APPRENTISSAGE ARTIFICIEL: SÉPARATRICES LINÉAIRES ET RÉSEAUX DE NEURONES

M. Serrurier
IRIT, Toulouse, France

LES DÉBUTS DE L'IA
// APPROCHE LOGIQUE VS APPROCHE CONNEXIONISTE

Acte de naissance : 1956, Darmouth College

Chaque aspect de l'apprentissage, ou toute autre caractéristique de l'intelligence peut en principe être décrit si précisément qu'il est possible de construire une machine pour le simuler

- Approche logique (John McCarthy)
  - Inspirée des travaux de Turing
  - Modéliser le raisonnement par la logique
- Applications : systèmes experts, recherche opérationnelle

- Approche par schéma
  - Mc Culloch
  - Pitts
- Applications : Neurone artificiel
  - Perceptron
  - Perceptron multicouche
  - Deep learning

UN EXEMPLE CLASSIQUE

// IRIS

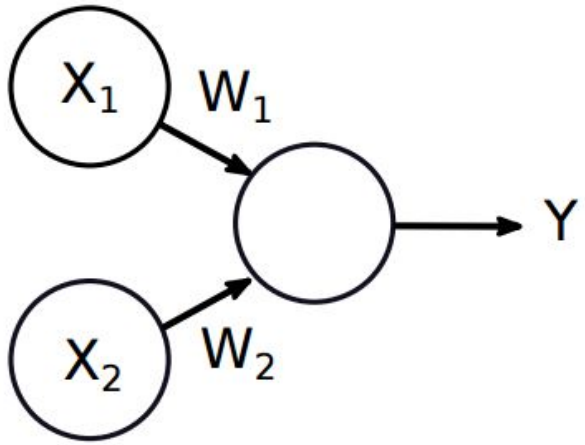
sépal		pétal		
long.	larg.	long.	larg.	type
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
.	.	.	.	.
7.0	3.2	4.7	1.4	versic.
6.4	3.2	4.5	1.5	versic.
.	.	.	.	.
6.3	3.3	6.0	2.5	virgin.
5.8	2.7	5.1	1.9	virgin.
.	.	.	.	.

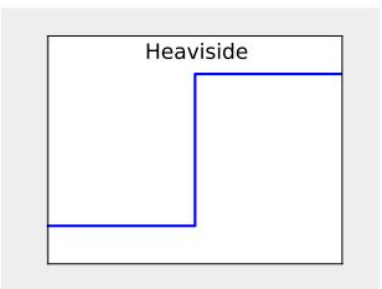
$$f(x) = \alpha \text{sepal.long} + \beta \text{sepal.larg} + \dots$$

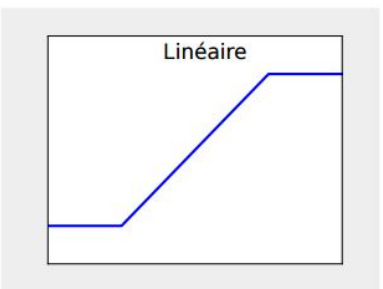
- si $f(x) < 5.1$ alors class=setosa
- si $5.1 \leq f(x) \leq 9.5$ alors class= versicolor
- si $9.5 < f(x)$ alors class= virginica

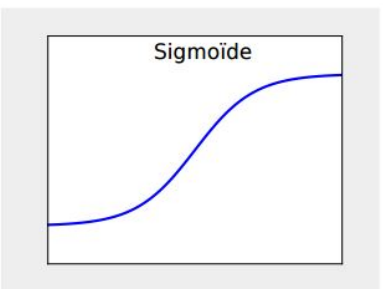
PERCEPTRON
// DEFINITION

- Un neurone possède des entrées
- Chaque entrée possède un poids
- La sortie est une fonction du poids et des entrées
  $Y = f(W_1 * X_1 + W_2 X_2)$



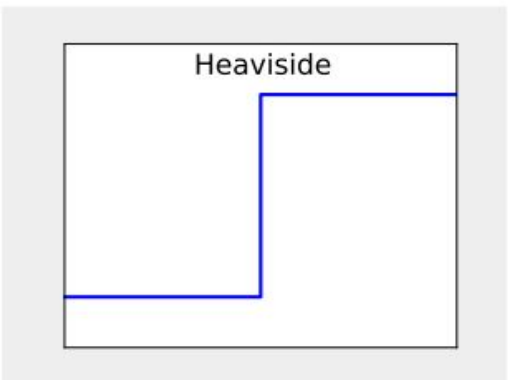


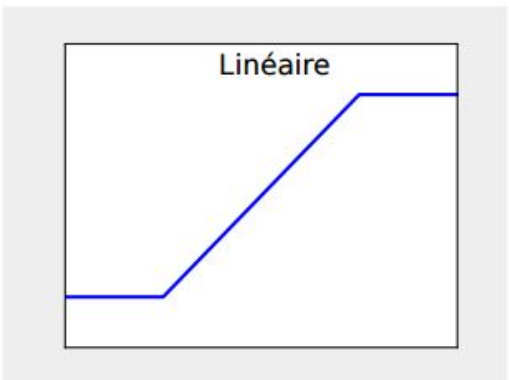


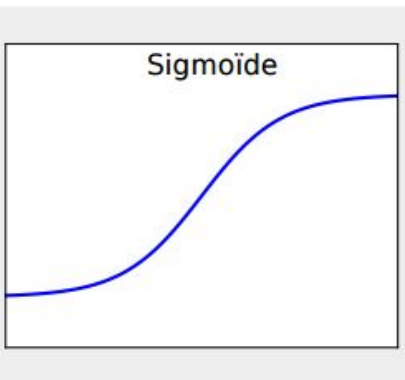


PERCEPTRON
// ACTIVATIONS

- Heaviside (seuil θ)
  - Si $x < \theta$ alors $f(x) = 0$
  - Si $x \geq \theta$ alors $f(x) = 1$
- Linéaire (seuils θ_1, θ_2)
  - Si $x < \theta_1$ ou $x > \theta_2$ alors $f(x) = 0$
  - Sinon $f(x) = x$
- Sigmoïd
  - $f(x) = \frac{1}{1+\exp(-x)}$

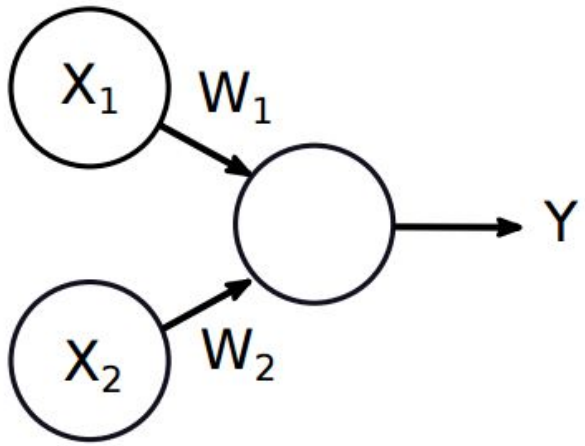






Soit la fonction logique suivante:

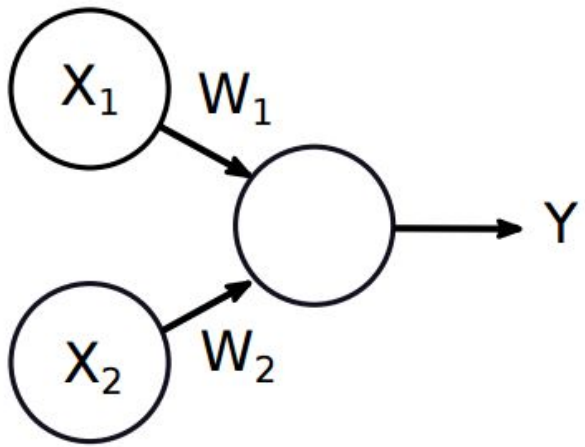
- $f(x_1 = 0, x_2 = 0) = 0$
- $f(x_1 = 1, x_2 = 0) = 1$
- $f(x_1 = 0, x_2 = 1) = 1$
- $f(x_1 = 1, x_2 = 1) = 1$



1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, trouver les poids w_1 et w_2. On prendra la fonction de Heaviside (seuil à fixer) comme fonction de transfert.

Soit la fonction logique suivante:

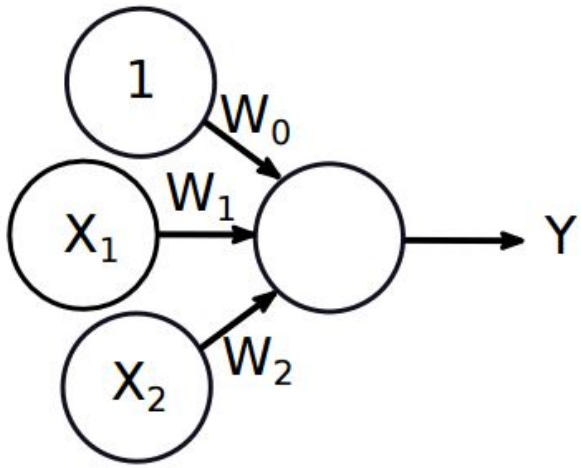
- $f(x_1 = 0, x_2 = 0) = 0$
- $f(x_1 = 1, x_2 = 0) = 0$
- $f(x_1 = 0, x_2 = 1) = 0$
- $f(x_1 = 1, x_2 = 1) = 1$



1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, essayer de trouver les poids w_1 et w_2. On prendra la fonction de Heaviside comme fonction de transfert (seuil $= 0$).

