L'IA/AA dans tous les médias aujourd'hui

L'IA/AA est attendue **partout** demain

- Aide au **diagnostic médical**
- Aide aux **juges**
- Octroi de **prêts**
- Aide au **choix des employés**
- Évaluation du **risque de criminalité** avant l'acte criminel
- Calcul des **primes d'assurance**

- **Assistant** personnel
- Véhicules **autonomes**
- Conduite des « **smart cities** »

- Sommes-nous prêts ?

Une perspective sur l'apprentissage artificiel

Antoine Cornuéjols

AgroParisTech – INRA MIA 518

antoine.cornuejols@agroparistech.fr

Motivation

Concepts difficult to hand-code

- Permissible moves for a robot
- Person to recruit / or not
- Predispositions for certain types of cancer

→ Learning **from examples**

Plan

Grands types d'apprentissages

Apprentissage prédictif (*supervisé*)

- A *learning set*

$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_m, y_m)\}$$

f

h

Prediction for new examples $x \text{ –}h\text{-> } y$?

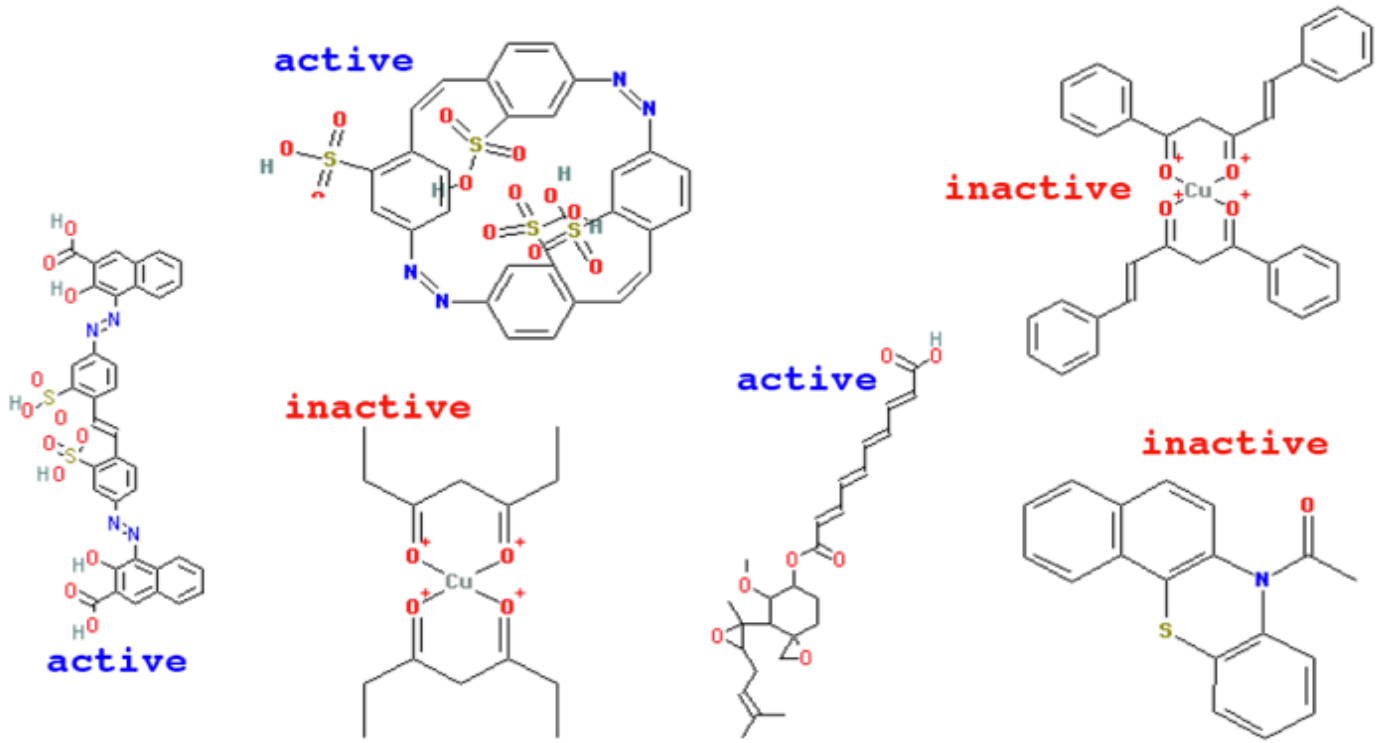
Examples

- Learn to **diagnose a disease**
  - x = description of the patient (symptoms, results of examinations, …)
  - y = disease (or recommanded therapy)

- Part-of-Speech tagging
  - x = a sentence (e.g. « *a star was born* »)
  - y = Role of the words in the sentence

- Face recognition
  - x = bitmap view of the face
  - y = name of the person (or emotion: *frightful, angry*, …)

Apprentissage prédictif



NCI AIDS screen results (from http://cactus.nci.nih.gov).

Apprentissage prédictif

- If *f* is a ***continuous*** *function*
  - Regression
  - Density estimation

- If *f* is a ***discrete*** *function*
  - Classification

- If *f* is a ***binary*** *function* (Boolean)
  - Concept learning

Apprentissage descriptif

À propos d'un *échantillon d'apprentissage* $s = \{(x_i)\}_{1,m}$

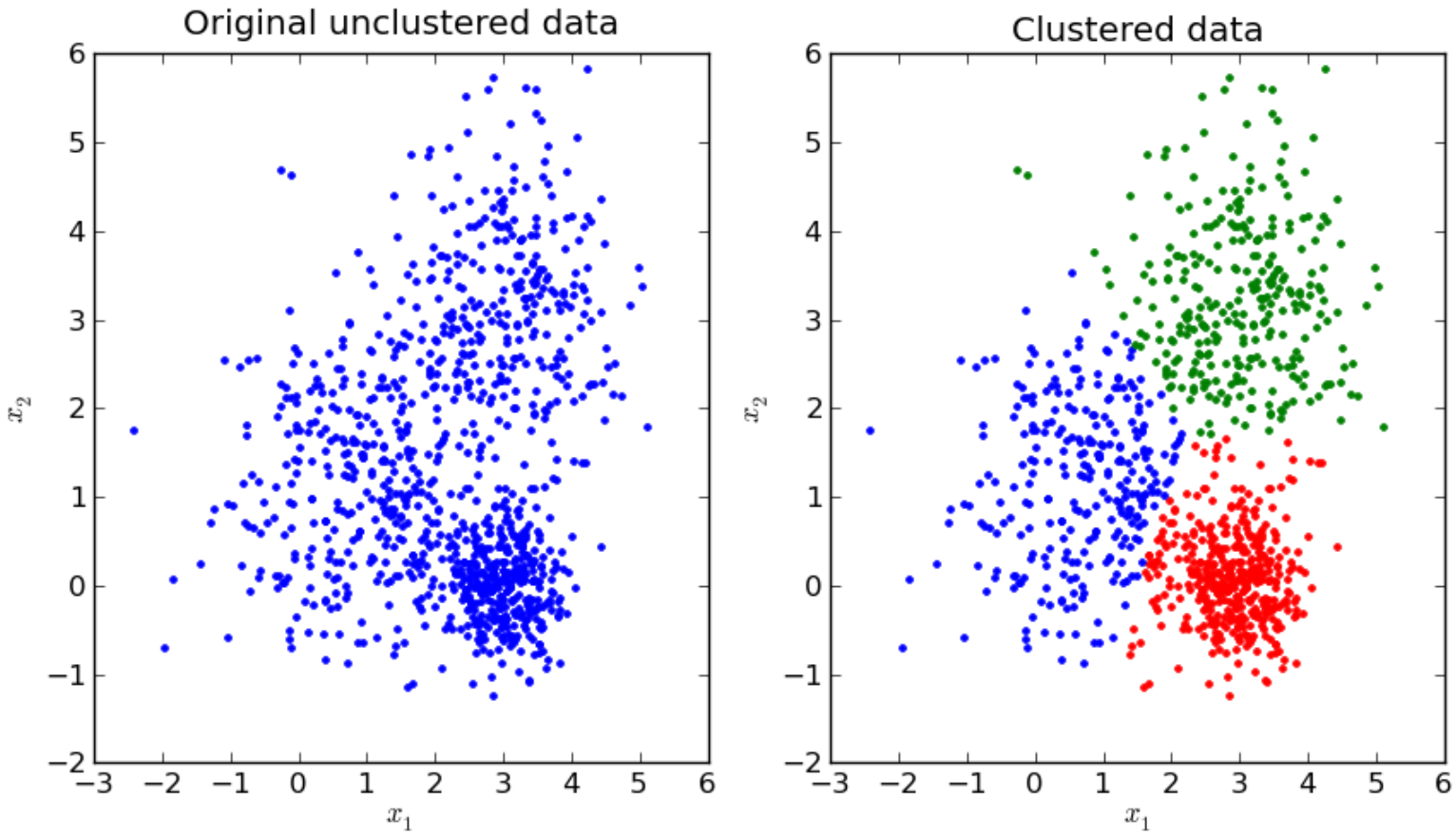
identifier des régularités rendant compte de *S*

- E.g. under the form of clusters (e.g. *mixture of Gaussians*)
  - CLUSTERING
- E.g. sous la forme de motifs fréquents (fouille de données)

pour résumer, suggérer des régularités, comprendre …

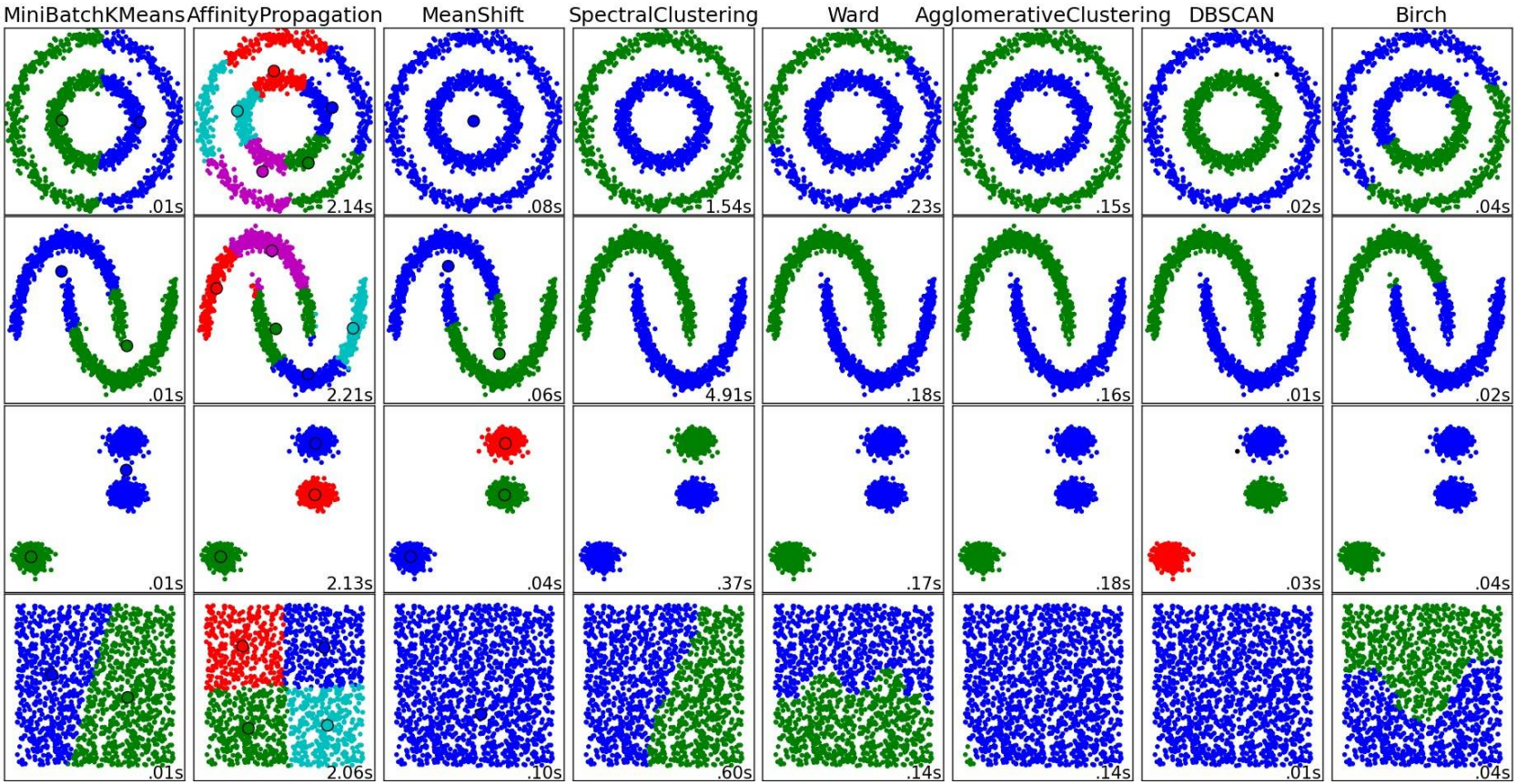
Les **grands types** d'apprentissage

- Apprentissage « **descriptif** » (non supervisé)



Clustering

Dépend beaucoup des biais a priori



Apprentissage prescriptif

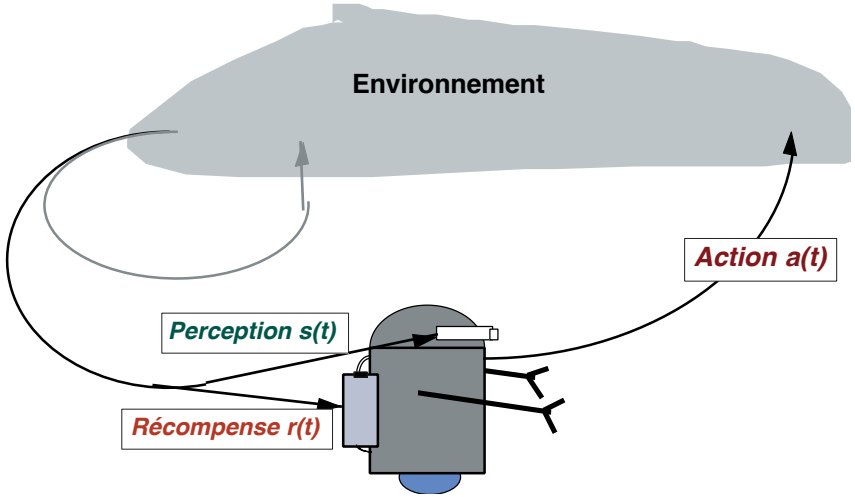
- Apprentissage « **prescriptif** » (recherche de ***causalités***)

1. J'observe que les gens qui mangent des glaces
   sont souvent en maillot de bain
2. Je voudrais vendre davantage de glaces

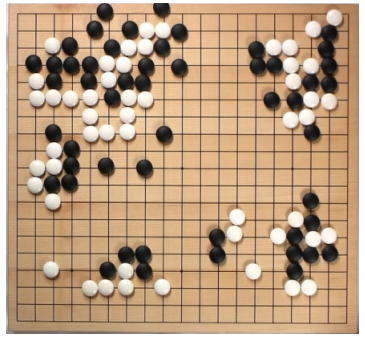
→ Je demande aux gens de se mettre en maillot de bain

Apprentissage « par **renforcement** »

- comment (ré)agir



1. Piloter un hélicoptère
2. Apprendre à jouer au tennis de table
3. Battre le champion de back-gammon (1992), de Go (2016)
4. Gérer un porte-feuille d'investissements
5. Contrôler une ferme



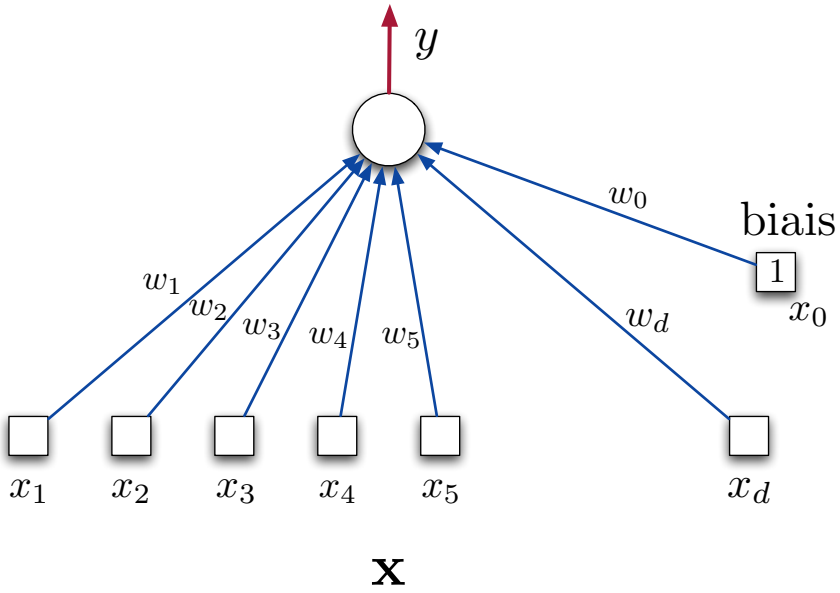
Plan

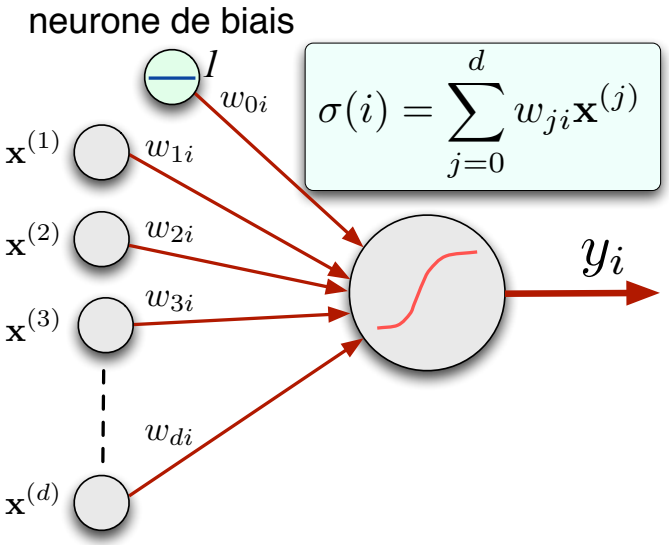
1. Grands types d'apprentissage
2. Apprentissage prédictif par réseaux de neurones
3. Quelles garanties ?
4. Recette pour créer des algorithmes d'apprentissage
5. Les réseaux de neurones profonds
6. Ce que l'on sait faire et les défis à relever

Le cas du perceptron

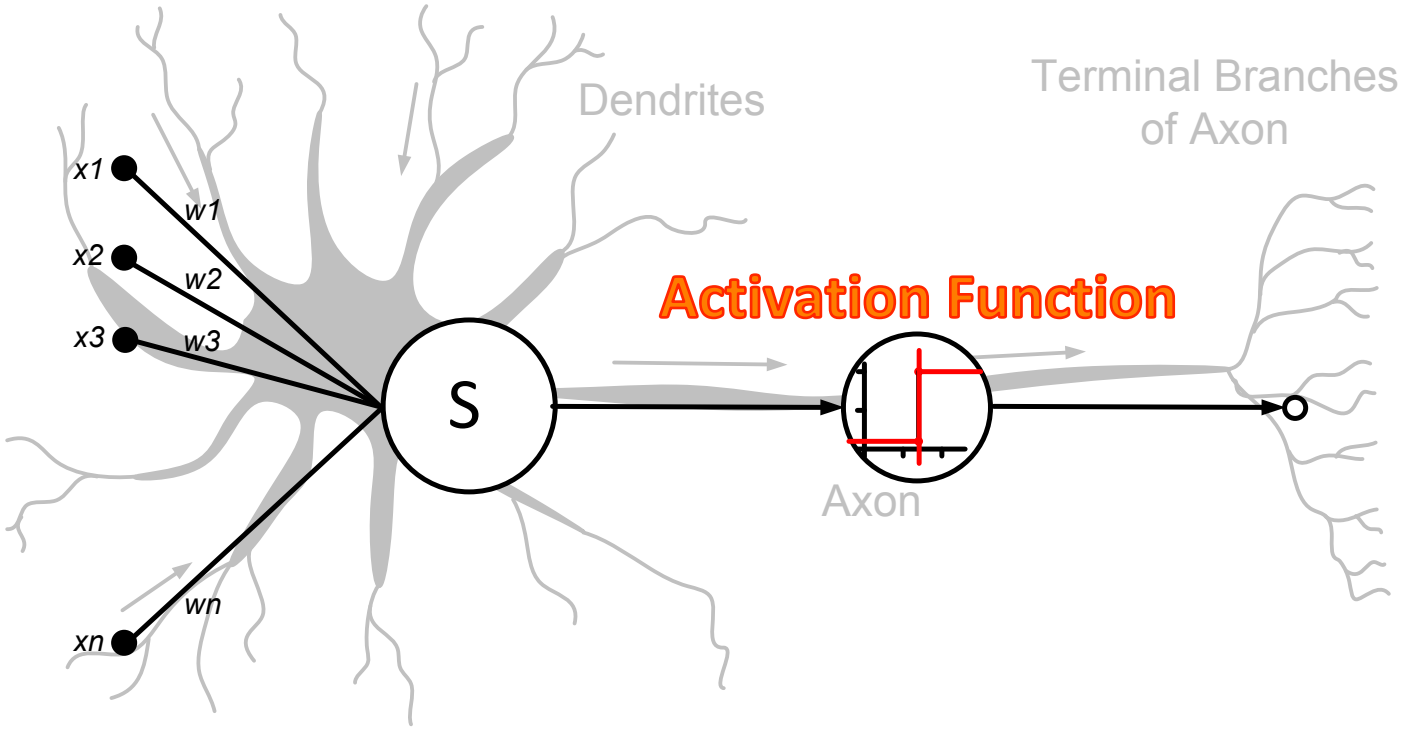
Le perceptron

– Rosenblatt (1958-1962)

