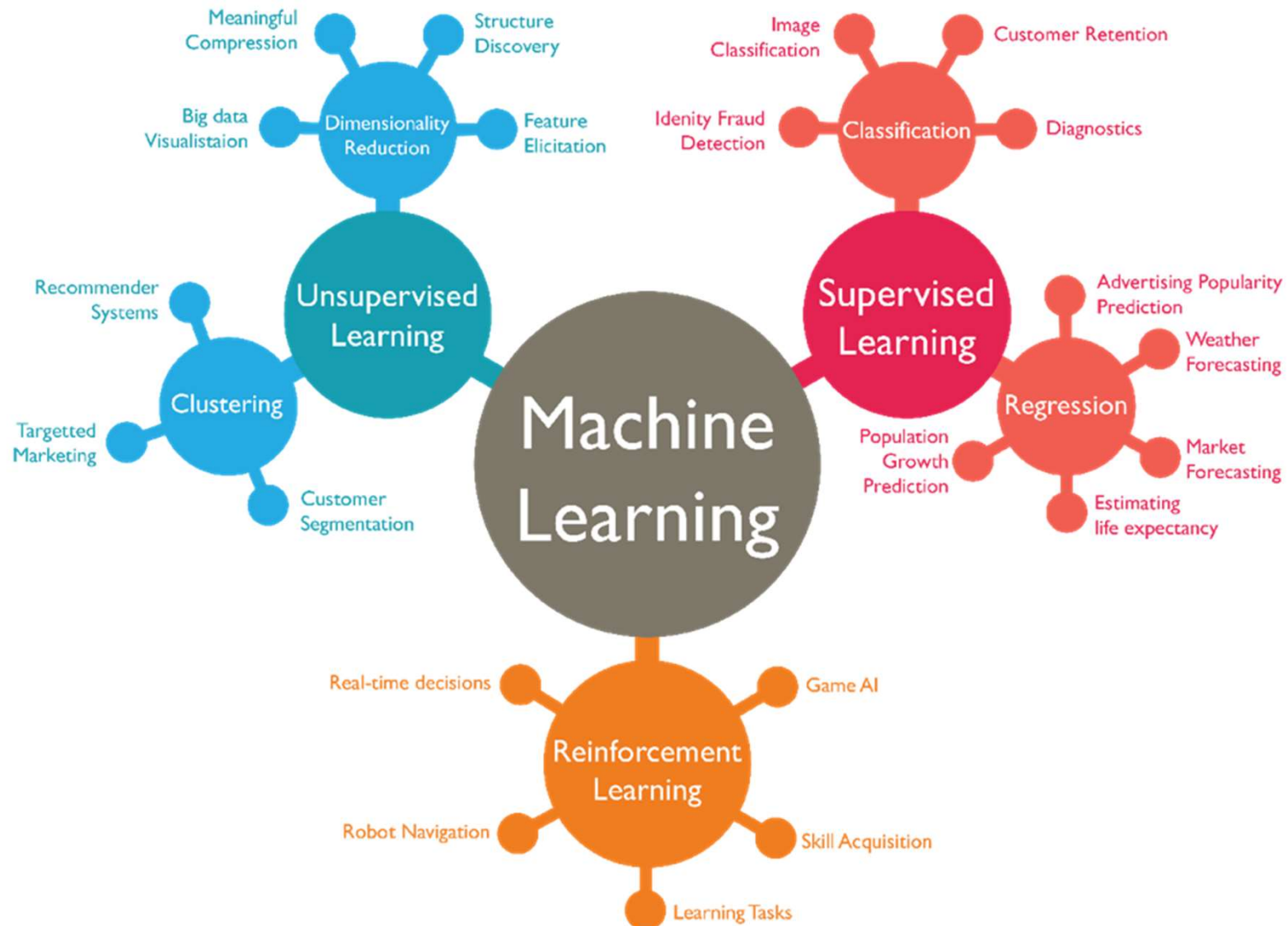
AI

- ...
- Machine Learning (apprentissage machine)
 - Random Forest
 - SVM
 - Bayésien
 - ...
 - Neural Network
 - Deep Learning (dans ce cours voir ça comme un « outils »)
 - Apprentissage par renforcement

IA / ML / DL



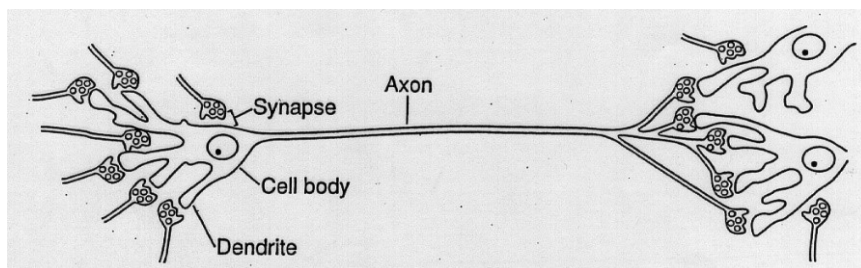
ML : Supervisé / Non supervisé



Réseau de neurones artificiel (RNA)

un modèle de calcul inspiré du cerveau humain

- Cerveau humain :
 - 10 milliards de neurones
 - 60 milliards de connexions (synapses)
 - Une synapse peut être inhibant ou excitant.
- RNA :
 - Un nombre fini de processeurs élémentaires (neurones).
 - Liens pondérés passant un signal d'un neurone vers d'autres.
 - Plusieurs signaux d'entrée par neurone

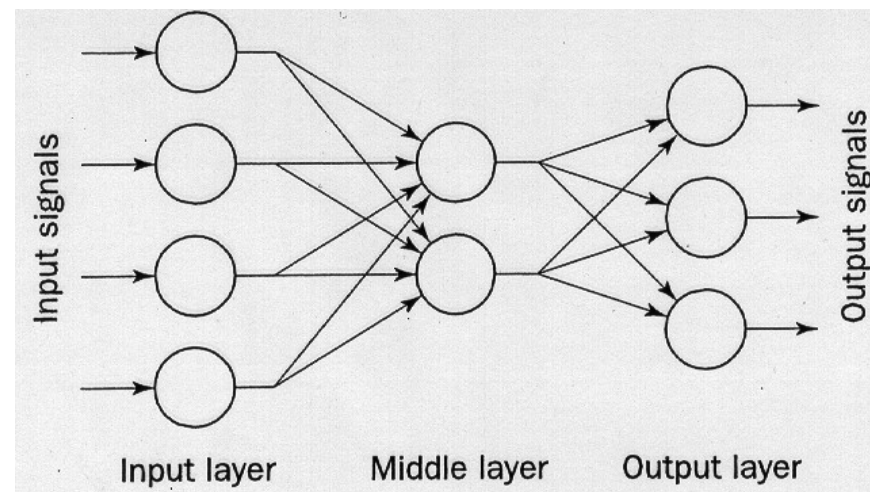


Cerveau

cellule (soma)
dendrites
synapses
axon

RNA

neurone
entrées
poids
sortie

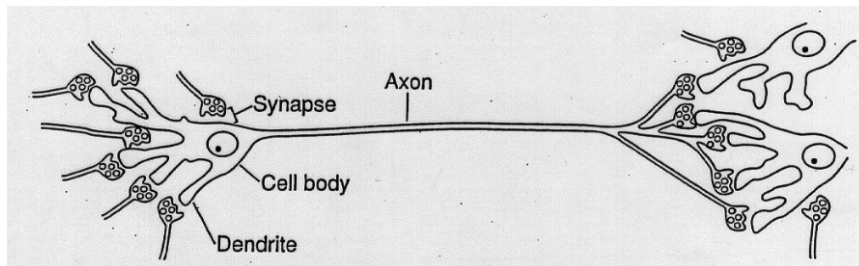


[McCulloch-Pitts, 1943]

Réseau de neurones artificiel (RNA)

un modèle de calcul inspiré du cerveau humain

- Cerveau humain :
 - 10 milliards de neurones
 - 60 milliards de connexions (synapses)
 - Une synapse peut être inhibant ou excitant.



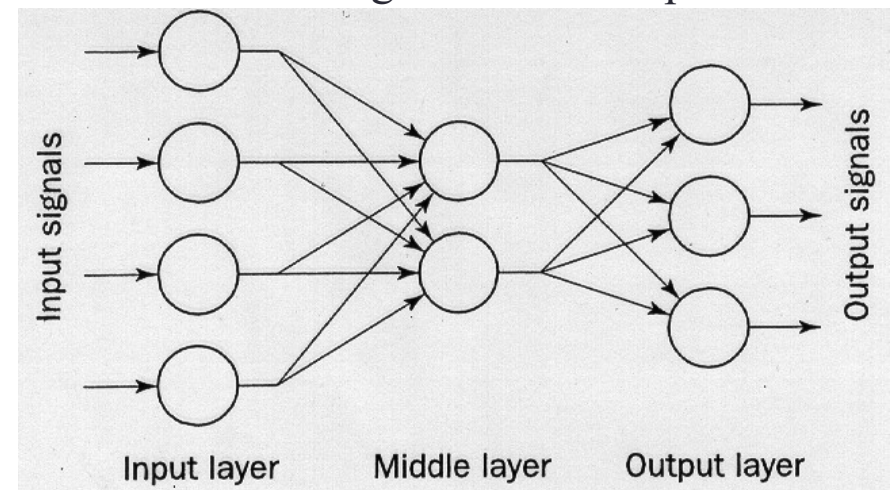
Cerveau

cellule (soma)
dendrites
synapses
axon

RNA

neurone
entrées
poids
sortie

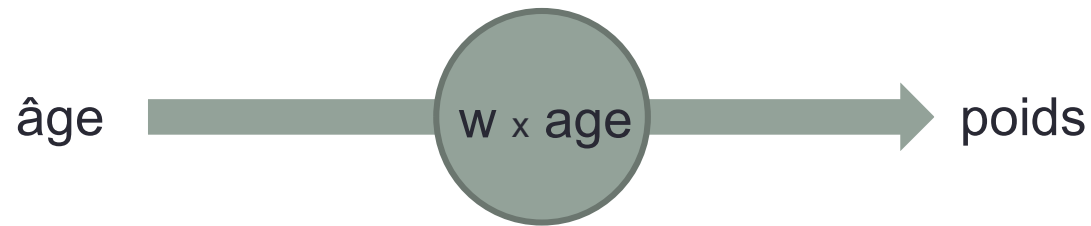
- RNA :
 - Un nombre fini de processeurs élémentaires (neurones).
 - Liens pondérés passant un signal d'un neurone vers d'autres.
 - Plusieurs signaux d'entrée par neurone