Soit la fonction logique suivante:

- $f(x_1 = 0, x_2 = 0) = 0$
- $f(x_1 = 1, x_2 = 0) = 0$
- $f(x_1 = 0, x_2 = 1) = 0$
- $f(x_1 = 1, x_2 = 1) = 1$

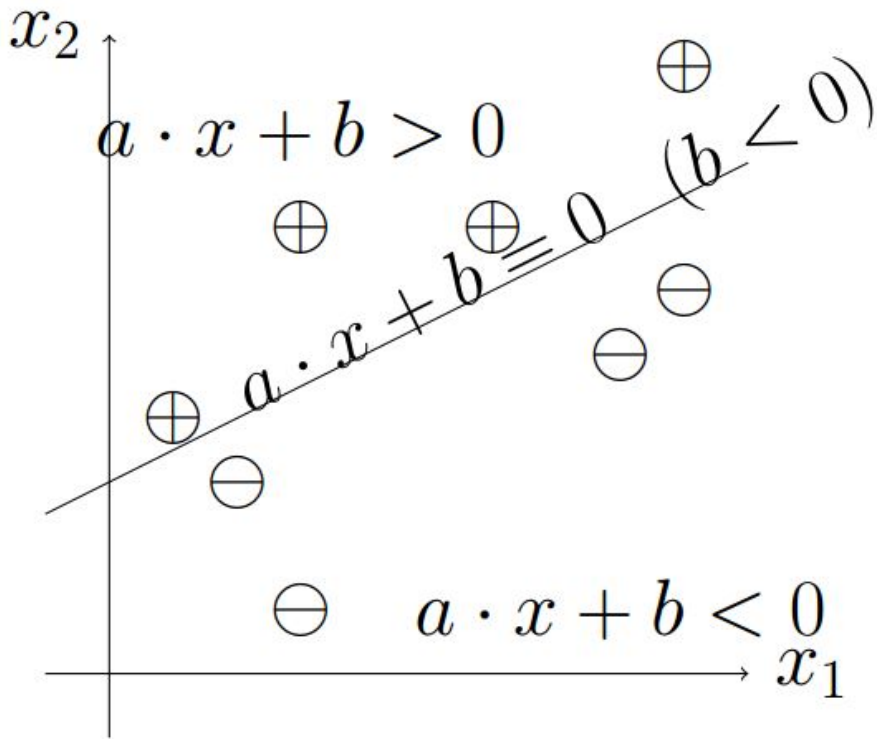


1. Essayer avec cette architecture.

PERCEPTRON
// INTERPRÉTATION

Le perceptron (Heaviside seuil=0) est une séparatrice linéaire qui a pour équation

$f(x) = w_1 * x1 + w2 * x2 + w0 = 0$

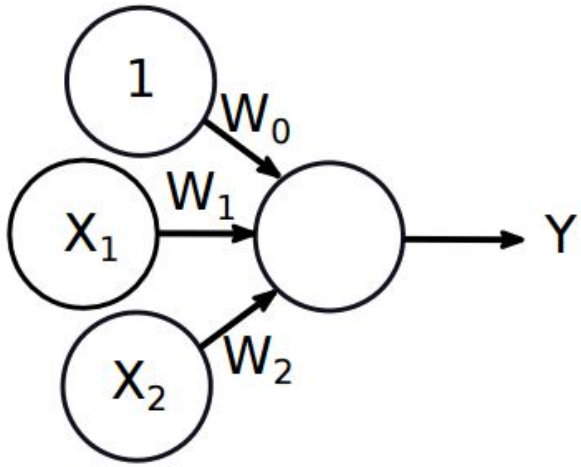


On veut séparer des exemples (x, u) avec $x = (x_1, \ldots, x_n)$ et $u \in \{-1, 1\}$. Le perceptron f est une séparatrice linéaire ssi:

$$\forall (x, u)(w_0 + \sum_{1}^{n} w_i * x_i) * u > 0$$

Soit la fonction logique suivante:

- $f(x_1 = 0, x_2 = 0) = 0$
- $f(x_1 = 1, x_2 = 0) = 0$
- $f(x_1 = 0, x_2 = 1) = 0$
- $f(x_1 = 1, x_2 = 1) = 1$



1. Tracer les points et la droite représentant le perceptron trouvé précédement
2. Verifier que c'est bien une séparatrice linéaire (avec 0 correspondant au -1)

perceptron(E)

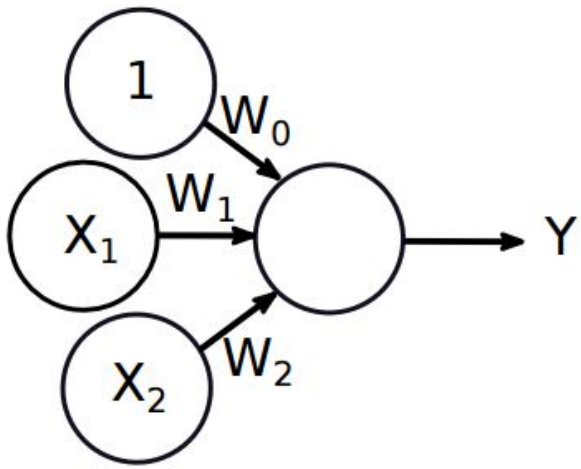
1. $w = (0, \ldots, 0)$; $w_0 = 0$; pas_fini = vrai;
2. Tant que pas_fini faire
   - 2.1 pas_fini = faux;
   - 2.2 pour chaque $(x, u) \in \mathcal{E}$, si $(w \cdot x + b) \times u \leq 0$ alors
     $w_i = w_i + \tau \times u \times x_i$;
     $w_0 = w_0 + \tau \times u$;
     pas_fini = vrai;
3. Retourner (w, w_0).

($\tau > 0$ permet de régler la vitesse de convergence)

APPRENTISSAGE PERCEPTRON
// EXERCICE 1

Soit la fonction logique suivante:

- $f(x_1 = 0, x_2 = 0) = -$
- $f(x_1 = 1, x_2 = 0) = +$
- $f(x_1 = 0, x_2 = 1) = +$
- $f(x_1 = 1, x_2 = 1) = +$

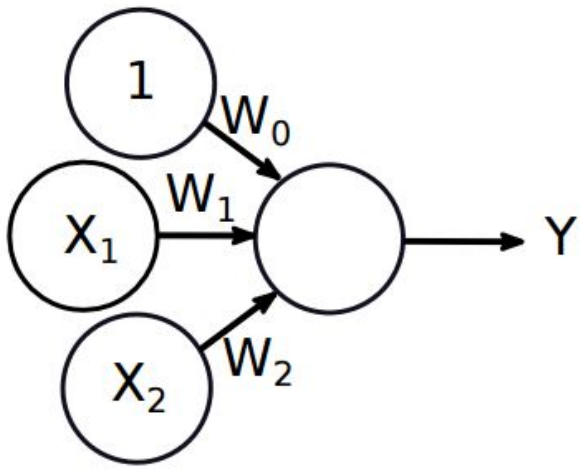


1. Utiliser l'algorithme du perceptron pour apprendre les valeurs du réseau $\tau = 0.5$

APPRENTISSAGE PERCEPTRON
// EXERCICE 2

Soit la fonction logique suivante:

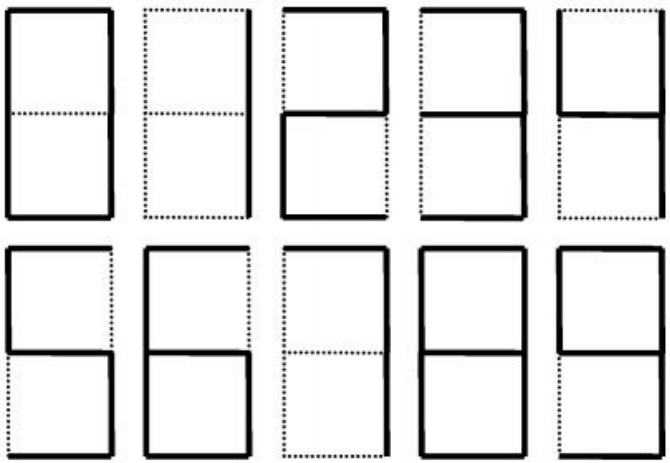
- $f(x_1 = 0, x_2 = 0) = -$
- $f(x_1 = 1, x_2 = 0) = -$
- $f(x_1 = 0, x_2 = 1) = -$
- $f(x_1 = 1, x_2 = 1) = +$