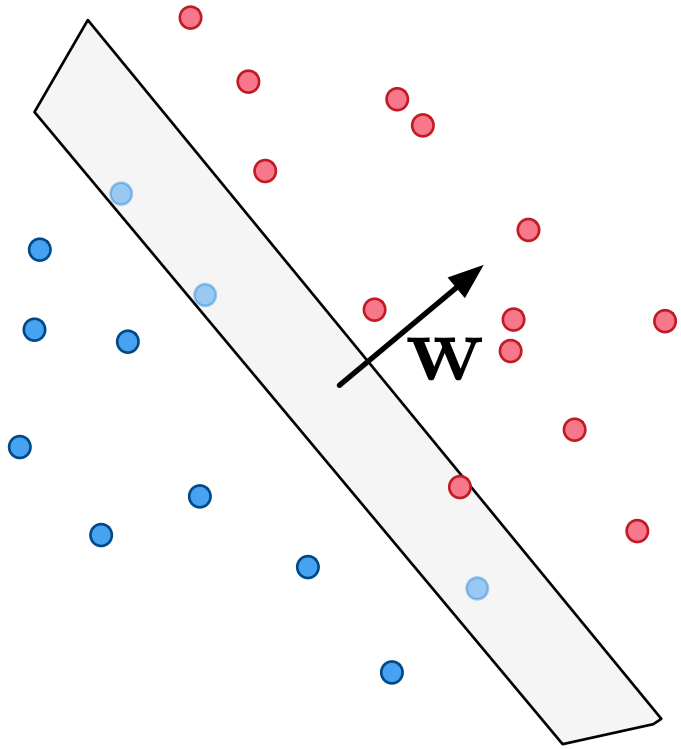




Artificial Neural Networks (ANN)



Le perceptron : un discriminant linéaire



Le perceptron

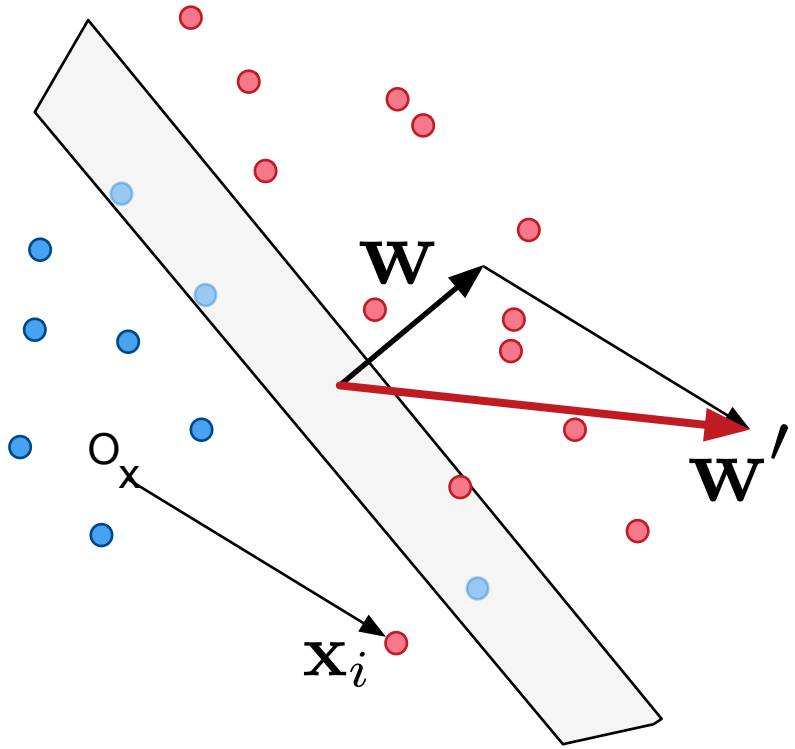
- **Apprentissage des poids** w_i
  - Principe (*règle de Hebb*) : en cas de succès, ajouter à chaque connexion quelque chose de proportionnel à l'entrée et à la sortie

Règle du perceptron : apprendre seulement en cas d'échec

Algorithme 1 : Algorithme d'apprentissage du perceptron

tant que *non convergence* **faire**
  si *la forme d'entrée est correctement classée* **alors**
    ne rien faire
  sinon
    $\mathbf{w}(t+1) = \mathbf{w}(t) \stackrel{+}{-} \eta\, \mathbf{x}_i\, y_i$
  fin
  Passer à la forme d'apprentissage suivante
fin

The perceptron learning algorithm: intuition



$$\mathbf{w}' = \mathbf{w} + \eta\, y_i\, \mathbf{x}_i$$

Des propriétés remarquables !!

- **Convergence** en un nombre fini d'étapes
  - Indépendamment du **nombre** d'exemples
  - Indépendamment de la **distribution** des exemples
  - (quasi)indépendamment de la **dimension** de l'espace d'entrée

!!!

Si il existe au moins *une séparatrice linéaire des exemples*

Garantie de généralisation ??

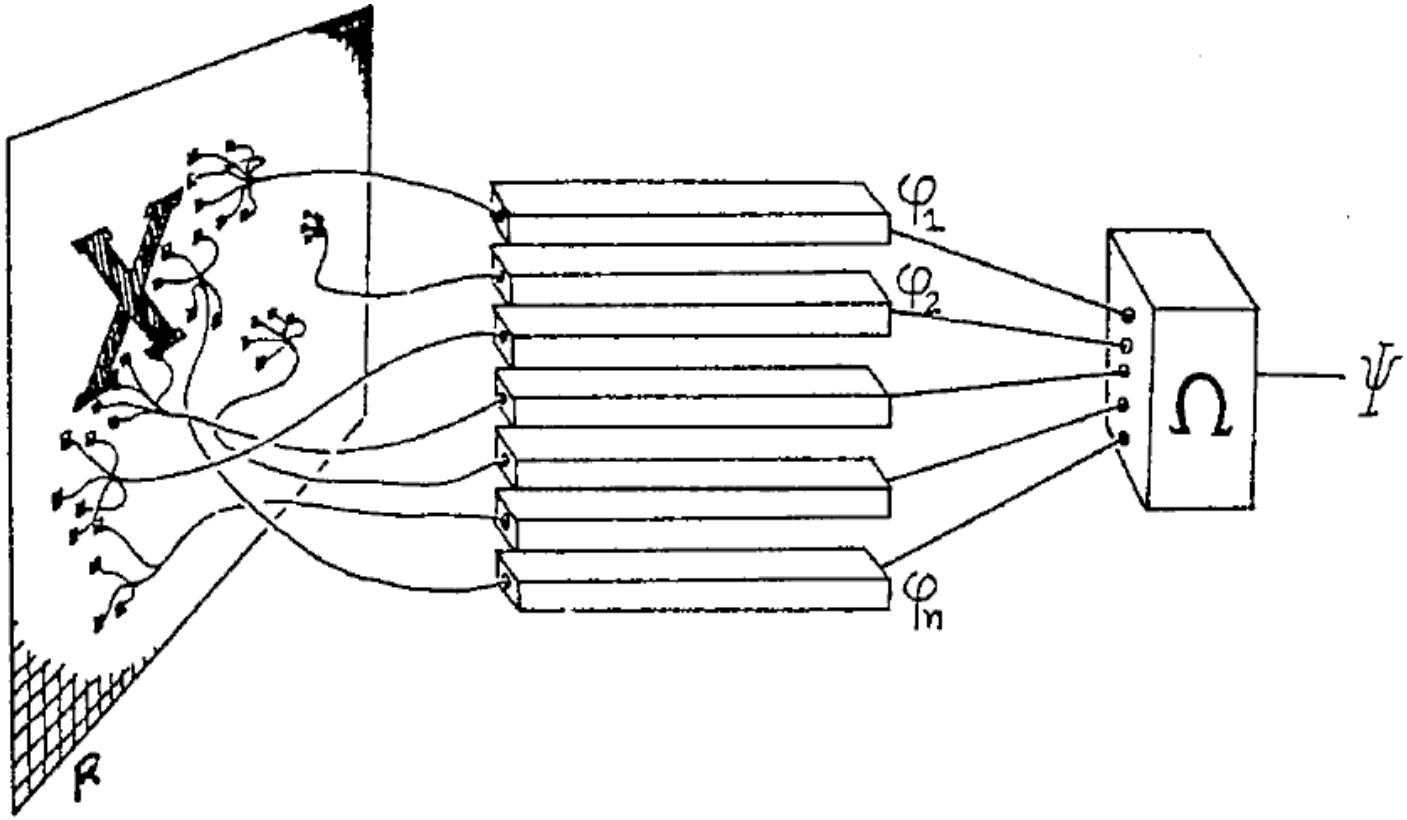
- Théorèmes sur la performance

  par rapport à l'échantillon d'apprentissage

- Mais qu'en est-il pour des **exemples à venir** ?

Le Perceptron

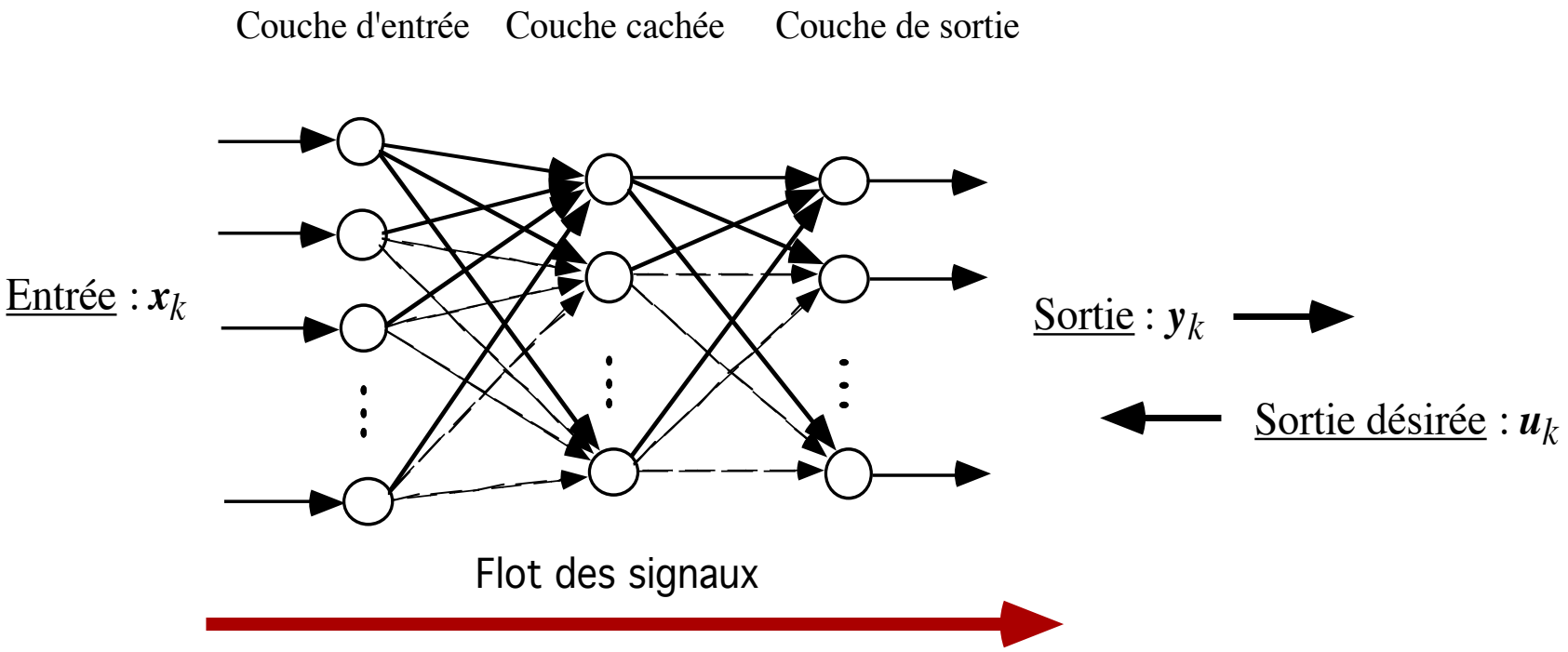
– Rosenblatt (1958-1962)



Les perceptrons multi-couches

Le perceptron multi-couche

- Topologie typique



Le Perceptron Multi-Couches : propagation

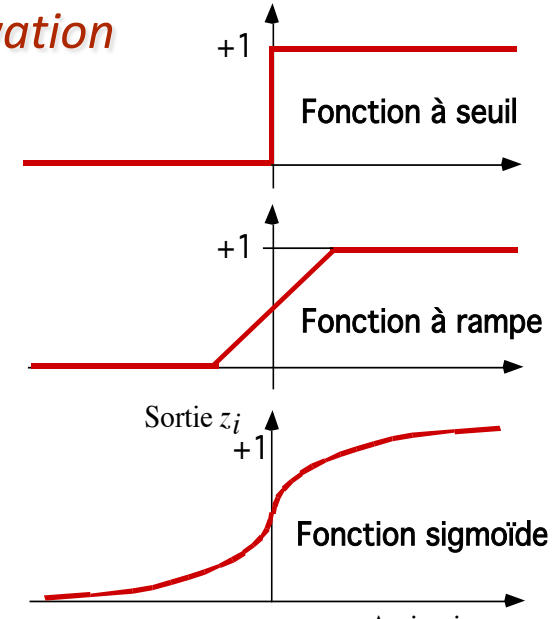
- Pour chaque neurone :

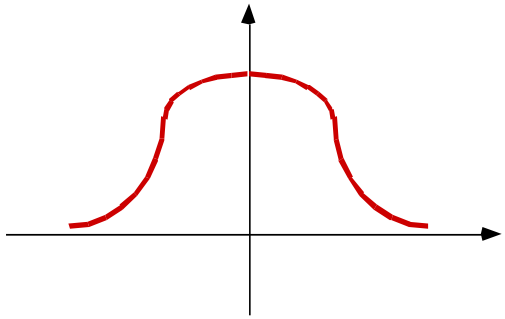
$$y_l = g\left(\sum_{j=0,d} w_{jk}\ \phi_j\right) = g(a_k)$$

  - w_{jk} : *poids* de la connexion de la cellule j à la cellule k
  - a_k : *activation* de la cellule k
  - g : *fonction d'activation*

$$g(a) = \frac{1}{1 + e^{-a}}$$

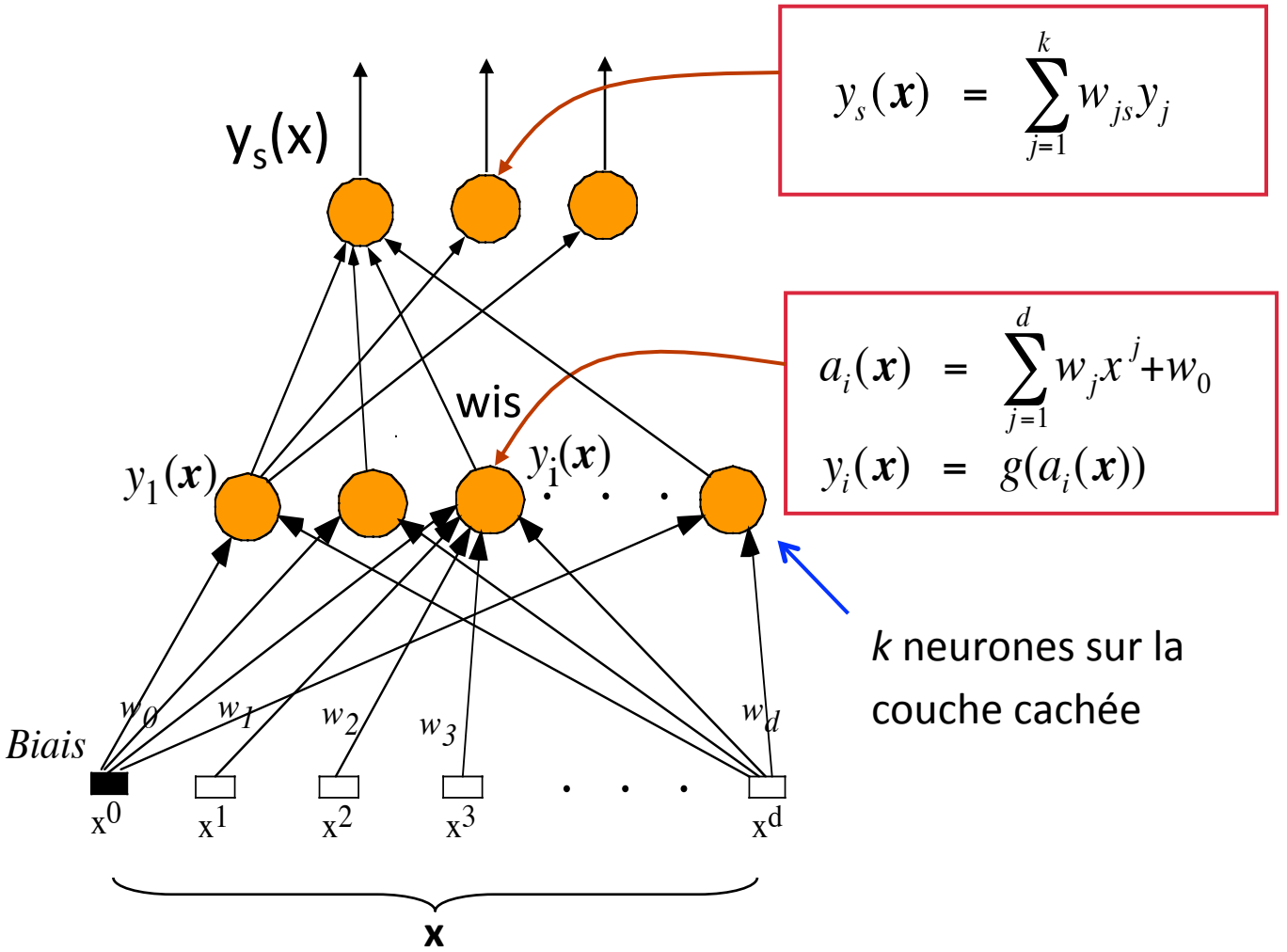
g'(a) = g(a)(1-g(a))





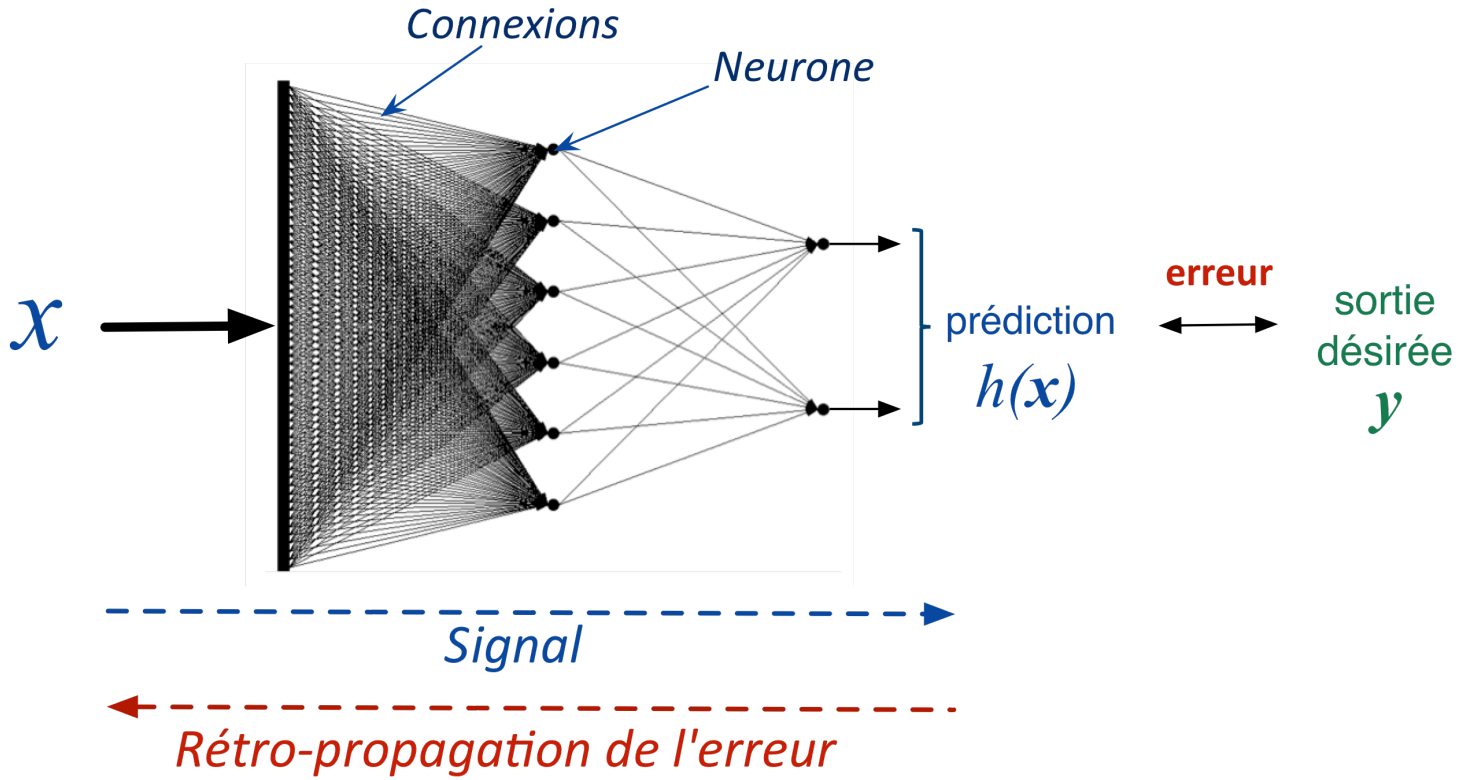
Fonction à base radiale

Le PMC : passes avant et arrière (résumé)



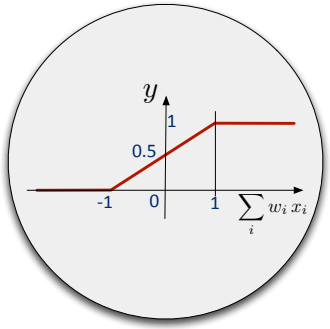
Learning with Multi-Layer Perceptrons

- Questions:
  - How to learn the **parameters** (weights of the connections) ?
  - How to set the **architecture** of the network?



