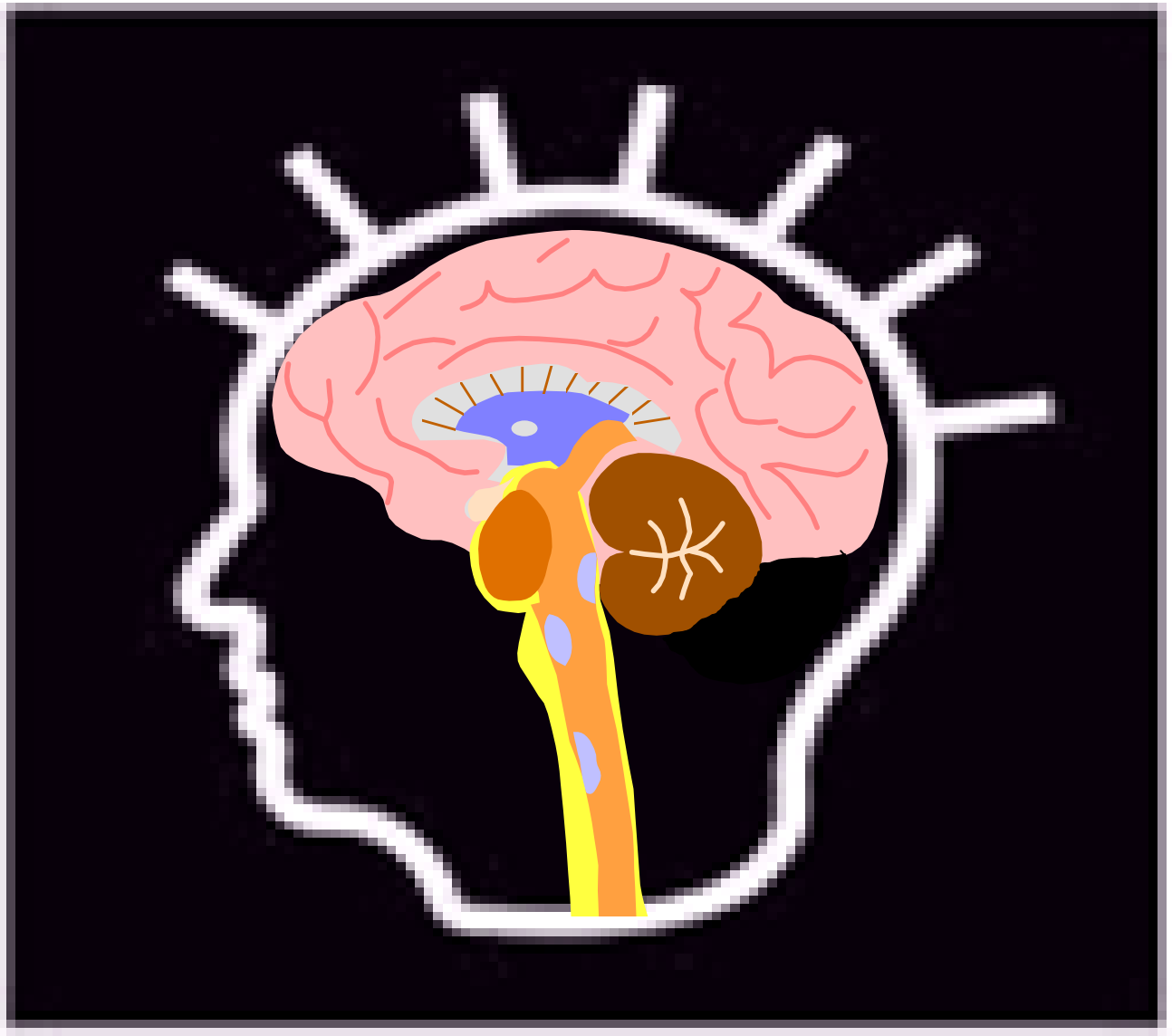
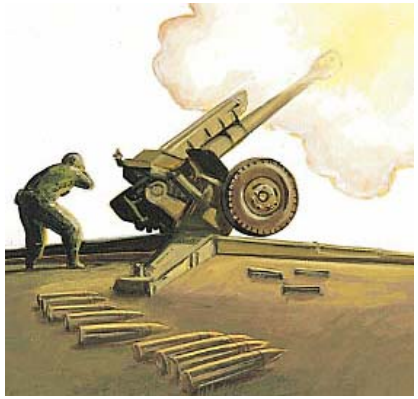
LES RESEAUX DE NEURONES ARTIFICIELS



Par Prof. N. Naït-Saïd 2005/2009

LES RNA, est-ce nécessaire?