1. Utiliser l'algorithme du perceptron pour apprendre les valeurs du réseau $\tau = 0.5$.

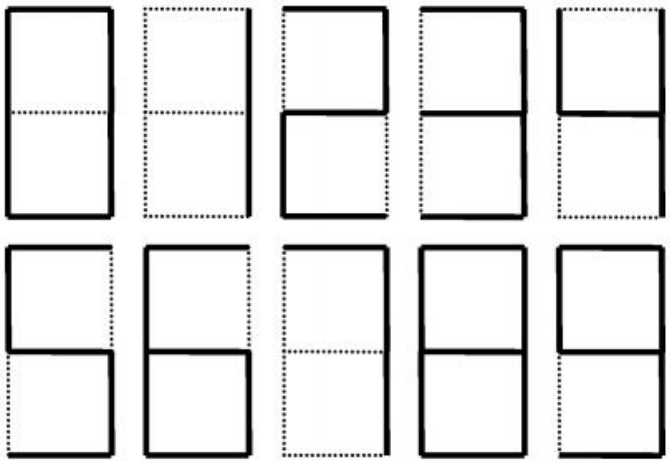
APPRENTISAGE PERCEPTRON

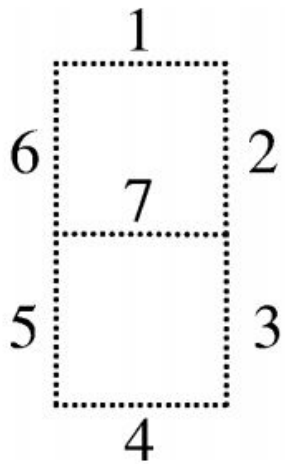
// PARITÉ



1. Proposer un codage binaire de chaque chiffre
2. Trouver les poids permettant de décider si le chiffre est pair ou non

APPRENTISAGE PERCEPTRON
// PARITÉ

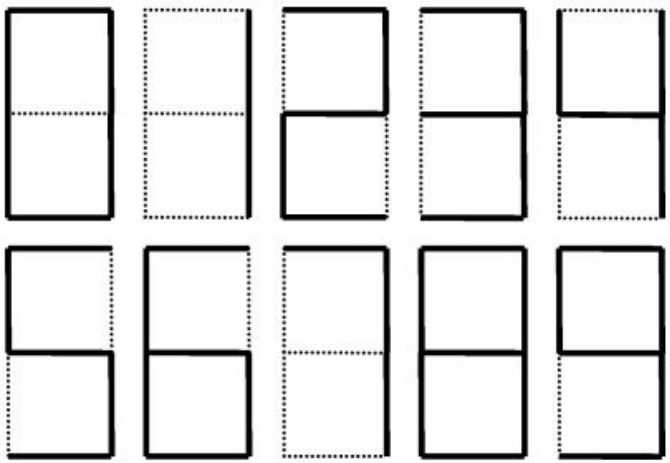


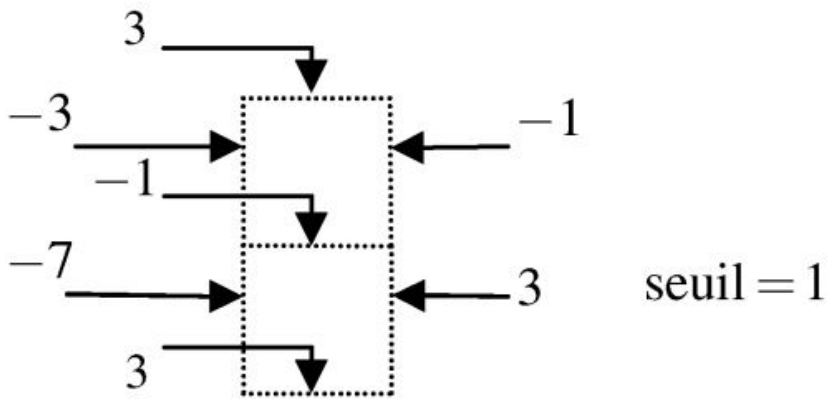


0 se code 1111110,
1 se code 0110000, e

1. Proposer un codage binaire de chaque chiffre
2. Trouver les poids permettant de décider si le chiffre est pair ou non

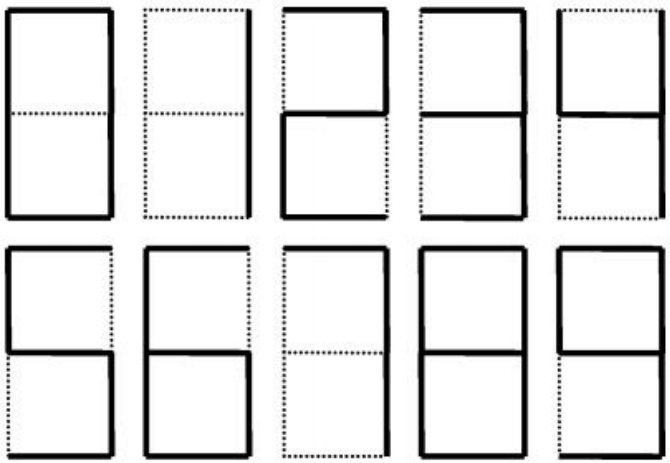
APPRENTISAGE PERCEPTRON
// PARITÉ





1. Proposer un codage binaire de chaque chiffre
2. Trouver les poids permettant de décider si le chiffre est pair ou non

APPRENTISAGE PERCEPTRON
// RECONNAISSANCE



1. Proposer une architecture permettant de reconnaître le chiffre
2. Trouver les poids correspondants

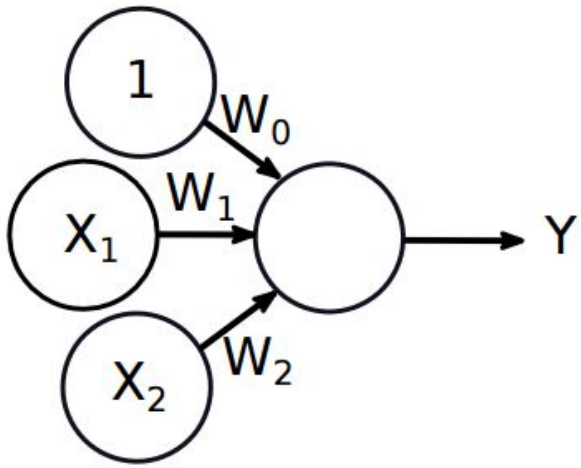
ALGORITHME PERCEPTRON
// PROPRIÉTÉS

- algorithme incrémental : il suffit de le relancer si de nouveaux exemples sont ajoutés.
- convergence garantie si $\mathcal{E}$ est linéairement séparable, c'est-à-dire s'il existe une hypothèse linéaire qui permet de classer $\mathcal{E}$;
- ne converge pas dans le cas contraire.

APPRENTISSAGE PERCEPTRON
// EXERCICE 4

Soit la fonction logique suivante:

- $f(x_1 = 0, x_2 = 0) = -$
- $f(x_1 = 1, x_2 = 0) = +$
- $f(x_1 = 0, x_2 = 1) = +$
- $f(x_1 = 1, x_2 = 1) = -$