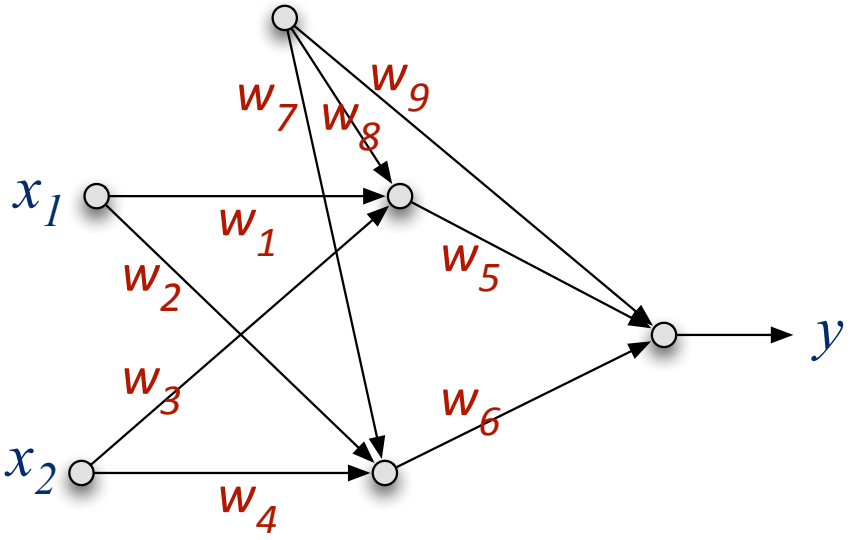
Compute the weights such that …

Calcul de la fonction XOR

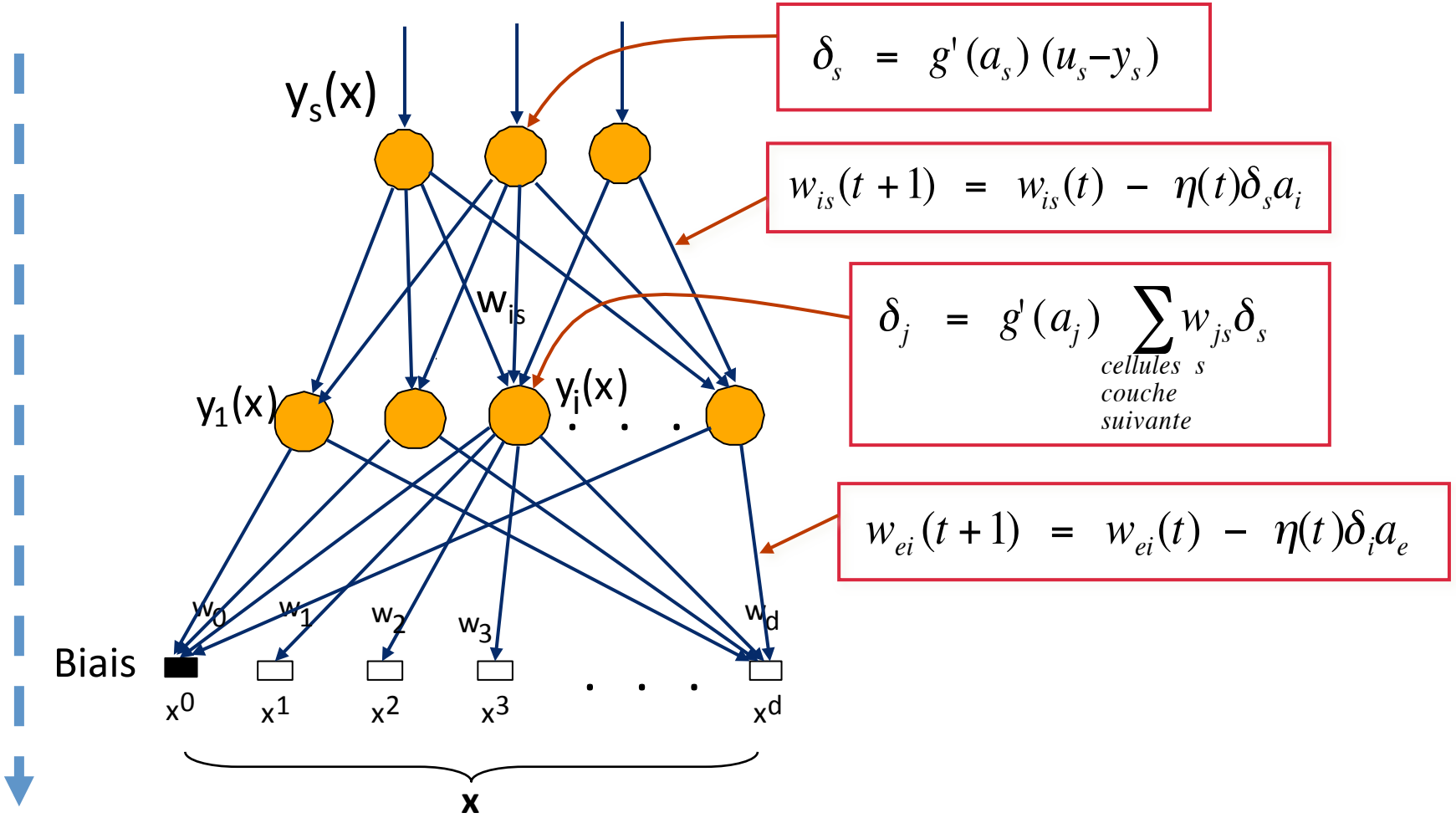


Entrée x_1	Entrée x_2	Sortie y
0	0	0
0	1	1
1	0	1
1	1	0

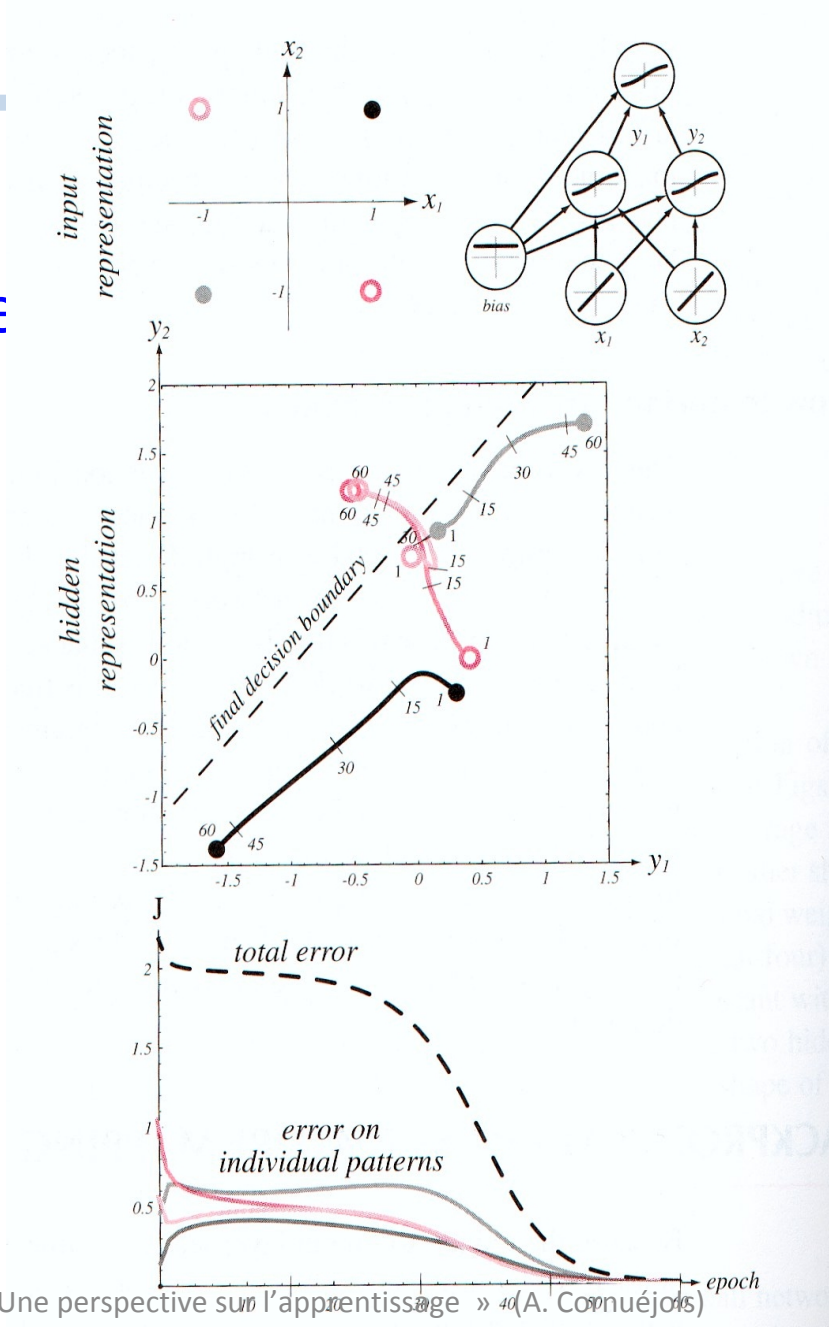
W_1 =
W_2 =
W_3 =
W_4 =
W_5 =
W_6 =
W_7 =
W_8 =
W_9 =



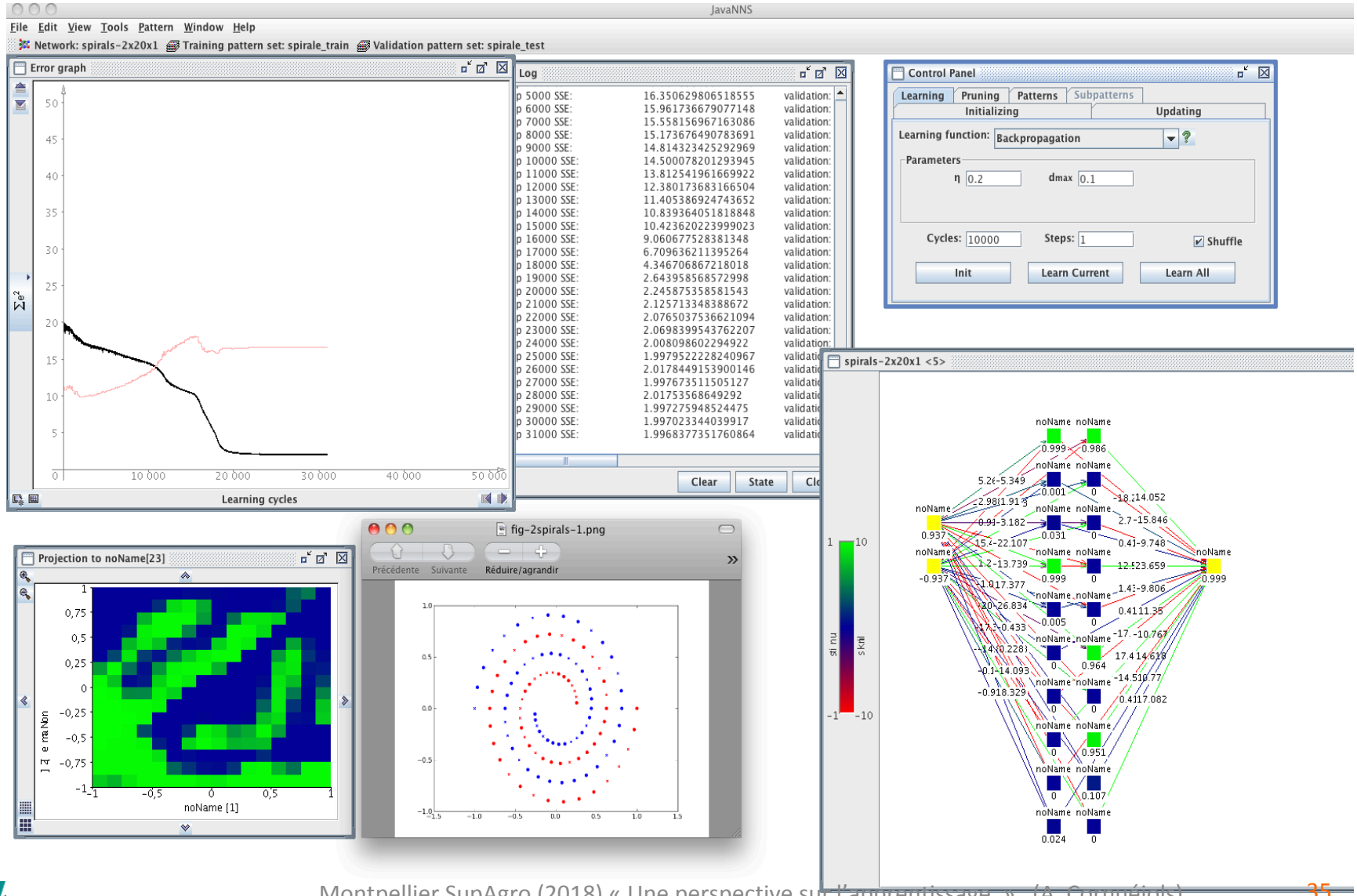
Le PMC : passes avant et arrière (résumé)



