

Master Humanités Numériques

Machine Learning pour les données textuelles
Apprentissage automatique pour le texte

Julien Velcin
Laboratoire ERIC – Université Lyon 2
<http://eric.univ-lyon2.fr/jvelcin>

Apprentissage automatique pour le texte

RAPPEL SUR L'APPRENTISSAGE AUTOMATIQUE

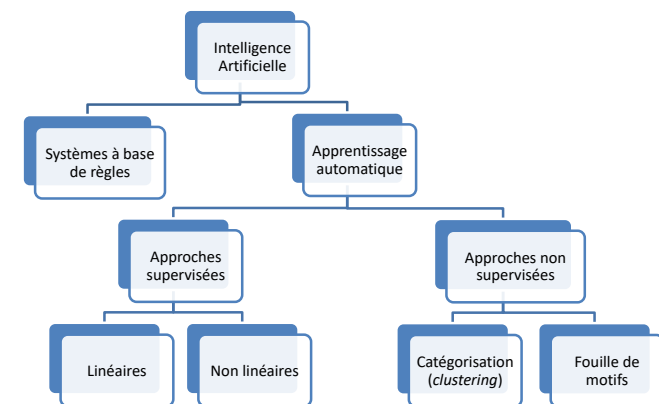
2

Objectifs du « machine learning »

- L'objectif principal consiste à apprendre automatiquement à **généraliser** à partir d'exemples observés afin de pouvoir faire de l'inférence sur de **nouveaux exemples** jamais été observé auparavant (principe inductif, opposé à celui de déduction)
- Différentes familles d'algorithmes :
 - apprentissage par cœur (pas de généralisation)
 - apprentissage par cas (faibles capacités de généralisation)
 - apprentissage par renforcement (principe essais-erreurs)
 - classification non supervisée / catégorisation (*clustering*)
 - classification supervisée, régression
 - algorithmes génératifs (apprendre à générer des données)

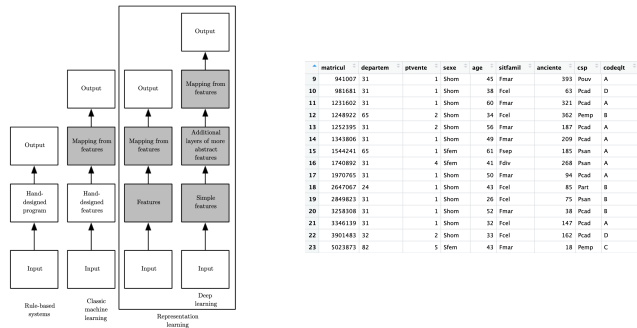
3

Positionnement vis-à-vis de l'IA



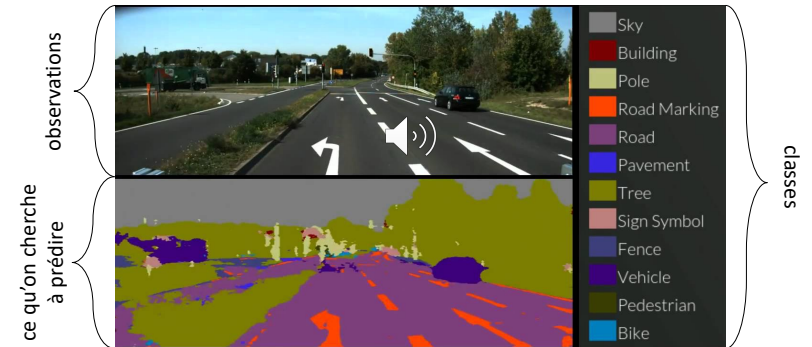
4

Apprentissage de représentations



	municipal	departem	premiere	sexe	age	sitfamit	anciennete	csp	codeqch
9	941007	31	1	Shom	45	Fmar	393	Pouv	A
10	981881	31	1	Shom	38	Fcol	63	Poad	D
11	1211662	31	1	Shom	60	Fmar	211	Poad	A
12	1248022	65	2	Shom	94	Fcol	362	Pemp	B
13	1252395	31	2	Shom	56	Fmar	187	Poad	A
14	1343806	31	1	Shom	49	Fmar	209	Poad	A
15	1544241	45	1	Shom	61	Fmar	185	Poad	A
16	1740862	31	4	Shom	41	Fcol	268	Poad	A
17	1970765	31	1	Shom	50	Fmar	94	Poad	A
18	2647987	24	1	Shom	43	Fcol	85	Poad	B
19	2698023	31	1	Shom	26	Fcol	75	Poad	B
20	3258058	31	1	Shom	52	Fmar	38	Poad	B
21	3346139	31	1	Shom	32	Fcol	147	Poad	A
22	3901483	32	2	Shom	33	Fcol	162	Poad	D
23	5023879	82	5	Shom	43	Fmar	18	Pemp	C

Exemple de la classification pour la reconnaissance d'objets dans des images

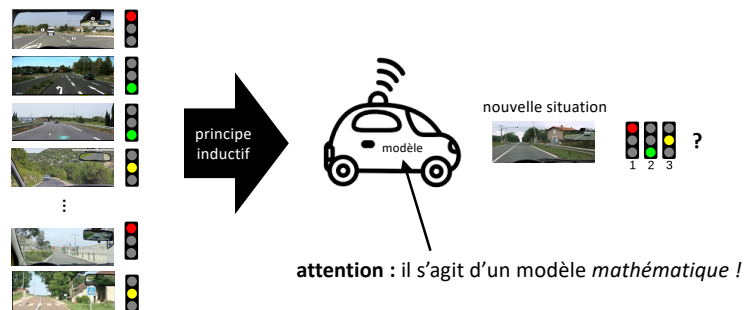


Deep Learning, by I. Goodfellow, Y. Bengio and A. Courville, MIT Press, 2016: <http://www.deeplearningbook.org>

