

# Apprentissage profond

## Introduction et applications en physique

Patrice Béchar

Département d'informatique et de recherche opérationnelle

24 octobre 2018

L'IA dans la culture populaire

Histoire de l'IA et de l'apprentissage automatique

Bases de l'apprentissage automatique

Réseaux de neurones et apprentissage profond

Principes et fondements

Démonstration

Les physiciens dans l'ère du Deep Learning

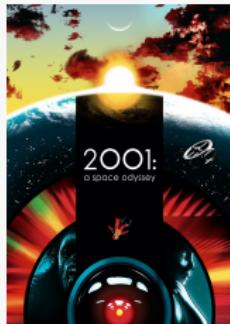
Beaucoup d'applications...

...Et beaucoup de travail en théorie

Pour aller plus loin...

Librairies Python pour l'apprentissage automatique

Cours et autres ressources



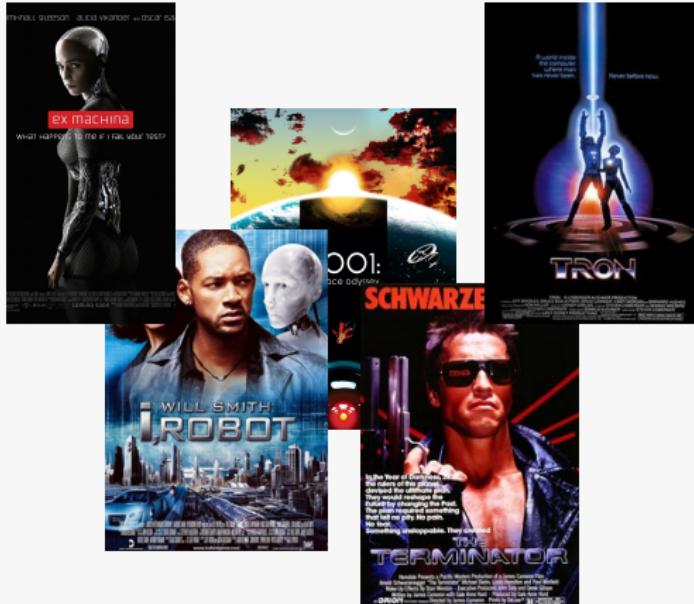




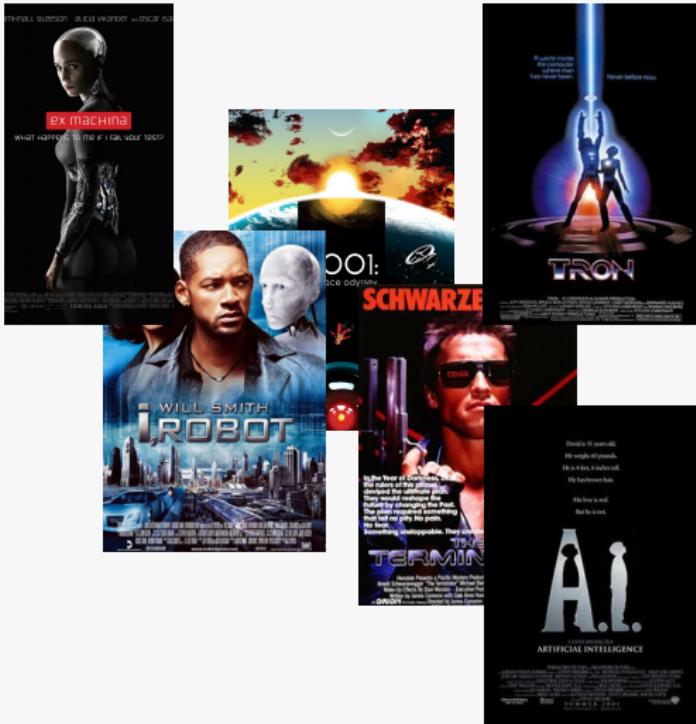
# L'IA dans la culture populaire



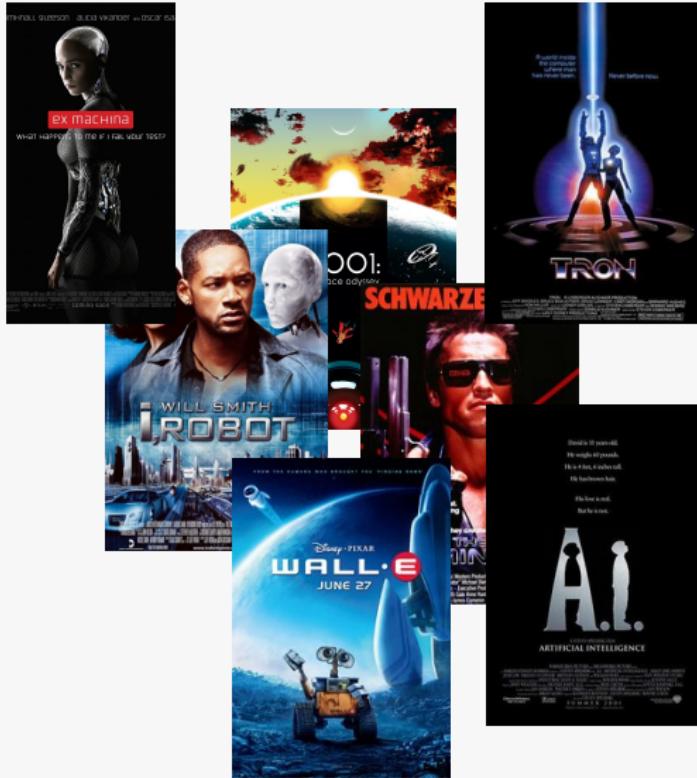
# L'IA dans la culture populaire



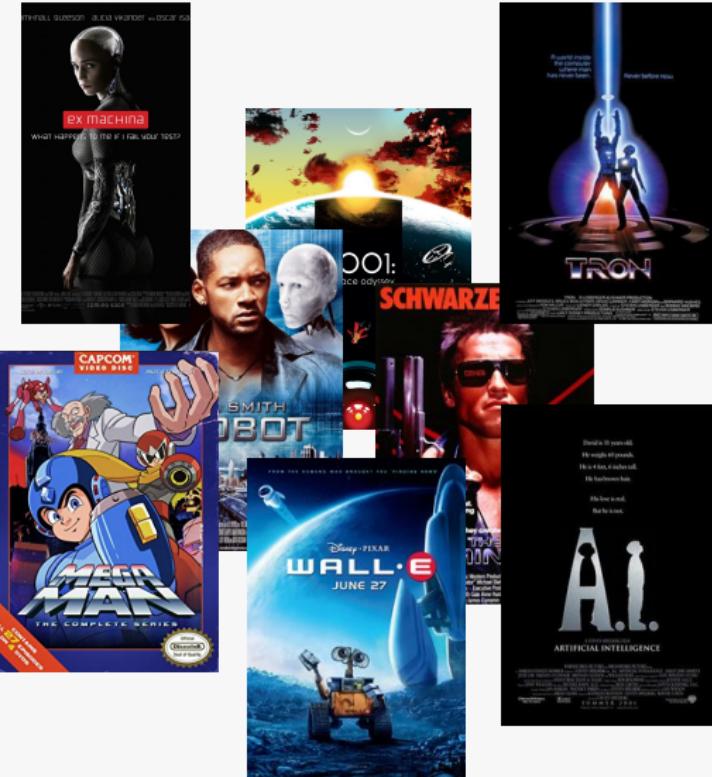
