

Apprentissage Par Problème [APP]

Comprendre et utiliser le *Machine learning*

Jean-Luc.Charles@ENSAM.EU



septembre 2023



L'aspect historique...

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.



DEEP LEARNING

Deep learning breakthroughs drive AI boom.



1950's

1960's

1970's

1980's

1990's

2000's

2010's

(crédit : developer.nvidia.com/deep-learning)

Intelligence Artificielle ?

Intelligence Artificielle¹ : reste un terme ambigu aux définitions multiples :

- *"...the science of making computers do things that require intelligence when done by humans."* [Alan Turing, 1940](#)
- *"the field of study that gives computers the ability to learn without being explicitly programmed."* [Arthur Samuel, 1959](#)
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ."* [Tom Mitchell, 1997](#)
- Notion d'agent intelligent ou d'agent rationnel
"...agent qui agit de manière à atteindre la meilleure solution ou, dans un environnement incertain, la meilleure solution prévisible." [Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015](#)

¹ utilisé la première fois en 1956 par [John McCarthy](#), chercheur à Stanford lors de la conférence de Dartmouth

Intelligences Artificielles ?

IA Forte (*Strong AI*)

- Vise à concevoir des systèmes qui pensent exactement comme les humains.
- Peut contribuer à expliquer comment les humains pensent...
- On en est encore loin...

IA Faible (*Weak AI*)

- Vise à concevoir des systèmes qui peuvent “se comporter” comme des humains.
- Ne nous dit rien sur la façon dont les humains pensent.
- On y est déjà... On l'utilise tous les jours !
reconnaissance faciale, vocale, anti-spam, traduction...

Machine Learning et IA

Page extraite de [medium.com/machine-learning-for-humans/...](https://medium.com/machine-learning-for-humans/)

Machine learning \subseteq artificial intelligence

ARTIFICIAL INTELLIGENCE

Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goal.

Subfields: vision, robotics, machine learning, natural language processing, planning, ...

MACHINE LEARNING

Gives "computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)

SUPERVISED LEARNING

Classification, regression

UNSUPERVISED LEARNING

Clustering, dimensionality
reduction, recommendation

REINFORCEMENT LEARNING

Reward maximization

Machine Learning et IA

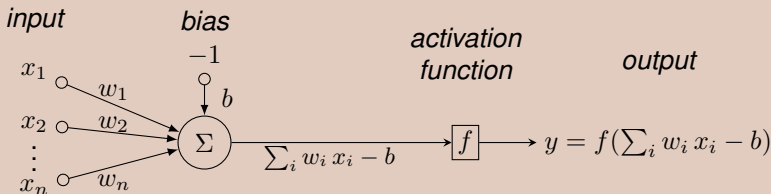
Plusieurs approches permettent de concevoir des algorithmes de *Machine Learning* :

- Programmation Génétique (*Genetic programming*)
- Inférence bayésienne (*Bayesian inference*)
- Logique Floue (*Fuzzy logic*)
- Réseaux de neurones (*Neural Networks*)
- ...

La suite traite uniquement des **Réseau de neurones artificiels**.

Neurone artificiel

Le modèle informatique du neurone artificiel



Un **neurone artificiel**:

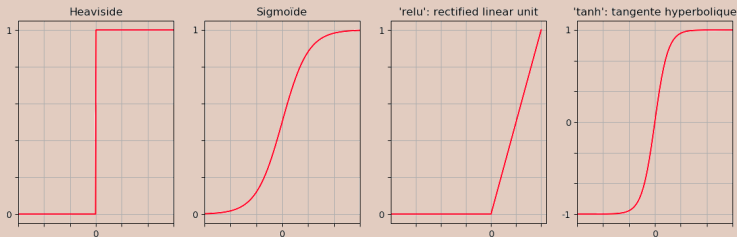
- Reçoit les données d'entrée $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$ (*weights*)
- Calcule la **somme pondérée** de ses entrées moins le biais : $\sum_i w_i x_i - b$
- Produit en sortie une **activation** $f(\sum_i w_i x_i - b)$, calculée avec une fonction d'activation f (en général non-linéaire).