5

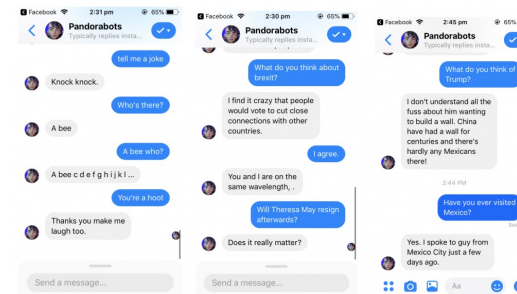
6

L'apprentissage automatique (ici, classification)



7

Robots conversationnels

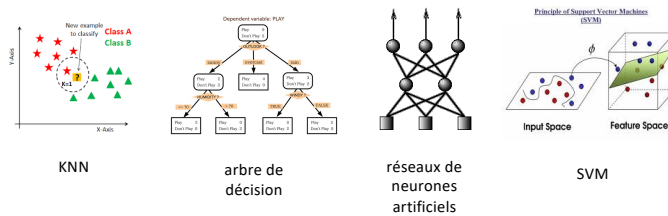


<https://www.pandorabots.com/mitsuku/>

8

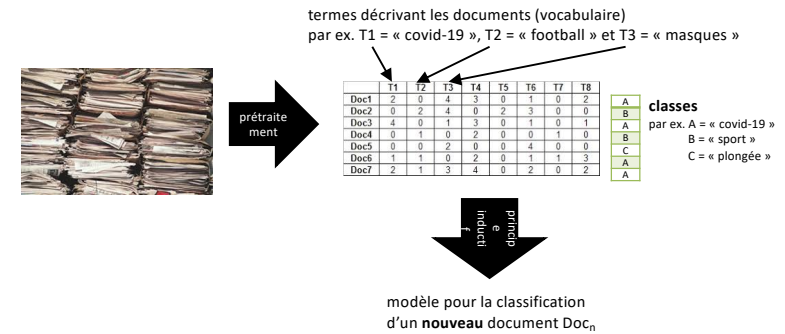
Classer des documents

- Par exemple détecter la **polarité** d'un tweet ou classer un article dans une **thématique** déterminée (ex. sport ou économie)
- De (très) nombreux algorithmes existent :



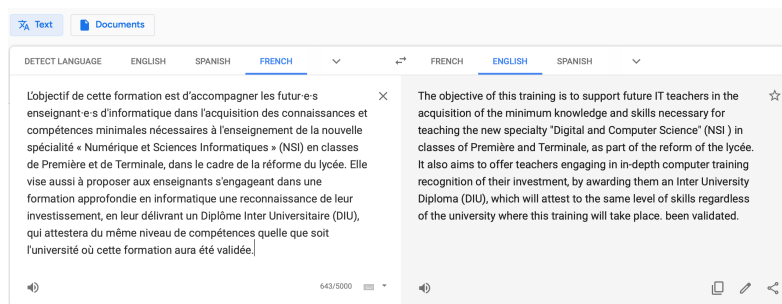
9

Procédure de traitement des données



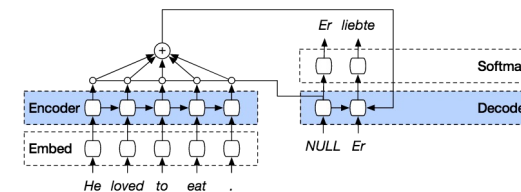
10

Traduire une langue en une autre



11

Réseaux de neurones profonds (deep learning)



source: https://smerity.com/articles/2016/google_nmt_arch.html

12

Quelques précautions à prendre

- Bien définir la tâche qu'on souhaite résoudre
- Identifier les données qui vont permettre à la machine d'apprendre (attention aux biais !)
- Préparer les données à l'apprentissage
- Apprendre à bien généraliser : méthodologie de l'apprentissage automatique et sur-apprentissage
- Souvent plusieurs critères : précision des résultats, interprétabilité, consommation et empreinte écologique...

13

Un panorama très riche

- Modèles simples basés sur le BoW
 - Multi-layer Perceptron (MLP)
- Modèles séquentiels avec mémoire
 - Recurrent Neural Network (RNN)
 - Long Short-Term Memory (LSTM)
- Modèles basés sur l'attention
 - Bidirectional Encoder Representations from Transformers (BERT)

15

Apprentissage automatique pour le texte

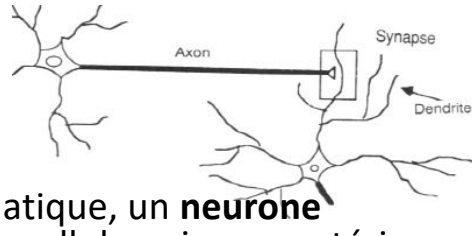
RÉSEAUX DE NEURONES ARTIFICIELS

14

Réseaux de Neurones (RN) (*Artificial Neuronal Networks -ANN-*)

- Les principaux réseaux se distinguent par l'organisation du graphe
 - = architecture ou topologie du RN :
 - o en couches
 - o complet...
- Complexité du RN :
 - o nombre de neurones
- Le type des neurones :
 - o leurs fonctions de transition