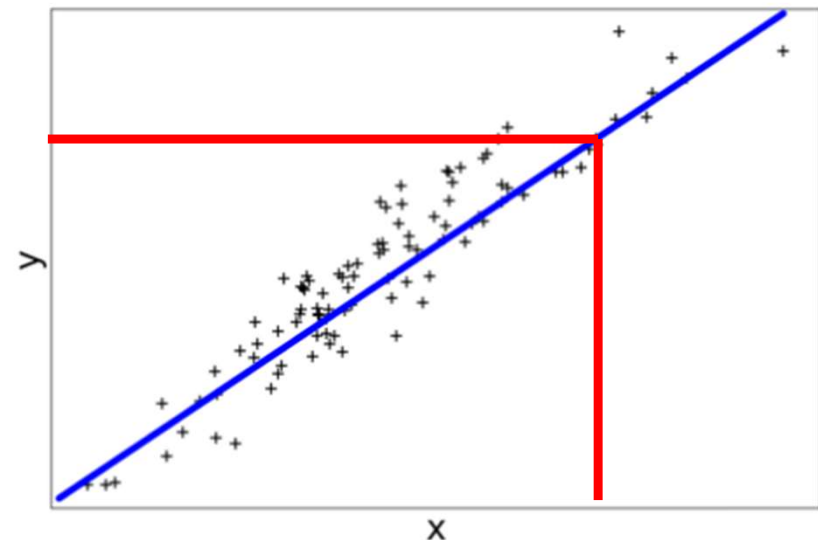
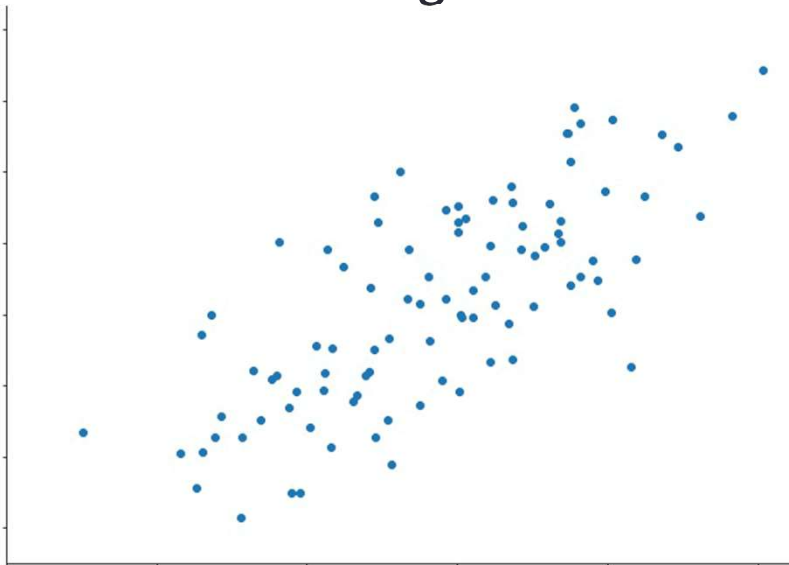
Une boîte noire qui transforme des nombres en d'autres nombres en « regardant » une base d'apprentissage. Après apprentissage on veut $f(\text{input}) = \text{output}$

Réseau de neurones artificiels

- Soit un problème simple, par exemple prédire le poids d'un enfant en fonction de son âge



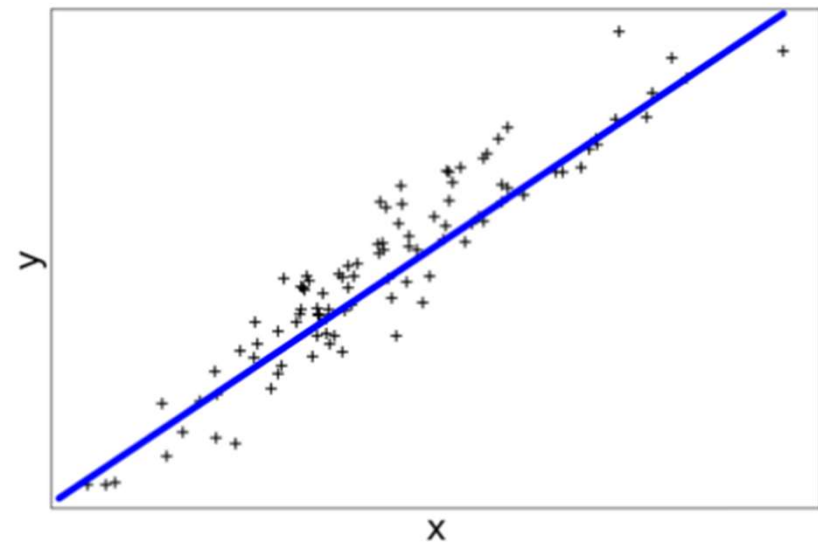
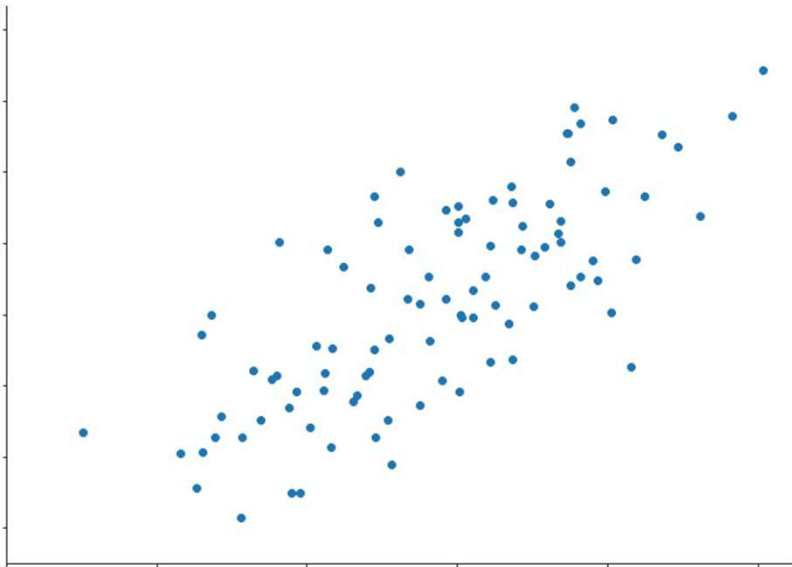
- Exploiter le neurone pour prédire
 - Pour un âge → estimation du poids



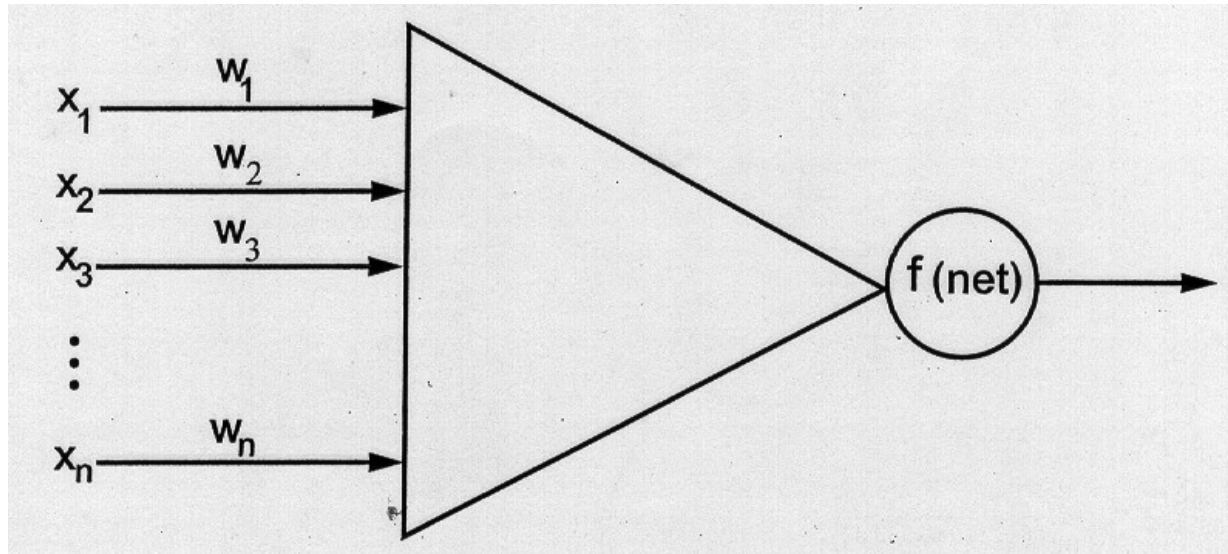
Introduction générale aux réseaux de neurones

Apprentissage consiste à trouver les poids w_{ij} par optimisation

- Base de connaissance, un jeu d'apprentissage
- une série de couple (entrée, sortie) donc de
- **Intuition : initialiser w_{ij} au hasard, puis descente de gradient (~)**



Introduction générale aux réseaux de neurones



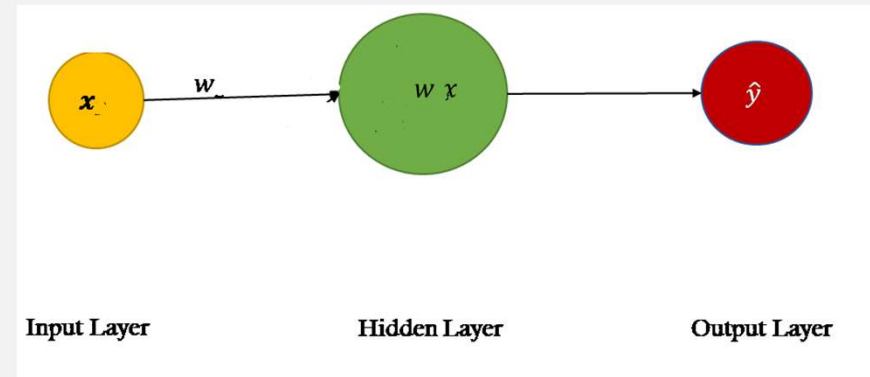
[McCulloch-Pitts, 1943]

Un type de neurone simple : $nD \rightarrow 1D$ avec $f=\text{sign}$

- $\text{net} = \sum w_i x_i$ x = données d'entrée, w =les poids
- $f(\text{net}) = +1$ si $\text{net} \geq 0$, -1 sinon ($\text{net} < 0$) f = Fonction d'activation du neurone
- C.à-d. ici : 1 neurone = $\text{sign}(\sum w_i x_i)$

Un exemple très simple

Input	Desired output
0	0
1	2
2	4
3	6
4	8



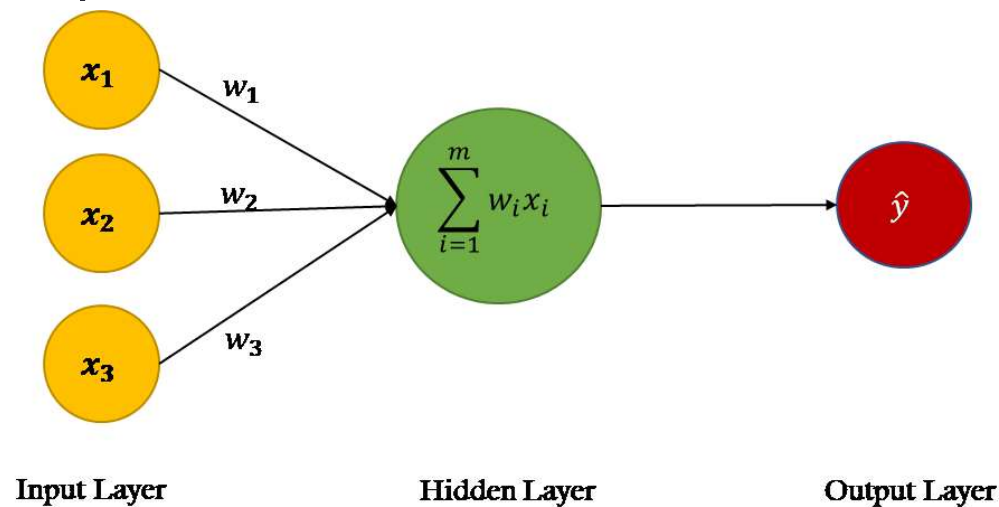
- 1 neurone ultra simple
 - sans activation
 - ni bias
 - input/output à 1D

$$w \cdot \text{input} = \text{output}$$

- Optimisation = entraînement = trouver w à partir de la base d'apprentissage (le tableau en haut à gauche)

Introduction générale aux réseaux de neurones

- Souvent un neurone à plusieurs entrées et plusieurs sorties
 - En générale problème non linéaire (mais dérivable)
 - ➔ Par exemple fonction d'activation non linéaire
 - Dimension élevée
 - Nombreux paramètres w



➔ Mais restons dans notre cas très simple

Un exemple très simple

- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$

Input	Actual output of model 1 ($y = 3 \cdot x$)
0	0
1	3
2	6
3	9
4	12

Un exemple très simple

- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$
- Fonction d'erreur : loss

Input	Actual output	Desired output
0	0	0
1	3	2
2	6	4
3	9	6
4	12	8

Un exemple très simple

- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$
- Fonction d'erreur : loss