Apport … par rapport au conventionnel



Gaspillage temps de calcul & mémoire calculateur pour résoudre un problème sans mérite

IMPORTANT

Justifier de l'utilisation des RNA :

✓ prouver les limites des techniques conventionnelles pour un problème donné (difficile ou impossible à appréhender le problème);

✓ ajouter un plus par rapport au conventionnel.

Concurrence ~~Concurrence~~ mais Complémentarité

Profiter de la puissance de calcul des RNA à travers :

- ***Parallélisme*** (Temps de propagation de l'information)
- ***Généralisation*** (Pour des entrées non rencontrées auparavant)

DEFINITIONS (1)

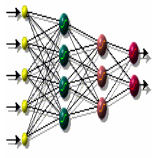
(Dictionnaire LE PETIT ROBERT)

Intelligence (*nom*)	Artificielle (*Adj.*)
Faculté de connaître, de comprendre *Didact.* Aptitude d'un être vivant à s'adapter à des situations nouvelles	Qui est le produit de l'habilité humaine

Ensemble des théories et techniques développant des programmes informatiques complexes capable de résoudre des problèmes sans que les algorithmes de résolution soient explicitement fournis

DEFINITIONS (2)

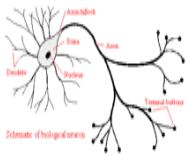
RNA



Les réseaux de neurones (ou neuronaux) artificiels (**RNA**) consistent en des modèles plus ou moins inspirés du fonctionnement cérébral de l'être humain (cerveau), en se basant sur le concept de neurone[1].

Un réseau de neurones (ou neuronal) est un processeur massivement **parallèle** et **distribué** constitué d'unités de traitement simples (neurones) qui ont la propriété naturelle de **stocker l'information** (connaissances) et de la mettre disponible pour l'utilisation [2].

Neurone



Le neurone est l'unité élémentaire de traitement. Il est doté d'entrées (**dendrites**) pouvant recevoir des signaux et une sortie (**axone**) distribuant le résultat du traitement.

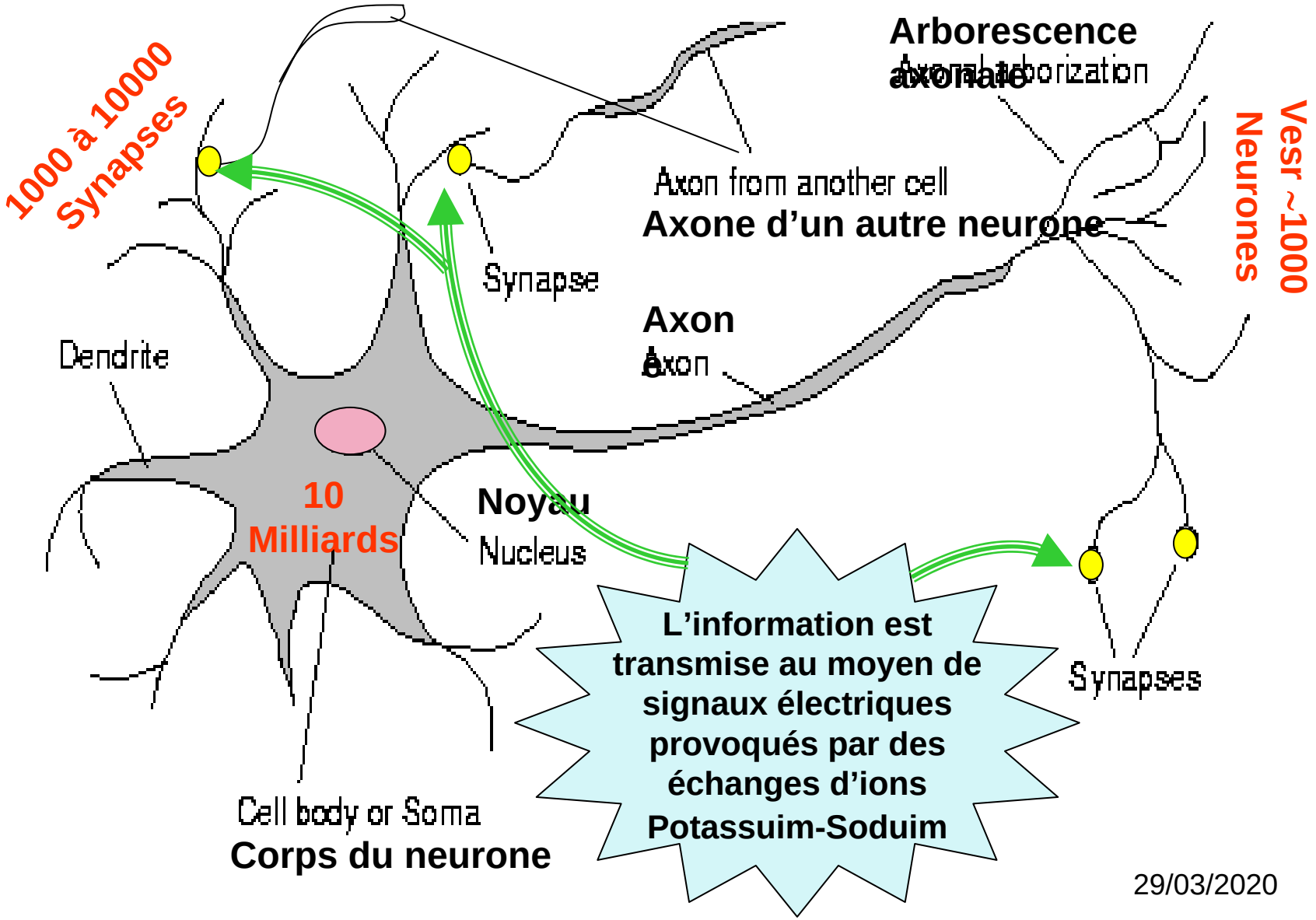
Les échanges de signaux se font au niveau des **synapses** reliant les axones aux dendrites des autres cellules (neurones).