1. Est-il possible de trouver des poids pour ce problème ? justifier

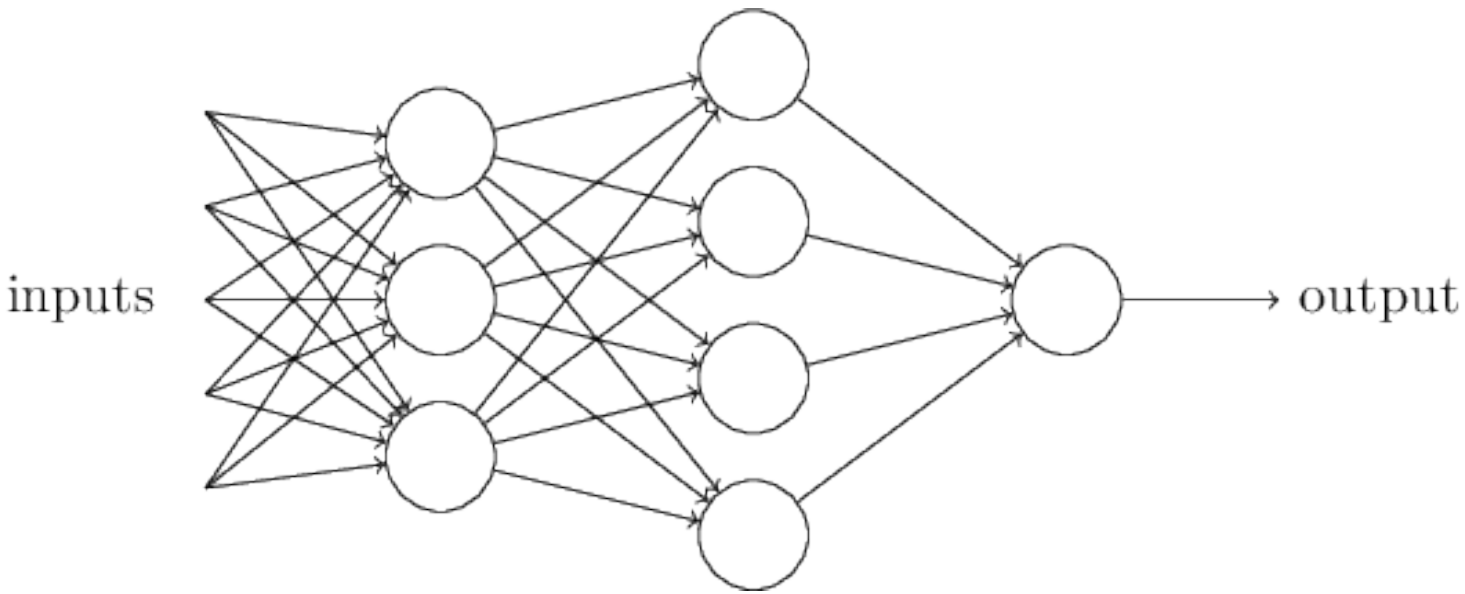
PERCEPTRON SIMPLE

limites

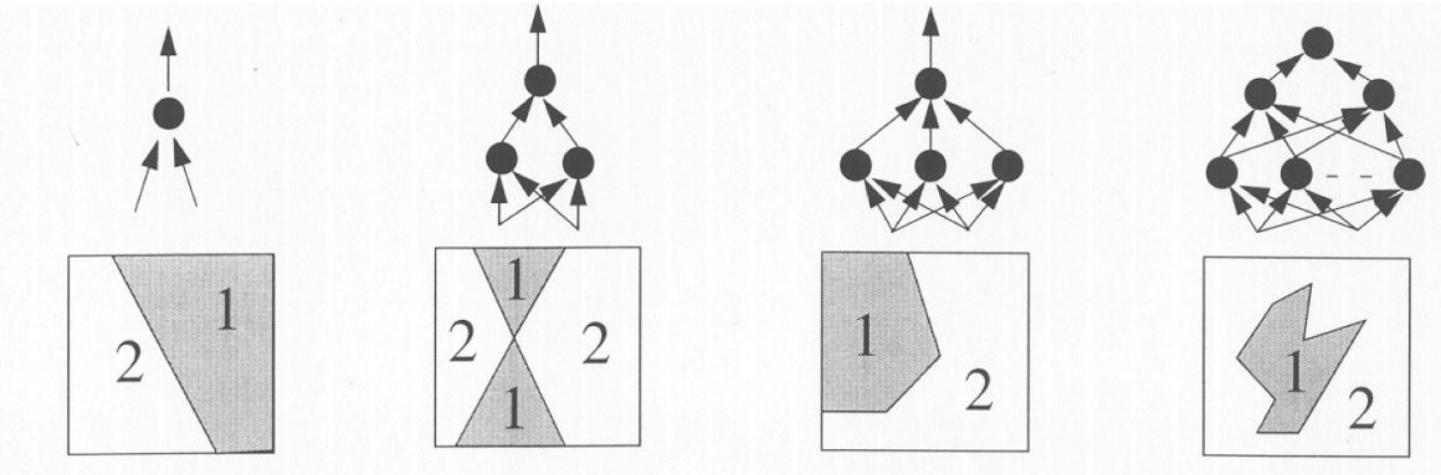
Le perceptron simple ne peut résoudre que des problèmes linéairement séparables. Pour aller plus loin, il est nécessaire d'ajouter des couches.

PERCEPTRON MULTICOUCHE

// PRÉSENTATION



Pouvoir séparateur



Propriété

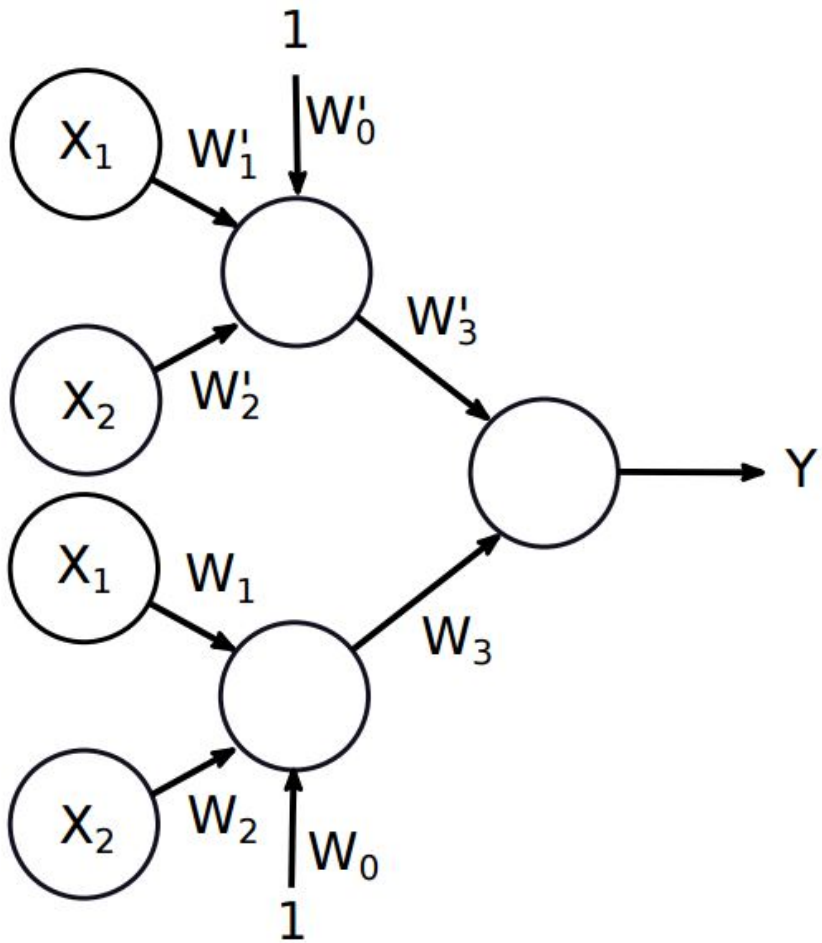
L'augmentation du nombre de couches et du nombre de neurones accroît le pouvoir de séparation

PERCEPTRON MULTICOUCHE

// EXERCICE : OU EXCLUSIF

On peut remarquer que :

$$A \oplus B = (A \vee B) \wedge \neg(A \wedge B).$$



En combinant les deux réseaux (OU et ET), réaliser la fonction ou exclusif

Rétropropagation du gradient

Le problème de l'apprentissage dans les perceptrons multi-couches est de connaître la contribution de chaque poids dans l'erreur globale du réseau. L'algorithme de rétro-propagation de l'erreur permet de faire cela.

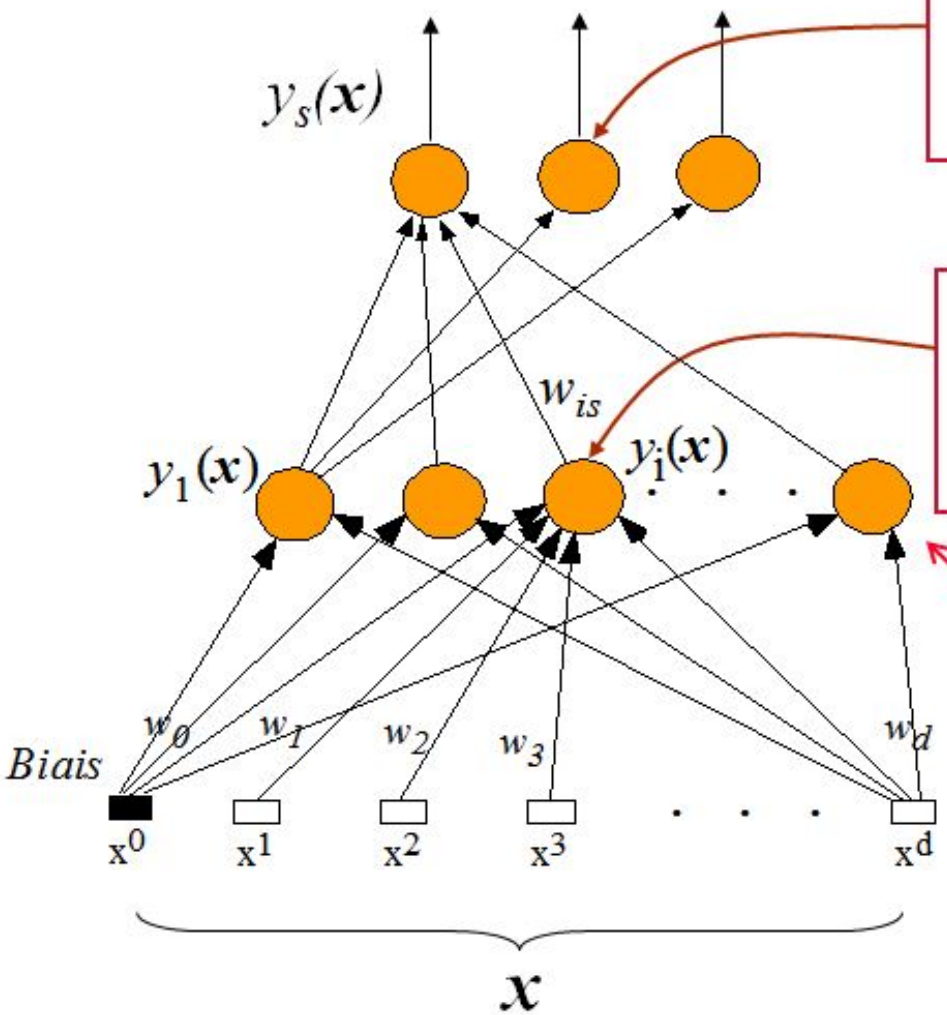
1. Propagation de l'entrée jusqu'à la sortie
2. Calcul de l'erreur en sortie
3. Rétro-propagation de l'erreur jusqu'aux entrées

Conditions

Il faut une fonction d'activation dérivable car on a besoin de la dérivé pour rétro-propager l'erreur.