Input	actual	Desired	Absolute Error	Square Error
0	0	0	0	0
1	3	2	1	1
2	6	4	2	4
3	9	6	3	9
4	12	8	4	16
Total:	-	-	10	30

Un exemple très simple

- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$
- Différentiation : $d\text{loss}/dw = (\text{loss}(w) - \text{loss}(w+\text{delta}))/\text{delta}$
 - Faire varier un peu $w \Rightarrow w=3.0001$
 - Calcul du gradient (dérivée partielle en cas de $\text{dim} > 1$)

Input	Output	W=3	sq.e (3)	W=3.0001	sq.e
0	0	0	0	0	0
1	2	3	1	3.0001	1.0002
2	4	6	4	6.0002	4.0008
3	6	9	9	9.0003	9.0018
4	8	12	16	12.0004	16.0032
Total:	-	-	30	-	30.006

→ Descente de gradient

$d\text{loss}/dw = (30.006-30)/0.0001 > 0 \rightarrow$ Augmenter w augmente l'erreur

→ Avancer dans le sens opposé au gradient $w_{\text{new}} = w - \text{lambda.gradient}$

Un exemple très simple

On verra plus tard
Learning rate

- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$
- Différentiation : $d\text{loss}/dw = (\text{loss}(w) - \text{loss}(w+\text{delta}))/\text{delta}$
 - Faire varier un peu $w \Rightarrow w=3.0001$
 - Calcul du gradient (dérivée partielle en cas de $\text{dim} > 1$)

Input	Output	W=3	sq.e (3)	W=3.0001	sq.e
0	0	0	0	0	0
1	2	3	1	3.0001	1.0002
2	4	6	4	6.0002	4.0008
3	6	9	9	9.0003	9.0018
4	8	12	16	12.0004	16.0032
Total:	-	-	30	-	30.006

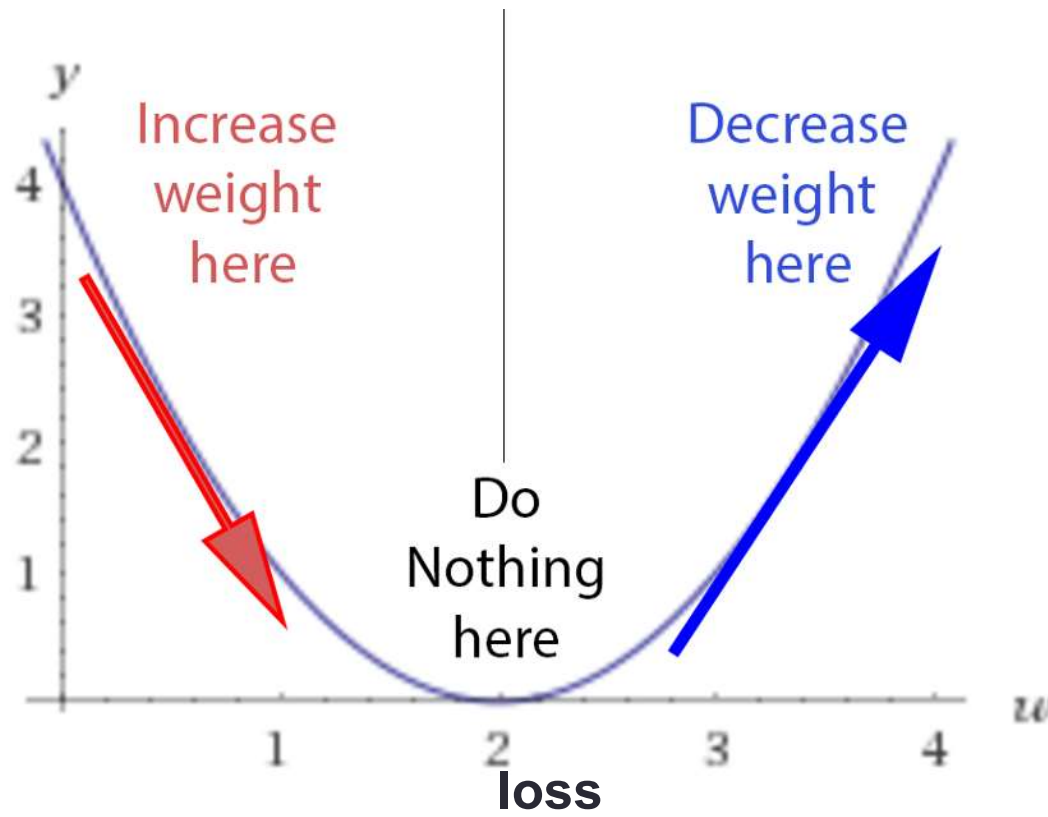
→ Descente de gradient

$d\text{loss}/dw = (30.006-30)/0.0001 > 0 \rightarrow$ Augmenter w augmente l'erreur

→ Avancer dans le sens opposé au gradient $w_{\text{new}} = w - \text{lambda.gradient}$

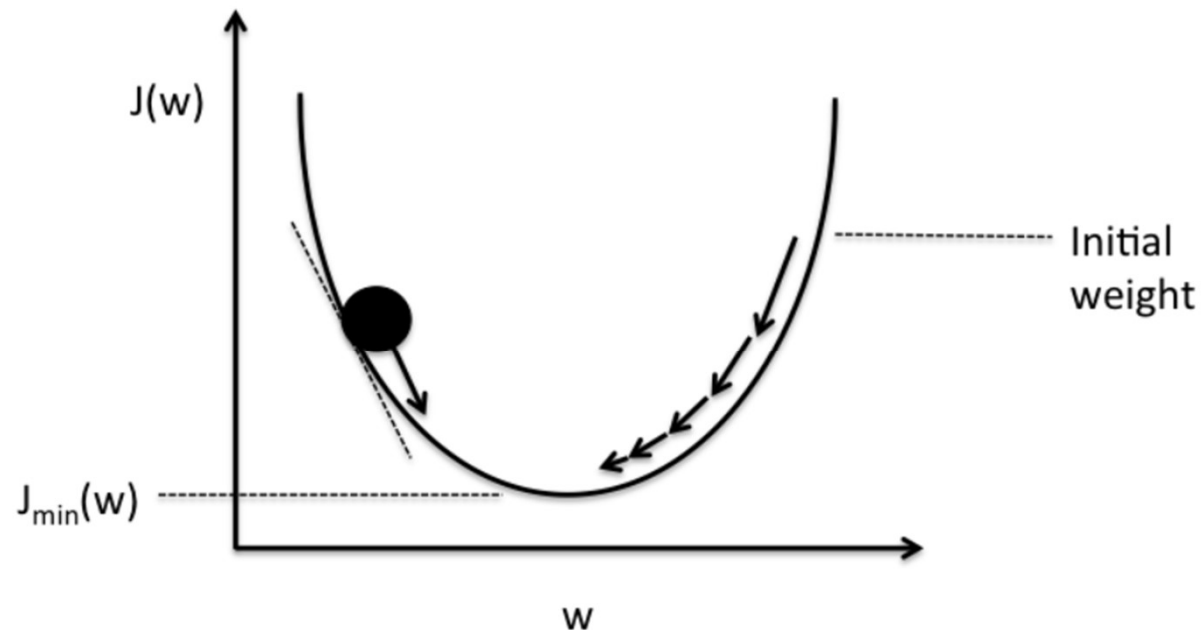
Un exemple très simple

- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$
- Différentiation : $df/dw = (f(w) - f(w+\text{delta}))/\text{delta}$
 - Faire varier un peu $w \Rightarrow w=3.0001$



Un exemple très simple

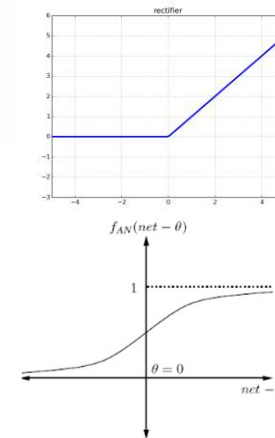
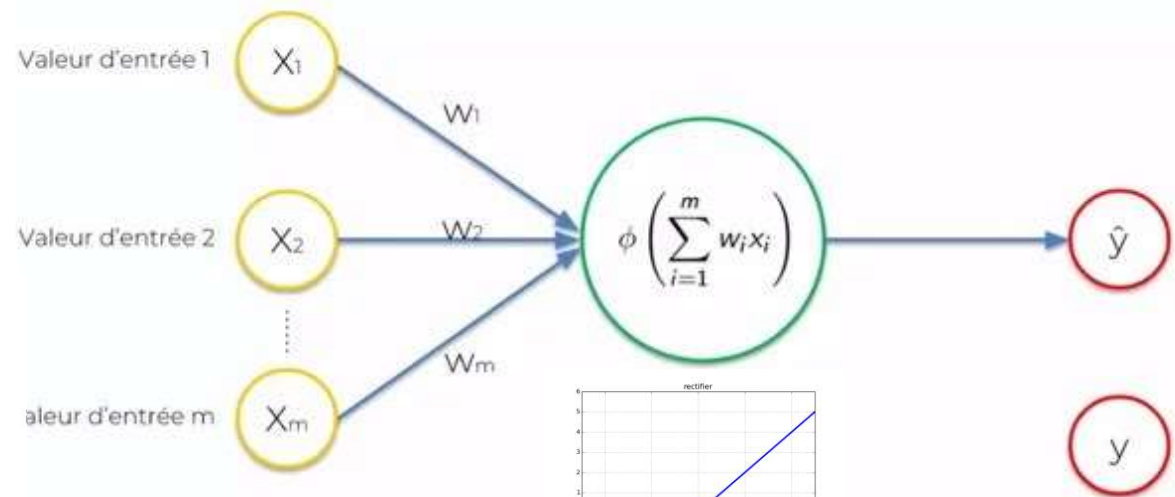
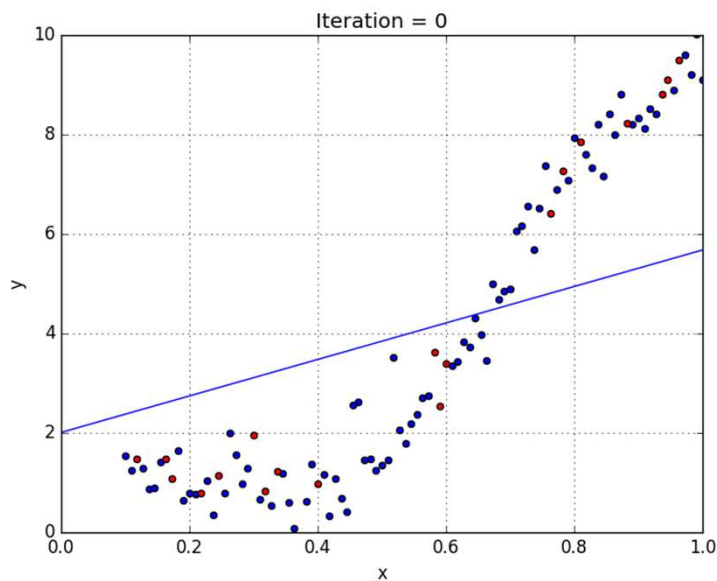
- Initialisation au hasard de w à 3
 $w \cdot \text{input} = \text{output}$
- Différentiation : $df/dw = (f(w) - f(w+\text{delta}))/\text{delta}$
 - Faire varier un peu $w \Rightarrow w=3.0001$



Schematic of gradient descent.