# L'IA dans la culture populaire



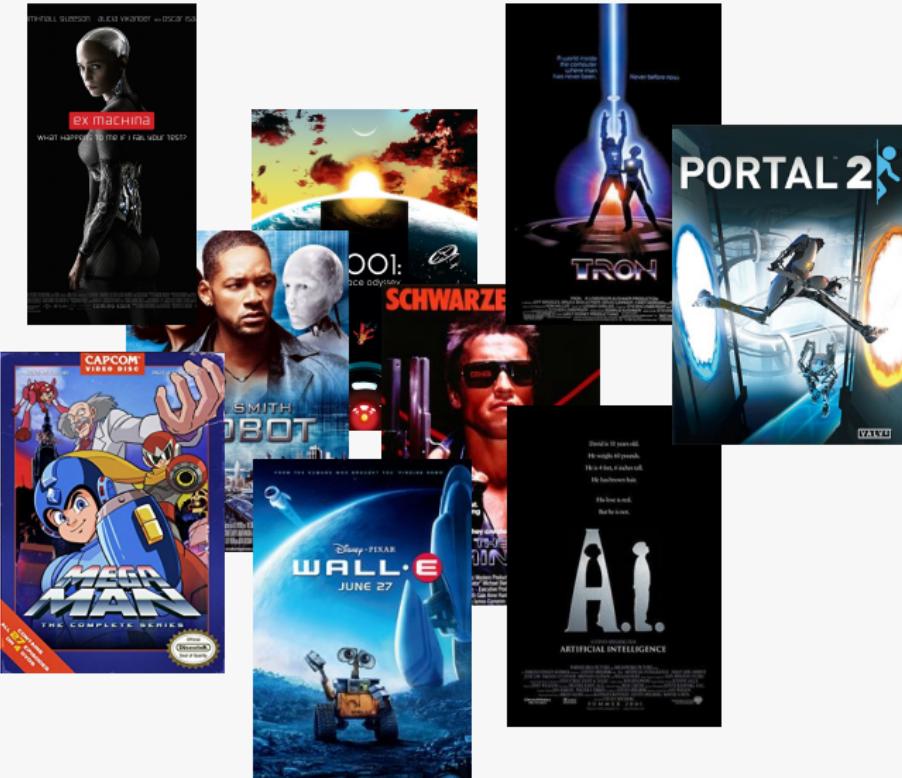
# L'IA dans la culture populaire



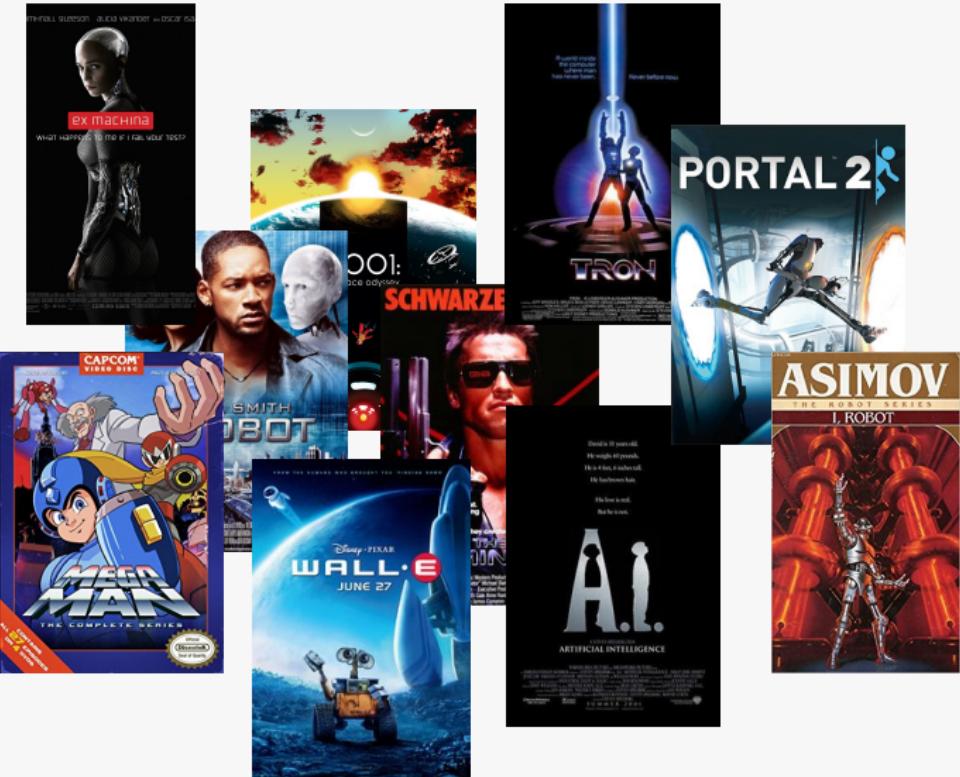
# L'IA dans la culture populaire



# L'IA dans la culture populaire



# L'IA dans la culture populaire



**Antiquité** Idée des robots conscients / automates (Égypte, Grèce)

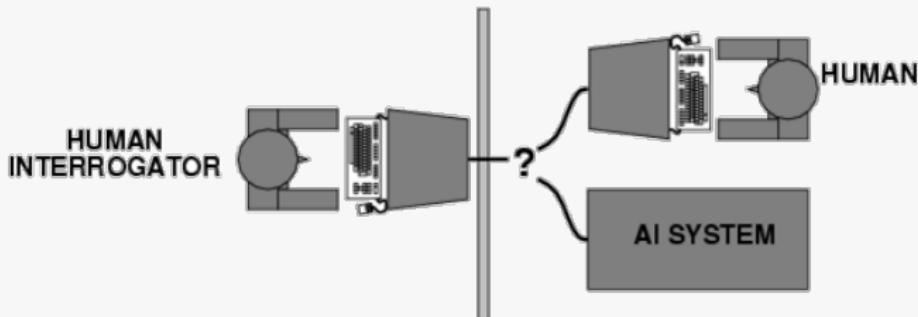
**17<sup>e</sup> siècle** Descartes, Hobbes et la philosophie mécaniste.  
Leibniz et la Caractéristique universelle.

**20<sup>e</sup> siècle** Boole, Russell et les fondements de la logique.  
Hilbert : Peut-on formaliser tout raisonnement mathématique ?  
Réponses de Gödel, Turing, Church...

## Conclusions :

- ▶ Il existe des limites à ce que l'on peut accomplir avec la logique.
- ▶ Dans ces limites, toute forme de raisonnement mathématique peut être mécanisé.