PERCEPTRON MULTICOUCHE

// INFÉRENCE



$$y_s(x) = \sum_{j=1}^{k} w_{js} y_j$$

$$a_i(x) = \sum_{j=1}^{d} w_j x^j + w_0$$

$$y_i(x) = g(a_i(x))$$

k neurones sur la couche cachée

objectif : minimiser

$$w^* = \text{argmin}_w \frac{1}{m} \sum_{l=1}^{m} [y(x_l, w) - u(x_l)]^2$$

- On utilise la retro-propagation du gradient :

$$w_{ij}^t = w_{ij}^{t-1} - \eta \delta_j$$

- avec
  - les cellules de la couche de sortie

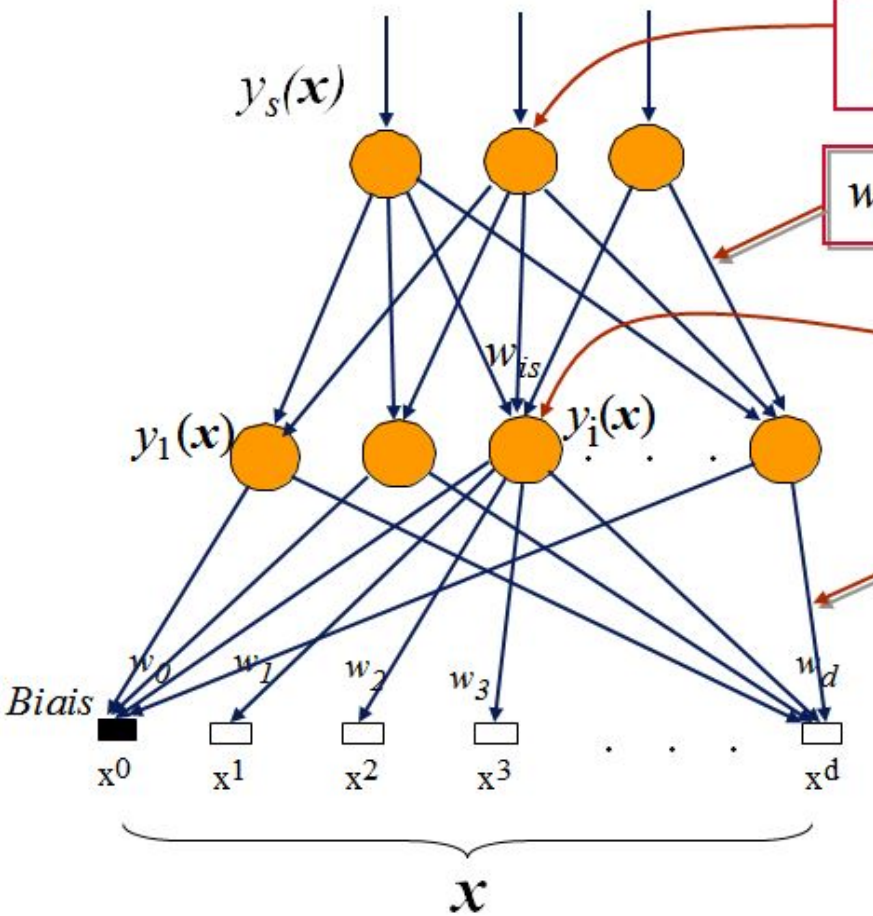
$$\delta_j = g'(a_j).(u_j(x_l) - y_j)$$

  - les cellules des couches intermédiaires

$$\delta_j = g'(a_j). \sum_k w_{jk} \delta_k$$

PERCEPTRON MULTICOUCHE

// RÉTRO-PROPAGATION DU GRADIENT



$$\delta_s = g'(a_s)(u_s - y_s)$$

$$w_{is}(t+1) = w_{is}(t) - \eta(t)\delta_s a_i$$

$$\delta_j = g'(a_j) \sum_{\text{cellules s couche suivante}} w_{js}\delta_s$$

$$w_{ei}(t+1) = w_{ei}(t) - \eta(t)\delta_i a_e$$

PERCEPTRON MULTICOUCHE
// ATTRAIT ET LIMITES

- Les attraits pratiques
  - D'emploi très général (en fonction de la structure du réseau) :
    - Classification
    - Régression
    - Transfert
    - Renforcement
  - Algorithme simple
  - Algorithme incrémental et parallélisable
- Limites
  - Opacité des "raisonnements" et des résultats
  - Difficile a apprendre avec des données complexes
  - Tendance au sur-apprentissage

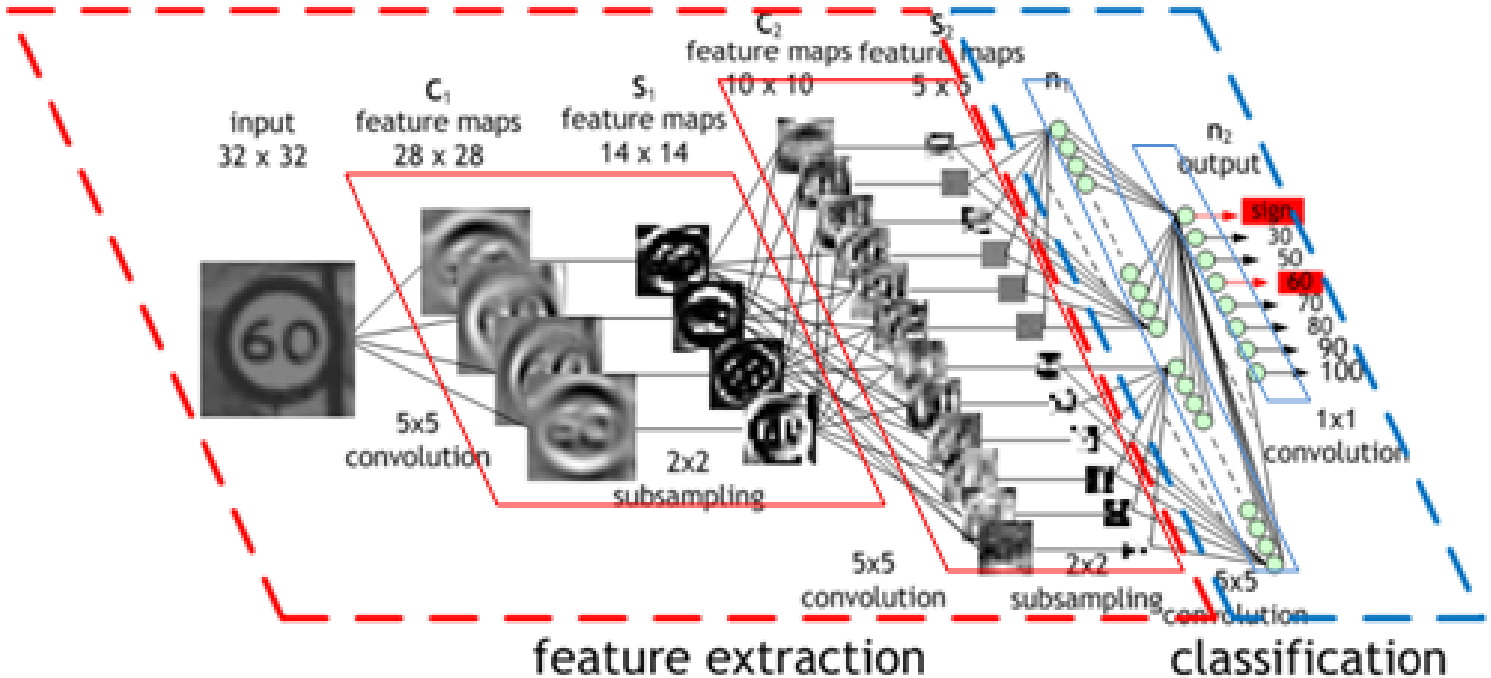
Autour des années 2010 : grand boum de l'apprentissage des réseaux de neurones : deep learning