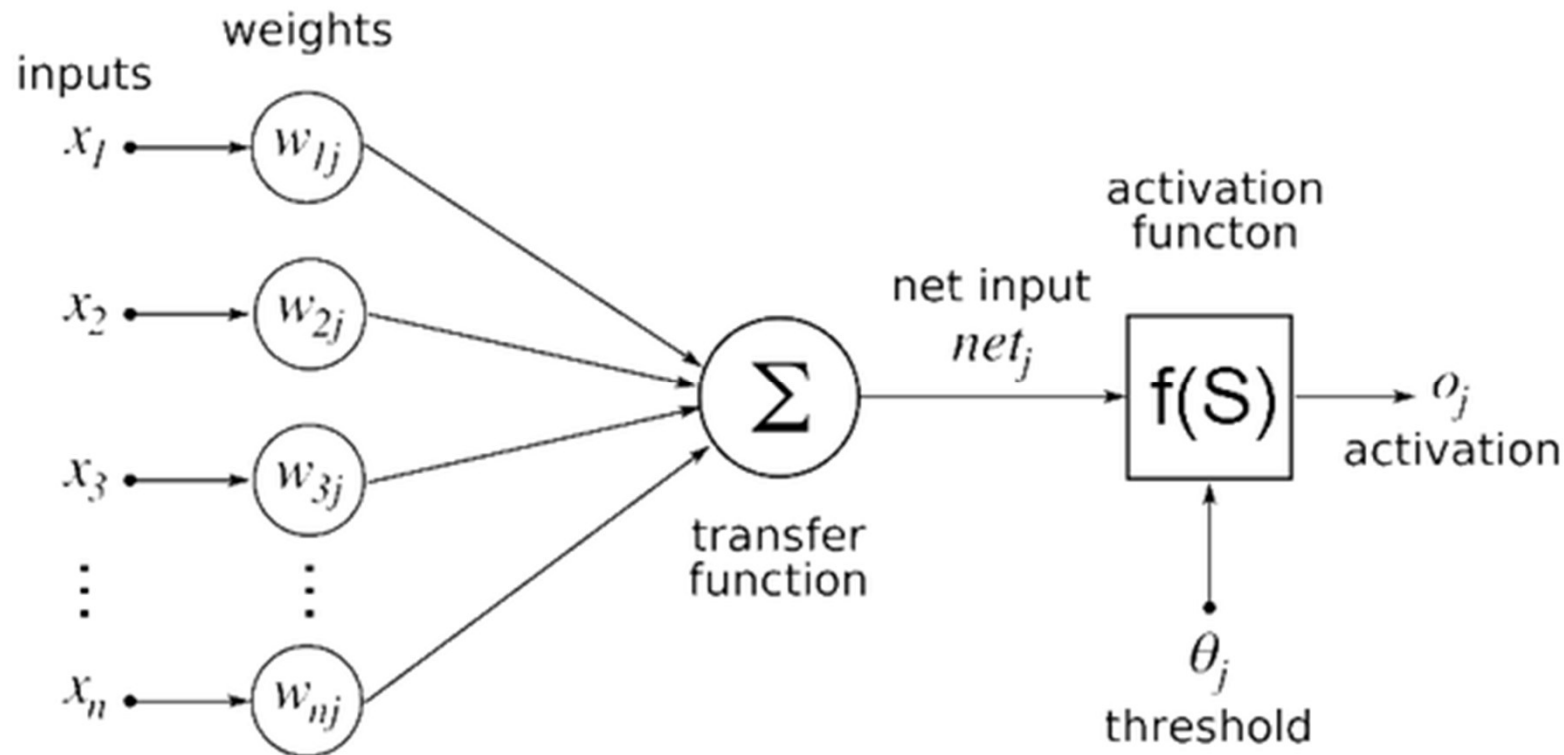
Réseau de neurones artificiels

- Seulement souvent le problème est moins simple
 - Moins linéaire, plus de dimensions
 - Ajout d'une fonction d'activation (non linéaire)
 - Et de plusieurs neurones



(d) Sigmoid function

Introduction générale aux réseaux de neurones



Apprentissage consiste à trouver les poids w_{ij} par optimisation

La fonction d'activation (non linéaire) permet d'approximer un problème/une fonction générale non linéaire. Rappel : le problème ou la fonction sont représentée par des données discrètes.

Introduction générale aux réseaux de neurones

Pour chaque neurone

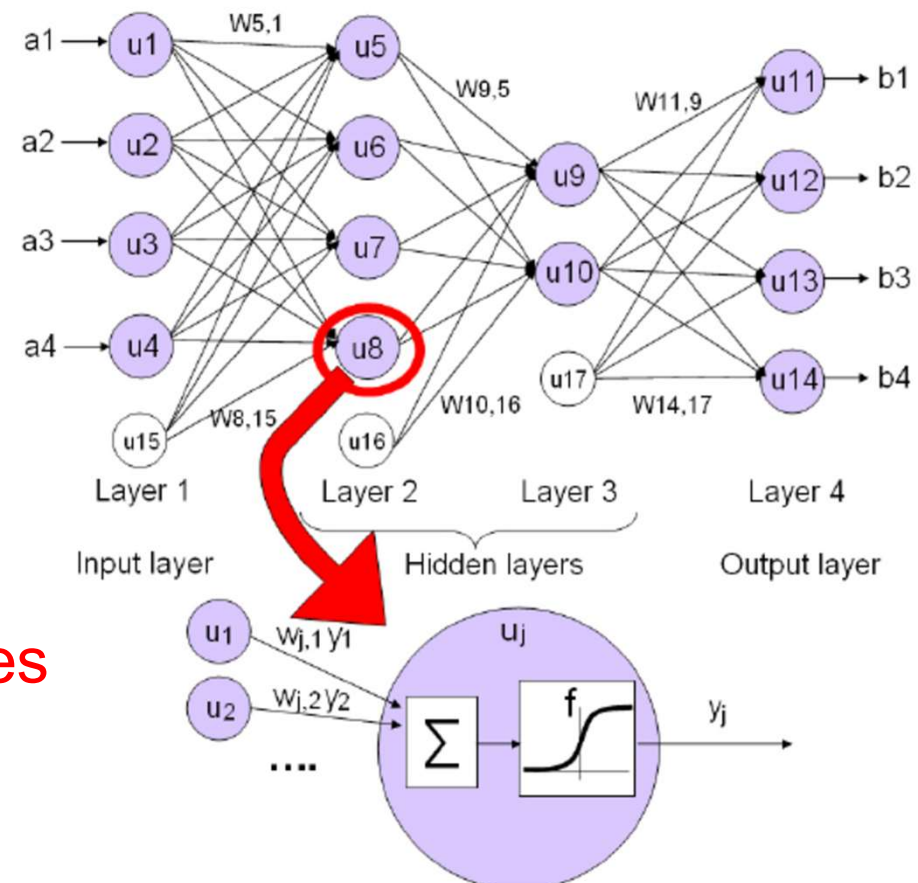
- n Dimensions en entrée → m Dimensions en sortie
- Les poids w_{ij} forment une matrice A

- $\text{sortie} = A \cdot \text{entrée} + B$

- $y_n = A \cdot x_m + B$

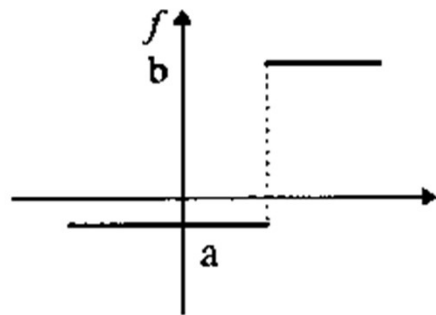
- La fonction d'erreur « loss » est calculée de manière globale (à la sortie output)

- Optimisation de tous les neurones en même temps

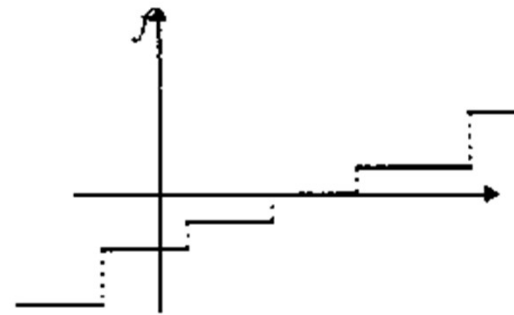


Introduction générale aux réseaux de neurones

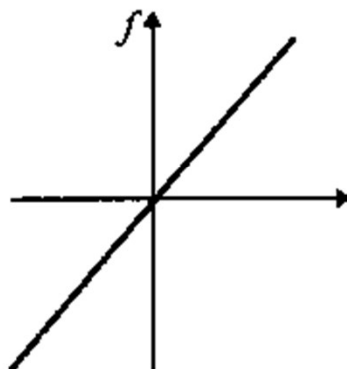
Les fonctions d'activation classique (possibilité d'avoir une fonction différente pour chaque neurone)



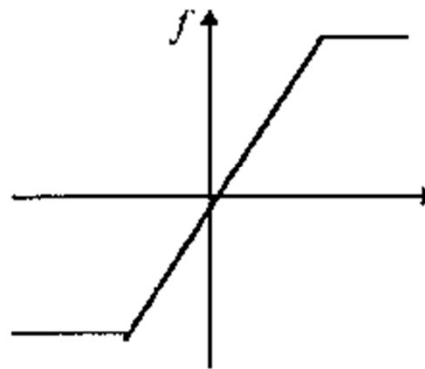
Fonction à seuil



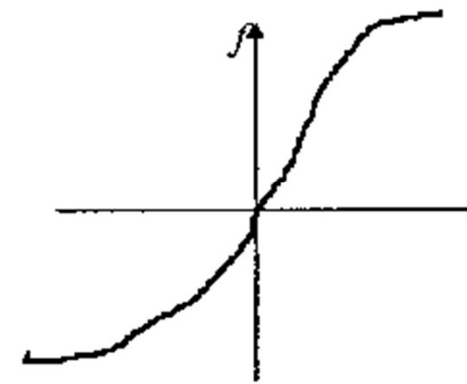
Fonction à valeurs discrètes



Linéaire



Saturation



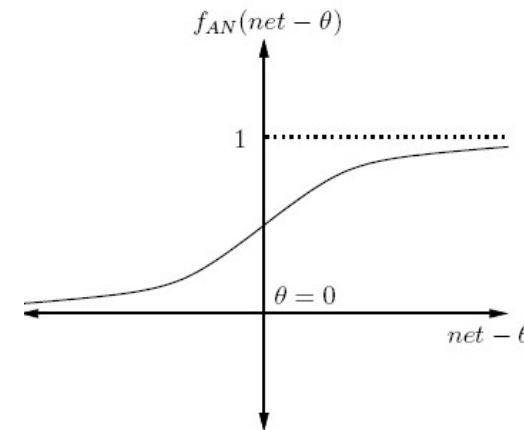
Sigmoïde

Introduction générale aux réseaux de neurones

La fonction sigmoïde

$$f_{AN}(\tilde{net}) \in (0,1)$$

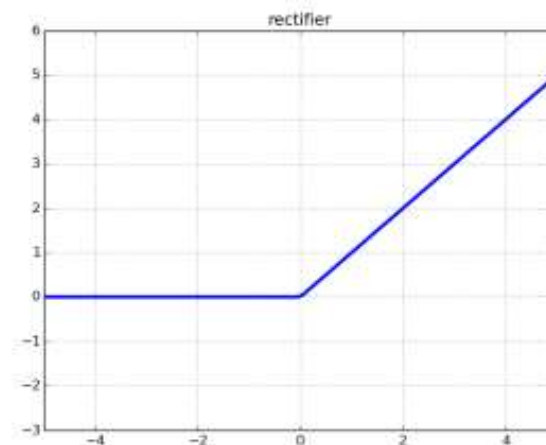
$$f_{AN}(net - \theta) = \frac{1}{1 + e^{-\lambda(net - \theta)}}$$