Neurone artificiel

La fonction d'activation d'un neurone :

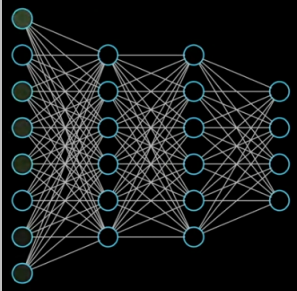
- introduit un comportement non-linéaire,
- fixe la plage de la sortie du neurone, par exemple $[-1, 1]$, $[0, 1]$ ou encore $[0, \infty[$.

Exemples de fonctions d'activation souvent utilisées



- Le biais b fixe le seuil d'activation du neurone.

Réseaux de neurones étudiés

- Les réseaux de neurones sont des assemblages plus ou moins complexes de neurones artificiels
- 
- The diagram illustrates a dense neural network with four layers of nodes. The first layer on the left has 8 nodes, the second has 6, the third has 4, and the fourth has 3. Every node in one layer is connected to every node in the next layer by a line, representing a fully connected architecture. The nodes are represented by small circles, and the connections are thin lines.
- Deux architectures souvent utilisées pour la classification par réseau de neurones :
 - Les **réseaux denses**, simples, généralistes.
 - Les **réseaux convolutifs** plus complexes, spécialisés dans le traitement des images.

La suite traite uniquement des **réseaux de neurones denses**.

Données utilisées pour l'APP

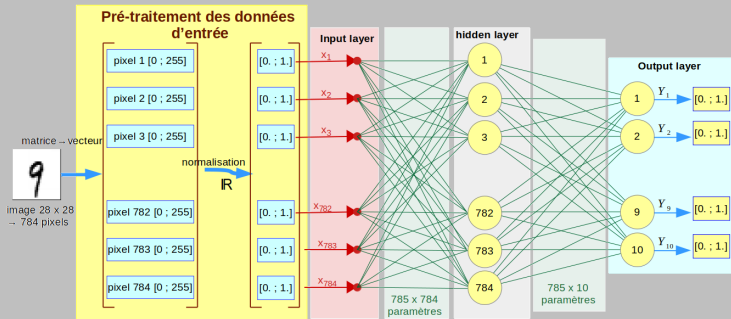
- **MNIST** : banque de 70000 **images labellisées**



- Images en ton de gris de 28×28 pixels
- 60000 images d'entraînement et 10000 images de test.

Réseau de neurones dense

Chaque matrice $28 \times 28 \rightsquigarrow$ vecteur normalisé de 784 composantes `float` $\in [0; 1]$.

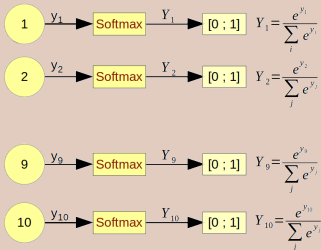


- Couche d'entrée (*Input layer*) : fixe la dimension des entrées du réseau, ne comporte aucun neurone.
- Couche "cachée" (*Hidden layer*) de 784 neurone (on pourrait essayer plus, ou moins...) : reçoit les données d'entrées.
- Couche de sortie (*Output layer*) : 10 neurones, un pour chaque chiffre à reconnaître.

Réseau de neurones dense

- Dans les couches intermédiaires la fonction d'activation **relu** favorise l'apprentissage du réseau ².
- La classification (dernière couche) utilise la fonction **softmax** :

Fonction d'activation *softmax*



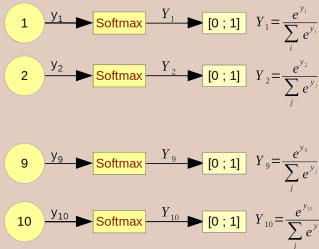
- L'activation du neurone k est $Y_k = e^{y_k} / \sum_i e^{y_i}$ avec $y_k = \sum_i \omega_i x_i - b$ calculé par le neurone k .
- Les sorties des neurones s'interprètent comme des probabilités dans l'intervalle $[0, 1]$.

² évite le *vanishing gradient* qui apparaît dans l'algorithme de *back propagation*

Réseau de neurones dense

- Dans les couches intermédiaires la fonction d'activation **relu** favorise l'apprentissage du réseau ².
- La classification (dernière couche) utilise la fonction **softmax** :

Fonction d'activation *softmax*



- L'activation du neurone k est $Y_k = e^{y_k} / \sum_i e^{y_i}$ avec $y_k = \sum_i \omega_i x_i - b$ calculé par le neurone k .
- Les sorties des neurones s'interprètent comme des probabilités dans l'intervalle $[0, 1]$.