- 1943 Pitts & McCulloch proposent l'idée du neurone artificiel [13]
- 1950 Test de Turing [21]
- 1957 Rosenblatt propose l'algorithme du perceptron [19]



1960 - 2000 Hauts et bas dans le milieu.

## Problèmes

- ▶ Puissance des ordinateurs toujours limitée
- ▶ Données souvent insuffisantes
- ▶ Paradoxe de Moravec

## Paradoxe de Moravec

*"It is comparatively easy to make computers exhibit adult-level performance in solving problems on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility."*

(Hans Moravec[16])

## Avancées récentes

- 1998 Deep Blue bat Garry Kasparov aux échecs.
- 2011 Watson bat les deux plus grands champions à *Jeopardy!*
- 2016 AlphaGo bat Lee Sedol au jeu de Go.
- 2017 AlphaZero atteint un niveau surhumain aux jeux de Go, shogi et échecs sans données externes en 24h[20].



## Autres domaines où les progrès sont fulgurants

- ▶ Vision par ordinateur
- ▶ Reconnaissance de la parole
- ▶ Traduction automatique
- ▶ Génération d'images
- ▶ et beaucoup plus...

## Deux approches à l'intelligence artificielle

### IA «classique» symbolique

- ▶ Fondé sur le raisonnement logique
- ▶ Règles codées à la main (if... else...)
- ▶ Pas de gestion de l'incertain

### Apprentissage automatique

- ▶ Apprendre les paramètres à partir d'exemples
- ▶ Approche probabiliste
- ▶ Objectif : **généralisation**

## Types de problèmes

- ▶ Apprentissage supervisé
  - ▶ Régression
  - ▶ Classification
- ▶ Apprentissage non-supervisé
  - ▶ Réduction de dimensionnalité, clustering, détection d'anomalie, ...
- ▶ Apprentissage par renforcement

## Apprentissage supervisé

- ▶ Ensemble de données  
 $\mathcal{D}_{\text{train}} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}.$
- ▶ Le modèle prédit une sortie  $f(x^{(i)})$  en fonction de l'entrée  $x^{(i)}$ . On veut que le modèle prédise la cible  $y^{(i)}$ .
- ▶ Lors de la phase d'entraînement, le modèle ajuste ses paramètres  $\Theta$  dans le but de minimiser le risque empirique  $\hat{R}$  selon une fonction de coût  $\mathcal{L}(y, f(x))$

$$\hat{R} = \frac{1}{N} \sum_{i=1}^N \mathcal{L}\left(y^{(i)}, f(x^{(i)})\right) \quad \Theta^* = \arg \min_{\Theta} \hat{R}(f, \mathcal{D}_{\text{train}})$$

## Quelques fonctions de coût...

- ▶ Erreur quadratique (régression) :

$$\mathcal{L}(y, f(x)) = \|y - f(x)\|_2^2$$

- ▶ Erreur de classification :

$$\mathcal{L}(y, f(x)) = \mathbb{1}_{\{y \neq f(x)\}}$$

- ▶ Entropie croisée binaire :

$$\mathcal{L}(y, f(x)) = -y \log f(x) + (1 - y) \log(1 - f(x))$$

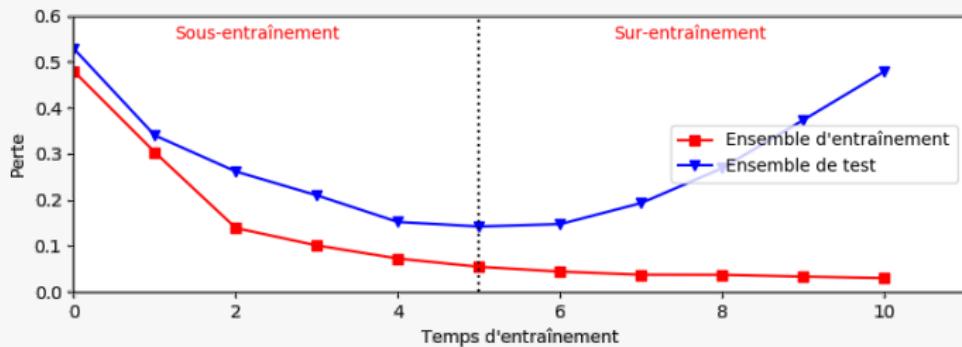
- ▶ Moins log-vraisemblance :

$$\mathcal{L}(y, f(x)) = -\log f(x)_y$$

## Estimer l'erreur de généralisation

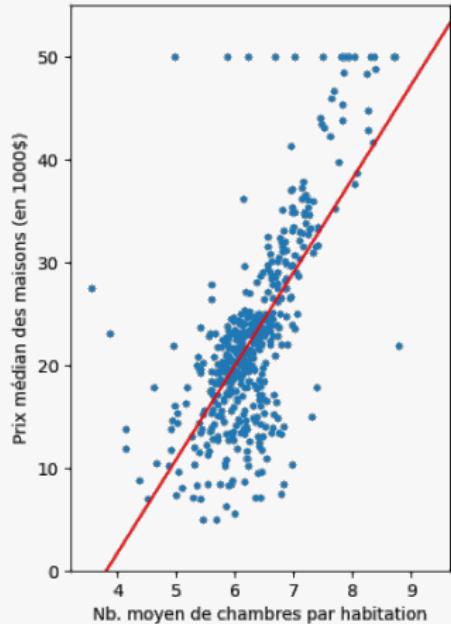
- ▶ But de l'apprentissage machine : **généralisation**
- ▶ On test le modèle entraîné sur des exemples jamais vus auparavant :

$$\mathcal{D}_{\text{test}} = \{(x^{(1')}, y^{(1')}), (x^{(2')}, y^{(2')}), \dots, (x^{(N')}, y^{(N')})\}$$



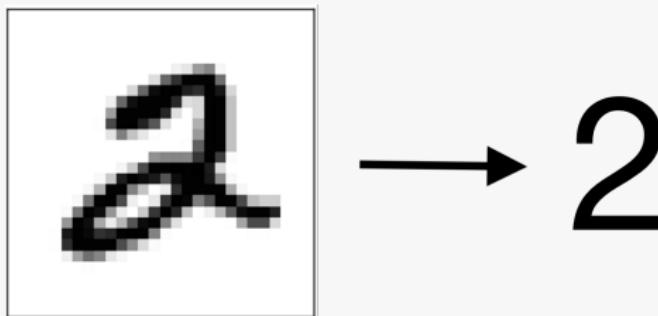
## Exemple : Prédiction du prix des maisons à Boston [6]

- ▶ Problème de régression
- ▶ Entrée : 13 traits caractéristiques ( $\in \mathbb{R}^{13}$ )  
ex : Nb. de chambres, Taux de criminalité, ...
- ▶ Sortie : Prix de la maison ( $\in \mathbb{R}$ )