- Causes :
  - Nouveaux algorithmes pour apprendre des réseaux complexes
    - Drop out
    - Mini-batch
    - Initialisation
  - Données disponibles en masse
  - Capacités de calcul accrues (cluster, GPU)

LA RECONNAISSANCE D'IMAGE
// PRINCIPE

- Première application industrielle des réseaux de neurones
- Première preuve d'efficacité du deep-learning
- Problème de classification :
  - couche d'entrées : pixel de l'image
  - Un neurone de sortie par classe
  - Utilisation d'un réseau de neurones particulier : le convolutional neural networks (CNN)
- Exemple classique : la base Mnist

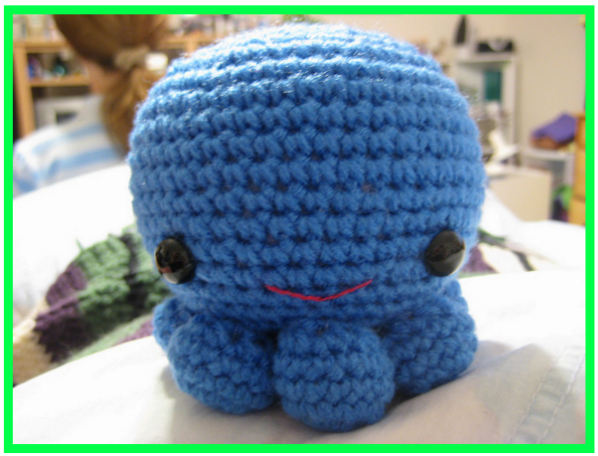
RECONNAISSANCE D'IMAGE
// CONVOLUTIONAL NEURAL NETWORKS



RECONNAISSANCE D'IMAGE
// CLASSIFICATION SIMPLE

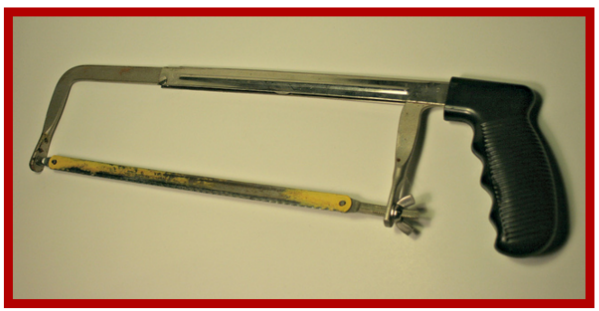
Your specialized huggable recommendation

Go for it! Hug it out. With a score of **0.695779**, we're somewhat sure.



Your specialized huggable recommendation

Don't hug that. Please. With a score of **0.999875**, we're pretty sure.



Your specialized huggable recommendation

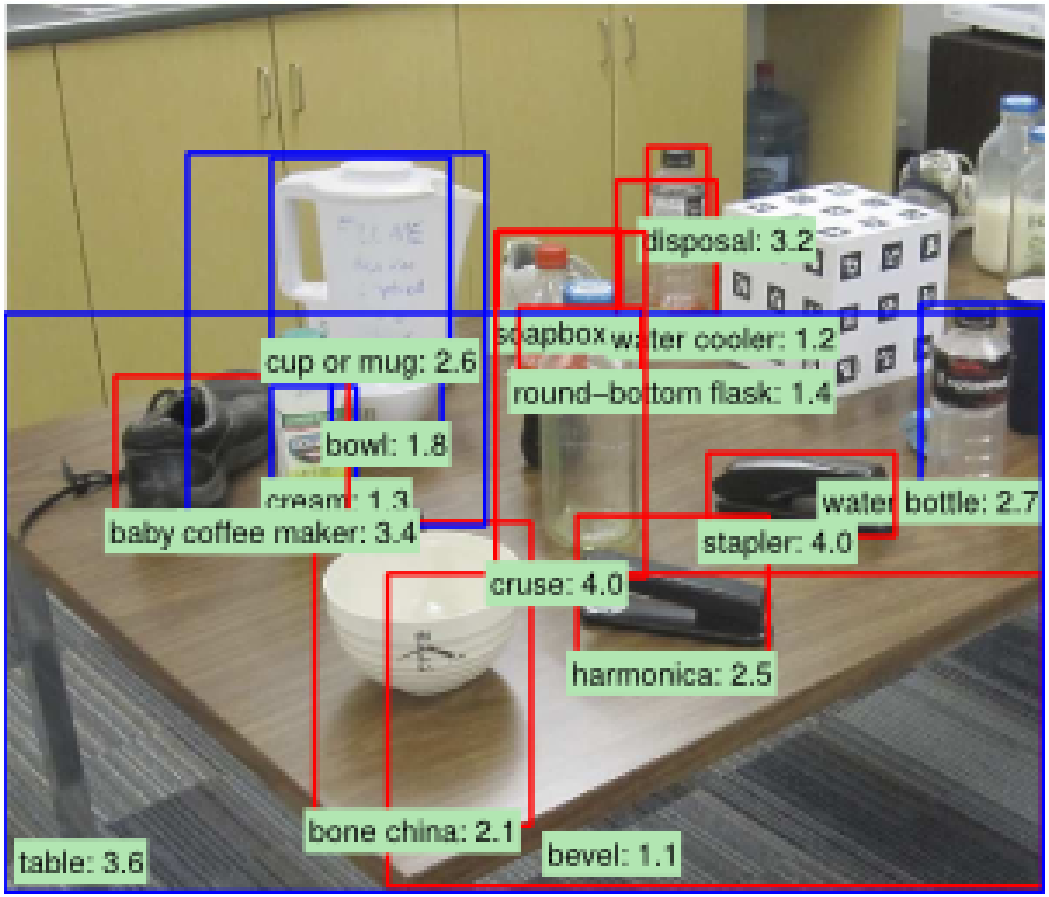
Don't hug that. Please. With a score of **0.778686**, we're pretty sure.

Your specialized huggable recommendation

Go for it! Hug it out. With a score of **0.958104**, we're pretty sure.