16

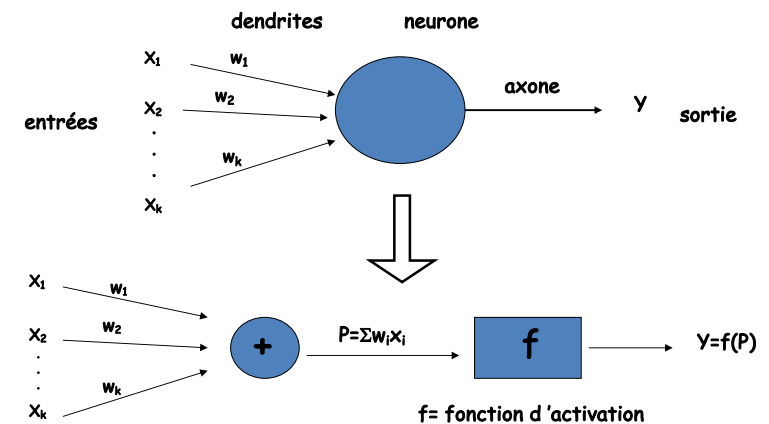


De façon très schématique, un **neurone biologique** est une cellule qui se caractérise par :

- des synapses = les points de connexion avec les autres neurones,
- des dendrites = les « entrées » du neurone,
- l'axone = la « sortie » du neurone vers d'autres neurones,
- le noyau qui active la sortie en fonction des stimuli présentés en entrée (dendrites).

17

Le modèle de McCulloch et Pitts

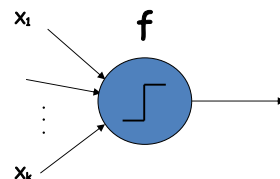


18

Fonction d'activation

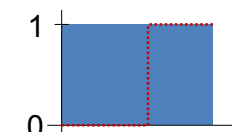
- Les premiers modèles de neurones étaient caractérisés par une fonction d'activation à seuil simple (ie. binaire : 0=inactif ; 1=actif).
- Le déclenchement de l'activité intervient si la somme des excitations dépasse un certain **seuil** (*threshold*) propre au neurone.

- Fonction à seuil :
 - o $f(x) = 1$, si $x > \text{SEUIL}$.
 - o $f(x) = 0$, sinon.

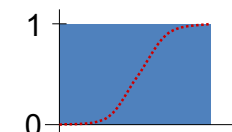


19

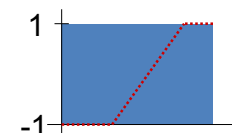
Plusieurs types de fonctions d'activation



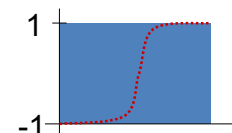
fonction à seuil :
 o $f(x) = 1$, si $x > \text{SEUIL}$,
 o $f(x) = 0$ sinon.



fonction sigmoïde exponentielle :
 o $f(x) = 1/(1+\text{EXP}(-x))$



fonction linéaire bornée :
 o $f(x) = -1$ ou $+1$ au-delà des bornes,
 o $f(x) = c \cdot x$ sinon.

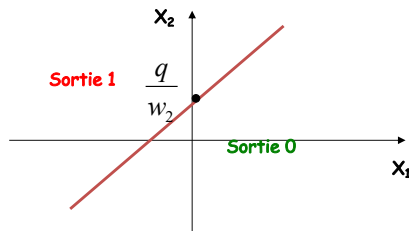


fonction sigmoïde tangentielle :
 o $f(x) = \text{TANH}(x) = \frac{e^x - 1}{e^x + 1}$

20

Exemple d'un neurone à 2 entrées

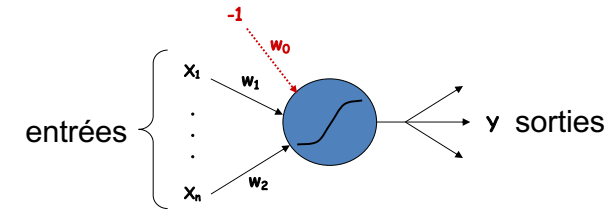
- Règle d'activation :
Si $P = w_1 \cdot x_1 + w_2 \cdot x_2 > q$, alors $y = 1$
Si $P = w_1 \cdot x_1 + w_2 \cdot x_2 \leq q$, alors $y = 0$
- Géométriquement :
Cela signifie qu'on a divisé le plan en deux régions par une droite d'équation $w_1 \cdot x_1 + w_2 \cdot x_2 - q = 0$.



21

Biais w_0

Poids fictif permettant de définir le seuil :



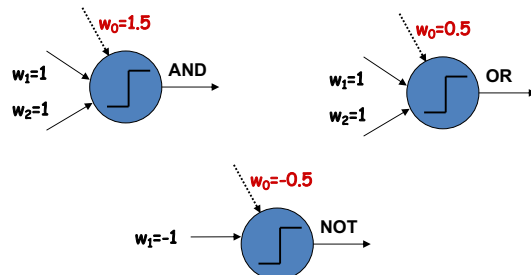
$$P = \sum_{i=0}^n w_i x_i = \sum_{i=1}^n w_i x_i - w_0$$

$$y = f(P) = 1 \text{ si } P > 0, 0 \text{ sinon} \Leftrightarrow y = 1 \text{ si } \sum_{i=1}^n w_i x_i > w_0, 0 \text{ sinon}$$