Réponse du réseau \leadsto label associé au neurone de plus grande probabilité.

² évite le *vanishing gradient* qui apparaît dans l'algorithme de *back propagation*

Réseau de neurones dense

Codage *One-hot* des labels

But : mettre les label des images au format de la sortie du réseau

- Labels des images : **nombres entiers** de 0 à 9.
- Sortie du réseau : **vecteur de 10 float** dans l'intervalle $[0,1]$ calculés par les fonctions *softmax* des 10 neurones de sortie.
- Codage *one-hot* d'un ensemble ordonné de N labels :
 - chaque label est représenté par un vecteur à N composantes toutes nulles sauf une égale à 1,
 - le rang du 1 dans le vecteur associé à un label est le rang du label.

Réseau de neurones dense

Codage *One-hot* des labels

But : mettre les label des images au format de la sortie du réseau

- Labels des images : **nombres entiers** de 0 à 9.
- Sortie du réseau : **vecteur de 10 float** dans l'intervalle $[0,1]$ calculés par les fonctions *softmax* des 10 neurones de sortie.

● Codage *one-hot* d'un ensemble ordonné de N labels :

chiffre	Y'_i : vecteur <i>one-hot</i>
0	[1 0 0 0 0 0 0 0 0 0]
1	[0 1 0 0 0 0 0 0 0 0]
2	[0 0 1 0 0 0 0 0 0 0]
3	[0 0 0 1 0 0 0 0 0 0]
4	[0 0 0 0 1 0 0 0 0 0]
5	[0 0 0 0 0 1 0 0 0 0]
6	[0 0 0 0 0 0 1 0 0 0]
7	[0 0 0 0 0 0 0 1 0 0]
8	[0 0 0 0 0 0 0 0 1 0]
9	[0 0 0 0 0 0 0 0 0 1]

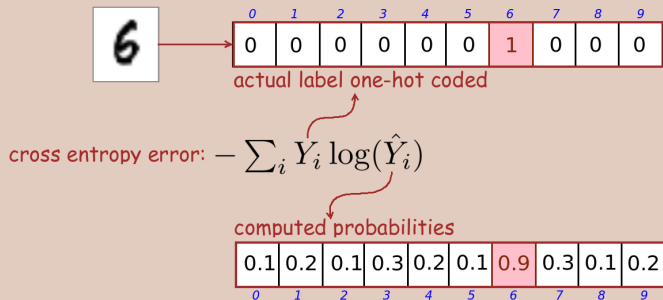
label est représenté par un vecteur à N composantes
toutes sauf une égale à 0,
le 1 dans le vecteur associé à un label est le rang du

Le codage *one-hot* des labels '0' à '9' donne un vecteur à 10 composantes, comme celui calculé par le réseau de neurones.

Réseau de neurones dense

Fonction d'erreur : *Cross entropy error*

- Une image traitée par le réseau \leadsto vecteur \hat{Y} de 10 float à comparer au codage *hot-one* Y du label de l'image.
- Fonction d'erreur *cross entropy* : $e(\hat{Y}, Y) = - \sum_i Y_i \log(\hat{Y}_i)$



Réseau de neurones dense

Optimisation et *Back Propagation*

- Pendant la phase d'apprentissage un algorithme d'optimisation calcule le gradient de la fonction de perte par rapport aux poids du réseau.

Réseau de neurones dense

Optimisation et *Back Propagation*

- Pendant la phase d'apprentissage un algorithme d'optimisation calcule le gradient de la fonction de perte par rapport aux poids du réseau.
- L'algorithme de *Back Propagation* **modifie** les poids du réseau couche par couche grâce au gradient de la fonction de perte, en itérant de la dernière couche à la première couche.