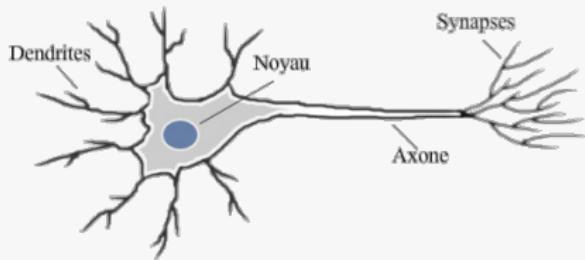


## Exemple : Classification de chiffres manuscrits [7]

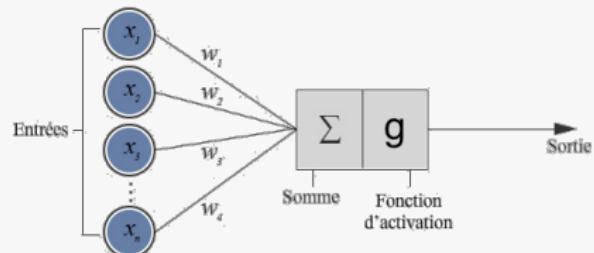
- ▶ Problème de classification
- ▶ Entrée : Image 28x28 px
- ▶ Sortie : Chiffre de 0 à 9



## Neurone biologique

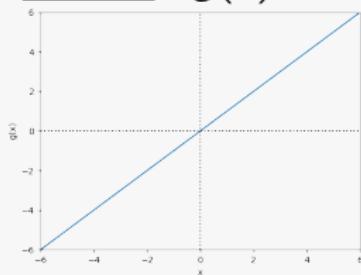


## Neurone artificiel

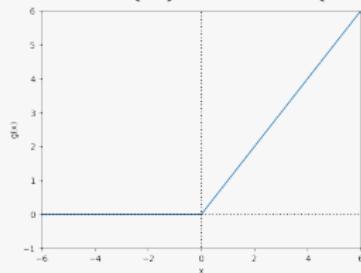


## Quelques fonctions d'activation

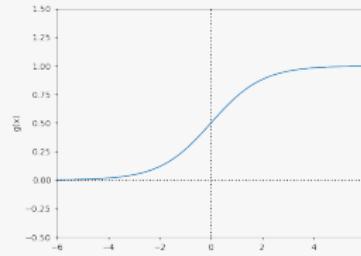
Linéaire :  $g(x) = x$



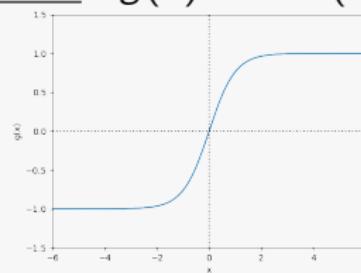
ReLU :  $g(x) = \max(0, x)$



Sigmoïde :  $g(x) = \frac{1}{1+\exp(-x)}$



Tanh :  $g(x) = \tanh(x)$

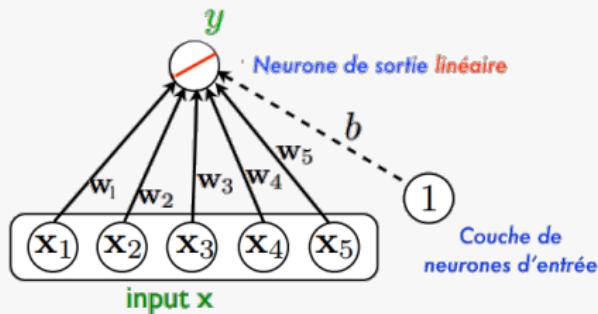


## Modèle linéaire : Régression linéaire

- ▶ Modèle pour la régression

$$f(x) = w^T x + b = \sum_i w_i x_i + b$$

Représentation neuronale :

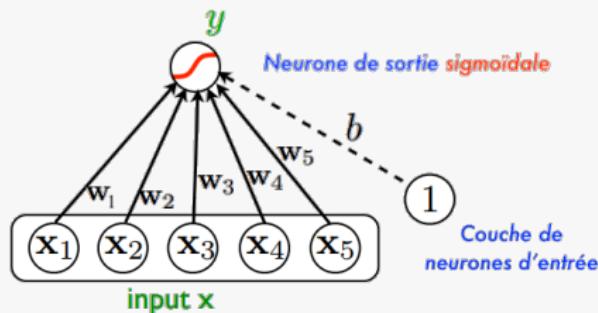


## Modèle linéaire : «Régression» logistique

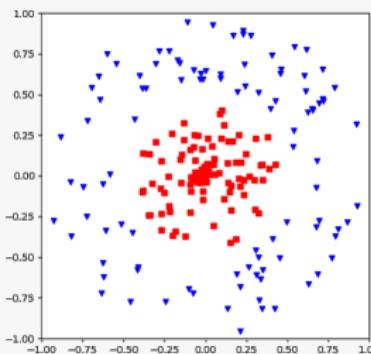
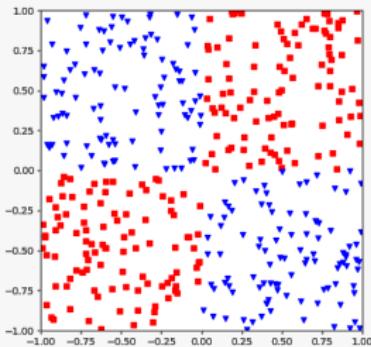
- Modèle pour la classification binaire

$$f(x) = \text{sigm} \left( w^T x + b \right) = \text{sigm} \left( \sum_i w_i x_i + b \right)$$

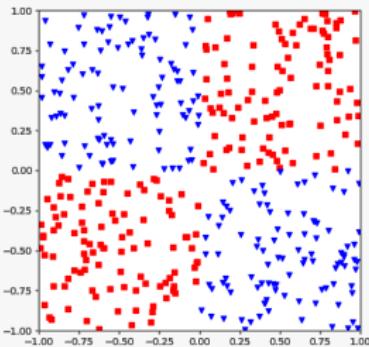
Représentation neuronale :



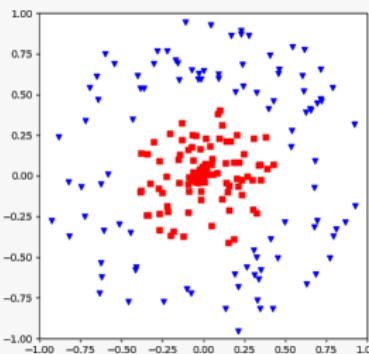
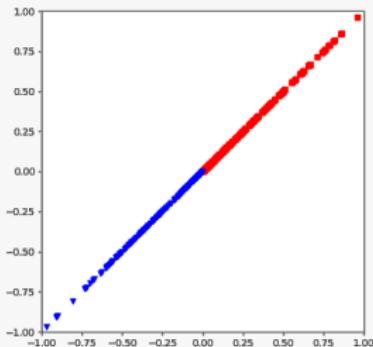
## Problème : ensembles de données non linéairement séparables