(d) Sigmoid function

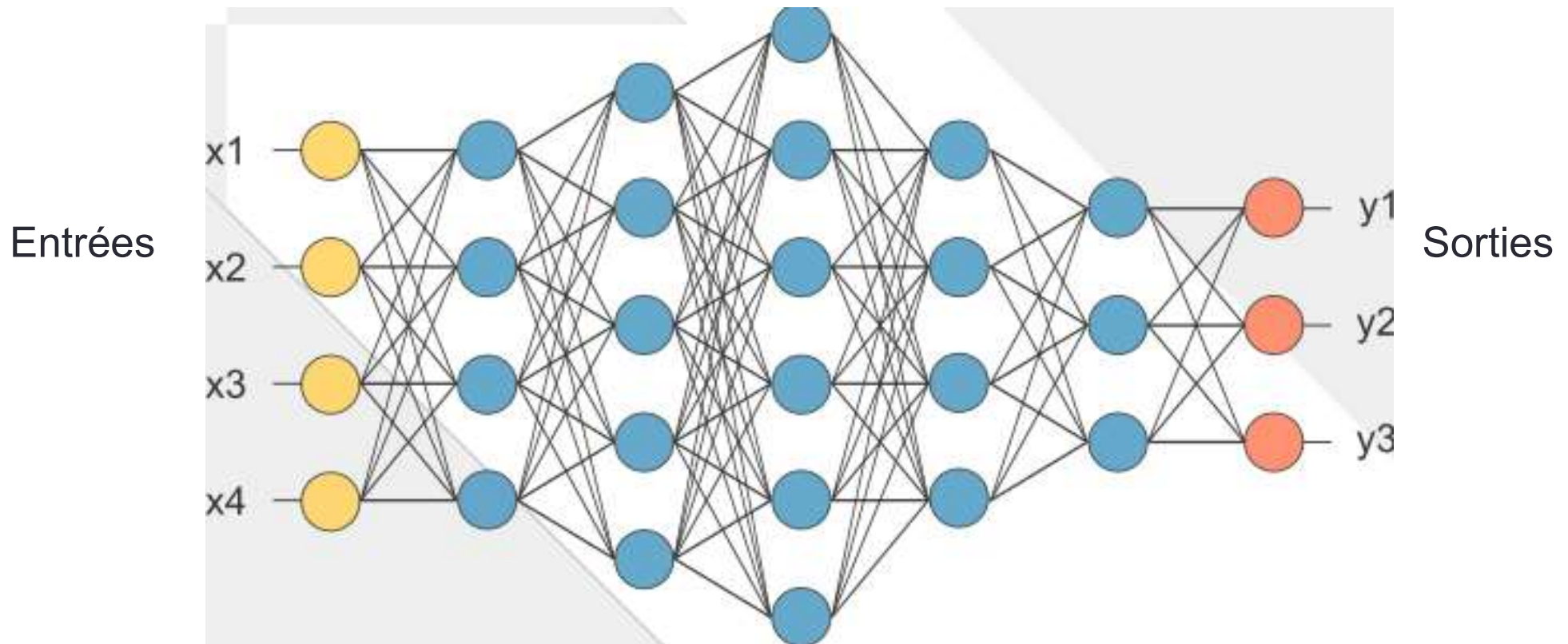
RELU (Rectified Linear)

if $x < 0$ then return 0
else return x



Réseau de neurones profonds

- Chaque cercle est un neurone $f(A.X+B)$
 - En bleu les couches cachées



La fonction d'erreur (LOSS)

- Entrainement ou apprentissage
= une optimisation pour trouver les bons poids W qui maximisent les résultats sur une base de connaissance (input X , output Y)
- La fonction d'erreur que l'on cherche à minimiser
 - Plusieurs types possibles suivant le problème

$$L1LossFunction = \sum_{i=1}^n |y_{true} - y_{predicted}|$$

$$L2LossFunction = \sum_{i=1}^n (y_{true} - y_{predicted})^2$$

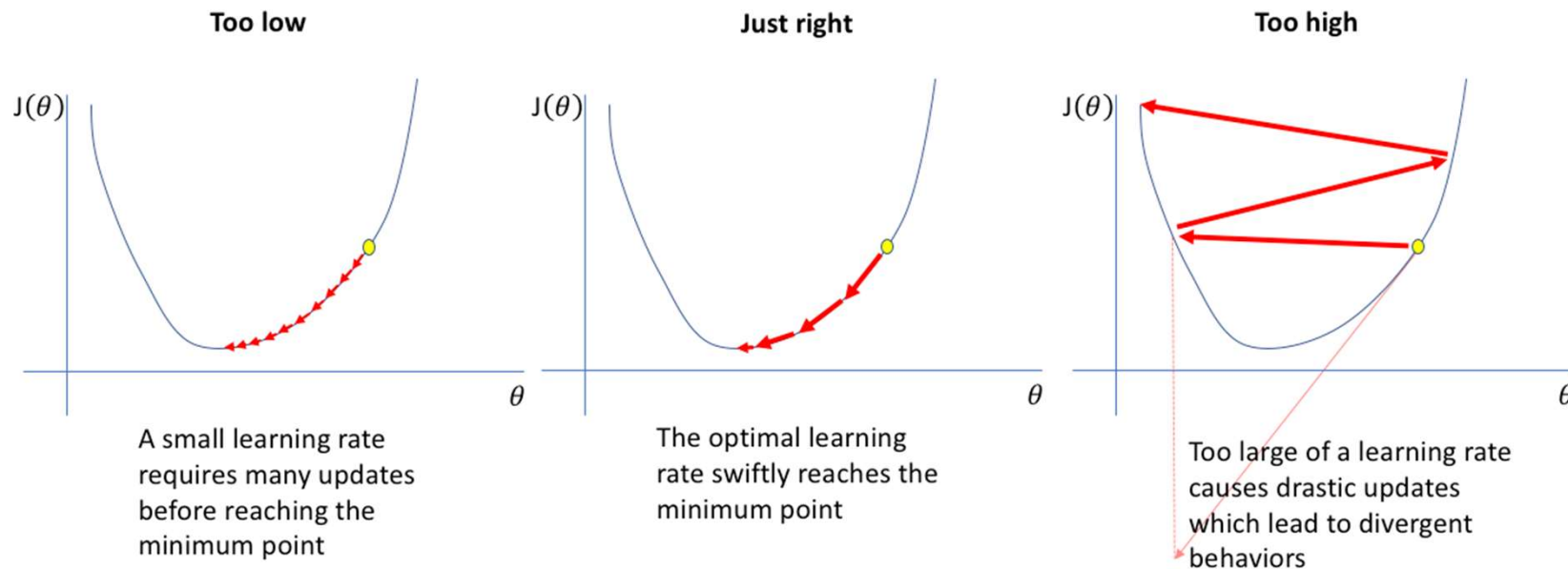
- Généralement (aberrantes) dans vos données la L1 peut être meilleur
- La fonction d'erreur peut comporter d'autres termes (somme)
- Même pour de la reconnaissance, de nombreuses autres fonctions d'erreur sont possibles : crossEntropy, etc.

Entrainement : backpropagation

- Dans un réseau le calcul « classique » du gradient peut être long, très long à calculer ! Avec N paramètres à optimiser (poids), le calcul du gradient demande N passes
 - Il faut calculer les N dérivées partielles
- Si la base de données d'apprentissage comporte M millions d'exemples, il faut passer M millions de fois dans le réseau (forward) pour avoir le résultat prédit.
- Donc $N \times M$ passage dans le réseau !
 - ➔ Des améliorations
 - SGD (Stochastic Gradient Descent)
 - Back propagation

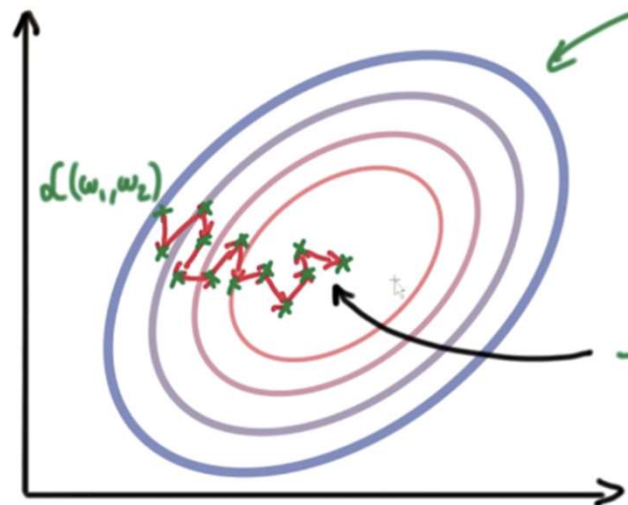
Réseau de neurones : entraînement

- Stochastic Gradient Descent (SGD)
 - **SGD est une Descente de gradient**
 - Learning rate = progression ou pas d'apprentissage
 - Il faut bien le choisir
 - Trop grand peut entrainer une dérive
 - Trop petit demandera plus de calcul et donc de temps



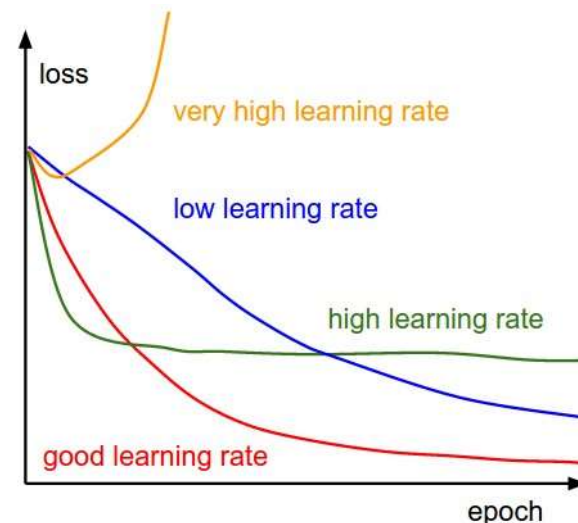
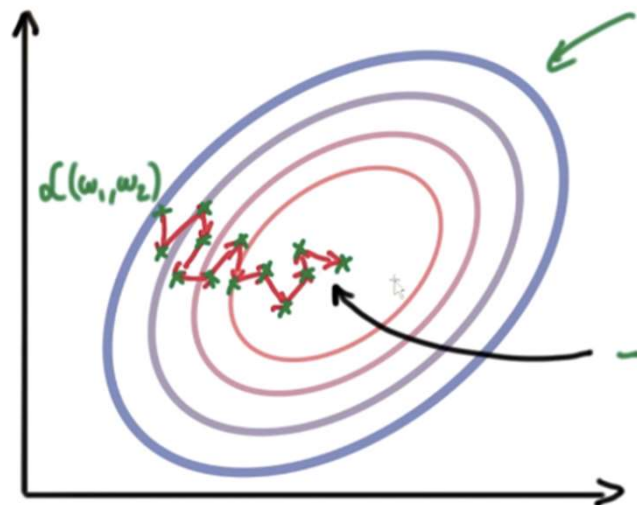
Réseau de neurones : entraînement

- Stochastic Gradient Descent (SGD)
 - Stochastic car la descente de gradient est calculée sur un sous-échantillonnage des données (epoch)
 - que M' exemples sont échantillonnés
 - Par exemple $M'=64$
 - ATTENTION : ces échantillons peuvent être très biaisés. La descente se fait alors en louvoyant.
 - Moment = comme stochastic le changement de direction se fait avec une inertie
 - Le gradient ne changera que lentement à chaque itération