22

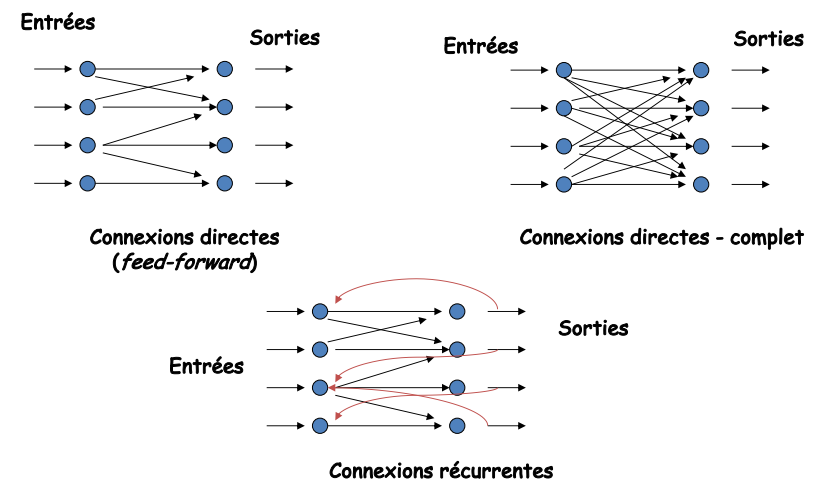
Comparaison avec les opérateurs logiques

L'une des motivations initiales était de pouvoir représenter des fonctions logiques :



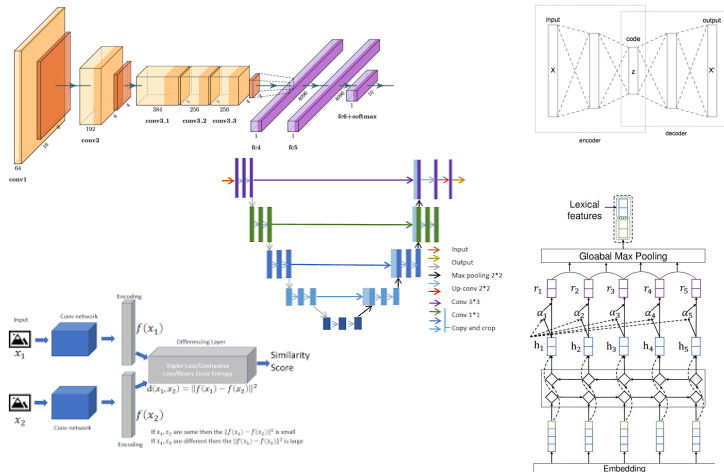
23

Architecture des réseaux de neurones



24

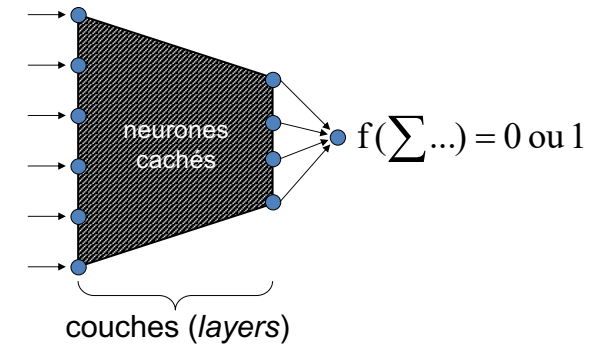
Aujourd'hui



25

Application des réseaux de neurones par connexions directes (1)

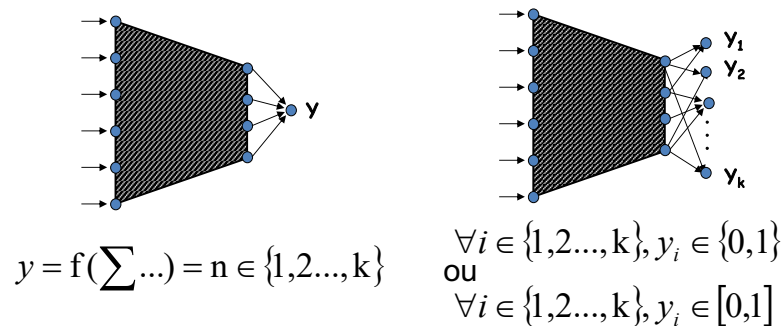
- Classification binaire :



26

Application des réseaux de neurones par connexions directes (2)

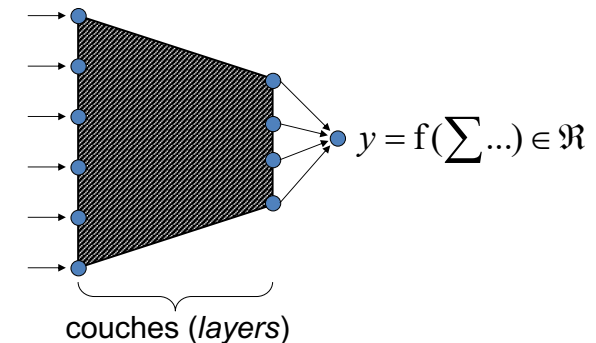
- Classification en k classes :



27

Application des réseaux de neurones par connexions directes (3)

- Régression :

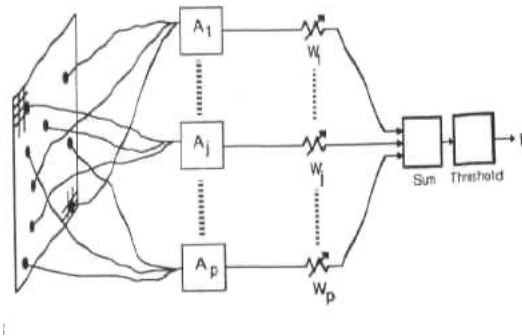


28

Le perceptron [Rosenblatt,1960]