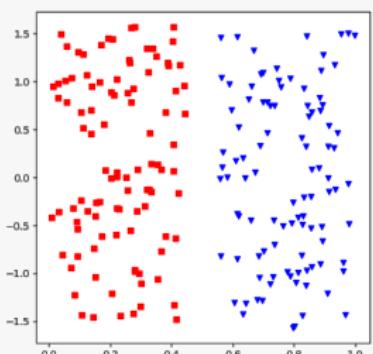
## Problème : ensembles de données non linéairement séparables



Multiplier les  
deux composantes



Coordonnées  
polaires



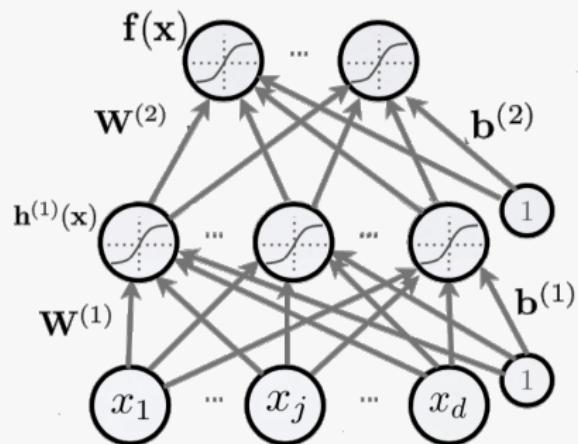
Comment définir ces nouvelles représentations pour chaque ensemble de données ?

Comment définir ces nouvelles représentations pour chaque ensemble de données ?

On laisse le modèle les apprendre !

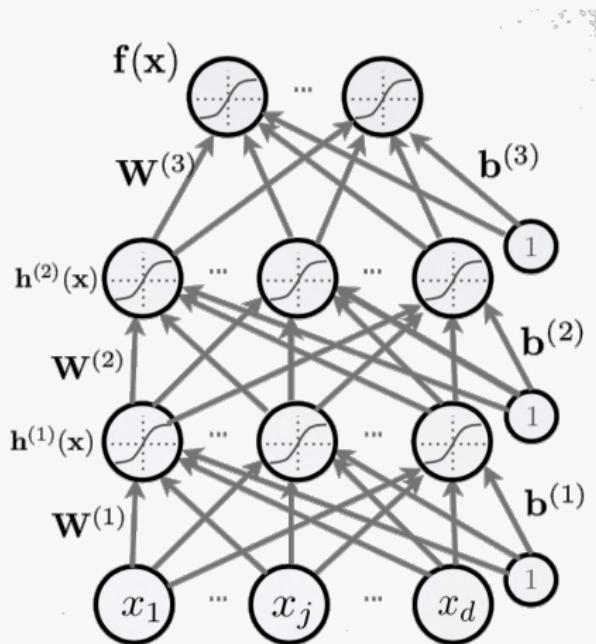
## Introduction d'une couche de neurones cachés

- ▶ Entrée :  $x \in \mathbb{R}^d$
- ▶ Paramètres :  
 $W^{(1)} \in \mathbb{R}^{d \times d_h}$ ,  $b^{(1)} \in \mathbb{R}^{d_h}$   
 $W^{(2)} \in \mathbb{R}^{d_h \times d_o}$ ,  $b^{(2)} \in \mathbb{R}^{d_o}$
- ▶ Propagation :  
$$h^{(1)}(x) = g^{(1)}(W^{(1),T}x + b^{(1)})$$
$$f(x) = g^{(2)}(W^{(2),T}h^{(1)}(x) + b^{(2)})$$



## On peut aussi avoir plus d'une couche cachée !

- ▶ C'est ce qu'on appelle **l'apprentissage profond**.
- ▶ Le nombre de couches cachées ainsi que le nombre de neurones dans chaque couche cachée sont des **hyperparamètres** que l'utilisateur doit ajuster manuellement.

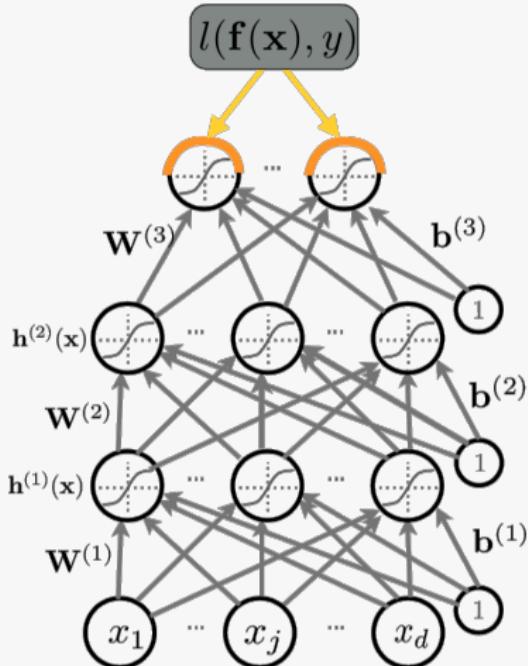


## Apprentissage des paramètres : algorithme de rétropropagation

- ▶ On calcule le gradient de la fonction de coût  $\nabla \mathcal{L}(f(x), y)$  pour chaque élément du réseau en utilisant la règle de dérivée en chaîne.
- ▶ On ajuste les paramètres dans la direction opposée du gradient :

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(f(x), y)^{(t)}$$

- ▶ Le **taux d'apprentissage**  $\eta$  est un hyperparamètre à déterminer.
- ▶ On répète pour chaque exemple jusqu'à convergence.

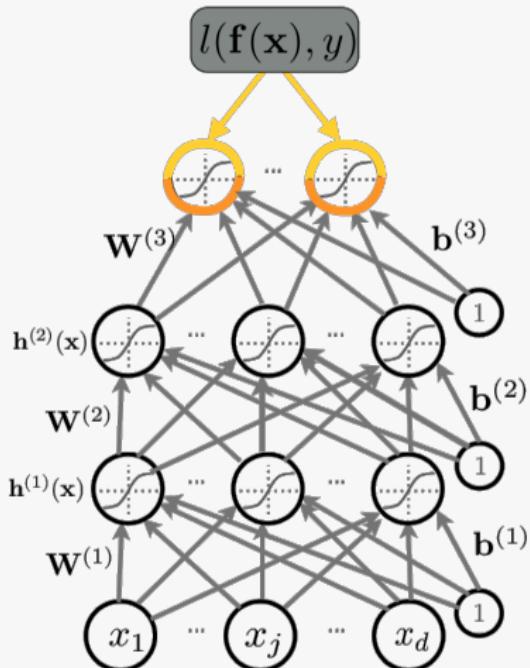


## Apprentissage des paramètres : algorithme de rétropropagation

- ▶ On calcule le gradient de la fonction de coût  $\nabla \mathcal{L}(f(x), y)$  pour chaque élément du réseau en utilisant la règle de dérivée en chaîne.
- ▶ On ajuste les paramètres dans la direction opposée du gradient :

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(f(x), y)^{(t)}$$

- ▶ Le **taux d'apprentissage**  $\eta$  est un hyperparamètre à déterminer.
- ▶ On répète pour chaque exemple jusqu'à convergence.