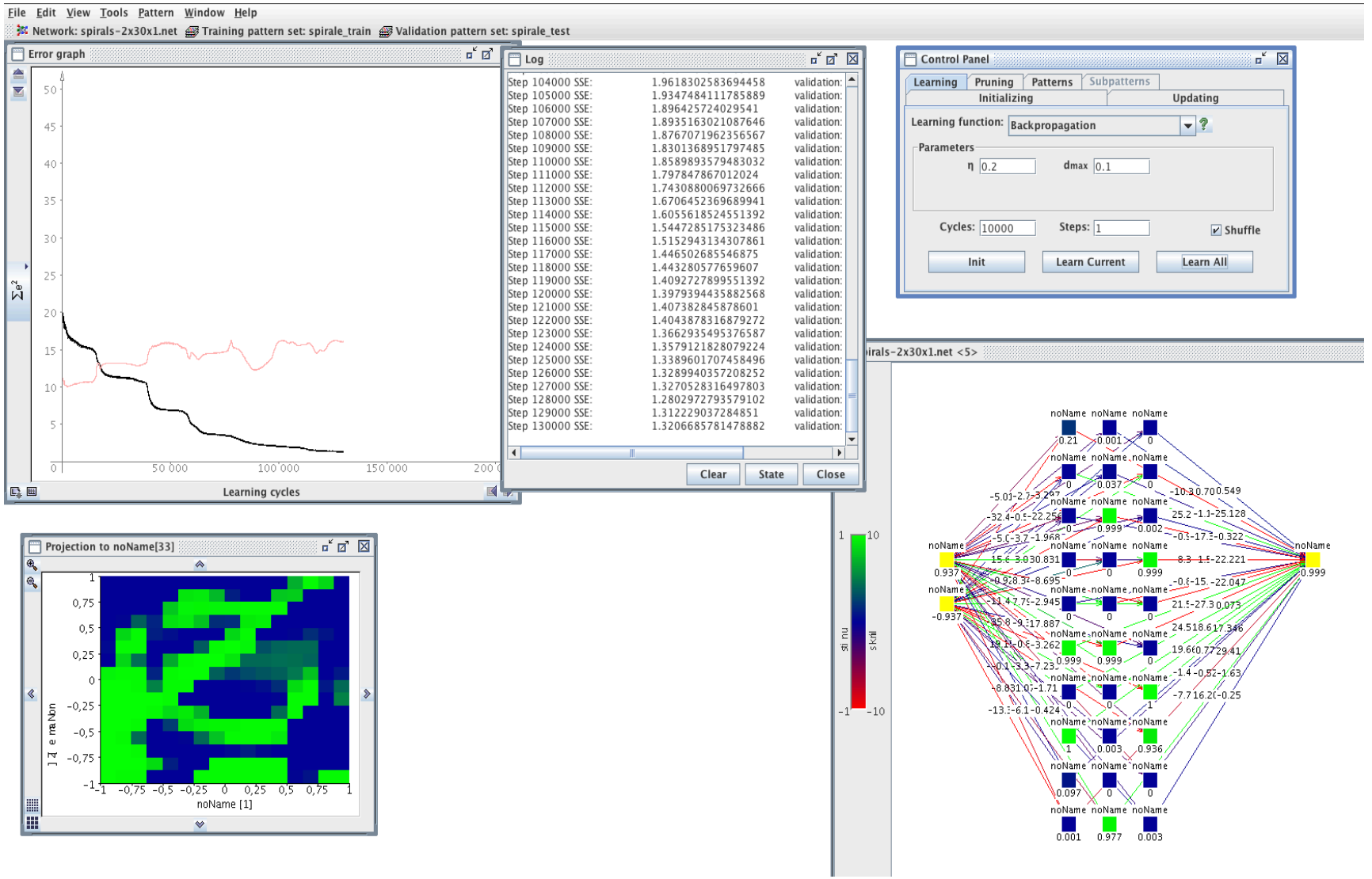
Rôle de la couche cachée



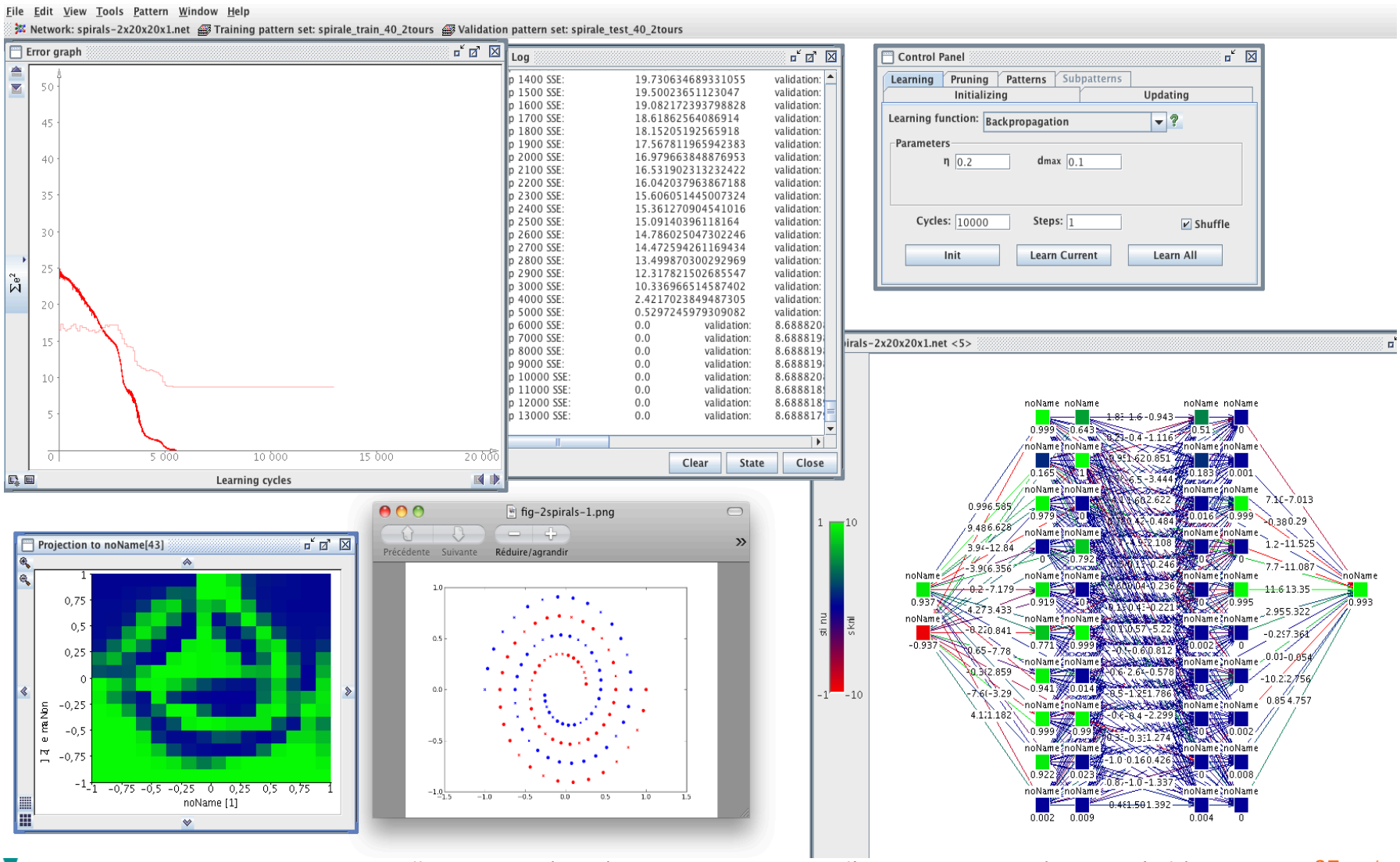
Illustration



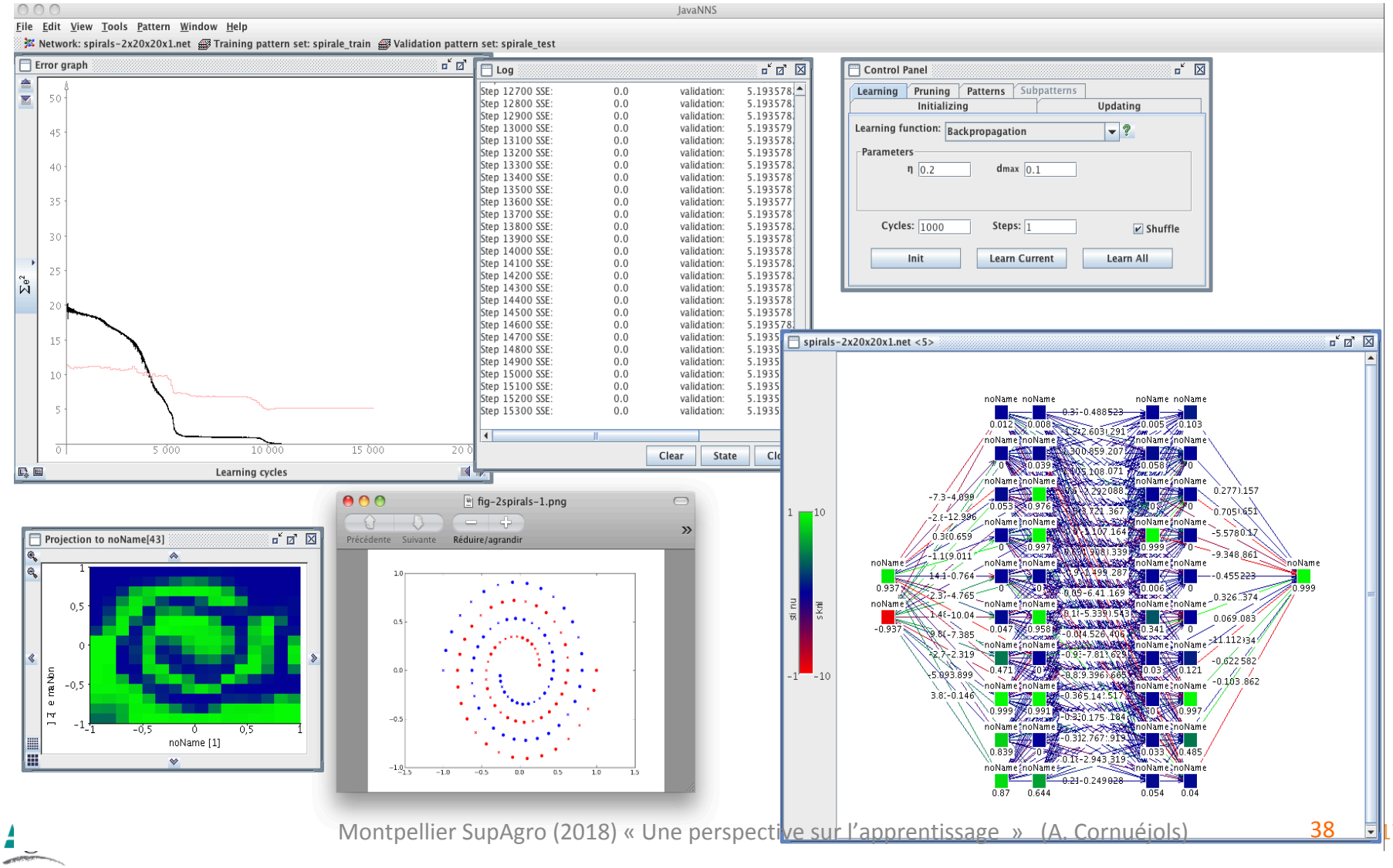
Illustration



Illustration

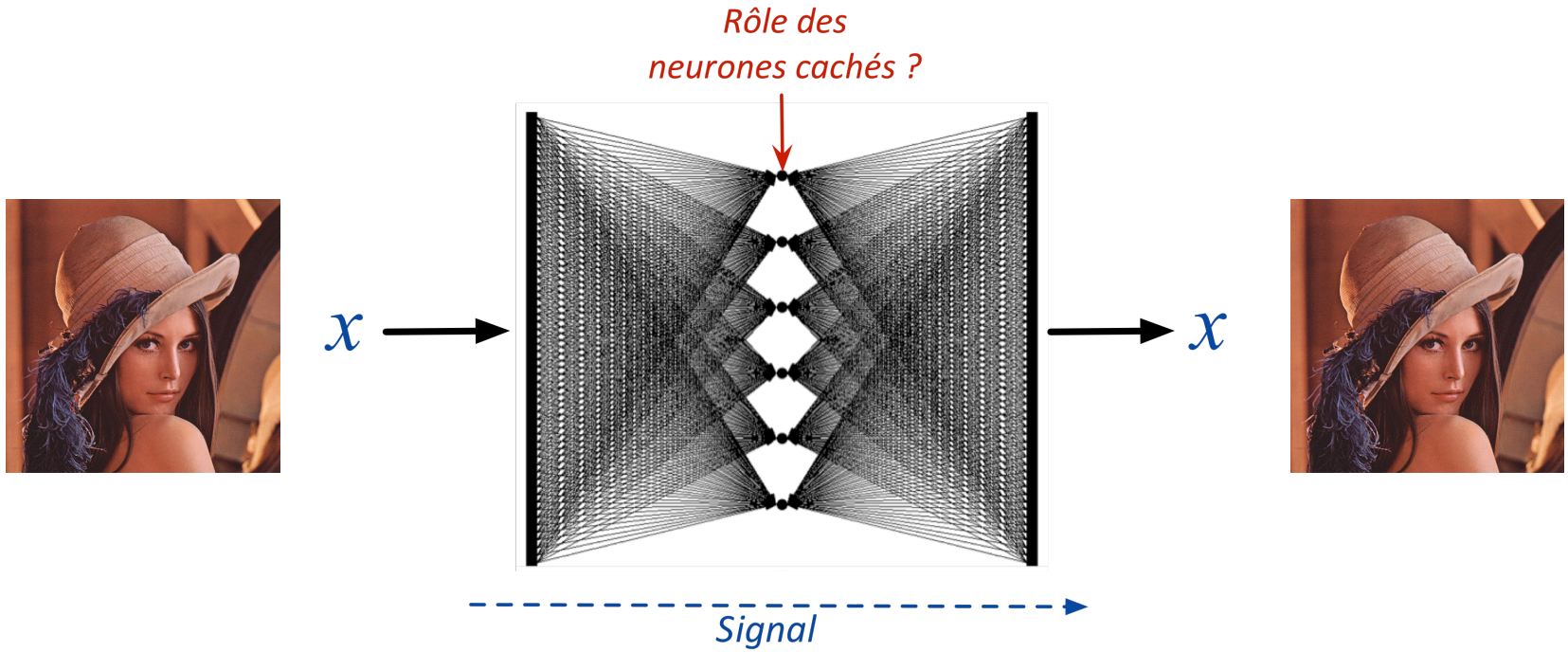


Illustration

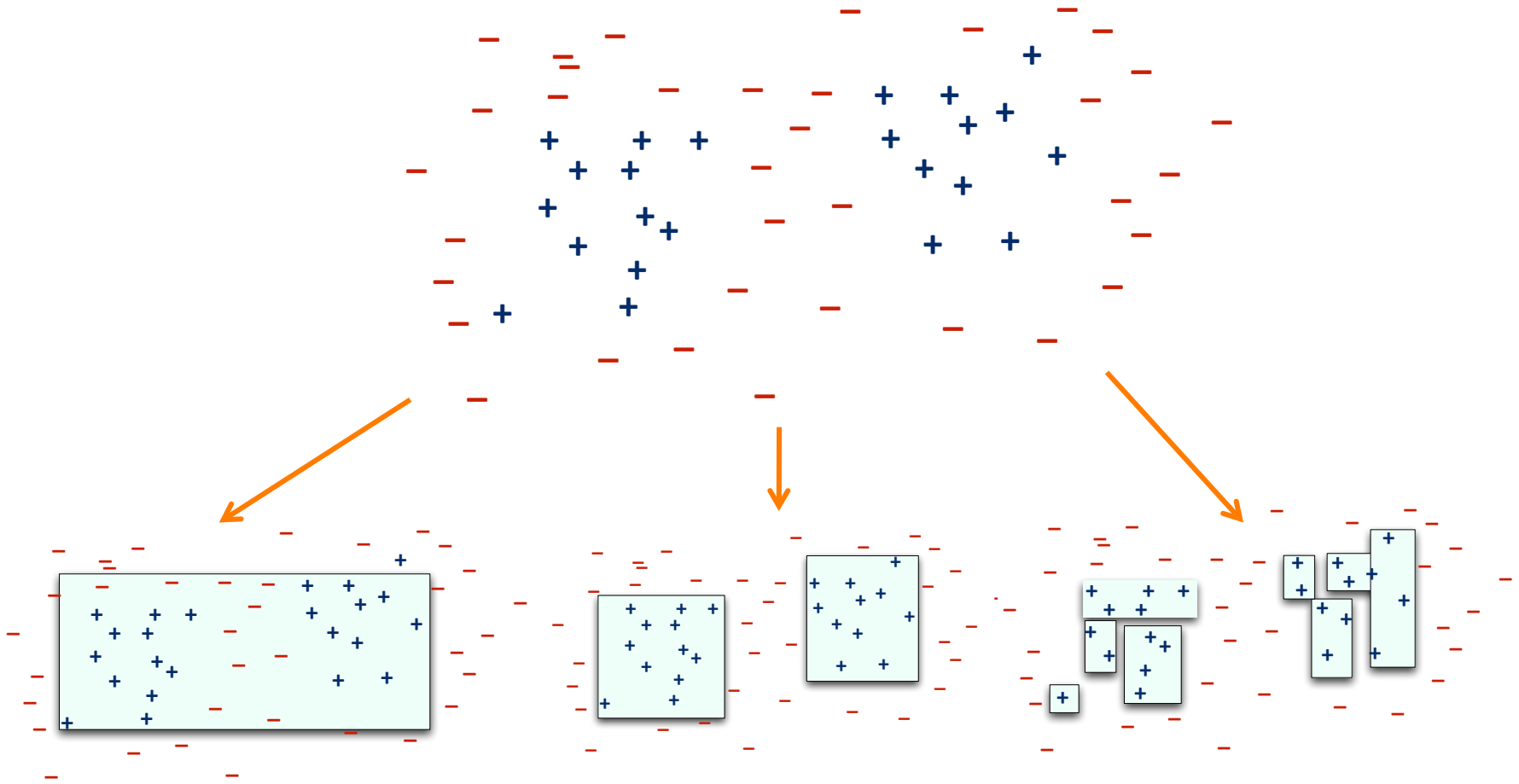


Change of representation: the role of hidden layer(s)

- **Which new representation** (latent variables)?
- How to choose the architecture?



Illustration

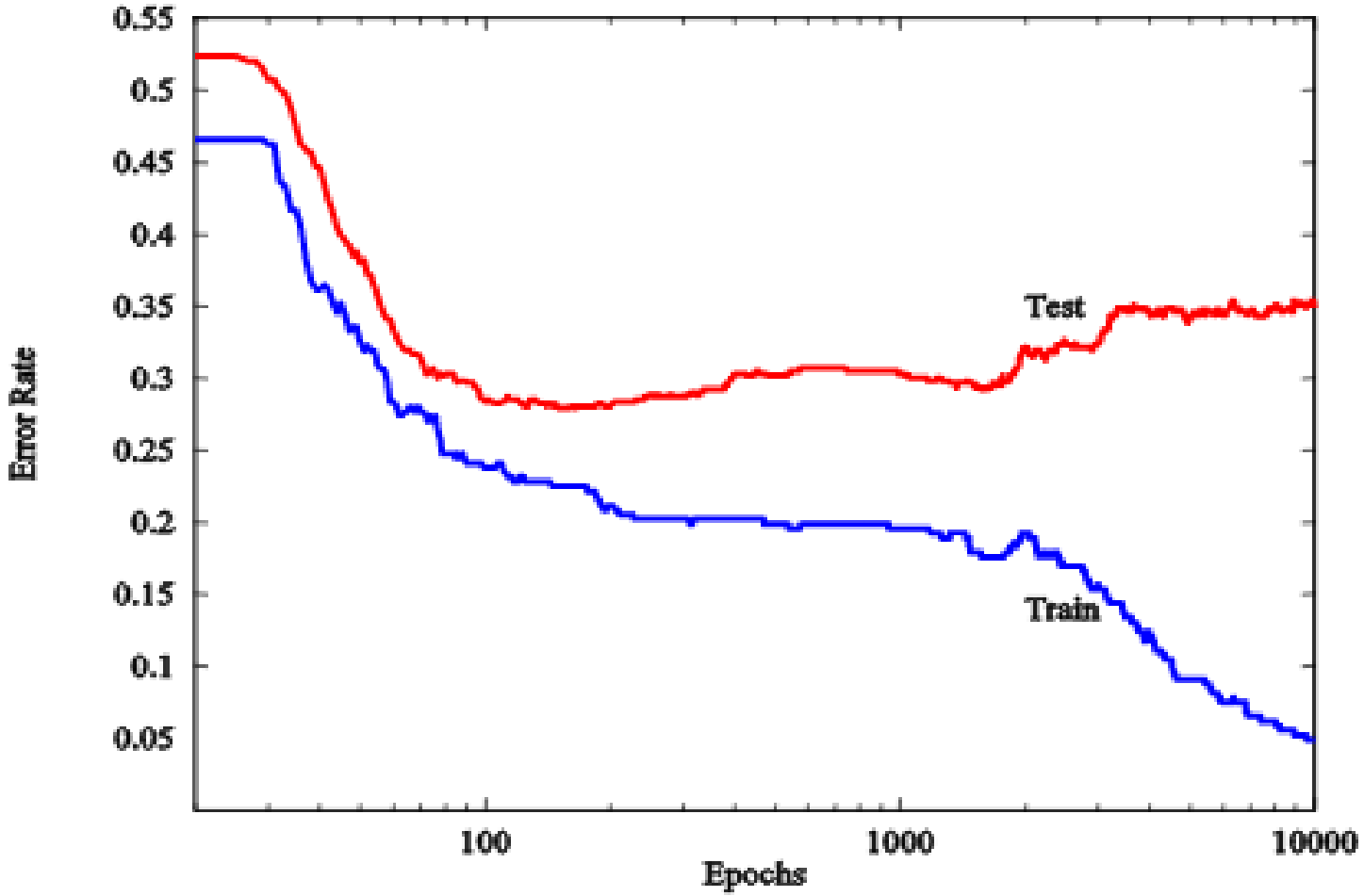


Sous-apprentissage

Modèle « correct »

Sur-apprentissage

Sur-apprentissage



Les trois ingrédients

de l'apprentissage artificiel

Trois ingrédients

1. Le choix de l'**espace des hypothèses** H
   - Généralement H ≠ F

2. Le **critère inductif**
   - Comment évaluer chaque hypothèse en fonction de S

3. La méthode d'**exploration** de H
   - Comment trouver une bonne (optimale ?) hypothèse

Plan

1. Grands types d'apprentissage
2. Apprentissage prédictif par réseaux de neurones
3. Quelles garanties ?
4. Recette pour créer des algorithmes d'apprentissage
5. Les réseaux de neurones profonds
6. Ce que l'on sait faire et les défis à relever

Comment fonder l'induction ?

Illustration : apprendre à **classer des exemples**

- Comment faire ?

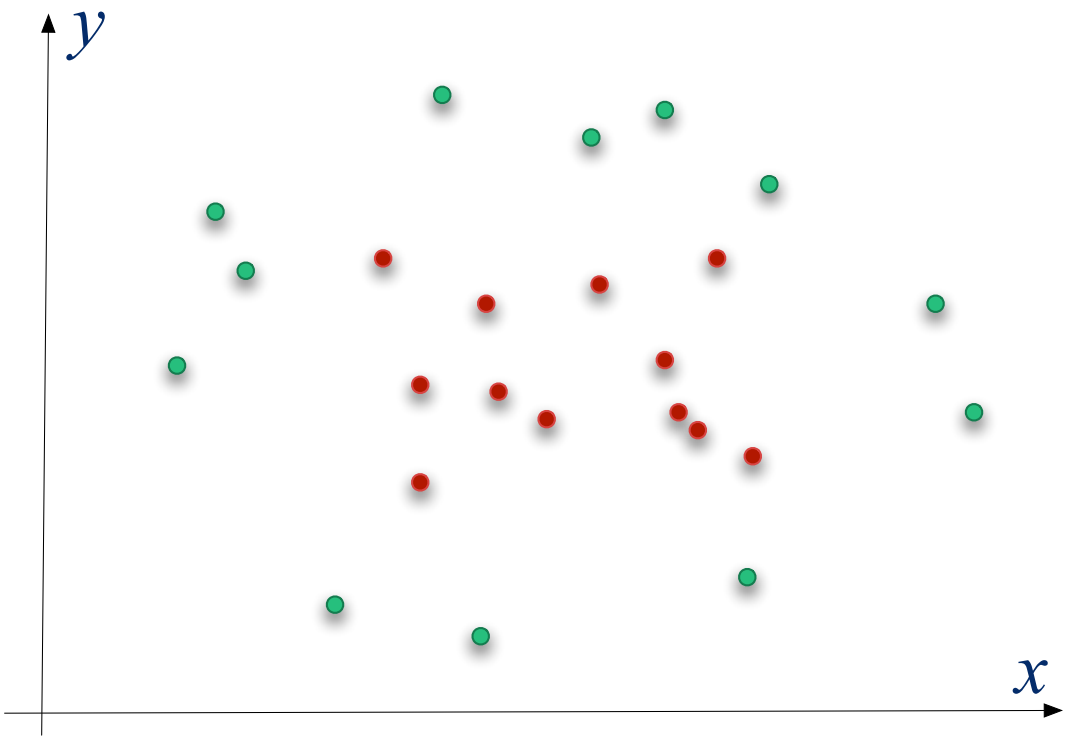


Illustration : apprendre à **classer des exemples**

- Comment faire ?

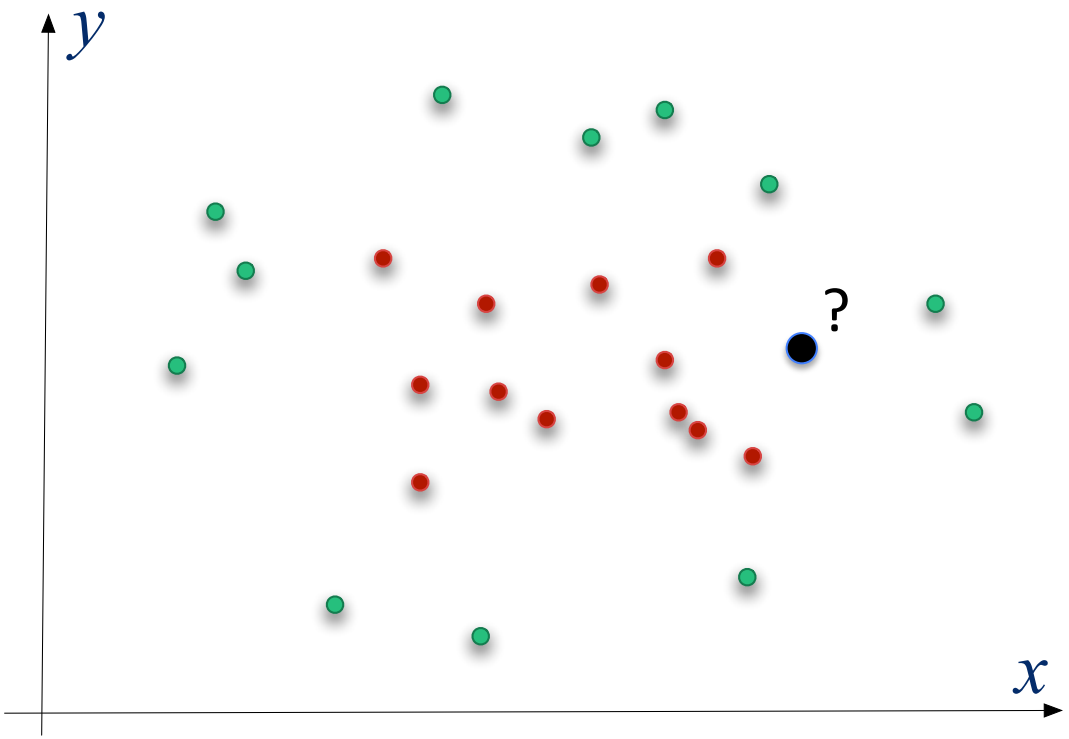
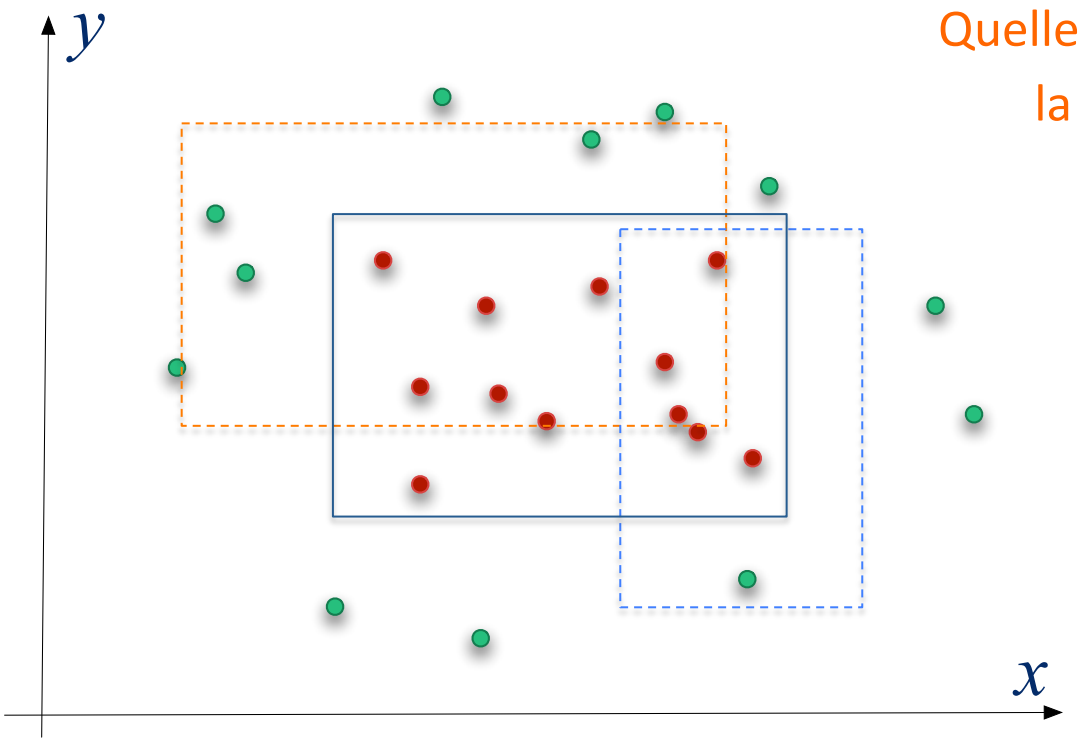


Illustration : apprendre à classer des exemples en 2D

- Comment faire ?



Quelle est l'hypothèse la meilleure ?

Quelle hypothèse choisir ?

Quelle qualité pour chaque hypothèse candidate ?

Le « critère inductif »

Quelle hypothèse choisir ?

Quelle performance ?

- Coût d'**une erreur** de prédiction
  - La ***fonction de perte***

$$\ell\big(h(\mathbf{x}), y\big)$$

Quelle hypothèse choisir ?

Quelle performance ?