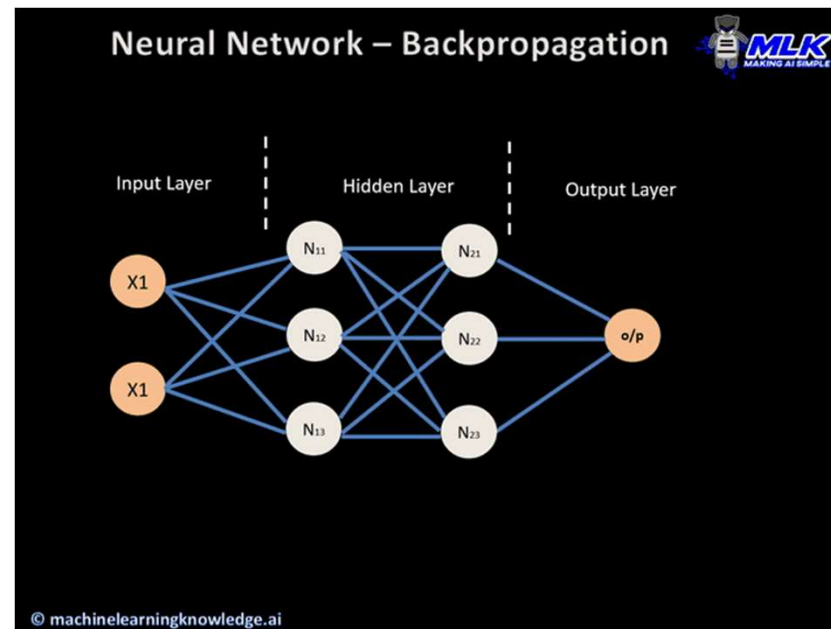
Réseau de neurones : entraînement

- Stochastic Gradient Descent (SGD)
 - Stochastic car la descente de gradient est calculée sur un sous-échantillonnage des données (epoch)
 - Learning rate = progression ou pas d'apprentissage
 - Moment = comme stochastic le changement de direction se fait avec une inertie
- Besoin des GPU pour la puissance de calcul
 - Par exemple, ImageNET 15 millions d'images avec 20 000 labels

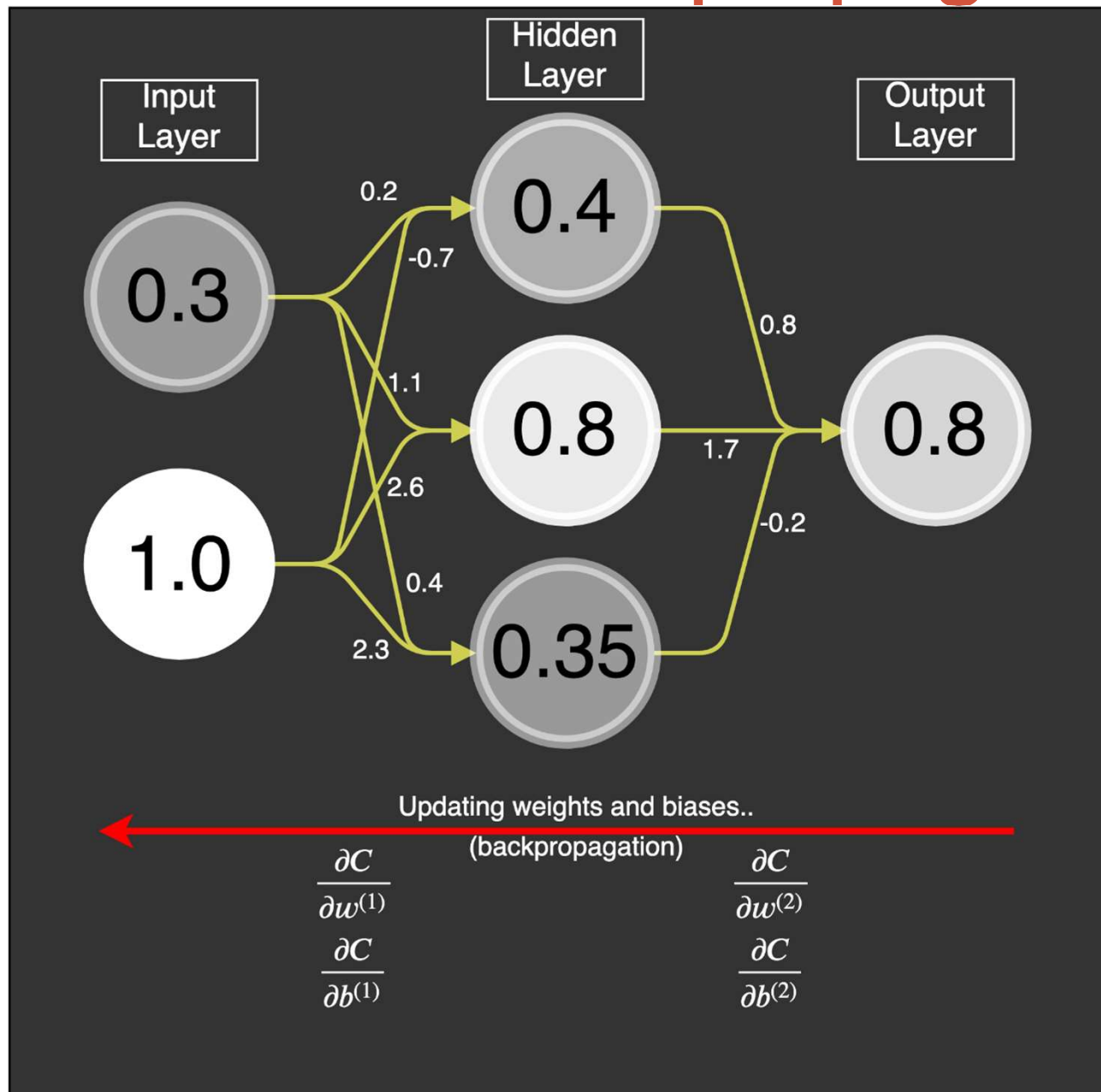


Entraînement : backpropagation

- Dans un réseau le calcul du gradient global est long !
 - On fait le calcul par neurone et la Back propagation permet de faire le calcul en remontant : en partant de l'erreur (la droite) et en remontant
 - Au lieu de calculer l'influence que va avoir une variation de w sur l'erreur, calcul de l'influence que va avoir la variation de l'erreur sur les poids
 - Si la fonction inverse de chaque nœud est différentiable
- ➔ voir le cours de Mathieu Lefort (Master IA) ...



Entrainement : backpropagation



Réseau de neurones : du code

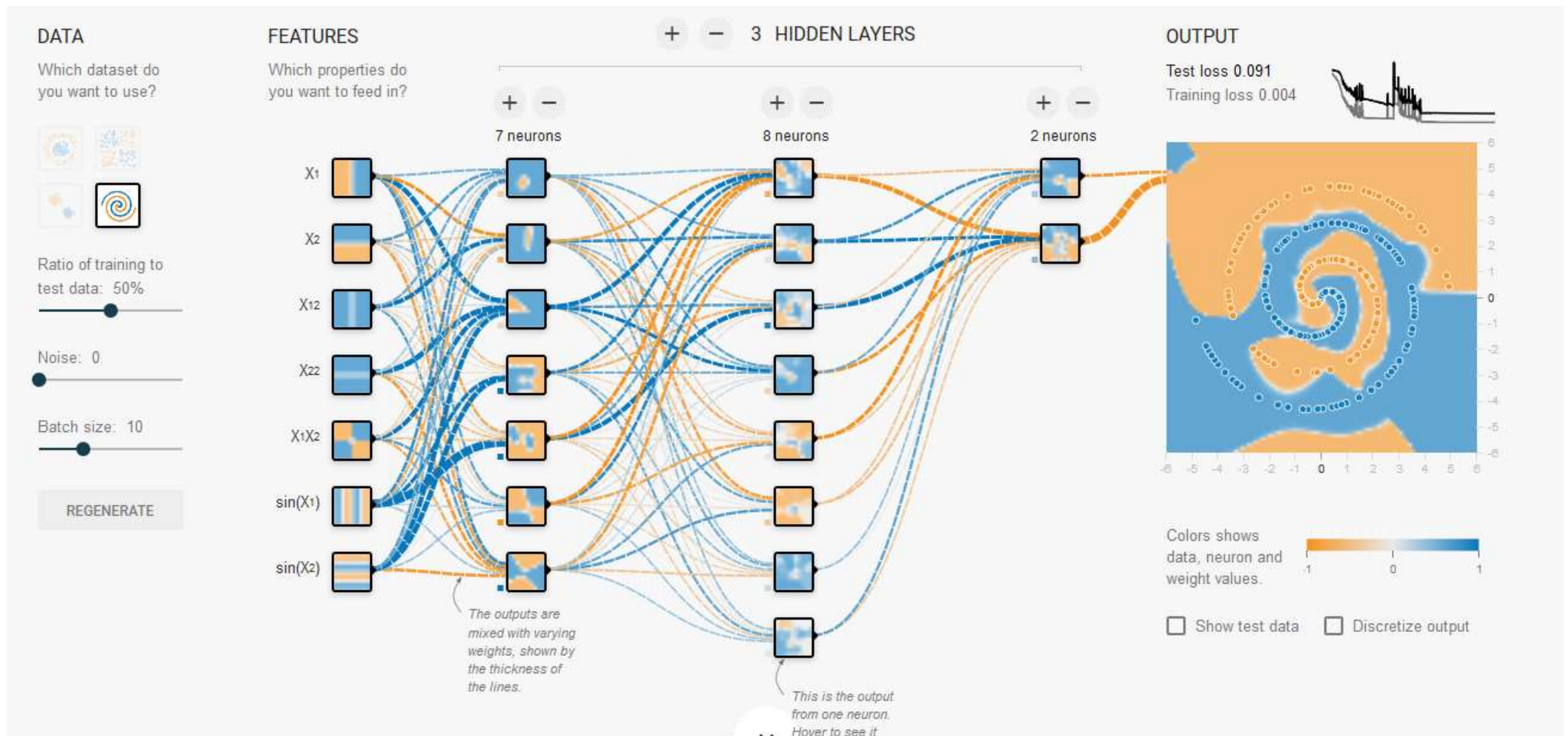
- Plateformes en python
 - TensorFlow (Google)
 - PyTorch (Yann Lecun, maintenant Facebook)
 - CNTK Microsoft Cognitive Toolkit
 - ...
- Accélération GPU
- Des exemples de code
 - Keras = une sur-couche de TensorFlow 2 facilitant le codage des réseaux
 - <https://keras.io/examples/>

➔ Une communauté immense, des exemples de code (github, etc.) par millions, reproductibilité du code, etc.

Réseau de neurones

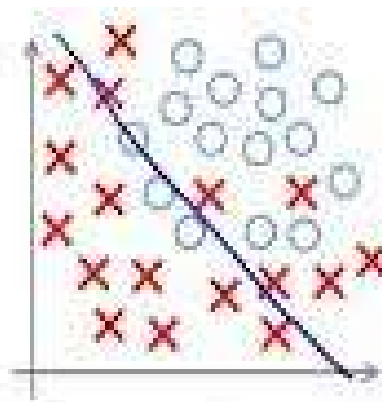
- Exemple de classification d'un nuage de points

<http://playground.tensorflow.org>



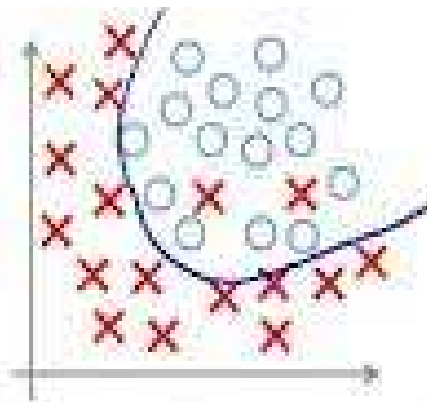
Surapprentissage / Overfitting

- Surapprentissage / Overfitting : c'est quoi ?
 - Quand un modèle apprend trop de détail/bruit sur les données avec pour conséquence d'être mauvais sur la généralisation (ie. Quand il verra des nouvelles données jamais vu avant)
 - Comme un étudiant qui « apprend par cœur » sans « comprendre »

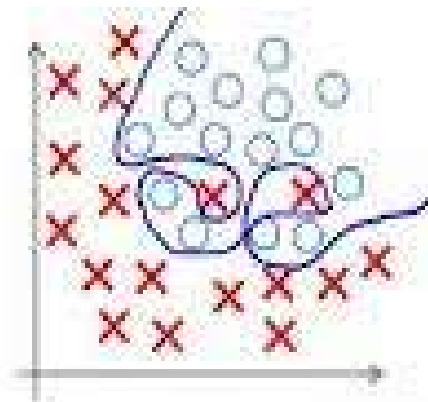


Under-fitting

(too simple to
explain the
variance)