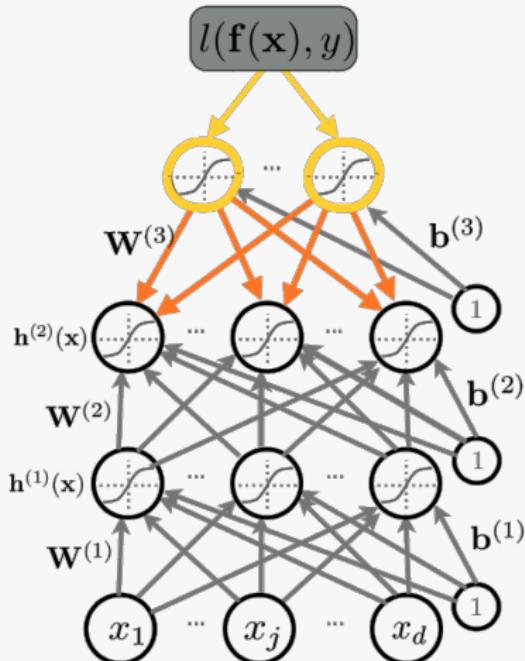


## Apprentissage des paramètres : algorithme de rétropropagation

- ▶ On calcule le gradient de la fonction de coût  $\nabla \mathcal{L}(f(x), y)$  pour chaque élément du réseau en utilisant la règle de dérivée en chaîne.
- ▶ On ajuste les paramètres dans la direction opposée du gradient :

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(f(x), y)^{(t)}$$

- ▶ Le **taux d'apprentissage**  $\eta$  est un hyperparamètre à déterminer.
- ▶ On répète pour chaque exemple jusqu'à convergence.

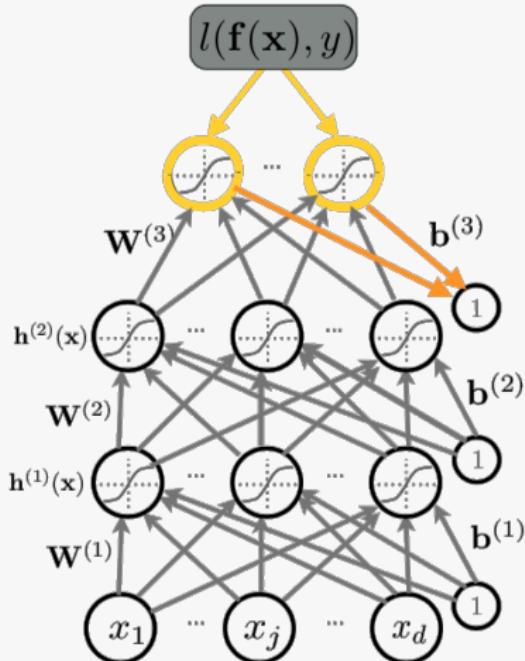


## Apprentissage des paramètres : algorithme de rétropropagation

- ▶ On calcule le gradient de la fonction de coût  $\nabla \mathcal{L}(f(x), y)$  pour chaque élément du réseau en utilisant la règle de dérivée en chaîne.
- ▶ On ajuste les paramètres dans la direction opposée du gradient :

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(f(x), y)^{(t)}$$

- ▶ Le **taux d'apprentissage**  $\eta$  est un hyperparamètre à déterminer.
- ▶ On répète pour chaque exemple jusqu'à convergence.

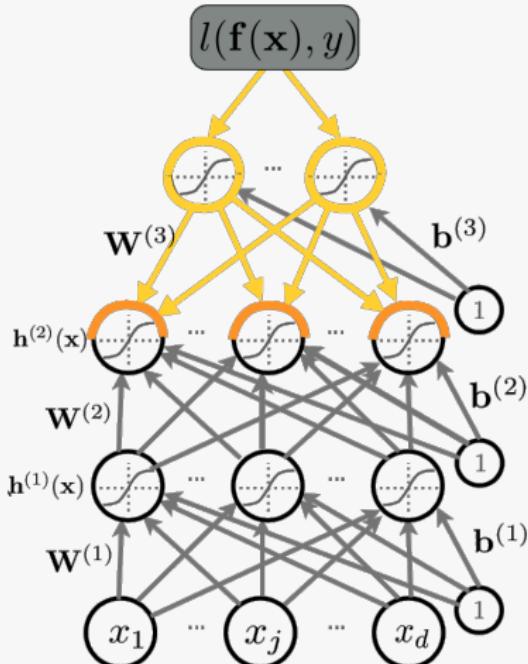


## Apprentissage des paramètres : algorithme de rétropropagation

- ▶ On calcule le gradient de la fonction de coût  $\nabla \mathcal{L}(f(x), y)$  pour chaque élément du réseau en utilisant la règle de dérivée en chaîne.
- ▶ On ajuste les paramètres dans la direction opposée du gradient :

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(f(x), y)^{(t)}$$

- ▶ Le **taux d'apprentissage**  $\eta$  est un hyperparamètre à déterminer.
- ▶ On répète pour chaque exemple jusqu'à convergence.