- Architecture : 2 couches

- o entrées (la rétine)
- o sorties

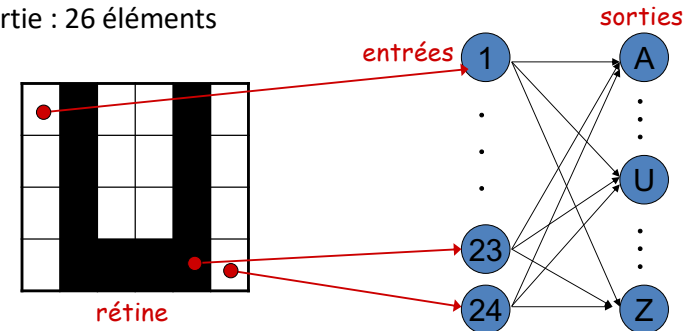


29

Le perceptron

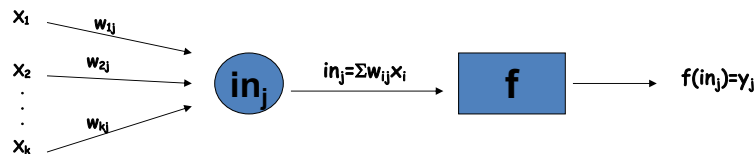
Objectif : association de **formes** présentées en entrée à des réponses.

Exemple § rétine 4 x 6 éléments (image d'une lettre) ; en sortie : 26 éléments



30

Fonctionnement du perceptron



- $x_i \in \{0,1\}$ sortie de la $i^{\text{ème}}$ cellule de la rétine.
- w_{ij} : intensité de connexion entre la $i^{\text{ème}}$ cellule et le $j^{\text{ème}}$ neurone.
- $f(in_j)$: activation de la cellule j.
- Règle d'activation : $y_j = 1$ si $in_j > \theta$, 0 sinon.
- Objectif de l'apprentissage : chercher les poids w_{ij} t.q. les entrées se traduisent par les sorties.

31

Apprentissage avec le perceptron

- Apprentissage supervisé :

Pour apprendre, le perceptron doit savoir qu'il a commis un **erreur** et il doit connaître la réponse qu'il aurait dû donner (intervention du professeur/oracle).

- La règle est locale :

Une cellule apprend sans avoir besoin de la réponse des autres cellules.

- Règle d'apprentissage :

Si la cellule de sortie est active quand elle devait être inactive, alors elle diminue l'intensité des synapses correspondant aux cellules de la rétine qui sont actives (ou inversement).

32

Minimiser l'erreur empirique

- Optimisation de l'erreur :

$$E = \frac{1}{2} Err^2 = \frac{1}{2} (\Phi(x) - f_W(x))^2$$

- Utilisation de la descente du gradient :

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j} \left(\Phi(x) - f \left(\sum_{j=0}^k W_j \cdot x_j \right) \right)$$

$$\frac{\partial E}{\partial W_j} = -Err \times f'(in) \times x_j$$

33

Algorithme du perceptron

function Perceptron-learning (examples E, network N, λ)

Input : E is a set of examples $\{(e_1, \Phi(e_1)), (e_2, \Phi(e_2)), \dots, (e_n, \Phi(e_n))\}$

N is a perceptron with weights w_{ij} , $i=0..k$, $j=1..p$ and activation function f

Repeat

for each e in E do

for each j in $\{1..k\}$ do

$$in \leftarrow \sum_{i=0}^k w_{ij} x_i(e)$$

$$err \leftarrow \Phi_j(e) - f(in)$$

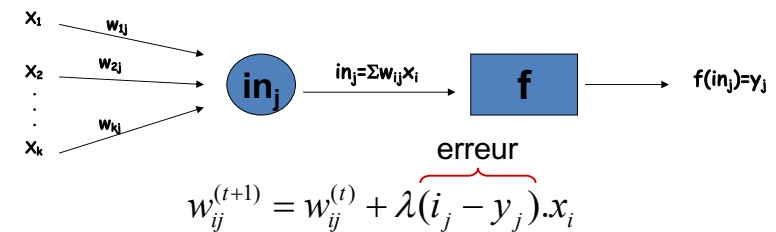
$$w_j \leftarrow w_j + \lambda \times err \times f'(in) \times x_i(e)$$

until some stopping criterion is satisfied

return new hypothesis W

35

Règle de Widrow-Hoff



- x_i : 0 ou 1, valeur de la sortie de i^{ème} cellule de la rétine,
- y_j : réponse de la j^{ème} cellule de la sortie (0 ou 1),
- i_j : réponse théorique, idéale de la sortie (cf. fonction Φ),
- $\lambda \in [0,1]$: constante d'apprentissage (« pas »).

34

Algorithme du perceptron

Cas d'une fonction à seuil

function Perceptron-learning (examples E, network N, λ)

Input : E is a set of examples $\{(e_1, \Phi(e_1)), (e_2, \Phi(e_2)), \dots, (e_n, \Phi(e_n))\}$

N is a perceptron with weights w_{ij} , $i=0..k$, $j=1..p$ and activation function f

Repeat

for each e in E do

for each j in $\{1..k\}$ do

$$in \leftarrow \sum_{i=0}^k w_{ij} x_i(e)$$

$$err \leftarrow \Phi_j(e) - f(in)$$

$$w_j \leftarrow w_j + \lambda \times err \times \cancel{f'(in)} \times x_i(e)$$

until some stopping criterion is satisfied

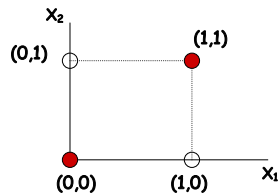
return new hypothesis W

36

Séparabilité linéaire (1)

- Le perceptron ne peut apprendre que si les catégories sont **linéairement séparables**.
- Exemple** § apprentissage de la fonction XOR :

X ₁	X ₂	Réponse
0	0	0
1	0	1
0	1	1
1	1	0

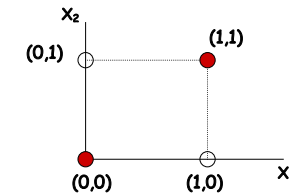


- On veut séparer les ● des ○.