- Coût d'**une erreur** de prédiction
  - La ***fonction de perte***

$$\ell\big(h(\mathbf{x}), y\big)$$

- Quel **coût à venir** (espérance) si je choisis *h* ?
  - Espérance de coût : le « ***risque réel*** »

$$R(h) = \int_{\mathcal{X}\times\mathcal{Y}} \ell\big(h(\mathbf{x}), y\big)\ \mathbf{p}_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, y)\ d\mathbf{x}\, dy$$

Quelle hypothèse choisir ?

Comment trouver *h** (ou une bonne hypothèse) alors que l'on n'a accès qu'à un **échantillon d'apprentissage limité** ?

Quelle hypothèse choisir ?

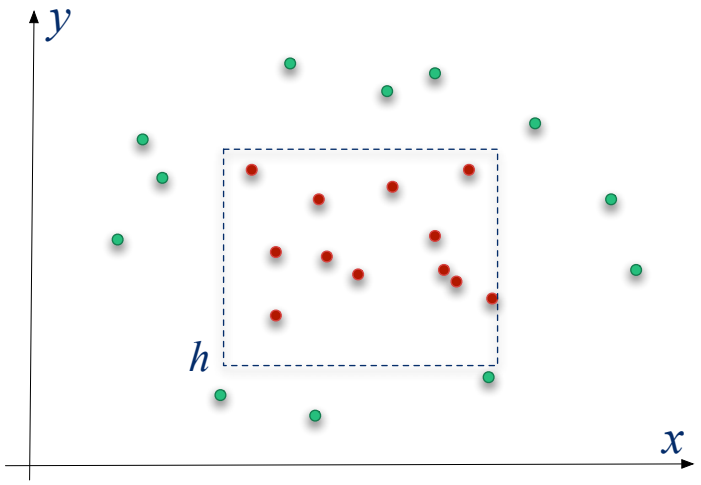
Comment trouver *h** (ou une bonne hypothèse) alors que l'on n'a accès qu'à un **échantillon d'apprentissage limité** ?

- Critère inductif : $\mathcal{H}$ x S -> valeur(h)

- Le plus naturel : ERM
  - La Minimisation du Risque Empirique

Quelle hypothèse choisir ?

- Quelle performance attendue pour *h* ?
  - Erreur moyenne sur l'échantillon d'apprentissage *S*



Le « risque empirique »

$$\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell\big(h(\mathbf{x}_i), y_i\big)$$

- A-t-on raison d'utiliser l'ERM ?

Un exemple qui en dit beaucoup …

- Exemples décrits par :

  Number (1 or 2); ***size*** (small or large); ***shape*** (circle or square); ***color*** (red or green)

Description	**Your** prediction	**True** class
1 large red square		-
1 large green square		+
2 small red squares		+
2 large red circles		-
1 large green circle		+
1 small red circle		+

How many possible functions altogether from *X* to *Y* ? $2^{2^4} = 2^{16} = 65{,}536$

How many functions do remain after 6 training examples? $2^{10} = 1024$

Un exemple qui en dit beaucoup …

- Examples described using:

  Number (1 or 2); ***size*** (small or large); ***shape*** (circle or square); ***color*** (red or green)

Description	**Your** prediction	**True** class
1 large red square		-
1 large green square		+
2 small red squares		+
2 large red circles		-
1 large green circle		+
1 small red circle		+
1 small green square		-
1 small red square		+
2 large green squares		+
2 small green squares		+
2 small red circles		+
1 small green circle		-
2 large green circles		-
2 small green circles		+
1 large red circle		-
2 large red squares	?	

15

How many remaining functions?

?

Induction: impossible de gagner ?

- **Un biais est nécessaire**

- **Types** de biais
  - **De représentation** (déclaratif)
  - **De recherche** (procédural)

Un exemple qui en dit beaucoup …

- Examples described using:

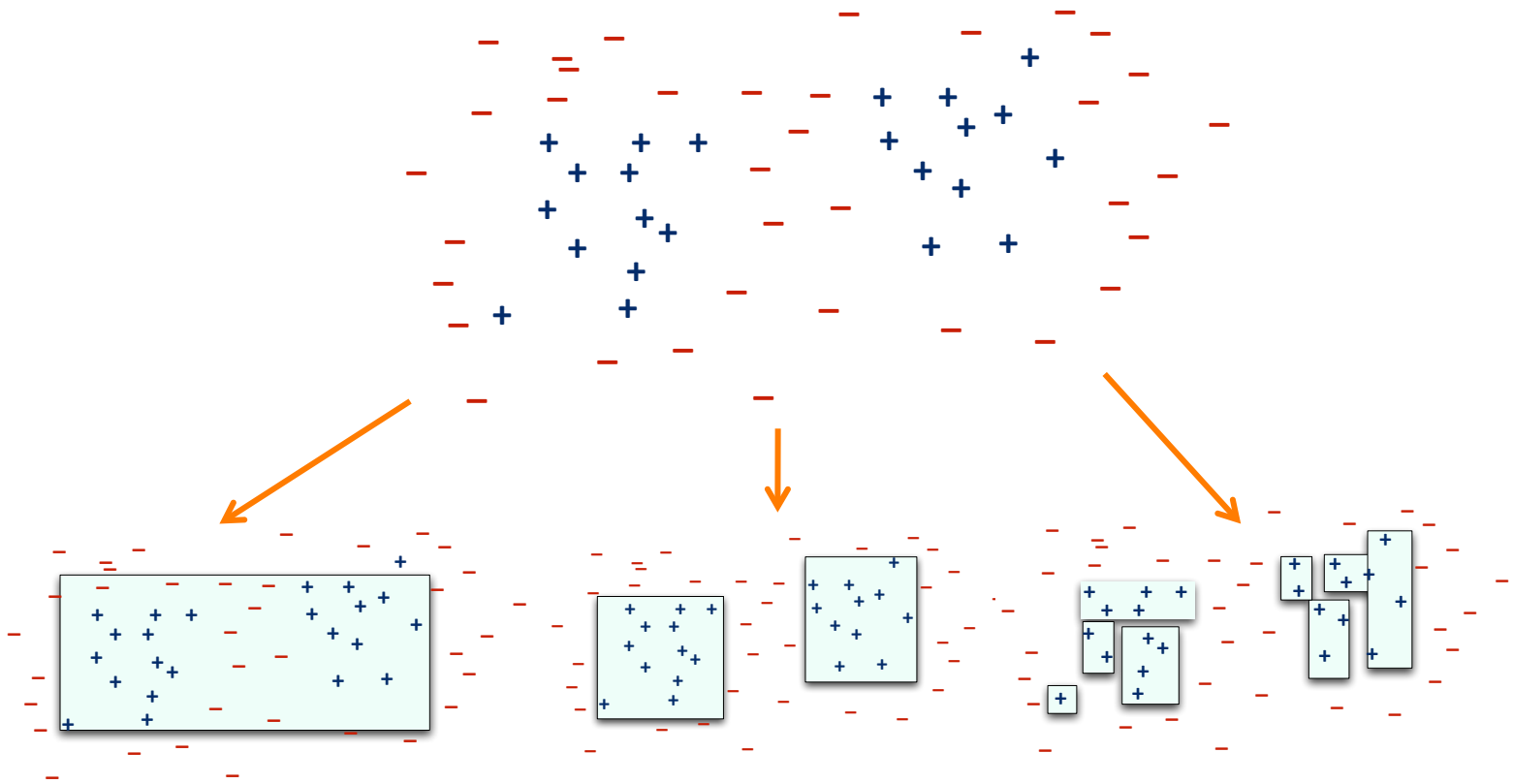
  ~~***Number***~~ (1 or 2); ***size*** (small or large); ~~***shape***~~ (circle or square); ***color*** (red or green)

Description	**Your** prediction	**True** class
~~1~~ large red ~~square~~		-
~~1~~ large green ~~square~~		+
~~2~~ small red ~~squares~~		+
~~2~~ large red ~~circles~~		-
~~1~~ large green ~~circle~~		+
~~1~~ small red ~~circle~~		+

How many possible functions with 2 descriptors from X to Y ? $2^{2^2} = 2^4 = 16$

How many functions do remain after 3 ≠ training examples? $2^1 = 2$

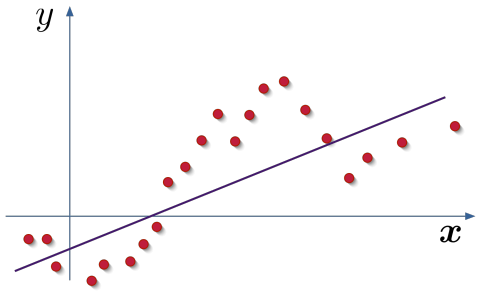
Comment garantir un niveau de performance ?



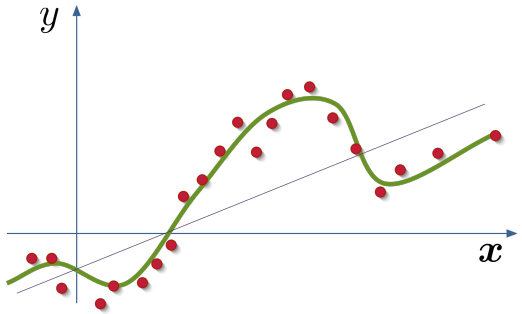
Sous-apprentissage **Bon**-apprentissage Sur-apprentissage

x

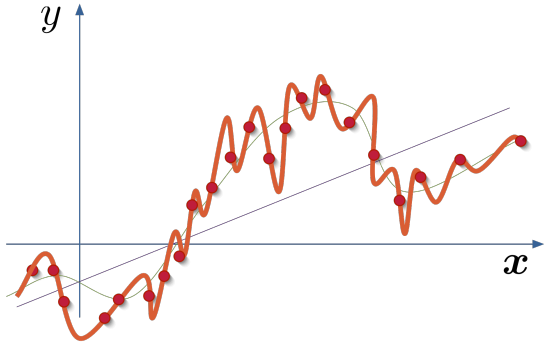
Comment garantir un niveau de performance ?



Sous-apprentissage



Bon-apprentissage



Sur-apprentissage

Question centrale : le principe inductif

- Le principe de minimisation du risque empirique (ERM)

  … est-il sain ?

  - **Si** je choisis *h* telle que $\hat{h} = \underset{h \in \mathcal{H}}{\text{ArgMin}} \ \hat{R}(h)$
  - Est-ce que *h* est bonne relativement au risque réel ?

$$\hat{R}(\hat{h}) \overset{?}{\longleftrightarrow} R(\hat{h})$$

  - Est-ce que j'aurais pu faire beaucoup mieux ? $h^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} \ R(h)$

$$R(h^*) \overset{?}{\longleftrightarrow} R(\hat{h})$$

L'analyse « PAC learning »

- On arrive à :

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad \mathbf{P}^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \overbrace{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m}}^{\varepsilon} \right] > 1 - \delta$$

Le principe de minimisation du risque empirique

n'est sain **que si** il y a des contraintes sur l'espace des hypothèses

Plan

1. Grands types d'apprentissage
2. Apprentissage prédictif par réseaux de neurones
3. Quelles garanties ?
4. Recette pour créer des algorithmes d'apprentissage
5. Les réseaux de neurones profonds
6. Ce que l'on sait faire et les défis à relever

Recette pour … **concevoir des algorithmes** d'apprentissage

1. Définir un **critère inductif régularisé**
   a. Exprimer le coût d'erreur de prédiction en une fonction de perte
   b. Définir un **terme de régularisation** qui exprime
      les attendus sur les régularités du monde
   c. Si possible, rendre convexe le problème d'**optimisation** résultant

$$h_{opt} = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left[\underbrace{\frac{1}{m} \sum_{i=1}^{m} l(h(\mathbf{x}_i), y_i)}_{\text{empirical risk}} + \lambda \underbrace{reg(\mathcal{H})}_{\text{bias on the world}} \right]$$

2. Utiliser ou développer un **algorithme d'optimisation efficace**

Learning **sparse linear** approximator

- The **hypothesis** is of the form $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
- A priori assumption: few non zero coefficients

Ridge regression

$$\mathbf{w}^*_{\text{ridge}} = \underset{\mathbf{w}}{\text{Argmin}} \left\{ \sum_{i=1}^{m} (y_i - \mathbf{w}\,\mathbf{x}_i)^2 + \lambda ||\mathbf{w}||_2^2 \right\}$$

Lasso regression

$$\mathbf{w}^*_{\text{lasso}} = \underset{\mathbf{w}}{\text{Argmin}} \left\{ \sum_{i=1}^{m} (y_i - \mathbf{w}\,\mathbf{x}_i)^2 + \lambda ||\mathbf{w}||_1 \right\}$$

3.3 du chapitre 3. Ainsi, étant donnés un échantillon source étiqueté $S = \{(x_i^s, y_i^s)\}_{i=1}^m$ constitué de m exemples *i.i.d.* selon P_S et un échantillon cible non étiqueté $T = \{(x_i^t)\}_{i=1}^m$ composé de m exemples *i.i.d.* selon D_T, en posant $S_u = \{x_i^s\}_{i=1}^m$ l'échantillon S privé de ses étiquettes, on veut minimiser :

Risque empirique régularisé

$$\min_{\mathbf{w}} \; c\, m\, \mathrm{R}_S(G_{\rho_\mathbf{w}}) + a\, m\, \mathrm{dis}_{\rho_{\mathbf{w}^2}}(S_u, T_u) + \mathrm{KL}(\rho_\mathbf{w} \| \pi_0), \tag{7.5}$$

où $\mathrm{dis}_{\rho_{\mathbf{w}^2}}(S_u, T_u) = \left| \mathop{\mathrm{E}}_{(h,h')\sim\rho_{\mathbf{w}^2}} \mathrm{R}_{S_u}(h,h') - \mathop{\mathrm{E}}_{(h,h')\sim\rho_{\mathbf{w}^2}} \mathrm{R}_{T_u}(h,h') \right|$ est le désaccord empirique entre S_u et T_u spécialisé à une distribution $\rho_\mathbf{w}$ sur l'espace $\mathcal{H}$ des classifieurs linéaires considéré. Les réels $a > 0$ et $c > 0$ sont des hyperparamètres de l'algorithme. Notons que les constantes A et C du théorème 7.7 peuvent être retrouvées à partir de n'importe quelle valeur de a et c. Étant donnée la fonction $\ell_{\mathrm{dis}}(x) = 2\,\ell_{\mathbf{Erf}}(x)\,\ell_{\mathbf{Erf}}(-x)$ (illustrée sur la figure 7.1) , pour toute distribution D sur X, on a :

$$\begin{aligned}
\mathop{\mathrm{E}}_{(h,h')\sim\rho_{\mathbf{w}^2}} \mathrm{R}_D(h,h') &= \mathop{\mathrm{E}}_{\mathbf{x}\sim D} \mathop{\mathrm{E}}_{(h,h')\sim\rho_{\mathbf{w}^2}} \mathrm{I}\left[h(\mathbf{x}) \neq h'(\mathbf{x})\right] \\
&= 2 \mathop{\mathrm{E}}_{\mathbf{x}\sim D} \mathop{\mathrm{E}}_{(h,h')\sim\rho_{\mathbf{w}^2}} \mathrm{I}\left[h(\mathbf{x}) = 1\right] \mathrm{I}\left[h'(\mathbf{x}) = -1\right] \\
&= 2 \mathop{\mathrm{E}}_{\mathbf{x}\sim D} \mathop{\mathrm{E}}_{h\sim\rho_\mathbf{w}} \mathrm{I}\left[h(\mathbf{x}) = 1\right] \mathop{\mathrm{E}}_{h'\sim\rho_\mathbf{w}} \mathrm{I}\left[h'(\mathbf{x}) = -1\right] \\
&= 2 \mathop{\mathrm{E}}_{\mathbf{x}\sim D} \ell_{\mathbf{Erf}}\left(\frac{\langle \mathbf{w}, \mathbf{x}\rangle}{\|\mathbf{x}\|}\right) \ell_{\mathbf{Erf}}\left(-\frac{\langle \mathbf{w}, \mathbf{x}\rangle}{\|\mathbf{x}\|}\right) \\
&= \mathop{\mathrm{E}}_{\mathbf{x}\sim D} \ell_{\mathrm{dis}}\left(\frac{\langle \mathbf{w}, \mathbf{x}\rangle}{\|\mathbf{x}\|}\right).
\end{aligned}$$

Expression de **substitution** du risque régularisé

Ainsi, trouver la solution optimale de l'équation (7.5) revient à chercher le vecteur $\mathbf{w}$ qui minimise :

$$c \sum_{i=1}^m \ell_{\mathbf{Erf}}\left(y_i^s \frac{\langle \mathbf{w}, \mathbf{x}_i^s\rangle}{\|\mathbf{x}_i^s\|}\right) + a \left| \sum_{i=1}^m \left[\ell_{\mathrm{dis}}\left(\frac{\langle \mathbf{w}, \mathbf{x}_i^s\rangle}{\|\mathbf{x}_i^s\|}\right) - \ell_{\mathrm{dis}}\left(\frac{\langle \mathbf{w}, \mathbf{x}_i^t\rangle}{\|\mathbf{x}_i^t\|}\right) \right] \right| + \frac{\|\mathbf{w}\|^2}{2}. \tag{7.6}$$

Optimisation

L'équation précédente est fortement non convexe. Afin de rendre sa résolution plus facilement contrôlable, nous remplaçons la fonction $\ell_{\mathbf{Erf}}(\cdot)$ par sa relaxation convexe $\ell_{\mathbf{Erf}_{\mathrm{cvx}}}(\cdot)$ (comme pour PBGD3 et illustrée sur la figure 7.1). L'optimisation se réalise ensuite par une descente de gradient. Le gradient de l'équation 7.6 étant :

Un paradigme général

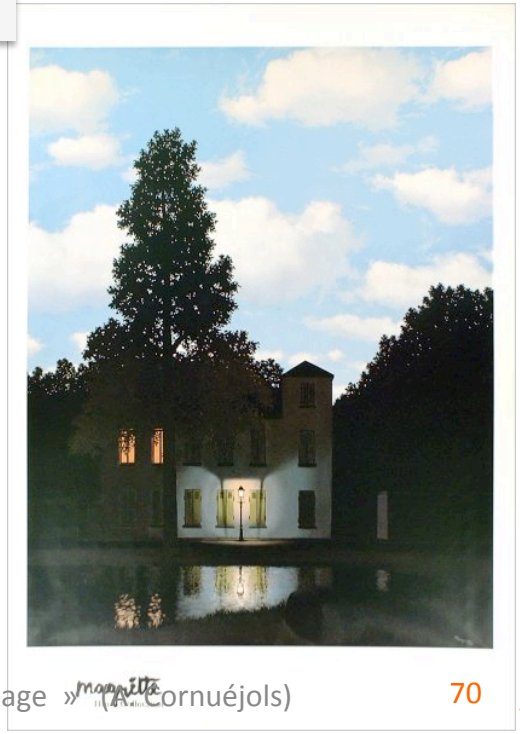
- Boosting
- Arbres de décisions (random forests)
- Régression logistique
- Réseaux de neurones
- Séparateurs à Vastes Marges (SVM)
- …

Des garanties « de lampadaire »

Des garanties « de lampadaire »

(*Quasi*) garantie que :

- **Si** le monde satisfait **mes attentes** sur lui
- **Alors** l'algorithme d'apprentissage produira une bonne hypothèse (proche de la vraie)



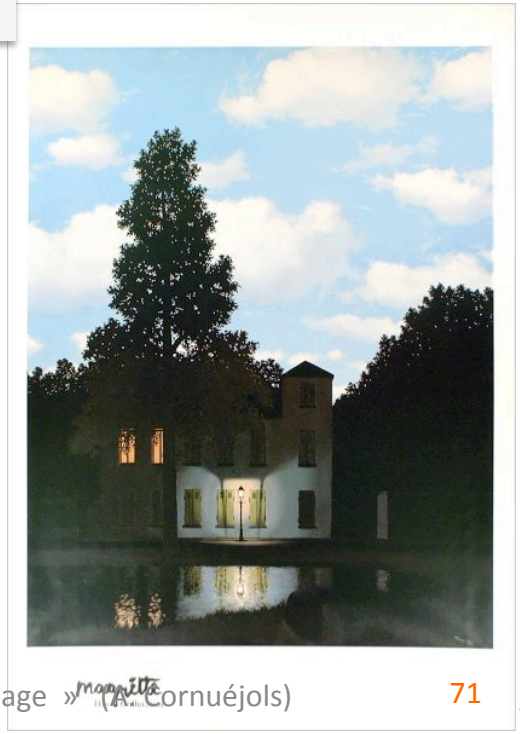
Des garanties « de lampadaire »

(*Quasi*) garantie que :

- **Si** le monde satisfait **mes attentes** sur lui
- **Alors** l'algorithme d'apprentissage produira une bonne hypothèse (proche de la vraie)

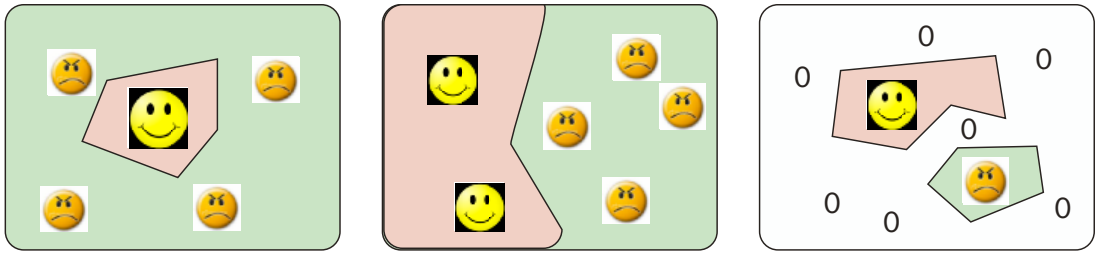
- **Autrement** l'apprentissage peut conduire à de très mauvaises hypothèses

  (ex. *Si le monde **n'est pas** parcimonieux*)