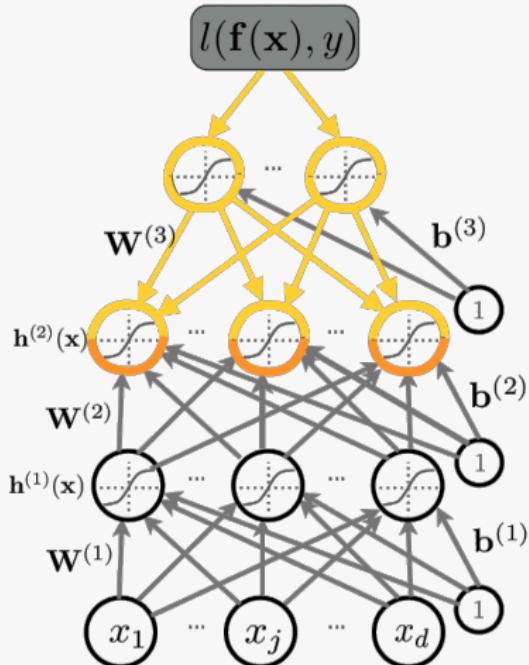


## Apprentissage des paramètres : algorithme de rétropropagation

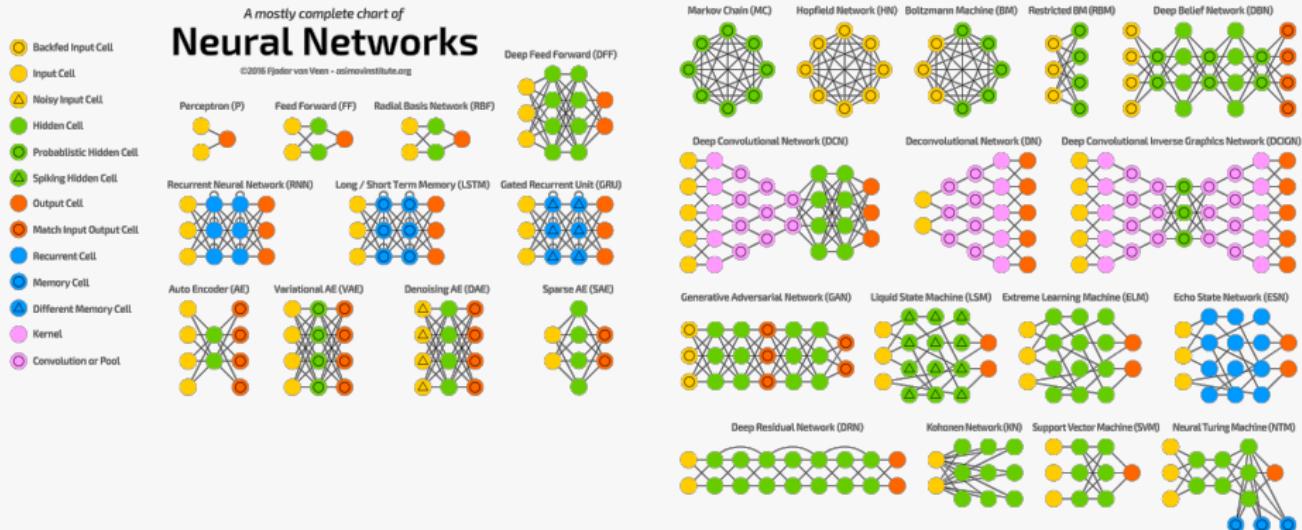
- ▶ On calcule le gradient de la fonction de coût  $\nabla \mathcal{L}(f(x), y)$  pour chaque élément du réseau en utilisant la règle de dérivée en chaîne.
- ▶ On ajuste les paramètres dans la direction opposée du gradient :

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(f(x), y)^{(t)}$$

- ▶ Le **taux d'apprentissage**  $\eta$  est un hyperparamètre à déterminer.
- ▶ On répète pour chaque exemple jusqu'à convergence.

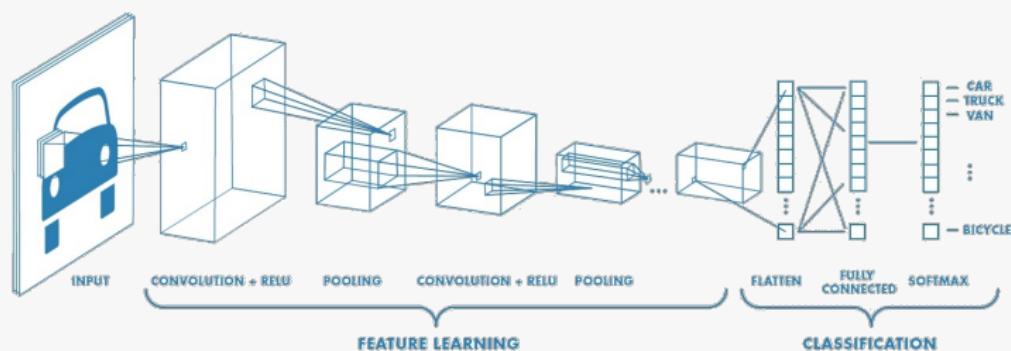


Beaucoup d'autres types de réseaux de neurones existent !



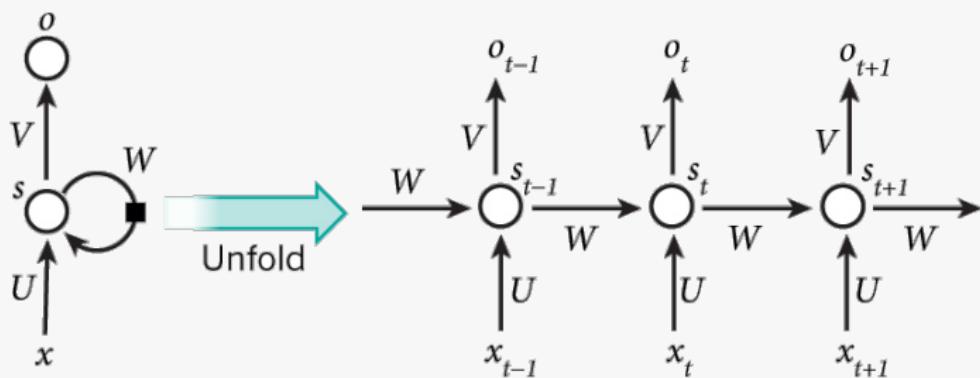
## Réseaux de neurones convolutifs (CNN)

- ▶ Grandement utilisé pour la vision par ordinateur
- ▶ Prend en compte de la connectivité locale des paramètres
- ▶ Permet d'introduire une invariance dans la compréhension des paramètres

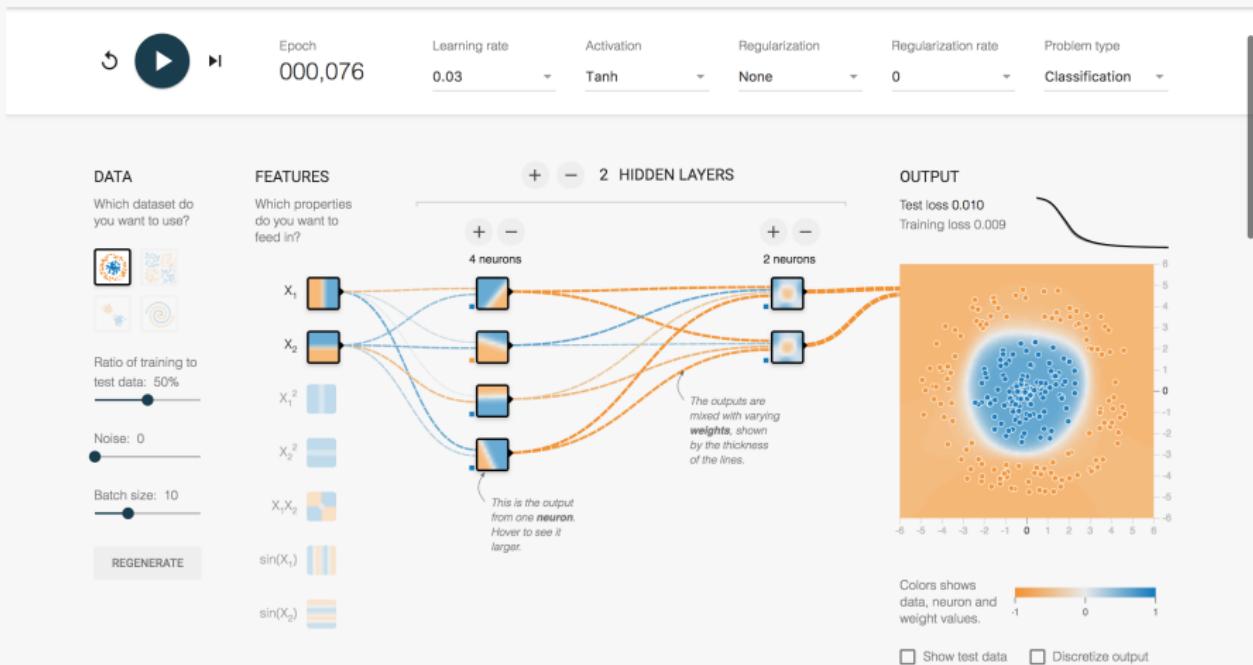


## Réseaux de neurones récurrents (RNN)

- ▶ Utilisé dans les données avec corrélation temporelle et dans le traitement des langues naturelles
- ▶ Capable de manipuler des entrées de différentes longueurs



## Démonstration : Tensorflow Playground



Quelle est la place des physiciens dans tout ça ?