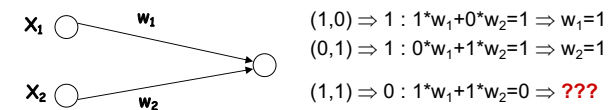
37

Séparabilité linéaire (2)

X ₁	X ₂	Réponse
0	0	0
1	0	1
0	1	1
1	1	0



- Soit un perceptron à deux cellules :



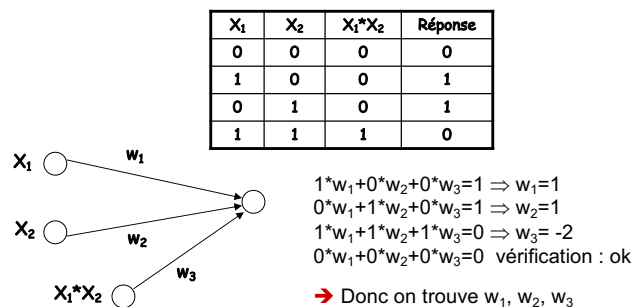
- Il est impossible de trouver des valeurs de W_i pour apprendre la fonction XOR...

38

Séparabilité linéaire (3)

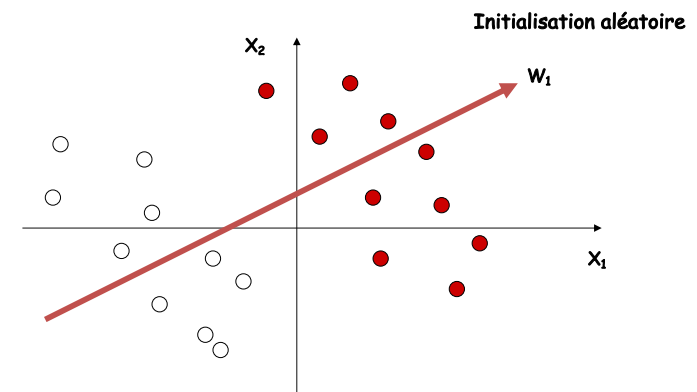
mais...

- On peut séparer les configurations si on utilise plus d'entrées au perceptron :



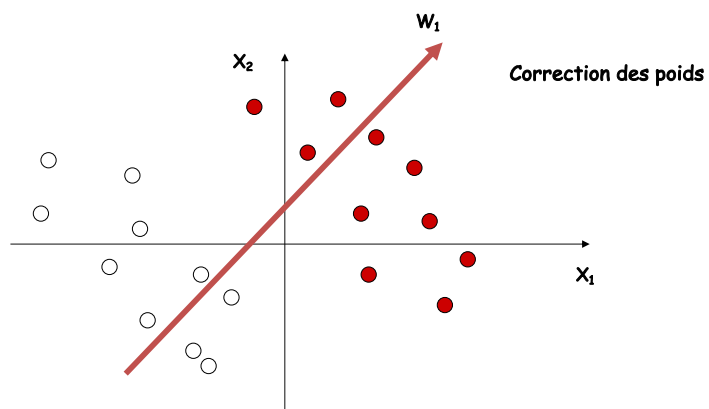
39

Analyse du perceptron (1)



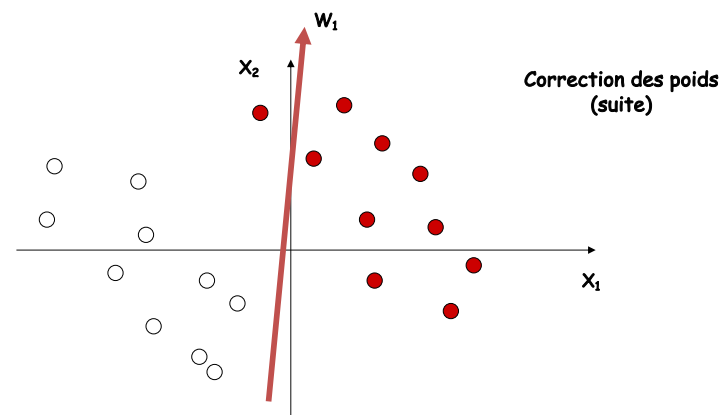
40

Analyse du perceptron (2)



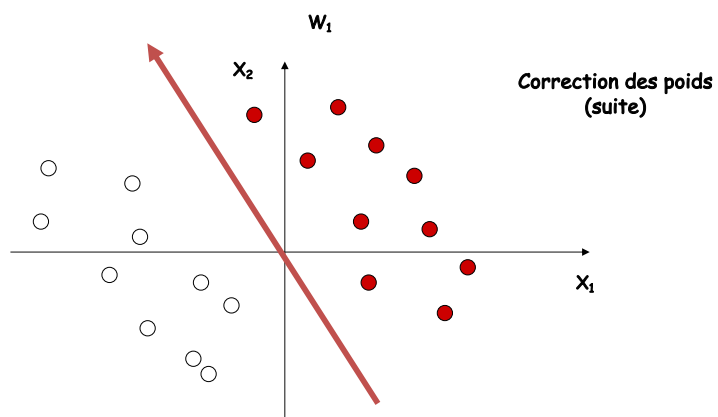
41

Analyse du perceptron (3)