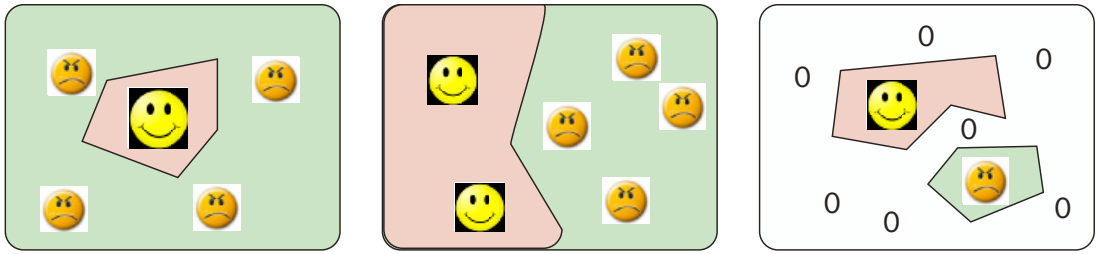
Le no-free-lunch theorem

Possible

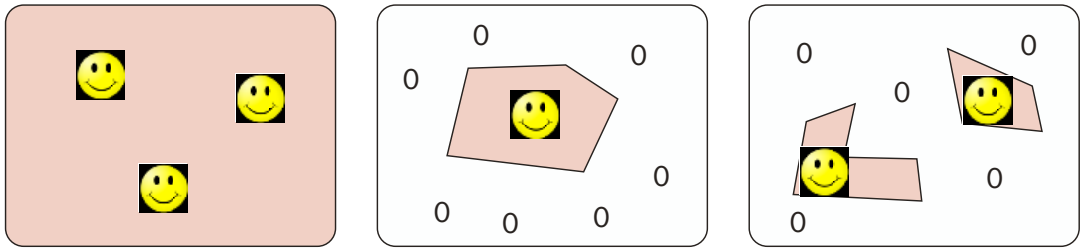


Le no-free-lunch theorem

Possible



Impossible

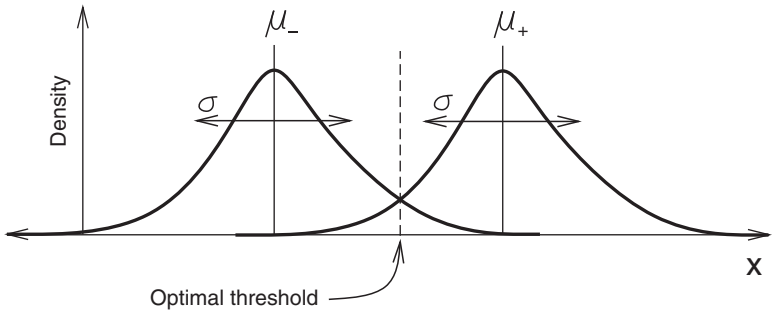


Il faut **choisir** le **bon algorithme** pour la **classe de problèmes** étudiée

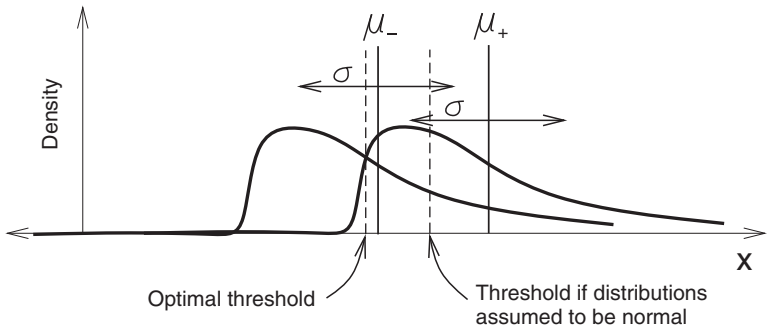
Différence entre …

- Apprentissage automatique
  - On cherche une règle / frontière de décision
  - Approche « discriminative »

- Approche statistique
  - On passe par la définition / estimation d'une distribution de probabilité
  - Et on applique la règle de Bayes
  - Approche « générative »



Une différence … significative

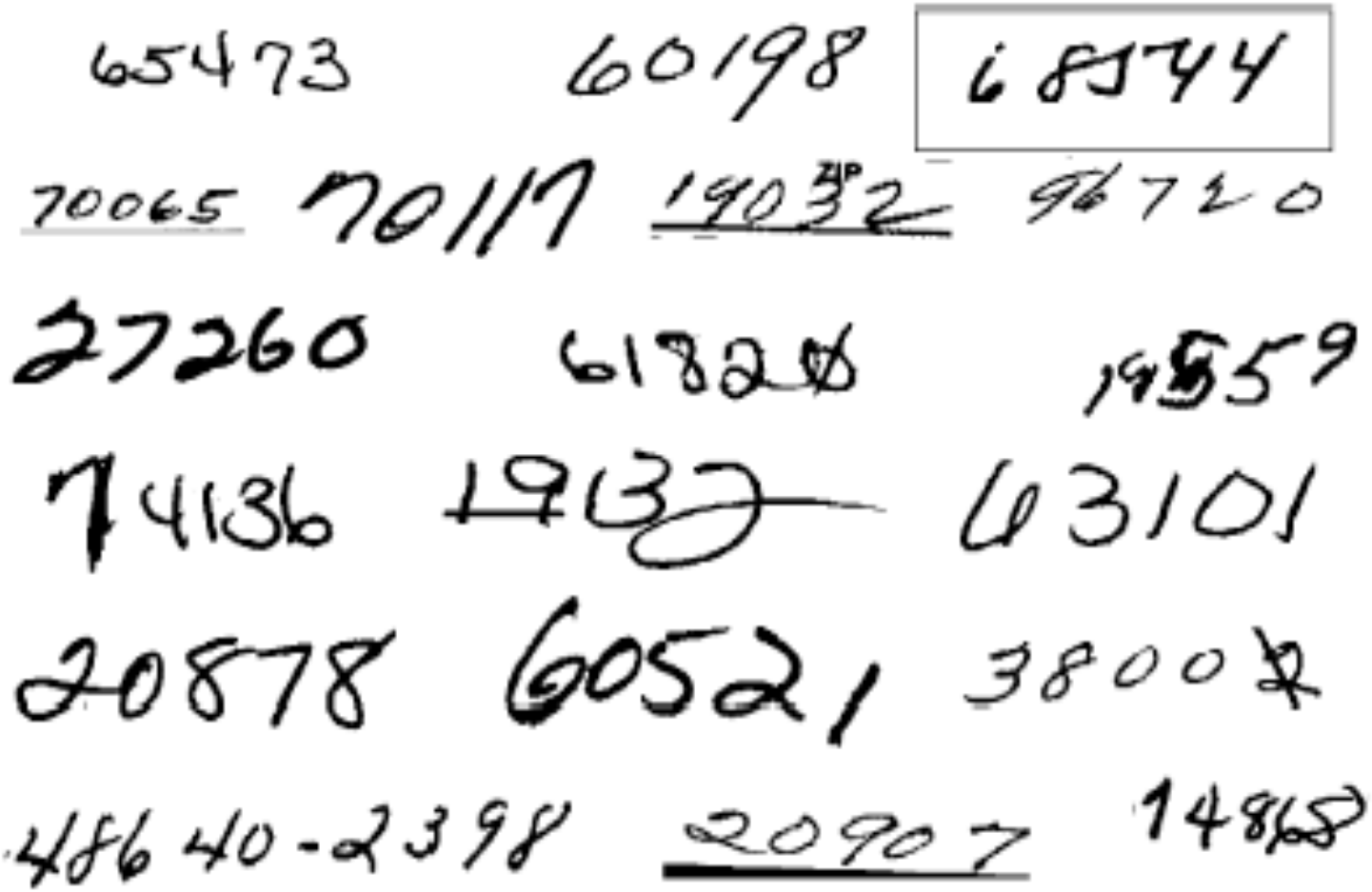


Plan

1. Grands types d'apprentissage
2. Apprentissage prédictif par réseaux de neurones
3. Quelles garanties ?
4. Recette pour créer des algorithmes d'apprentissage
5. Les réseaux de neurones profonds
6. Ce que l'on sait faire et les défis à relever

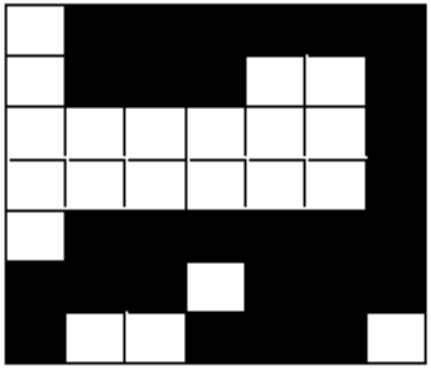
Les réseaux de neurones **profonds**

La base de données

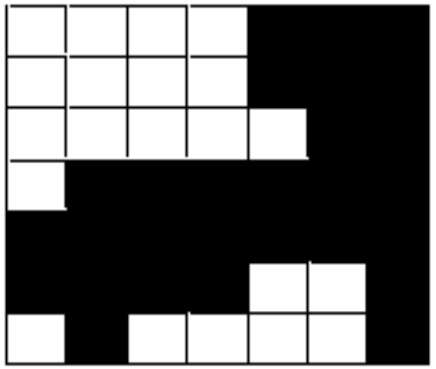


How to code the inputs

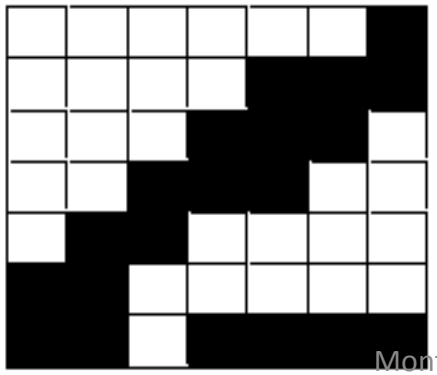
Learning is easy when we know what to look for



- Yes

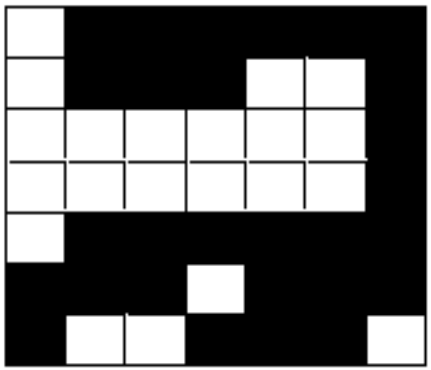


- Yes



- No

Inputs and prior knowledge



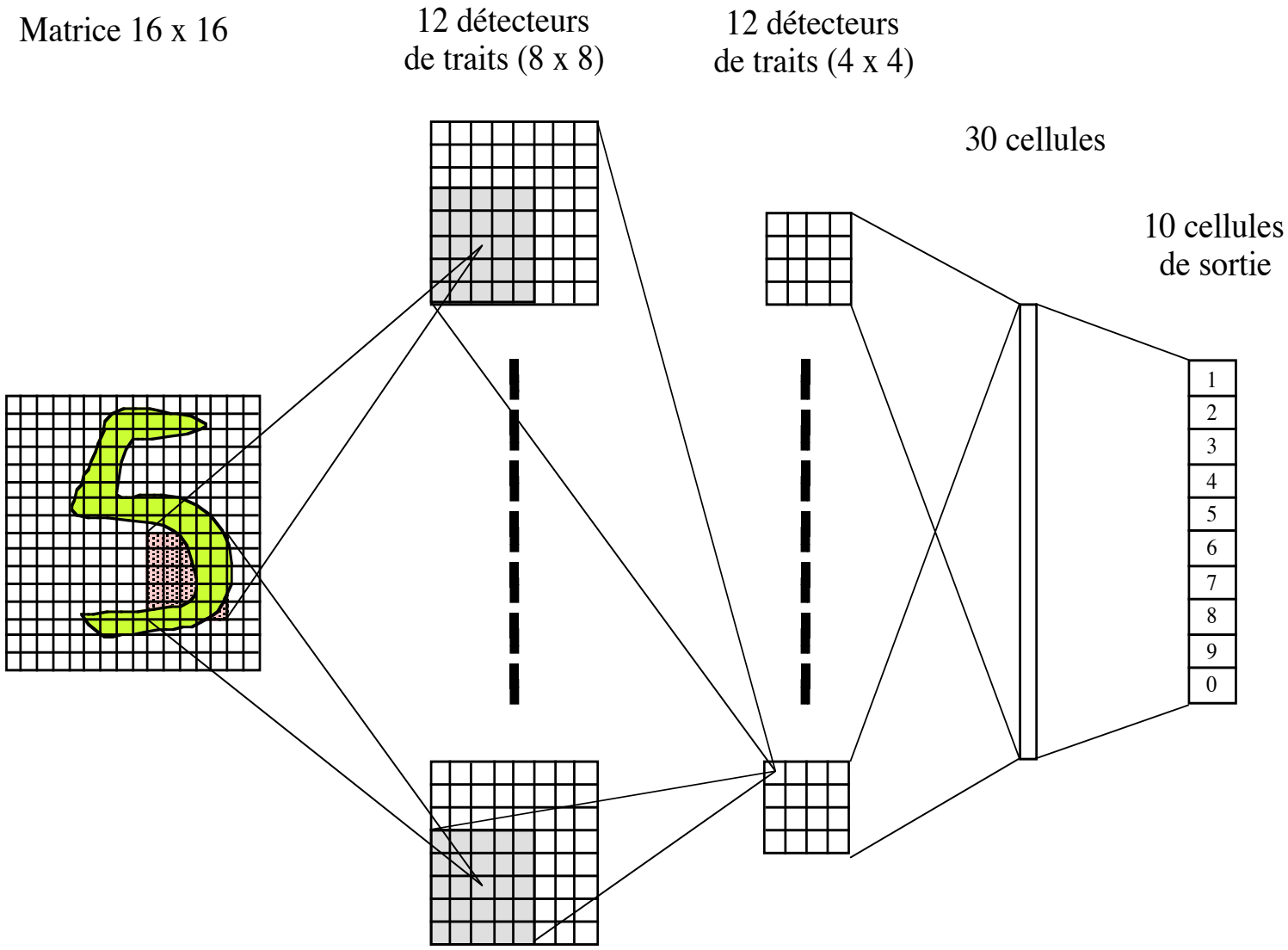
- Is it a pattern recognition task? A character recognition task? …
- *How to code the examples?*

0 1 1 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0

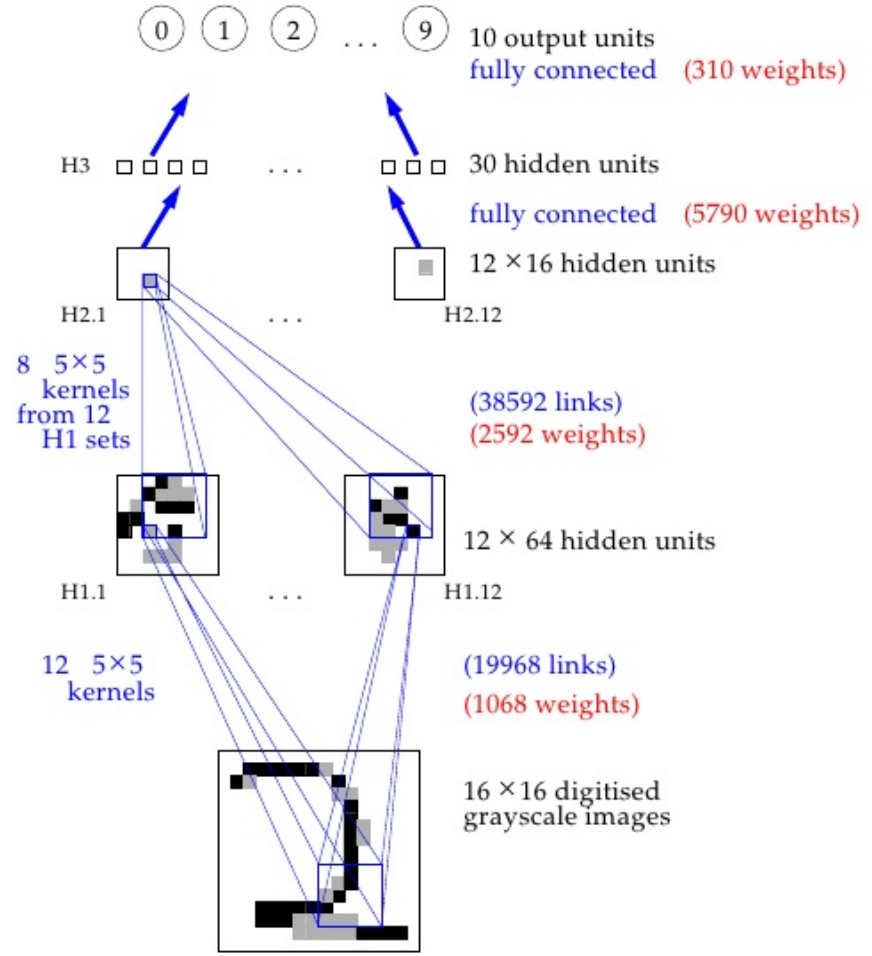
- A right choice of representation can render the learning task trivial

➠ But how can we know the right representation?

Réseaux à convolution : Application aux codes postaux

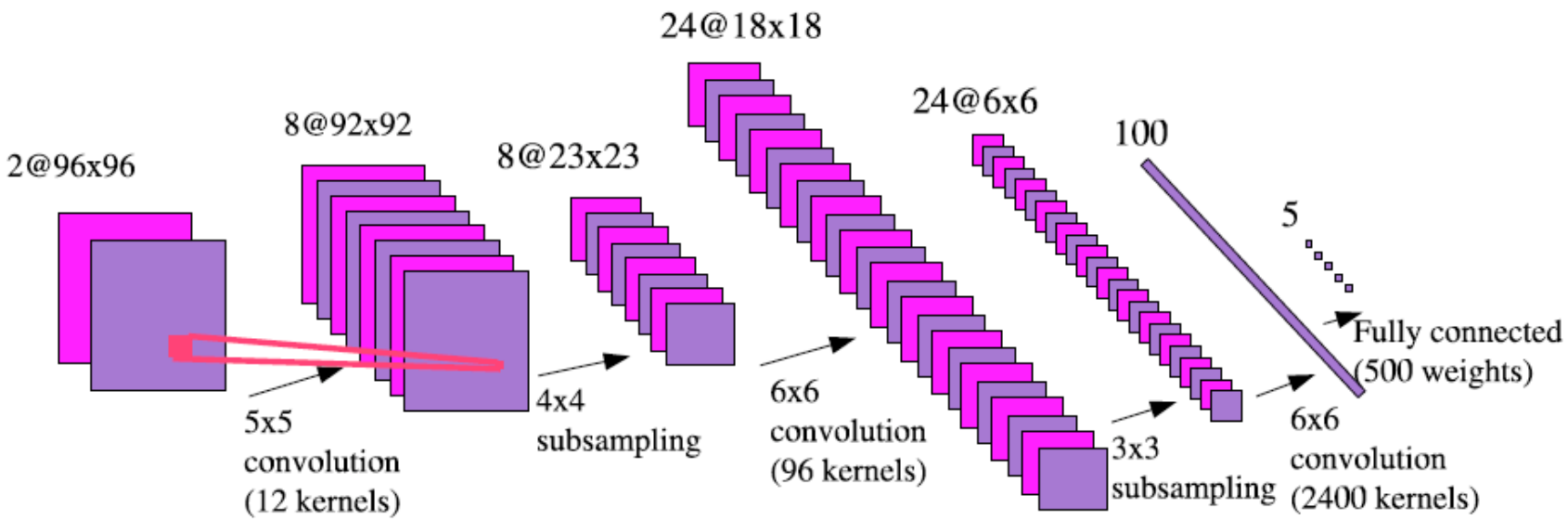


Réseaux à convolution : Application aux codes postaux

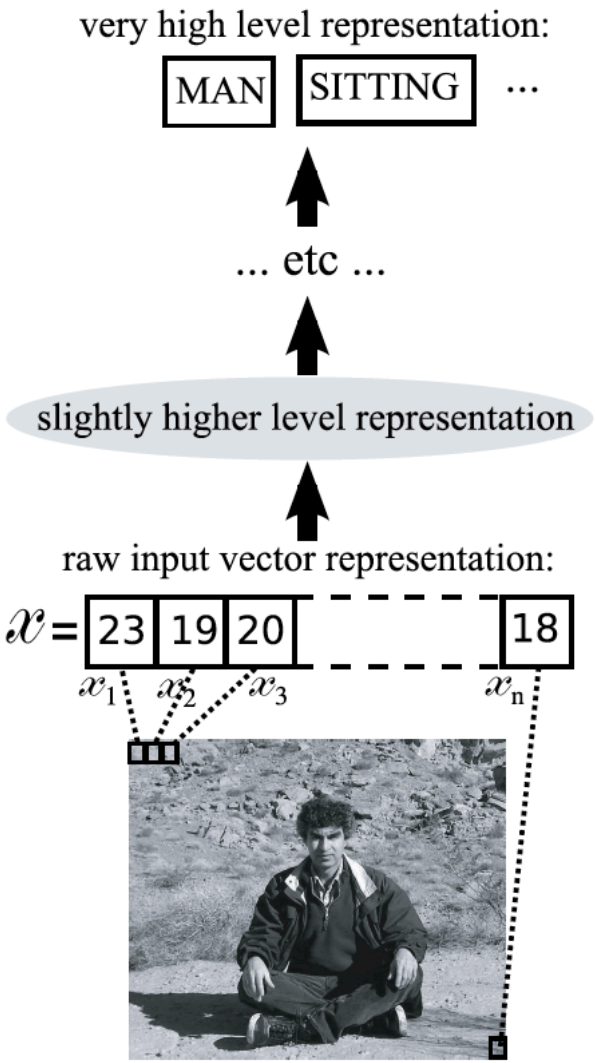


- Taken from http://image.slidesharecdn.com/bp2slides-090922011749-phpapp02/95/the-back-propagation-learning-algorithm-10-728.jpg?cb=1253582278

Réseau connexionniste pour la reconnaissance de chiffres manuscrits



« Deep belief networks »