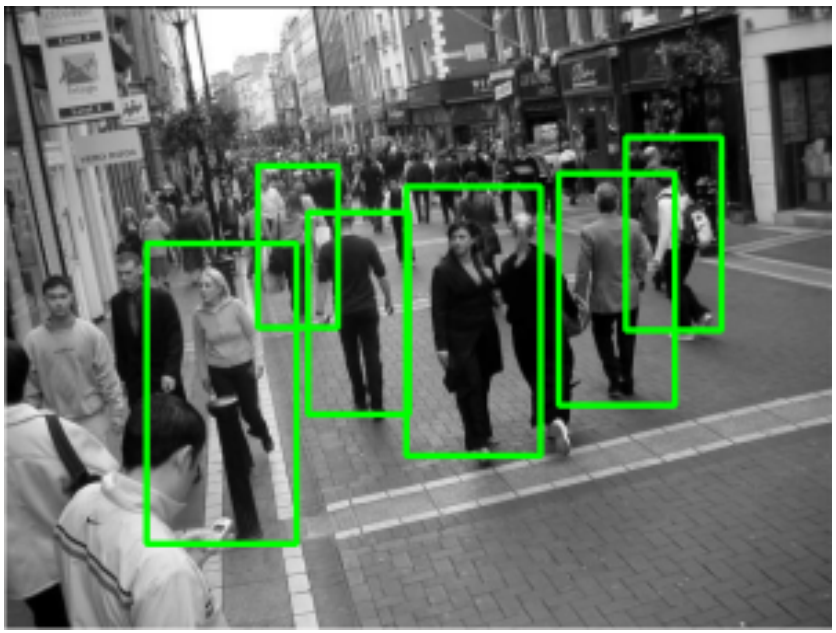
RECONNAISSANCE D'IMAGE

// CLASSIFICATION MULTI LABEL

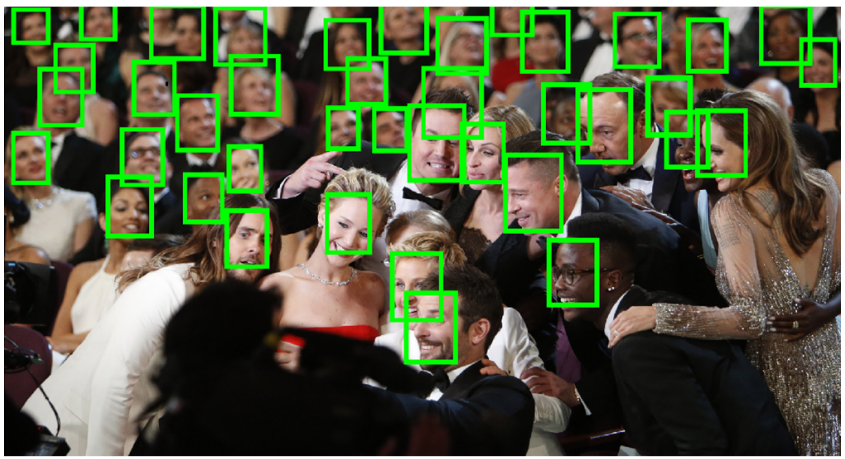


RECONNAISSANCE D'IMAGE
// RECONNAISSANCE DE PATTERNS

Caméra de sécurité

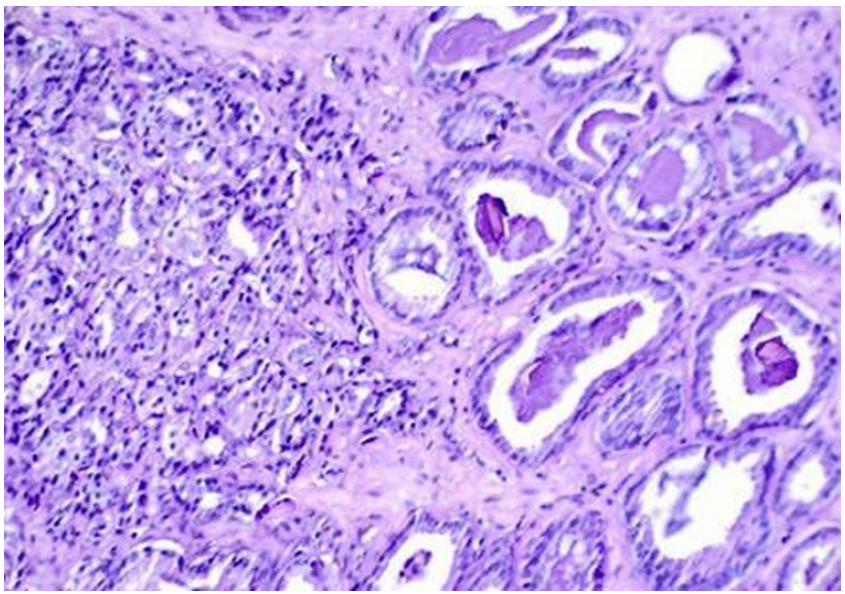


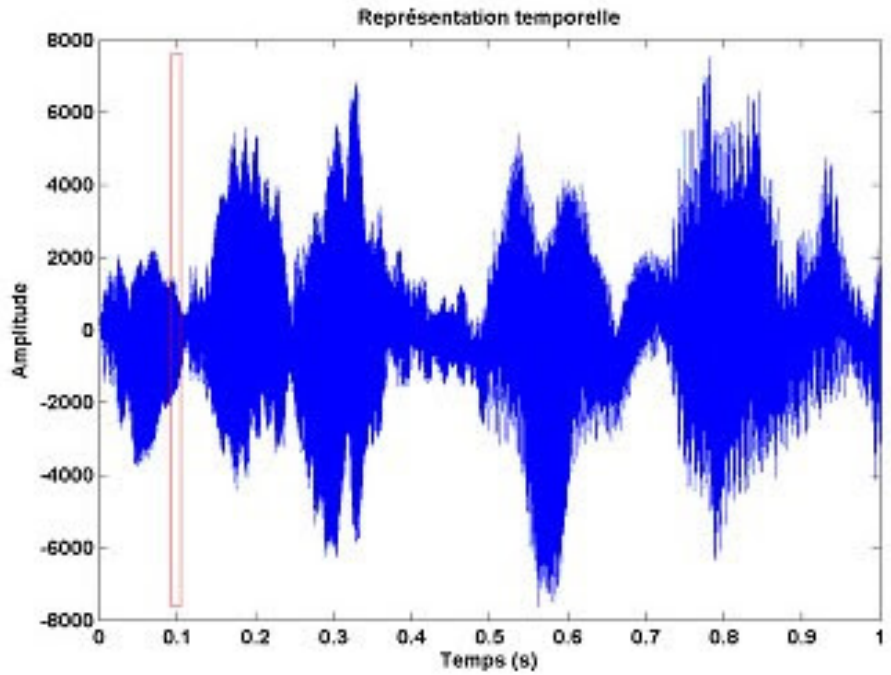
Facebook



RECONNAISSANCE D'IMAGE
// QUELQUES PROJETS

INSERM : detection et ratio de cellules cancéreuses





Classification de signaux bruités

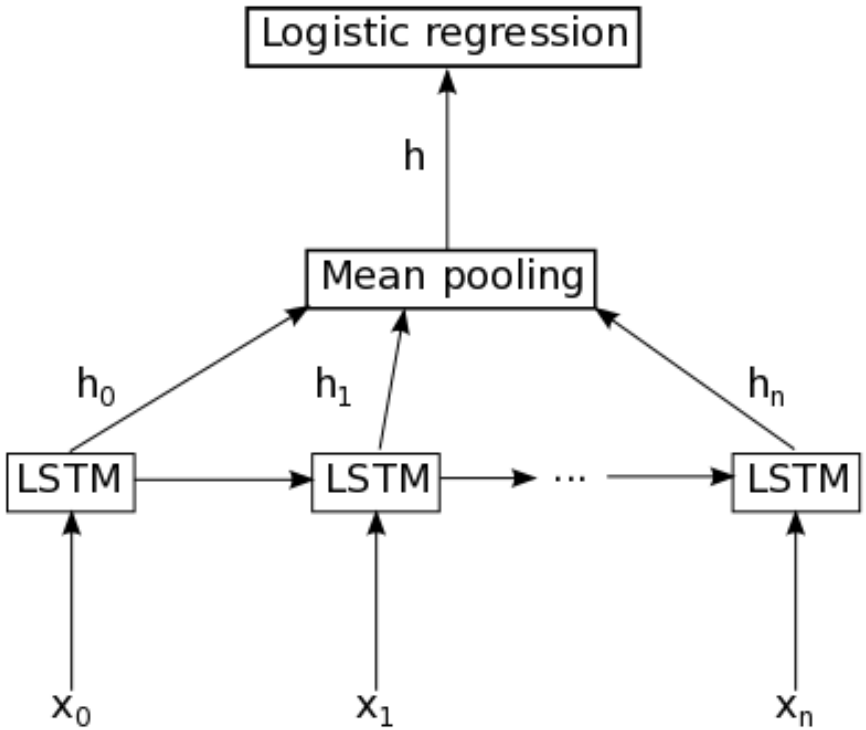
LA CLASSIFICATION DE TEXTE
// PRINCIPE

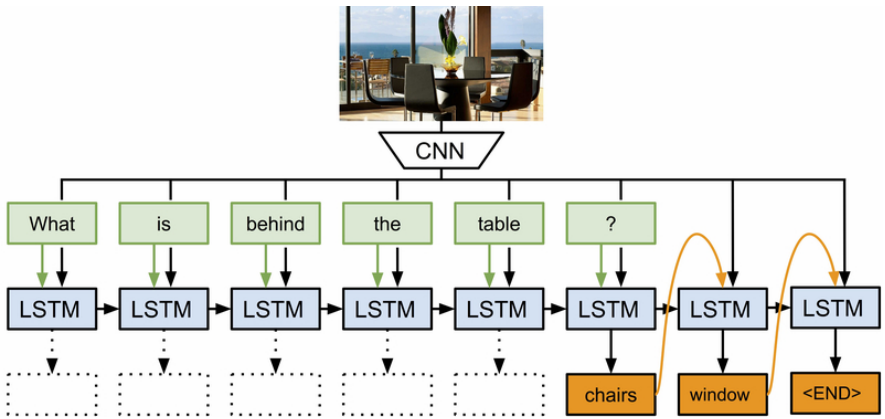
- Exemples :
  - Google actualité
  - Tendance sur Twitter
- Un problème de classification deux façons de considérer les textes :
  - Bag of of words (tout le texte d'un coup)
  - Mot par mot

Approche mot par mot

- Vecteur d'entrées = dictionnaire de la langue (données sparses)
- Nécessite un nouveau type de réseau : le réseau de neurones récurent (ou LSTM)