42

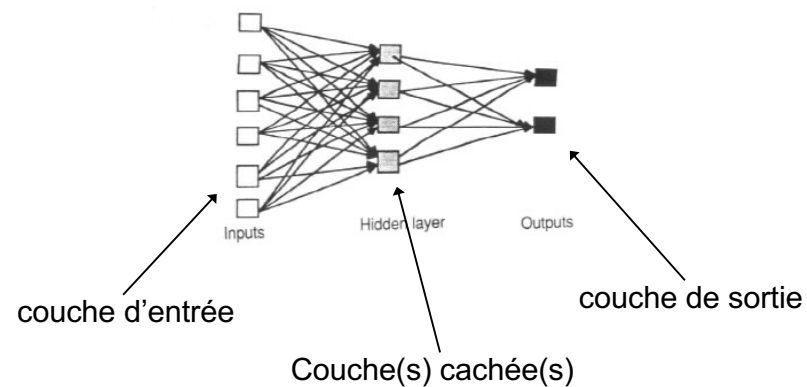
Analyse du perceptron (4)



43

Les réseaux multicouches

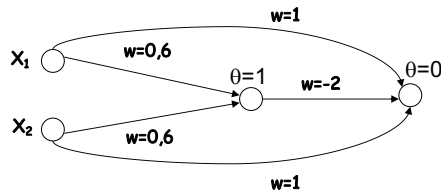
Les réseaux feed-forward



<http://playground.tensorflow.org/>

44

Retour sur le XOR

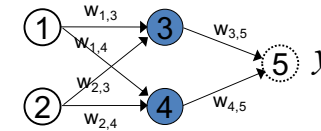


Entrées		Etat C interne	Etat C sortie
x_1	x_2		
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

- On peut donc apprendre la fonction XOR à l'aide d'un réseau à 1 couche cachée.

45

Réseau et apprentissage



$$y = f(w_{3,5} \cdot a_3 + w_{4,5} \cdot a_4)$$

$$y = f(w_{3,5} \cdot f(w_{1,3} \cdot a_1 + w_{2,3} \cdot a_2) + w_{4,5} \cdot f(w_{1,4} \cdot a_1 + w_{2,4} \cdot a_2))$$

Posons $a_1 = x_1$ et $a_2 = x_2$, $\langle x_1, x_2 \rangle$ étant le vecteur d'entrée.

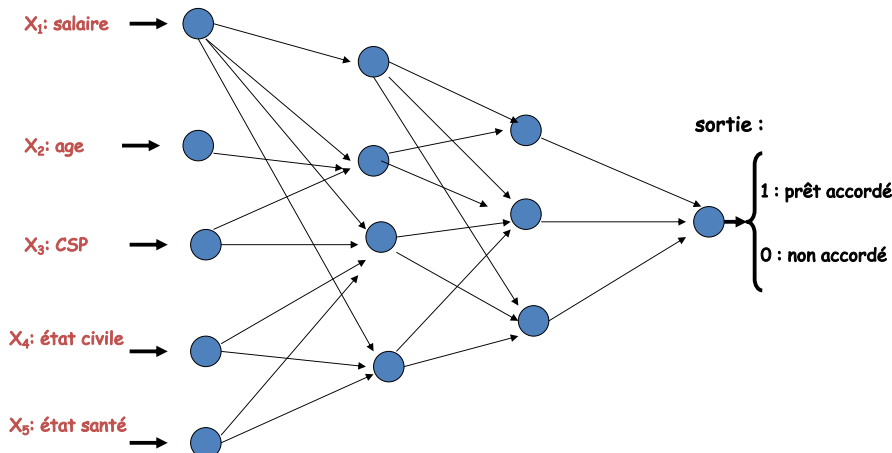
$$\text{Alors : } y = f_W(a_1, a_2) = f_W(x_1, x_2)$$

➔ Le réseau calcule une fonction $f_W(\vec{x})$

Modifier W équivaut à **changer** la fonction calculée.

46

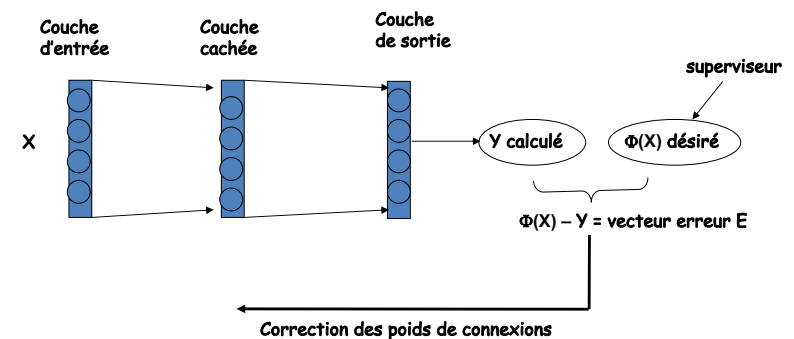
Exemple de problème d'apprentissage



47

Règle d'apprentissage

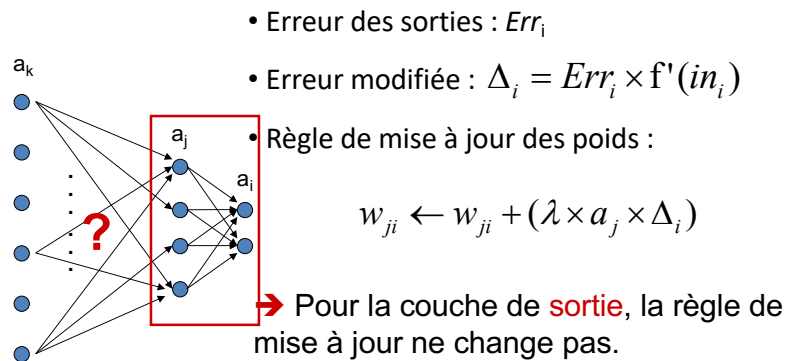
- Algorithme de rétropropagation du gradient (*back-propagation*) :



48

Approche intuitive

cas du Single Layer Perceptron (SLP)



Mais comment mettre à jour les poids antérieurs ?