Réseau de neurones dense

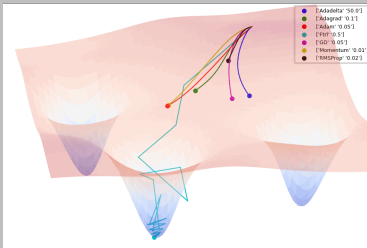
Optimisation et *Back Propagation*

- Pendant la phase d'apprentissage un algorithme d'optimisation calcule le gradient de la fonction de perte par rapport aux poids du réseau.
- L'algorithme de *Back Propagation* **modifie** les poids du réseau couche par couche grâce au gradient de la fonction de perte, en itérant de la dernière couche à la première couche.
- Exemples d'algorithme d'optimisation utilisés :
 - Descente de Gradient (*Gradient Descent (GD)*)
 - Descente de Gradient Stochastique (*Stochastic Gradient Descent (SGD)*)
 - *Adam* (version améliorée de descente de gradient)...

Le module [tf.keras.optimizers](#) propose l'implémentation Python de plusieurs algorithmes d'optimisation.

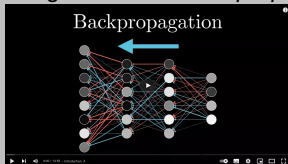
Réseau de neurones dense

Visualisation des itérations d'algorithmes de descente de gradient pour une fonction de perte ultra-simple à seulement 2 variables :



(source : github.com/Jaewan-Yun/optimizer-visualization)

Vidéo d'explication de l'algorithme de *back propagation* :



Mise en oeuvre dans l'APP-ML

Auto-formation

- Les trois *notebooks* [ML1_MNIST.ipynb](#), [ML2_DNN.ipynb](#) et [ML3_DNN_suite.ipynb](#) proposés sur SAVOIR visent les savoir-faire :
 - charger et pré-traiter les images du MNIST,
 - construire un réseau de neurones **dense**,
 - entraîner le réseau reconnaître les images du MNIST,
 - évaluer et exploiter le réseau entraîné.
- Les modules Python utilisés pour créer les réseaux de neurones et les entraîner sont [tensorflow](#) et [keras](#).
- Les scores obtenus avec des réseaux denses peuvent atteindre 98% de réussite dans les cas les plus favorables.

Mise en oeuvre dans l'APP-ML

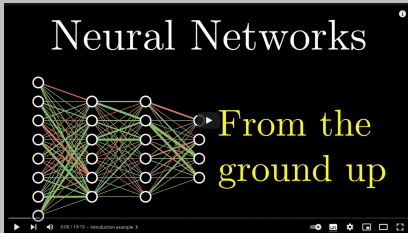
Résolution d'un problème de classification

- On dispose de données acquises sur un banc de perçage instrumenté.
- Une première étape de pré-traitement permet de calculer une quinzaine de données traitées (*features* à partir des données brutes.
- L'étude des pré-traitements sera abordée lors de séances de *traitement du signal* dédiées.
- Le **problème** proposé : entraîner un Réseau de Neurones Dense à reconnaître le matériau percé à partir des signaux traités obtenus pendant le perçage.

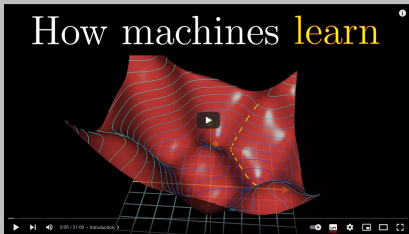
Vidéographie



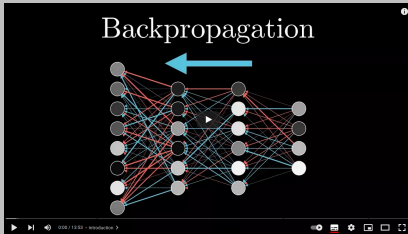
1/ Local: "Le deep learning - YouTube.webm"



2/ local : "But what is a neural network.webm"



3/ Local: "Gradient descent how neural networks learn.webm"



4/ Local: "What is backpropagation really doing .webm"

Bibliographie

- [1] *Intelligence Artificielle*, 3e édition, PEARSON Education, 2010, ISBN : 2-7440–7455–4, aima.cs.berkeley.edu
- [2] *What is artificial intelligence (AI), and what is the difference between general AI and narrow AI?*, Kris Hammond, 2015
www.computerworld.com/article/2906336/what-is-artificial-intelligence.html
- [3] *Stanford Encyclopedia of Philosophy*, plato.stanford.edu/entries/artificial-intelligence
- [4] *Deep Learning.*, Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016), MIT Pres, ISBN 9780262035613