Appropriate-fitting



Over-fitting

(forcefitting – too
good to be true)

Régularisation

Les techniques de **Regularisation** sont les techniques utilisées pour résoudre le surapprentissage (overfitting) en apprentissage machine

Par exemple

- L1 Regularization or Lasso Regularization

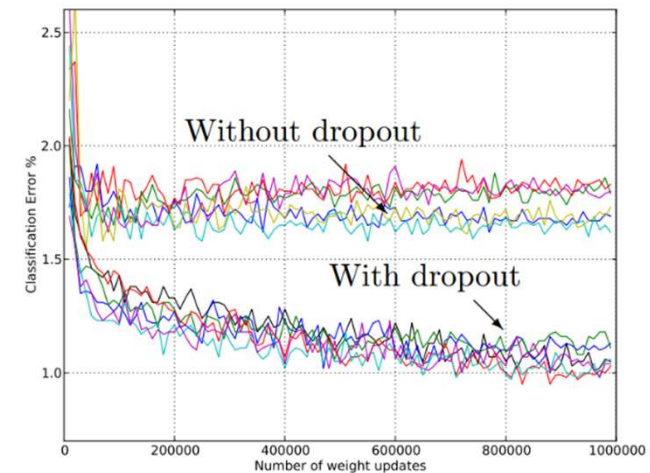
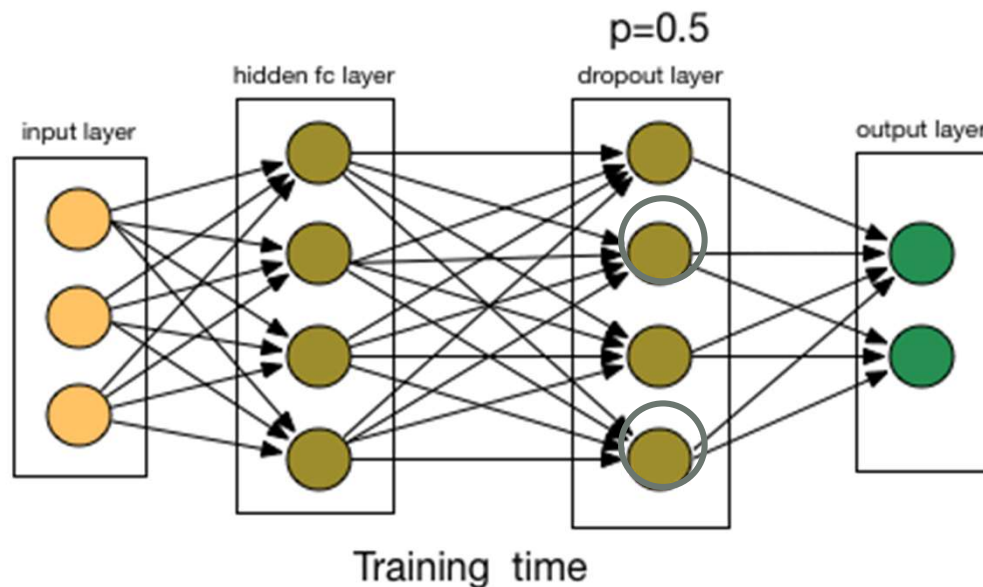
$$\text{Min}(\sum_{i=1}^n (y_i - w_i x_i)^2 + p \sum_{i=1}^n |w_i|)$$

- L2 Regularization or Ridge Regularization

$$\text{Min}(\sum_{i=1}^n (y_i - w_i x_i)^2 + p \sum_{i=1}^n (w_i)^2)$$

Drop Out

- Dropout aide pour éviter le surapprentissage (overfitting)
 - force le réseau à apprendre de manière plus robuste
 - technique de [régularisation](#)
- Dropout annule temporairement aléatoirement certain neurone avec une proba p



Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov

JMLR 2014

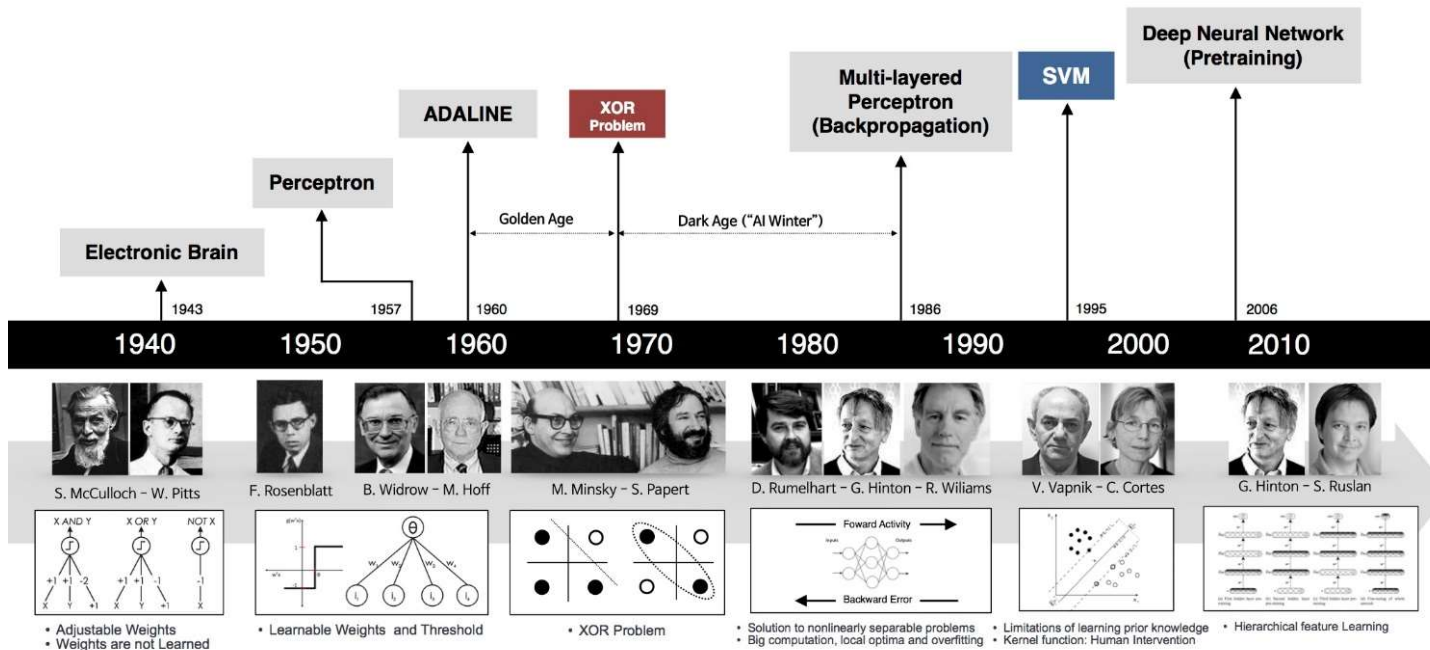
Explosion du deep learning

- Il faut des données : souvent images, texte
 - Computer vision
 - Natural Language Processing



Age d'or du
deep learning :
données + GPU

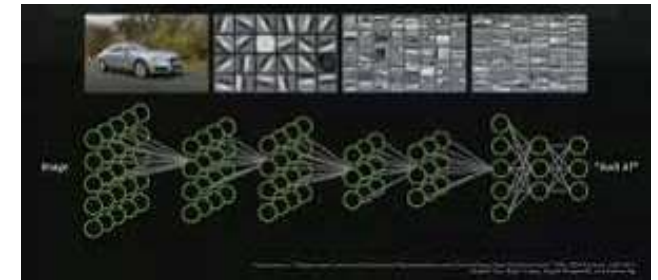
CNN, AE, GAN,
Transformer, ...



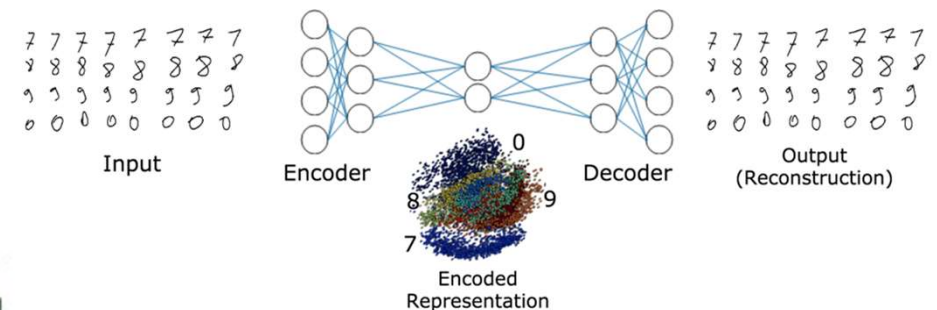
Explosion du deep learning

- Invention de différents type de réseaux

- ConvNN



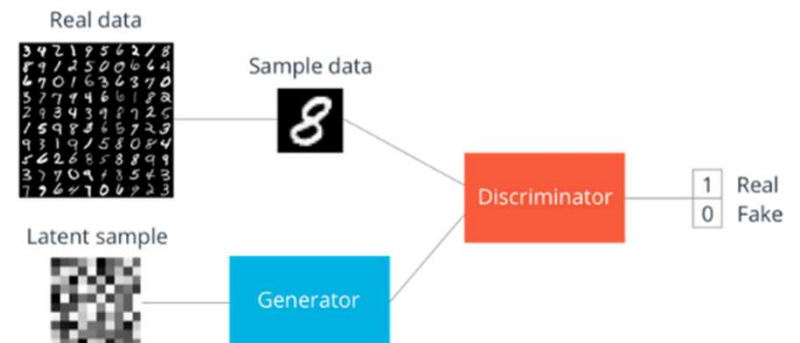
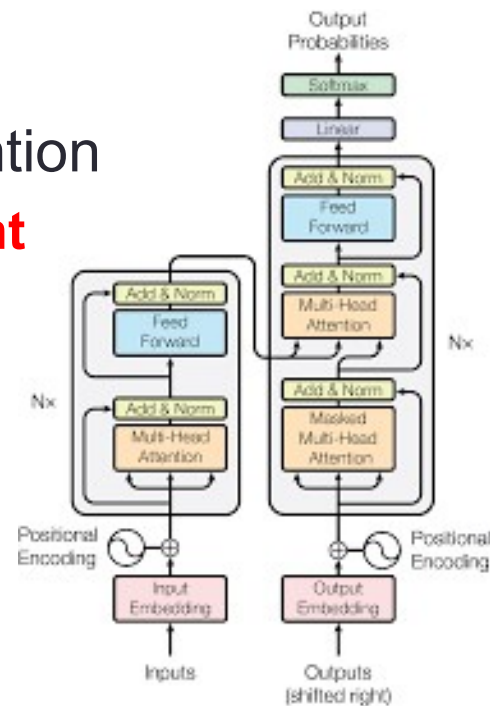
- Auto-encoder