49

Propagation de l'erreur

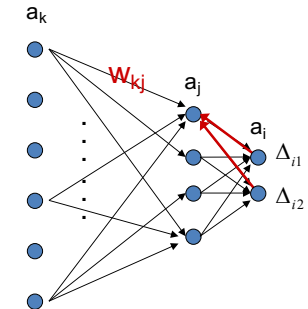
- Nécessité de définir une quantité analogue à l'erreur Err_i des neurones de la couche de sortie. D'où l'idée de la **rétropropagation**.

- Propagation de Δ_i :

$$\Delta_j = f'(in_j) \sum_i w_{ji} \Delta_i$$

- Règle de mise à jour :

$$w_{kj} \leftarrow w_{kj} + (\lambda \times a_k \times \Delta_j)$$



50

Algorithme de rétropropagation

function BackProp-learning (examples E, network N)

Input : E is a set of examples $\{(e_1, \Phi(e_1)), (e_2, \Phi(e_2)), \dots, (e_n, \Phi(e_n))\}$

N is a perceptron with L layers, weights W and activation function f

Repeat

for each e in E do

for each node j in the input layer do $a_j \leftarrow x_j[e]$

for l=2 to L do $in_i \leftarrow \sum_j w_{ji} a_j$
 $a_i \leftarrow f(in_i)$

for each node i in the output layer do

$\Delta_i \leftarrow f'(in_i) \times (\Phi_i[e] - a_i)$

for l=L-1 to 1 do

for each node j in layer l do

$\Delta_j \leftarrow f'(in_j) \sum_i w_{ji} \Delta_i$

for each node i in layer l+1 do
 $w_{ji} \leftarrow w_{ji} + (\lambda \times a_j \times \Delta_i)$

until some stopping criterion is satisfied

return new hypothesis W

51

Algorithme de rétropropagation

Quelques remarques

- « Pas » d'apprentissage : λ
 - o trop petit : convergence lente vers la solution,
 - o trop grand : risque d'oscillation.
- Heuristiques courantes :
 - o diminuer le pas au fur et à mesure
 - o Momentum
 - etc.
- Un ANN avec couches cachées est capable d'approximer **toute fonction booléenne** existante, pourvu que l'on fixe convenablement le nombre de neurones dans la couche cachée.

52

Avantages des ANN

- **Parallélisme** : le principe et le potentiel sont clairement affichés. De nouveaux formalismes et les bénéfices à en tirer restent à étudier...
- **Capacité d'adaptation** : possibilités d'auto-organisation.
- **Capacités de généralisation** : parfois spectaculaires, notamment en reconnaissance des formes et sur le texte.
- **Simplicité** de mise en oeuvre pour de nombreux problèmes.
- Intérêt général dans les problèmes pour lesquels on connaît **peu d'informations a priori**.

53

Limitations

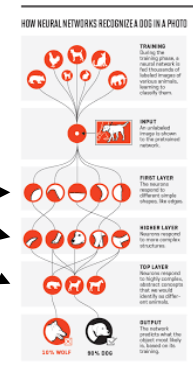
- **Performances** (temps de réponses) : notamment pour la machine de Boltzmann ou des algorithmes de type rétro-propagation.
- Aspect « **boîte noire** » : pouvoir explicatif très limité.
- Importance du travail préliminaire de **préparation des données**, notamment : choix des variables pertinentes, normalisation, élimination de données aberrantes, organisation du protocole d'apprentissage et de validation, etc.
- **Choix de l'architecture** du réseau (paramètres).
- Convergence vers un **optimum local**.

54

Des réseaux de plus en plus profonds



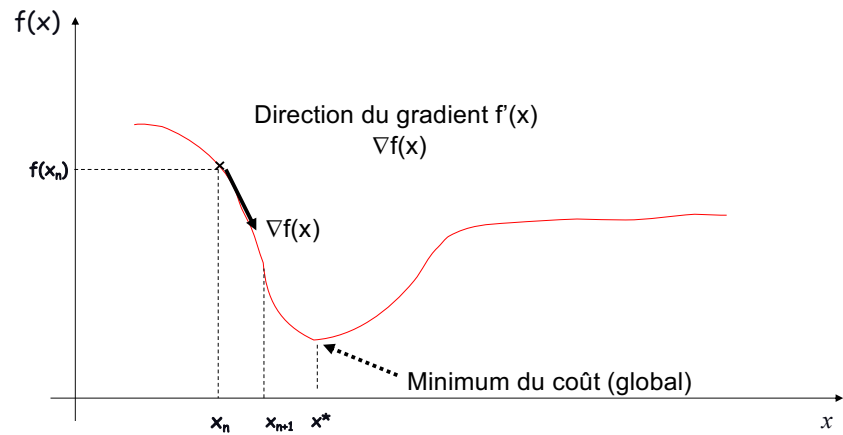
On observe que le réseau apprend à distinguer différentes granularités dans l'information



55

56

Problème des optima locaux



57

Pour conclure

- Les ANN sont aujourd'hui la base du **deep learning** utilisé dans de nombreuses applications en IA :
 - reconnaissance d'objets dans des images / vidéo
 - traduction automatique
 - aide à la programmation
 - prédiction de la structure 3D de protéines
 - jeux de société et jeux vidéo
 - génération automatique d'images, de musique, de texte...

58