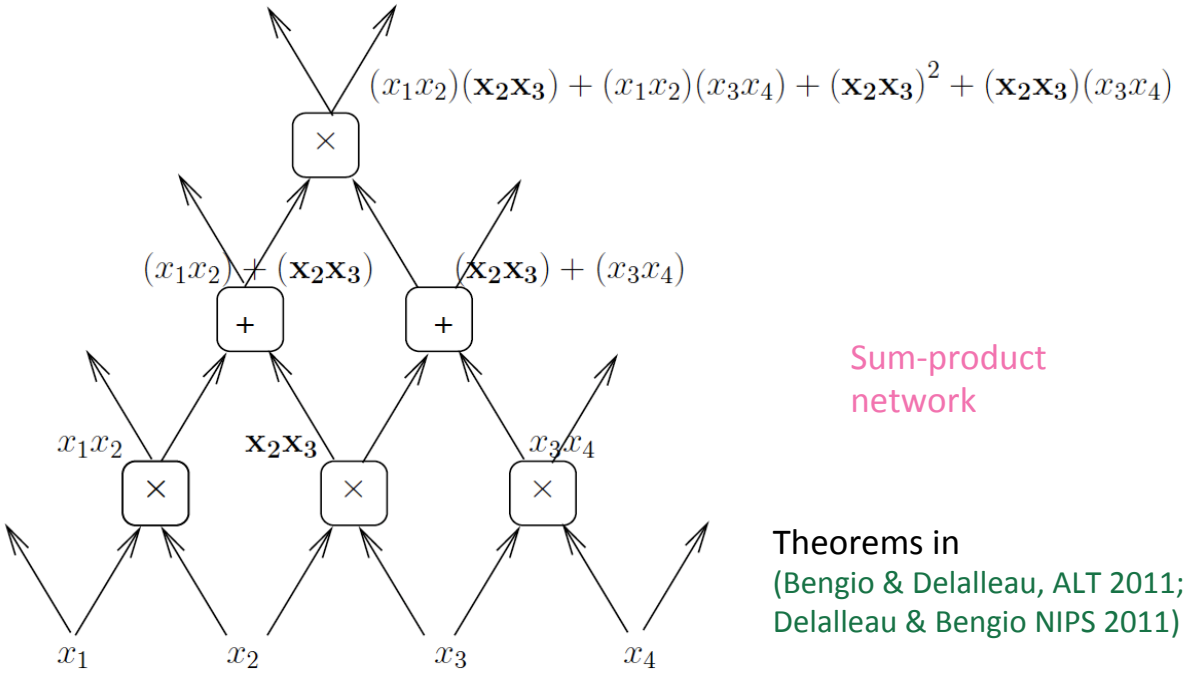
Les « réseaux de neurones profonds »

- Des réseaux de neurones artificiels

  1. à grand nombre de couches
  2. et **très grand nombre de paramètres**
  3. qui apprennent des **représentations hiérarchiques**
  4. et **décomposent les calculs**

Les « réseaux de neurones profonds »

- Des réseaux de neurones artificiels
  - et **décomposent les calculs**

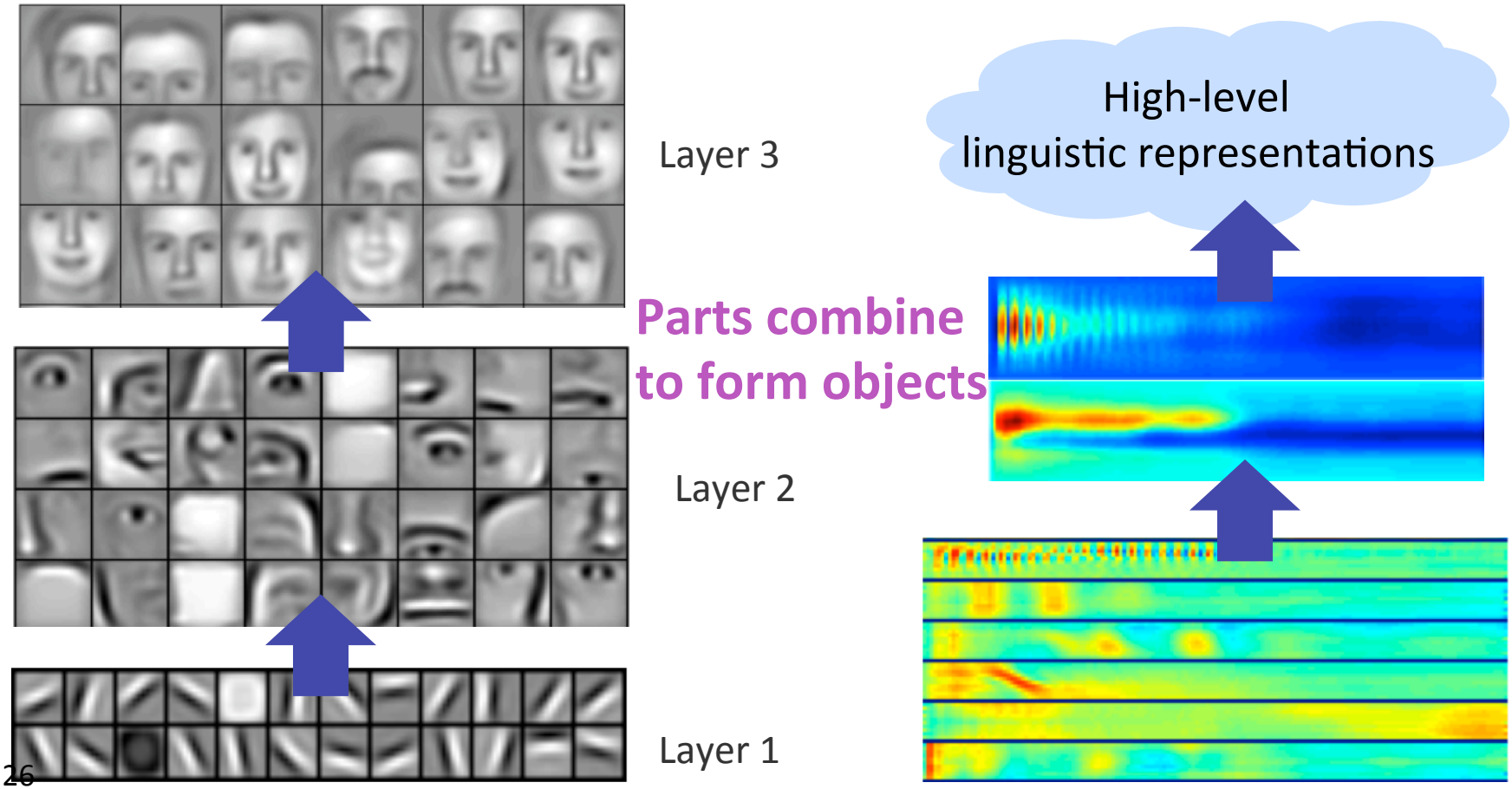


Sum-product network

Theorems in
(Bengio & Delalleau, ALT 2011;
Delalleau & Bengio NIPS 2011)

Apprentissage de représentations hiérarchiques

- Apprentissage de représentations hiérarchiques



26

Illustration : ImageNet

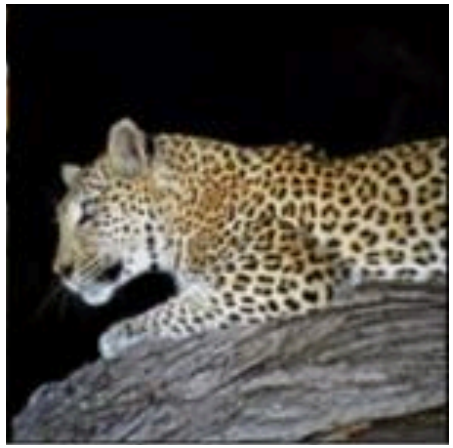
La compétition ImageNet

- Plus de **15M d'images** haute résolution étiquetées
- Environ **22K catégories**
- Récoltées sur le Web et étiquetées par Amazon Mechanical Turk

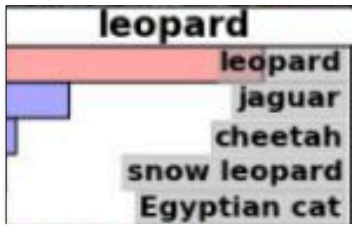
Illustration : ImageNet

La compétition ImageNet

- Plus de **15M d'images** haute résolution étiquetées
- Environ **22K catégories**
- Récoltées sur le Web et étiquetées par Amazon Mechanical Turk



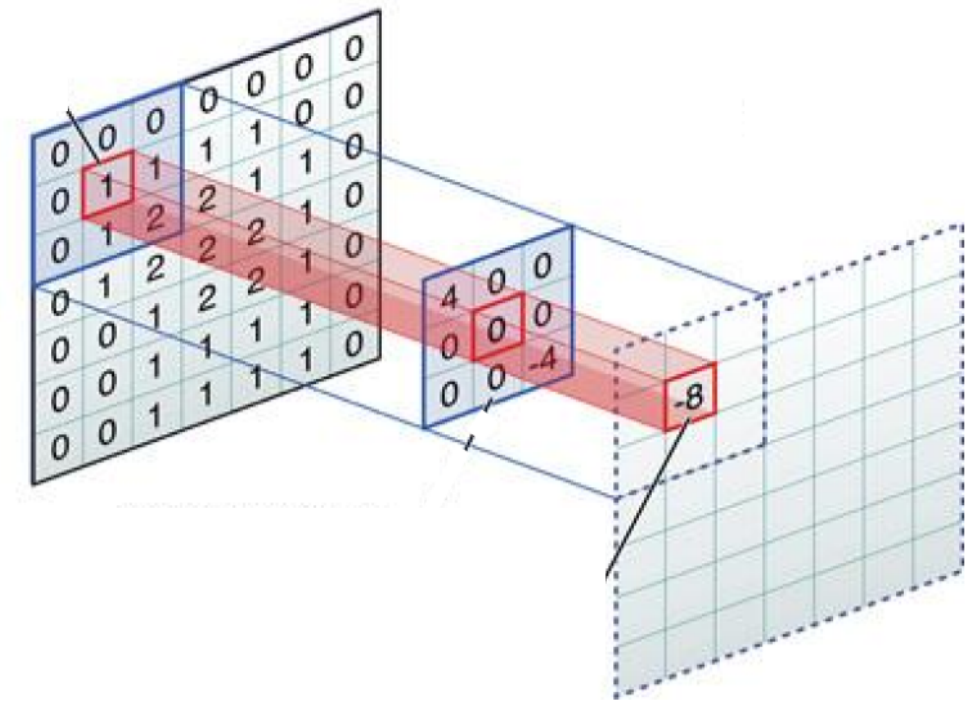
Classification



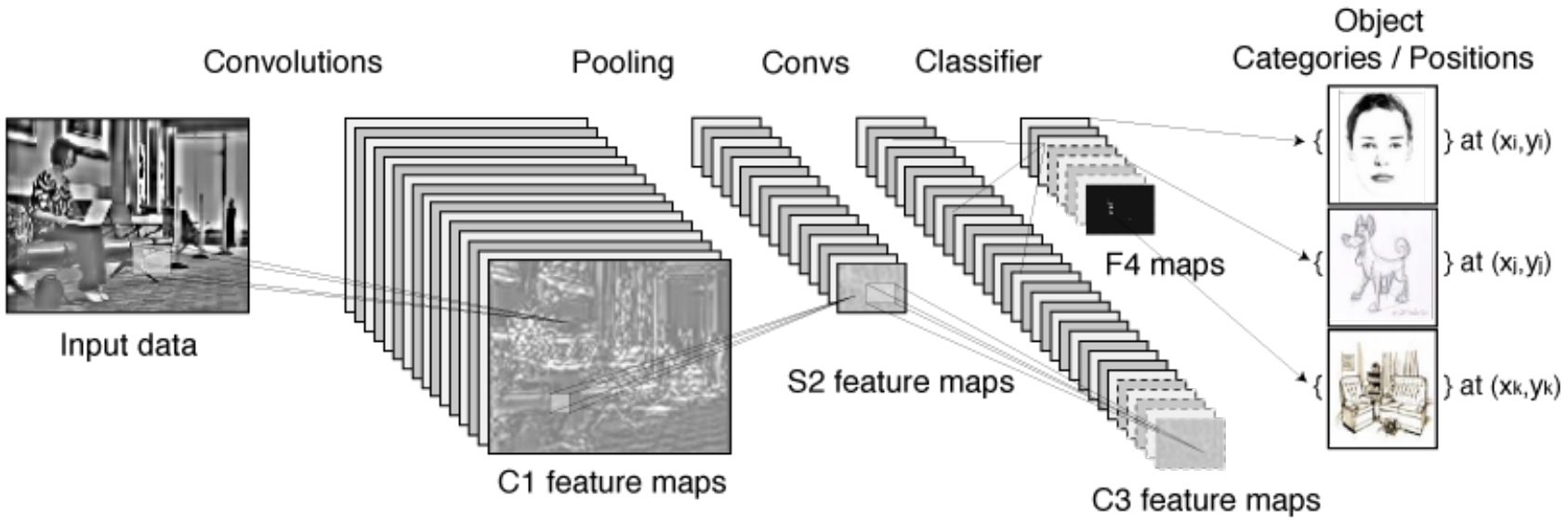
Convolutions

Same dimension between layers

(4x0) + (0x0) + (0x0) + (0x0) + (0x1) + (0x1) + (0x0) + (0x1)+ (-4x2) = -8



La structure des **réseaux de convolution**



1) Le **pooling** consiste réduire la résolution des images filtrées, par exemple en moyennant les pixels contigus *(ex : 4 pixels deviennent 1 seul pixel qui contient une valeur moyenne)* - étape S2

2) Convolution : des filtres sont appris sur les nouvelles images - étape C3

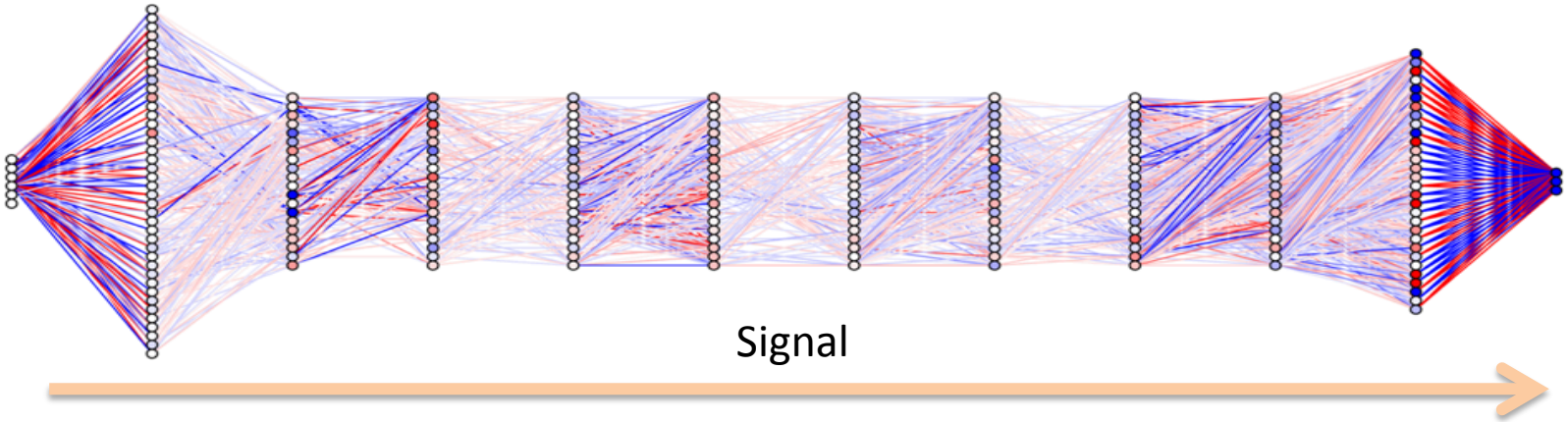
3) Classifieur : les dernières couches sont entièrement connectées *(i.e. MLP)* et apprennent à prédire la classe à partir des filtres appris *(i.e. des descripteurs générés automatiquement)* - étape F4

The SuperVision network

Image classification with deep convolutional neural networks

http://image-net.org/challenges/LSVRC/2012/supervision.pdf

- 7 hidden "weight" layers
- 650K neurons
- **60M** parameters
- 630M connections



The SuperVision network

Image classification with deep convolutional neural networks

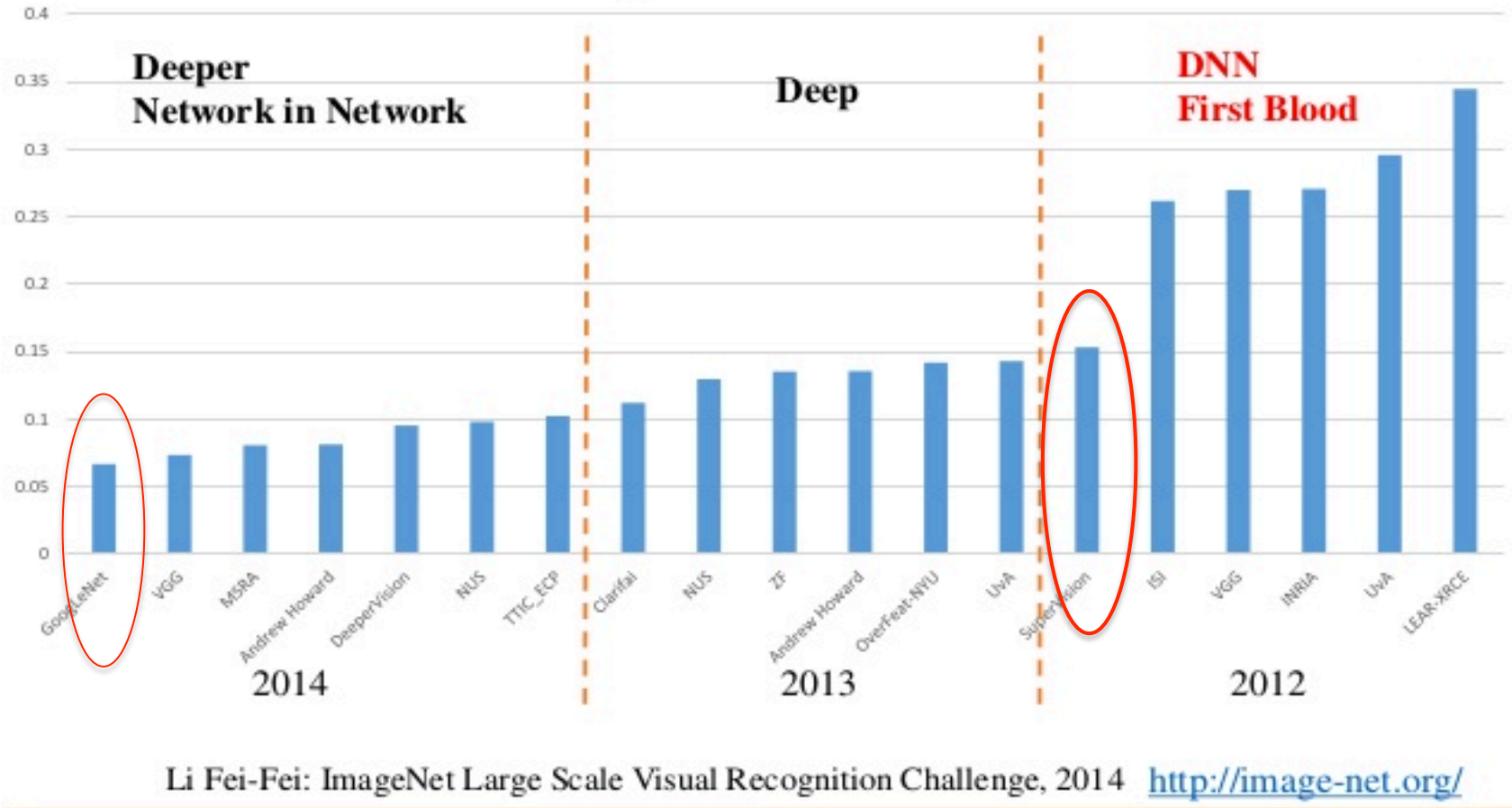
http://image-net.org/challenges/LSVRC/2012/supervision.pdf

- 7 hidden “weight” layers
- 650K neurons
- 60M parameters
- 630M connections

- Rectified Linear Units (ReLU)
- Overlapping pooling
- Dropout trick
- Randomly extracted 224x224 patches for more data

ILSVRC

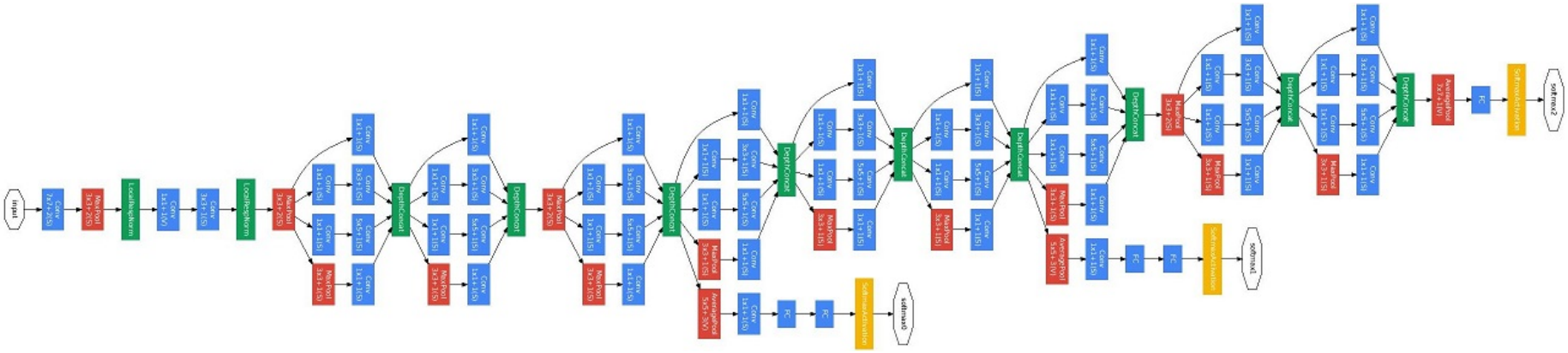
ImageNet classification error throughout years and groups



Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014 http://image-net.org/

GoogleNet

- Un **mécano** de réseaux de neurones

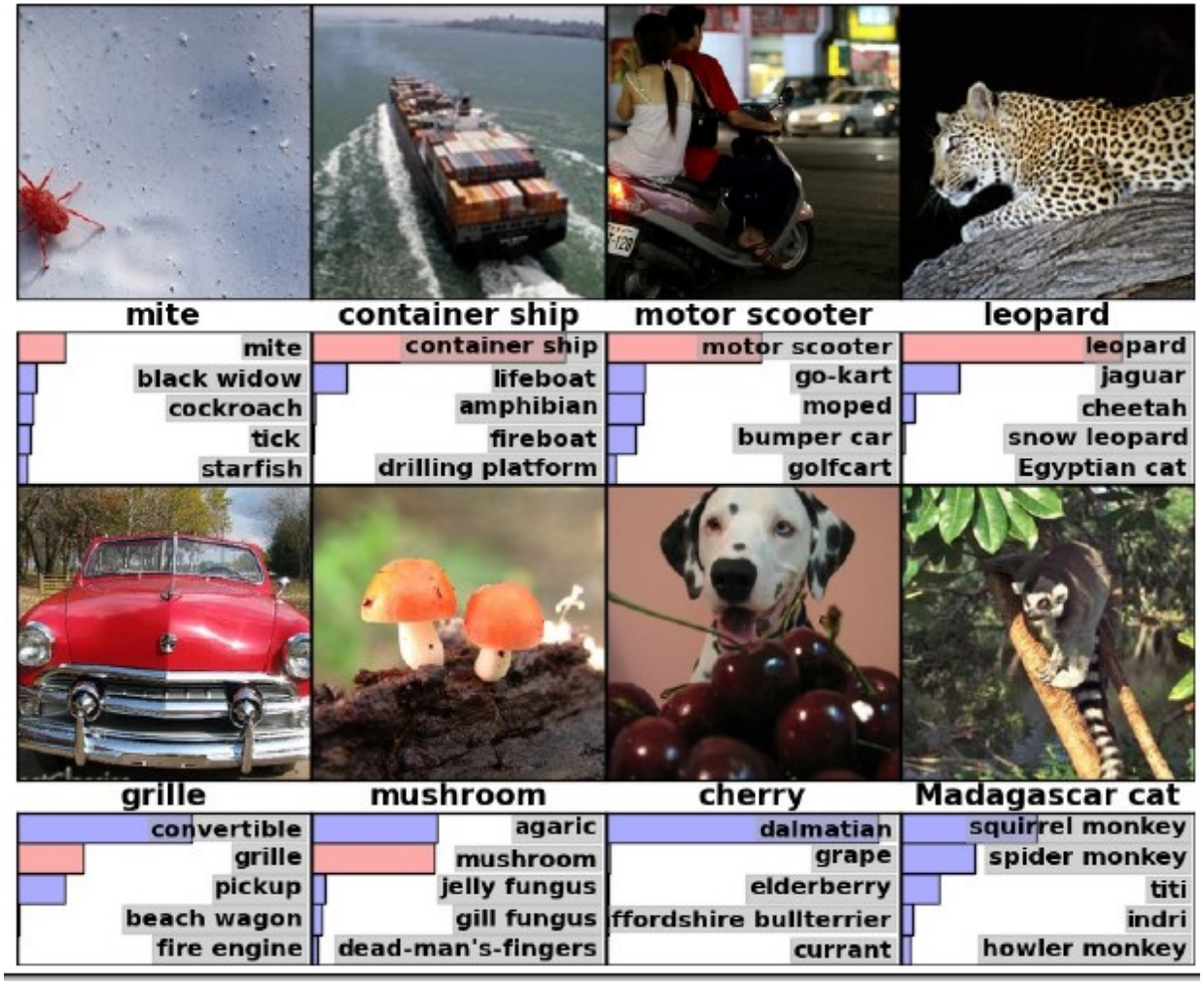


Illustration

Système développé par Google et U. de Stanford

- Reconnaissance de visages
  - Sous conditions de lumière diverses
  - Sous tout angle
- Apprentissage non supervisé
  - 9 couches ; 10^9 connexions
  - 10 millions d'images
  - 3 jours de calcul sur 16 000 processeurs
- Amélioration des performances de 70% / état de l'art

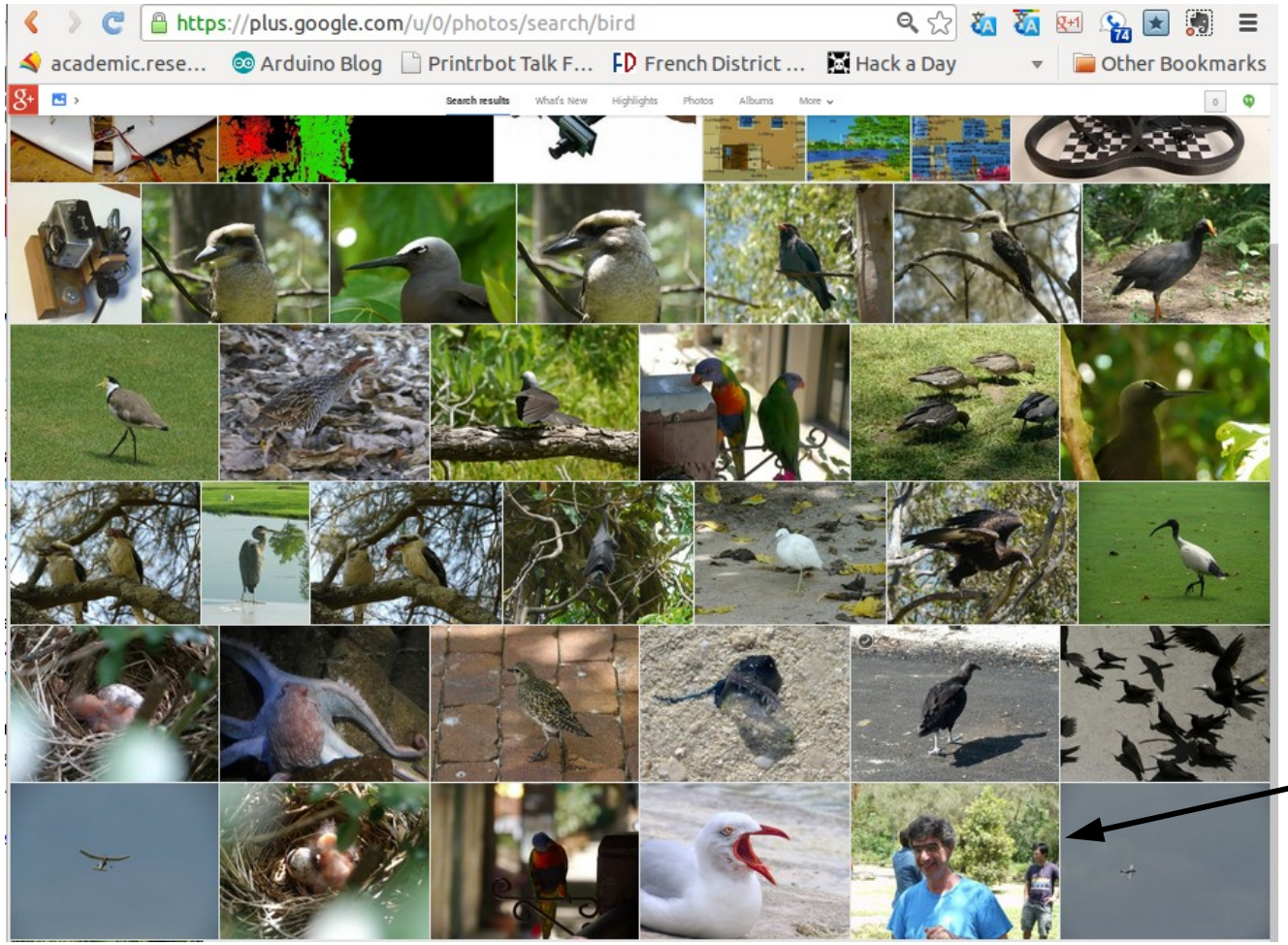
Object recognition



- [Krizhevsky, Sutskever and Hinton (2012)]

Object retrieval. ConvNet-Based Google+ Photo Tagger

Searched my personal collection for "bird"



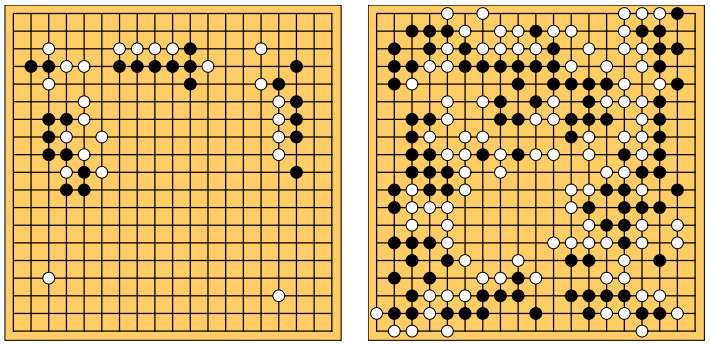
Annotation d'images

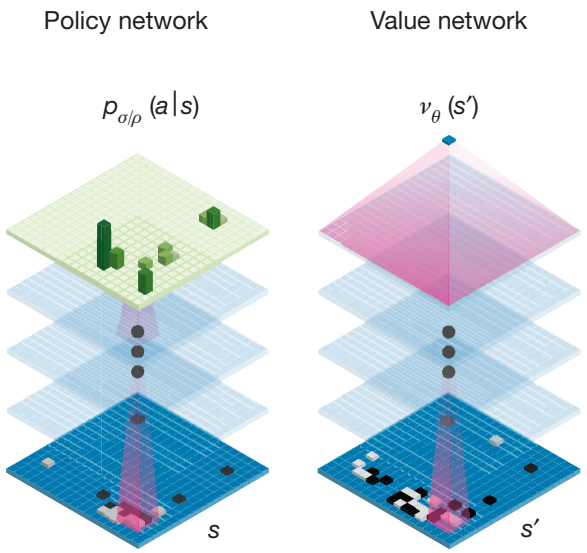
- 

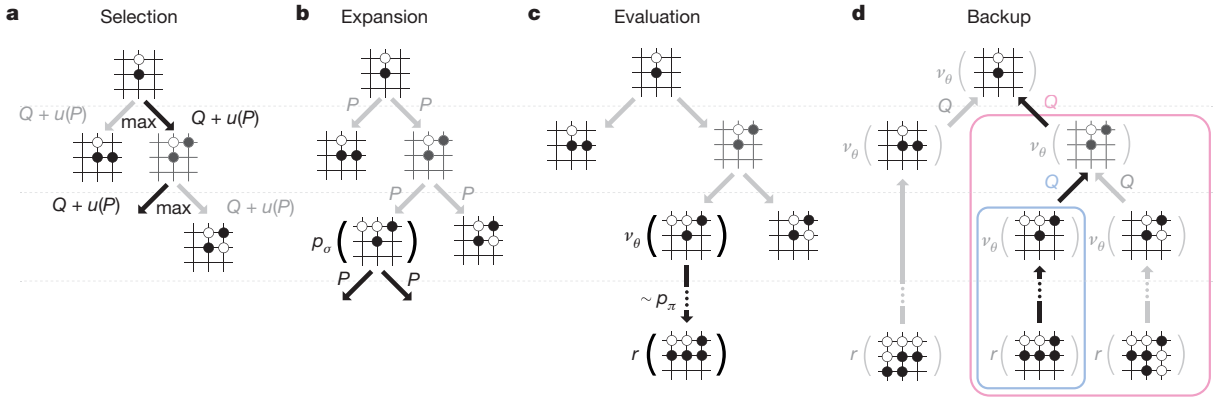
Figure 2.11: "A group of young people playing a game of frisbee"—that caption was written by a computer with no understanding of people, games or frisbees.

Game playing with Reinforcement Learning

- E.g. AlphaGo







Un « bolide » délicat à piloter

Requiert

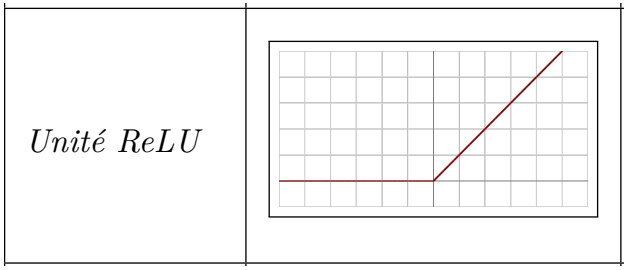
1. beaucoup de **données** (en général)
   - Des millions d'images
   - Des dizaines de milliers de documents
2. du **savoir-faire** (des data scientists)
   - Nombreuses « **astuces** » d'ingénierie
   - Utilisation de réseaux déjà appris (**transfert**)
   - L'état de l'art **progresse très vite**
3. des machines adaptées
   - Puissance **calcul** : clusters et/ou cartes graphiques
   - **Mémoire** centrale importante (≥ 128 Go)

Enseigné dans certaines écoles et universités

Beaucoup de « recettes de cuisine »

Les grandes idées nouvelles

- La **rétro-propagation classique ne marche pas** avec un grand nombre de couches (trop dilué)



Drop Out

- **Risque de sur-apprentissage** avec un nombre gigantesque de paramètres

Auto-encoder : couches apprises en non-supervisé

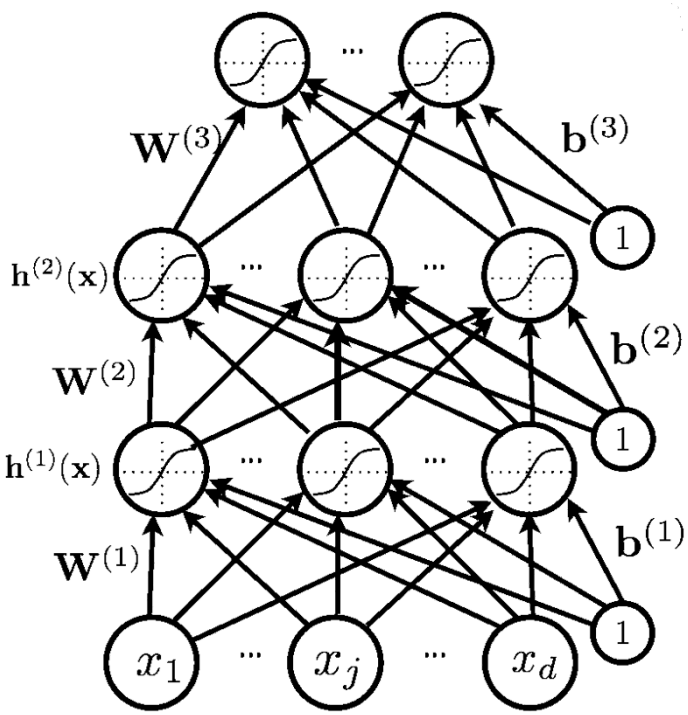
Réseaux à convolutions :
imposer une structure (avec motifs répétitifs) au réseau

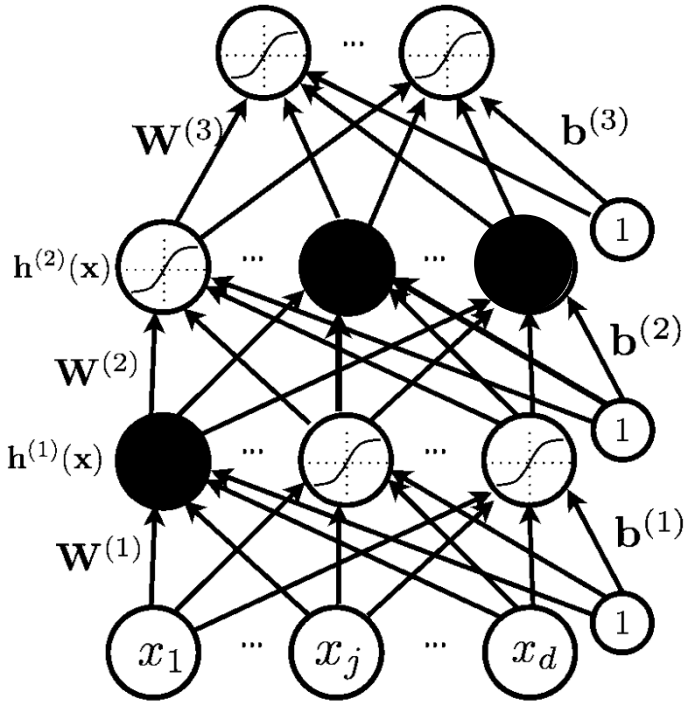
Le « Drop Out »

- Classiquement :
  - Les poids sont initialisés aléatoirement et difficiles à ajuster

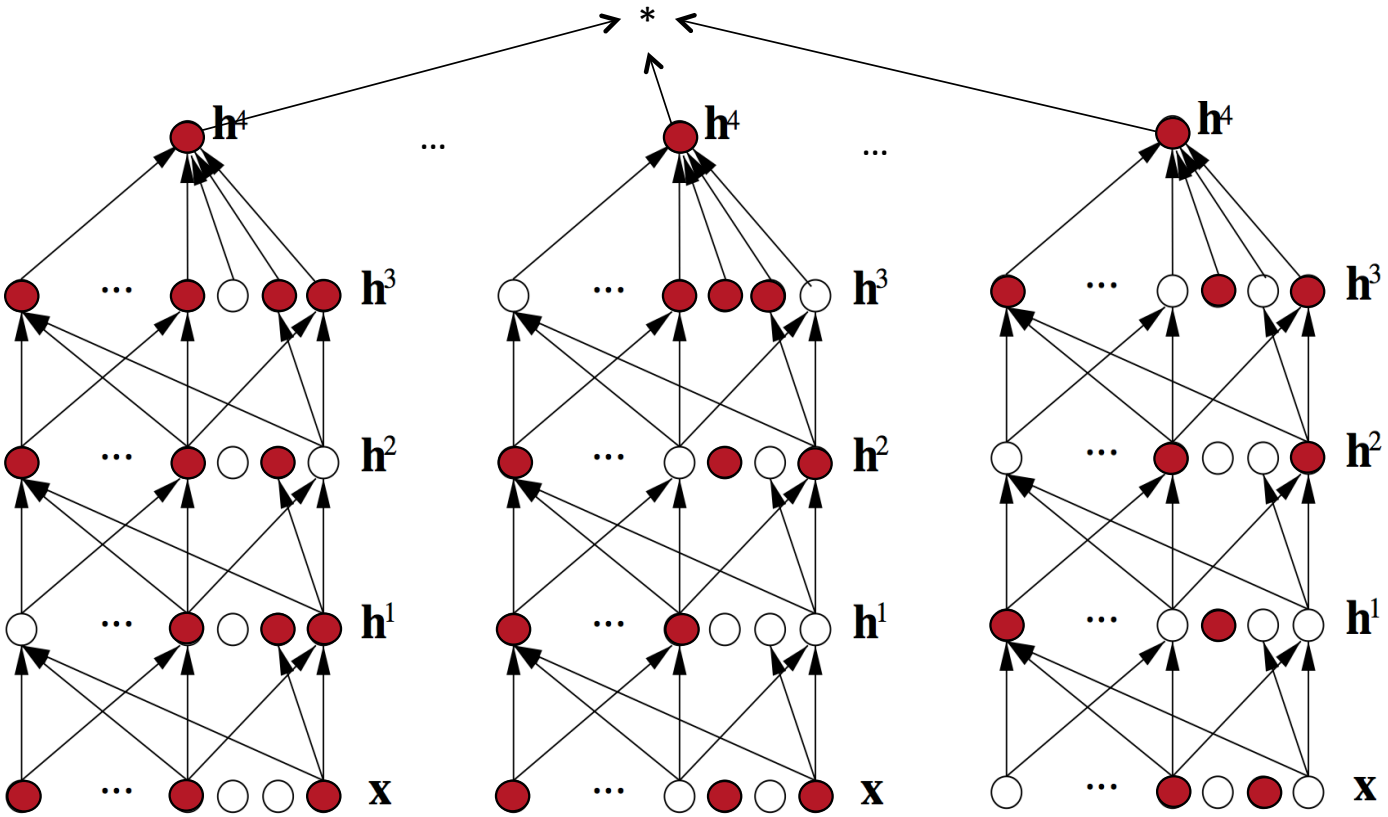
- **Principe** :
  - **Débrancher** des **neurones aléatoirement** lors de l'apprentissage (tirage aléatoire à chaque nouvel exemple)
  - Paramètre par défaut : 0.5

Le « Drop Out »


$\mathbf{W}^{(3)}$
$\mathbf{b}^{(3)}$
$\mathbf{h}^{(2)}(\mathbf{x})$
$\mathbf{W}^{(2)}$
$\mathbf{b}^{(2)}$
$\mathbf{h}^{(1)}(\mathbf{x})$
$\mathbf{W}^{(1)}$
$\mathbf{b}^{(1)}$
x_1
x_j
x_d
1


$\mathbf{W}^{(3)}$
$\mathbf{b}^{(3)}$
$\mathbf{h}^{(2)}(\mathbf{x})$
$\mathbf{W}^{(2)}$
$\mathbf{b}^{(2)}$
$\mathbf{h}^{(1)}(\mathbf{x})$
$\mathbf{W}^{(1)}$
$\mathbf{b}^{(1)}$
x_1
x_j
x_d
1

Dropout regularizer = super-efficient Bagging



Backprop in practice for deep architectures

- Use ReLU non-linearities (tanh and logistic are falling out of favor)
- Use cross-entropy loss for classification
- Use Stochastic Gradient Descent on minibatches
- Shuffle the training samples
- Normalize the input variables (zero mean, unit variance)
- Schedule to decrease the learning rate
- Use a bit of L1 or L2 regularization on the weights (or a combination)
  - But it's best to turn it on after a couple of epochs
- Use "dropout" for regularization
  - Hinton et al 2012 http://arxiv.org/abs/1207.0580
- Lots more in [LeCun et al. "Efficient Backprop" 1998]
- Lots, lots more in "Neural Networks, Tricks of the Trade" (2012 edition) edited by G. Montavon, G. B. Orr, and K-R Müller (Springer)

- From Yann Le Cun's slides

Du nouveau dans l'hardware

Technologie : du hardware spécifique

- **GPU** *(Graphics Processing Unit)* historiquement utilisé comme carte graphique pour les jeux vidéo
- Hardware spécialisé dans le **calcul matriciel** hautement **parallélisé**
- Des algorithmes très contraints : des **milliers « threads »** qui doivent exécuter **la même opération** simultanément

Des implémentations modernes des réseaux de neurones permettent de tirer partie des GPU *(ex: Torch 7, Cuda conv-net, Theano, TensorFlow …)*