Beaucoup d'applications possibles dans toutes les branches de la physique !

## Astronomie

- ▶ Beaucoup de données → potentiel énorme !
- ▶ exemples :
  - ▶ Détection d'exoplanètes [17]
  - ▶ Prédiction de phénomènes solaires [14]
  - ▶ Détection d'ondes gravitationnelles [5]

## Matière condensée

- ▶ Prédiction de l'énergie du niveau fondamental d'un électron [15]
- ▶ Identification des phases et transitions de phases [3]

## Physique des particules

- ▶ Détection de particules exotiques [1]
- ▶ Caractérisation de jets au LHC [12]

## Climatologie

- ▶ Détection de phénomènes météorologiques extrêmes [11]

## Physique médicale

- ▶ Détection d'anomalies et de maladies dans une panoplie d'images médicales [10]

## Biophysique

- ▶ Prédiction de repliement de protéines [22]

Et un grand besoin en théoriciens aussi !

*"Machine learning has become alchemy - alchemy worked, it helped invented many things. [...] If you are building photo sharing systems, alchemy is OK, but we are beyond that now. We are building systems that govern healthcare and mediate our civic dialogue, we influence elections. I would like to live in a society where systems are built on top of verifiable, rigorous thorough knowledge and not alchemy."*

(Ali Rahimi[18])

- ▶ **Pourquoi** est-ce que l'apprentissage profond fonctionne si bien ? [9]
- ▶ Liens entre l'apprentissage profond et l'intrication quantique [8]
- ▶ Liens entre l'apprentissage profond et la théorie des champs quantiques ? [4]

Pour aller plus loin...

Librairies Python pour l'apprentissage automatique



theano

PYTORCH

The Keras logo features a large red square containing a white stylized letter 'K'. To the right of the square, the word "Keras" is written in a large, bold, black, sans-serif font.

The scikit-learn logo consists of two overlapping circles: a blue circle on the left and an orange circle on the right. The word "scikit" is written in a small, black, sans-serif font inside the blue circle, and the word "learn" is written in a larger, black, sans-serif font inside the orange circle.

## Cours et autres ressources

- ▶ IFT3395/IFT6390 - Fondements de l'apprentissage machine
- ▶ Machine Learning sur Coursera
- ▶ Chaîne Youtube de Hugo Larochelle
- ▶ Chaîne Youtube de Siraj Raval
- ▶ Blog de Andrej Karpathy
- ▶ Blog de Chris Olah

- [1] P. Baldi, P. Sadowski, and D. Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, Nature Communications, 5 (2014), p. 4308.
- [2] B. G. Buchanan, *A (very) brief history of artificial intelligence*, Ai Magazine, 26 (2005), p. 53.
- [3] J. Carrasquilla and R. G. Melko, *Machine learning phases of matter*, Nature Physics, 13 (2017), p. 431.
- [4] R. Fok, A. An, and X. Wang, *Spontaneous Symmetry Breaking in Neural Networks*, ArXiv e-prints, (2017).
- [5] D. George and E. A. Huerta, *Deep Learning for Real-time Gravitational Wave Detection and Parameter Estimation : Results with Advanced LIGO Data*, Phys. Lett., B778 (2018), pp. 64–70.

- [6] D. Harrison and D. L. Rubinfeld, *Hedonic housing prices and the demand for clean air*, Journal of environmental economics and management, 5 (1978), pp. 81–102.
- [7] Y. LeCun, *The mnist database of handwritten digits*, <http://yann.lecun.com/exdb/mnist/>, (1998).
- [8] Y. Levine, D. Yakira, N. Cohen, and A. Shashua, *Deep Learning and Quantum Entanglement : Fundamental Connections with Implications to Network Design*, ArXiv e-prints, (2017).
- [9] H. W. Lin, M. Tegmark, and D. Rolnick, *Why Does Deep and Cheap Learning Work So Well?*, Journal of Statistical Physics, 168 (2017), pp. 1223–1247.

- 
- [10] G. Litjens, T. Kooi, B. Ehteshami Bejnordi, A. Arindra Adiyoso Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez, *A Survey on Deep Learning in Medical Image Analysis*, ArXiv e-prints, (2017).
  - [11] Y. Liu, E. Racah, Prabhat, J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. Wehner, and W. Collins, *Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets*, ArXiv e-prints, (2016).
  - [12] G. Louppe, K. Cho, C. Becot, and K. Cranmer, *QCD-Aware Recursive Neural Networks for Jet Physics*, ArXiv e-prints, (2017).
  - [13] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 5 (1943), pp. 115–133.

- 
- [14] S. McGregor, D. Dhuri, A. Berea, and A. Munoz-Jaramillo, *FlareNet : A Deep Learning Framework for Solar Phenomena Prediction*, in NIPS Workshop on Deep Learning for Physical Sciences, Long Beach, 2017.
  - [15] K. Mills, M. Spanner, and I. Tamblyn, *Deep learning and the Schrödinger equation*, ArXiv e-prints, (2017).
  - [16] H. Moravec, *Mind children : The future of robot and human intelligence*, Harvard University Press, 1988.
  - [17] K. A. Pearson, L. Palafox, and C. A. Griffith, *Searching for Exoplanets using Artificial Intelligence*, in AAS/Division for Planetary Sciences Meeting Abstracts #49, vol. 49 of AAS/Division for Planetary Sciences Meeting Abstracts, Oct. 2017, p. 421.02.

- 
- [18] A. Rahimi, *Nips 2017 test-of-time award presentation.*  
[https://www.youtube.com/watch?v=Qi1Yry33TQE&ab=&ab\\_channel=Alister](https://www.youtube.com/watch?v=Qi1Yry33TQE&ab=&ab_channel=Alister), 2017.
  - [19] F. Rosenblatt, *The perceptron : A probabilistic model for information storage and organization in the brain.*, Psychological review, 65 (1958), p. 386.
  - [20] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*, arXiv preprint arXiv :1712.01815, (2017).
  - [21] A. M. Turing, *Computing machinery and intelligence*, Mind, 59 (1950), pp. 433–460.

- 
- [22] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu, *Accurate de novo prediction of protein contact map by ultra-deep learning model*, PLOS Computational Biology, 13 (2017), pp. 1–34.