- GAN

- Transformer + attention

- **la star du moment**

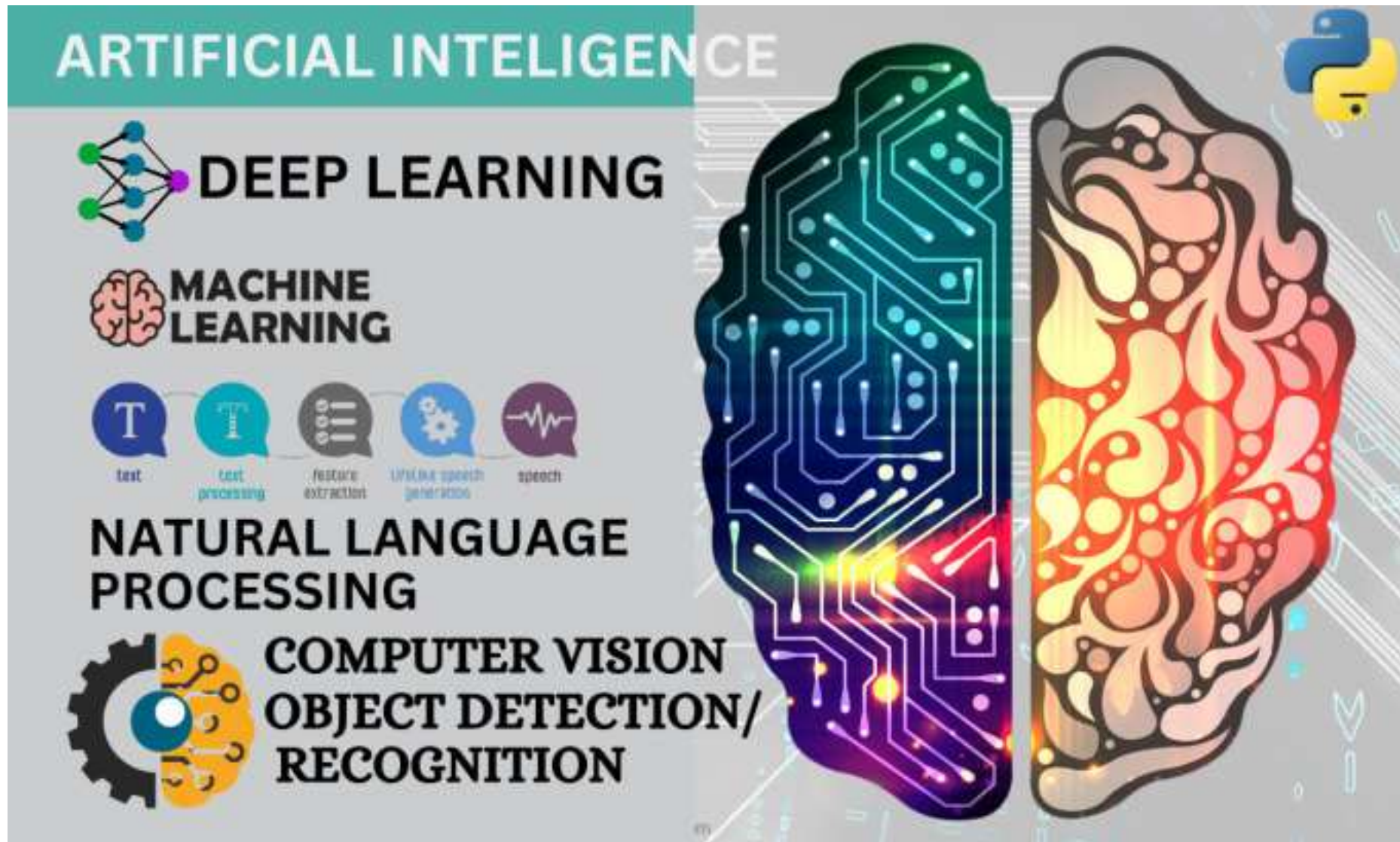


Explosion du deep learning

- Des données
- Du GPU
- Un algo de back-propagation rapide et facilement codable sur GPU
- Des algo, des algo, des algo ...



Conclusion 1



Conclusion 2

- Tous les domaines de la science ~~vont~~ sont touchés
- Les réseaux sont le moteur
- Les données le carburant



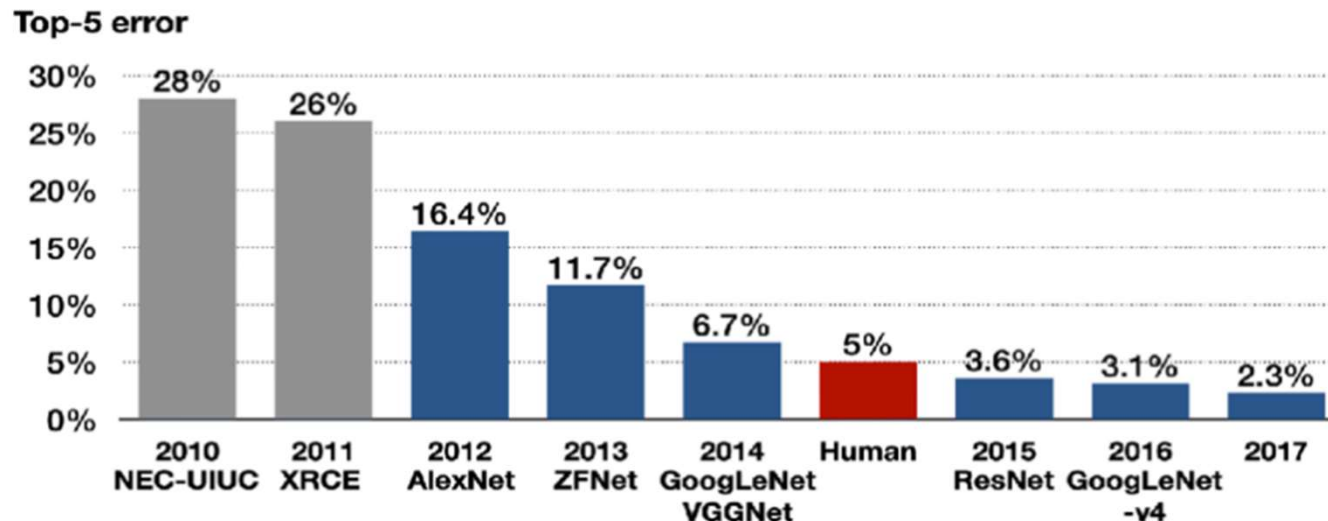
- Mais
 - Maintenant, les inventions se font en trouvant de nouvelles architectures : LSTM, AE, GAN, Transformer, Diffusion, etc.
 - Il reste beaucoup à faire ...
 - Explicabilité
 - Moins de données (frugal), renforcement, non supervisé, etc.
 - Transfert
 - Contrôle/ édition / interaction : apprentissage en continu, etc.
 - Je pense que les réseaux vont rester la brique de base, cela ne veut pas dire qu'il suffit des données et tout est réglé !

VRAC

Corpus d'images

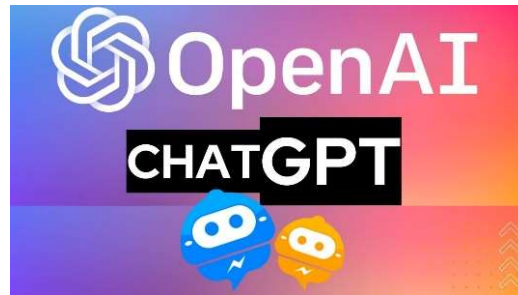


- Classification
 - Vision humaine : repérer un prédateur ou un membre de sa famille
 - Concours IMAGENET → mettre un label sur une image
 - 10 millions d'images avec 11000 classes, 1.4To



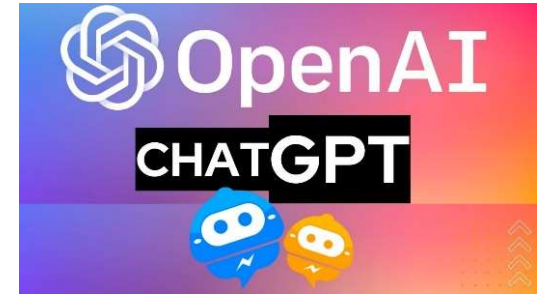
En bleu, méthodes à base de réseaux

ChatGPT Generative Pre-trained Transformer



- Dans son moteur, il y a deux parties
 - modèle géant de prédiction de texte combiné avec un modèle encyclopédique
 - modèle conversationnel
 - Apprentissage par renforcement à partir de retour humains
 - Seulement pour la conversation
- **Entrainé avec 500 milliards de mots**
- **175 Milliards de paramètres**
 - 100 milliards de neurone dans un cerveau humain mais ATTENTION à la comparaison

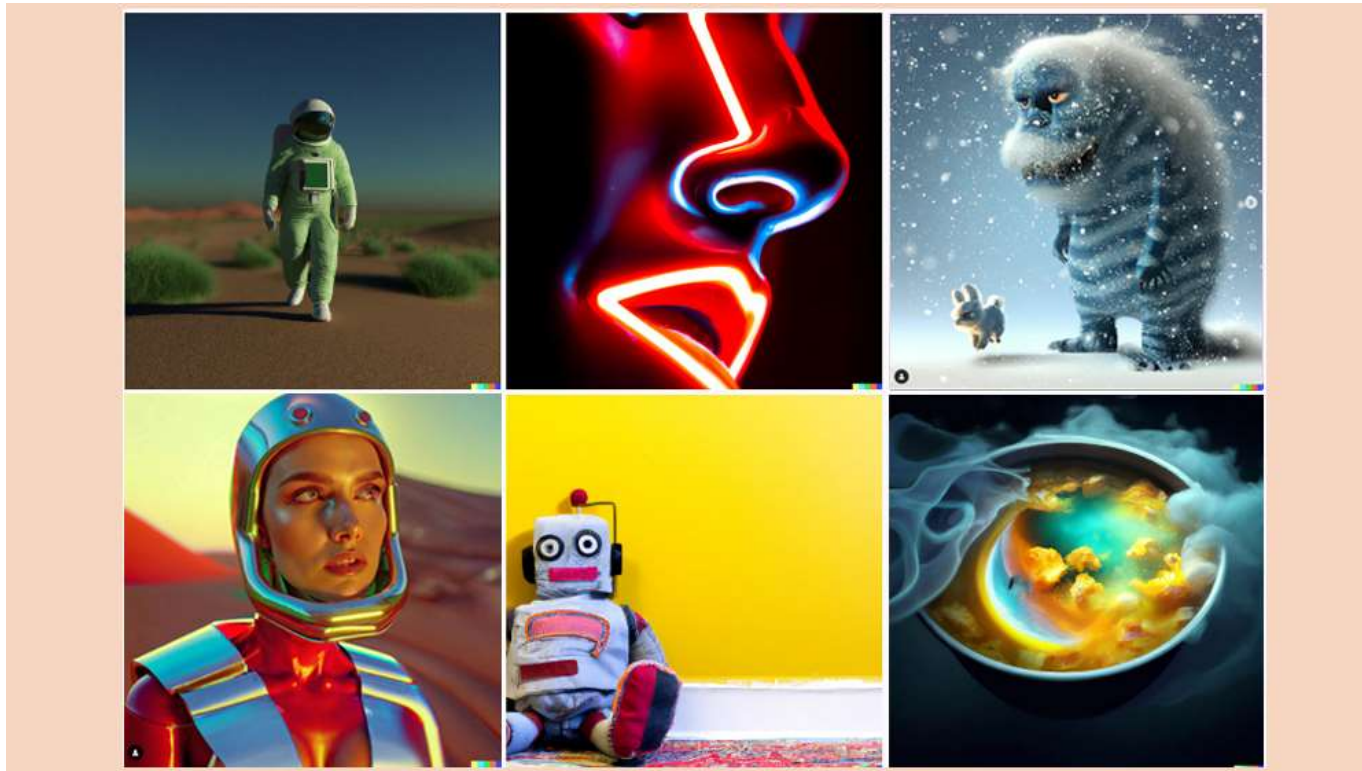
ChatGPT



- Existe dans les labos depuis 2018 mais
 - Grand public demandent de la puissance
 - Risque de mauvais buzz s'il se trompe !
 - Le modèle linguistique n'est pas actualisé (stop en 2021)
 - Le modèle conversationnel
 - Mémoire de 3000 mots pour la conversation
 - continue de s'affiner en fonction du retour des utilisateurs toutes les 3-4 semaines en moyenne
- ➔ pas de nouvelles connaissances mais générations phrases de meilleures qualités et, inversement, pénalise davantage les générations malvenues

Générateur d'images

- DALL-E : générateur d'images à partir de texte
 - Réseau de diffusion
 - auto-encoder basée sur un bruit inversible
 - Couplé à un modèle de texte



A suivre ...