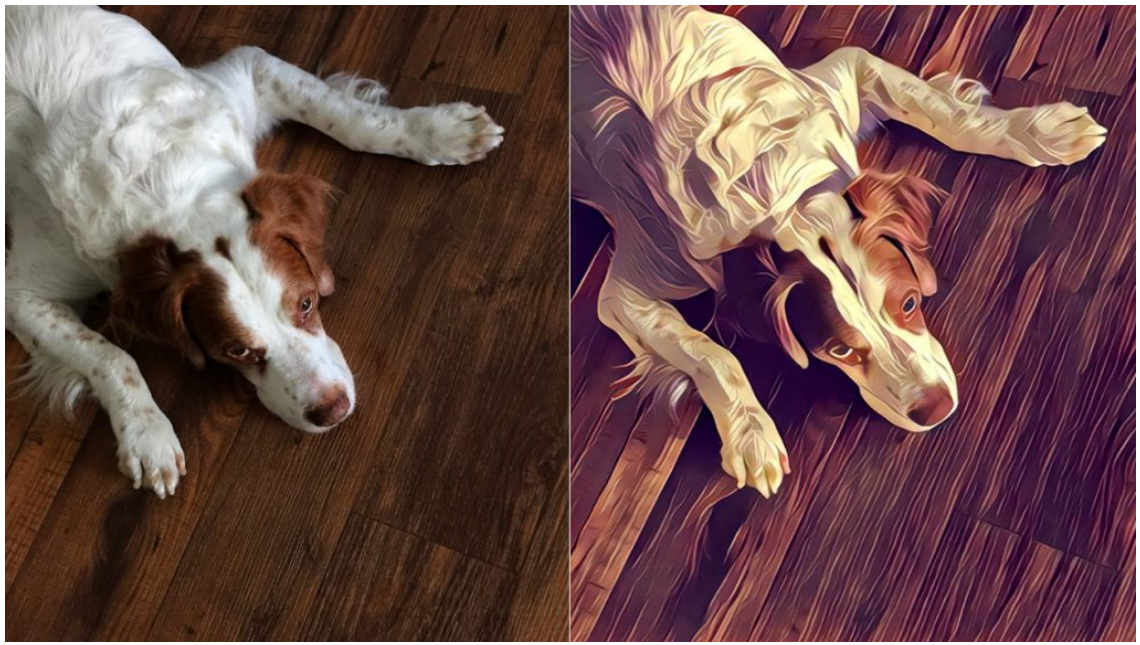
LA CLASSIFICATION DE TEXTE
// LSTM





APPRENTISSAGE DE FONCTIONS DE TRANSFERT

// PRINCIPE



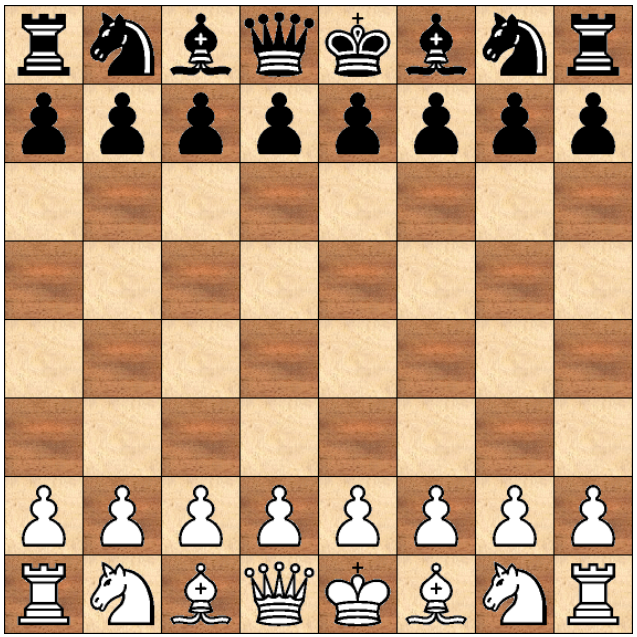
- Application :
  - Traduction / filtrage d'image (nouvel algorithme de google trad)
  - Primsa
- Problème de classification :
  - Couche d'entrée : pixel de l'image (ou mots du texte)
  - Autant de neurones en sortie qu'en entrée

APPRENTISSAGE DE FONCTIONS DE TRANSFERT

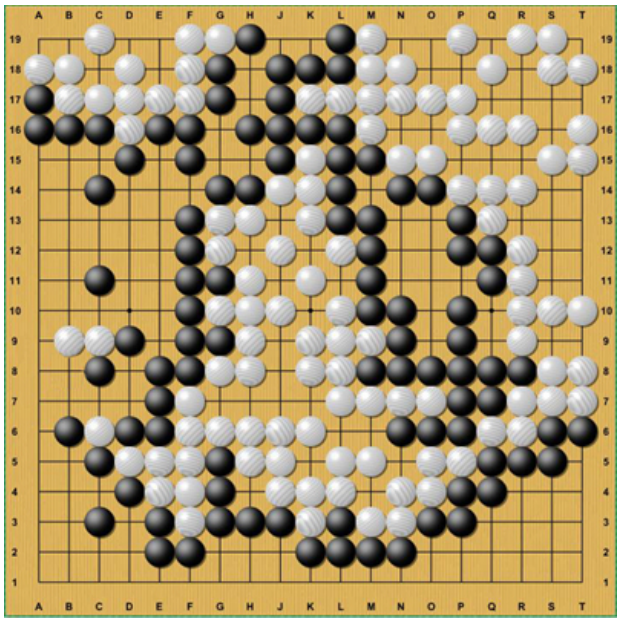
// EXEMPLE

L'APPRENTISSAGE PAR RENFORCEMENT

// ALPHA-GO VS DEEP BLUE



10^{120} possibilités



10^{761} possibilités

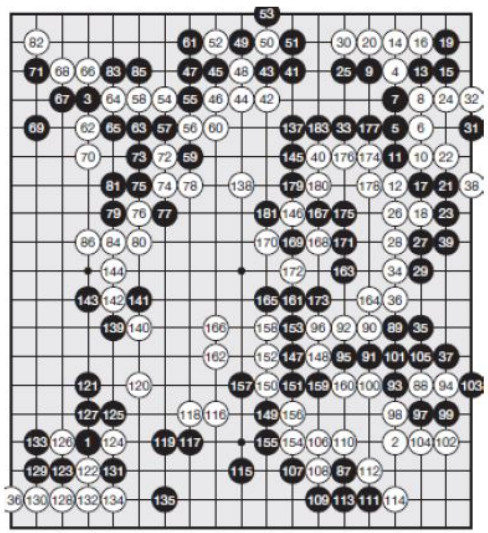
Problème très difficile

- Force brute impossible : approche deep blue ne marche pas
- Q learning classique impossible : espace trop grand

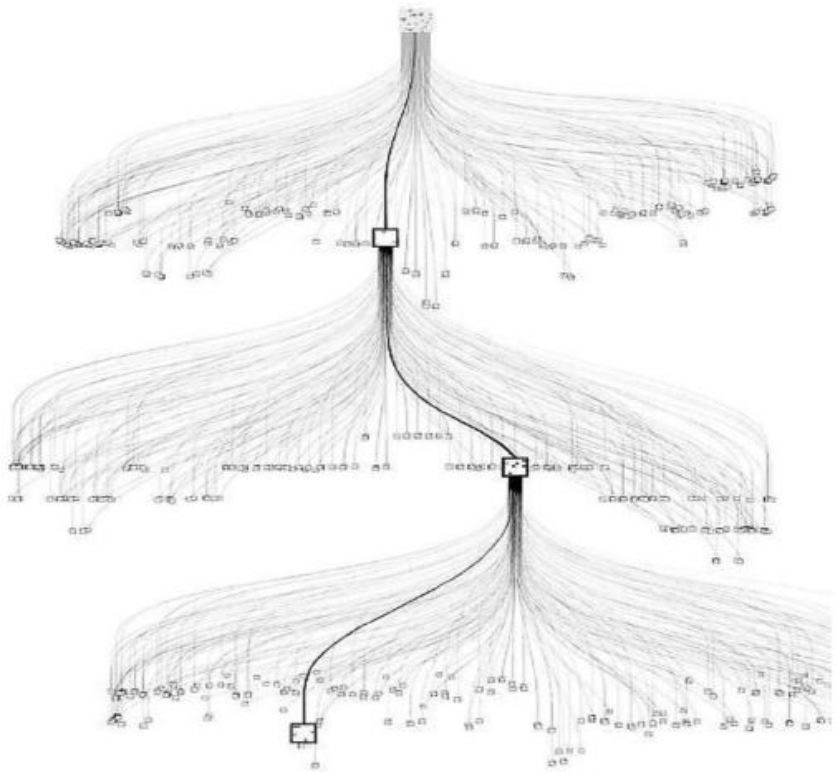
L'APPRENTISSAGE PAR RENFORCEMENT

// ALPHA GO

AI: AlphaGo



L'APPRENTISSAGE PAR RENFORCEMENT
// ALPHA-GO



- Considère le plateau comme une image
- Deux phases :
  - Entrainement du réseau à partir d'une base de partie
  - Entrainement du réseau face à lui même
- Raisonnement proche de l'humain (à débattre)

Prochain défi

Les jeux en temps réel (ex: Starcraft) en utilisant seulement les informations disponibles à l'écran.

DEEP LEARNING
// CONCLUSIONS

- Deep learning : une technologie mature
  - Algorithmes publiques
  - Infrastructures hardware

- Deep learning : une technologie efficace
  - Algorithmes très performant
  - Au coeur de l'industrie des données
- Mais :
  - Pas une baguette magique
  - Nécessite une culture du stockage des données et de l'expertise
  - Manque de garantie -> problèmes éthiques et juridiques


DEEP LEARNING...
DEEP LEARNING EVERYWHERE
memegenerator.net