---

1. Def. Algorithmes génétiques et réseaux de neurones, J.M. Rendres, Ed. Hermes
2. Def. Neural Networks, S. Haykin, Ed. Prentice Hall Inc.,

Le Neurone Biologique

Unité fondamentale, fonctionnelle et anatomique du tissu nerveux,



Parcours Visuel

The Visual Pathway

Decisions & Actions
(& Conscious Awareness?)

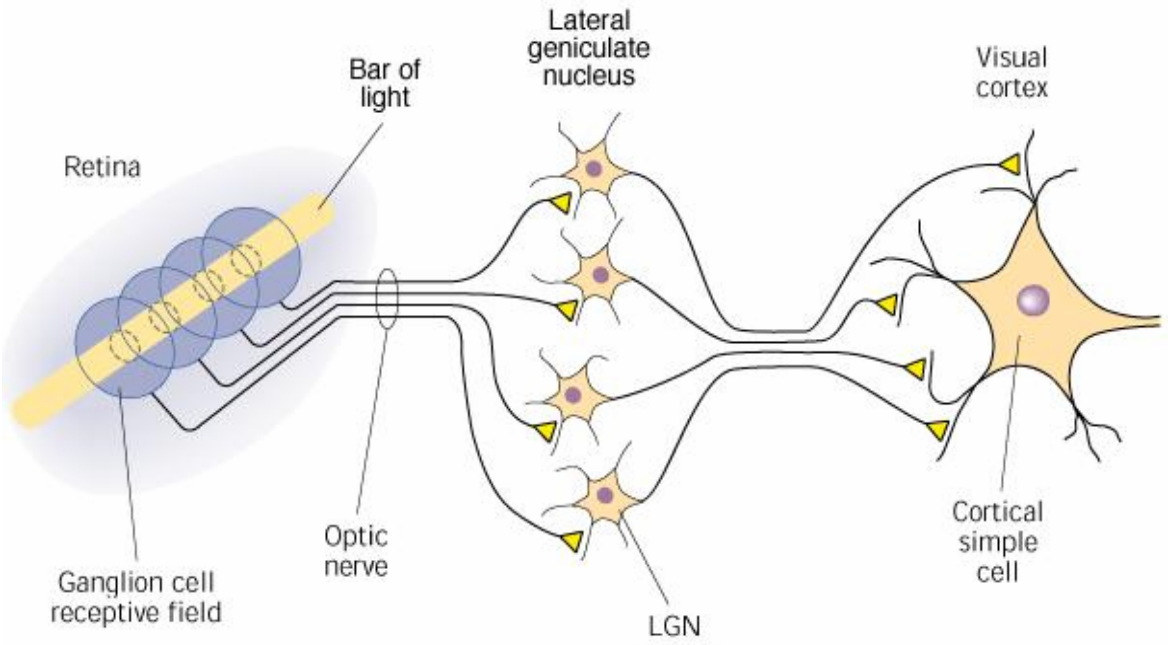
Prefrontal Areas
& Premotor Areas

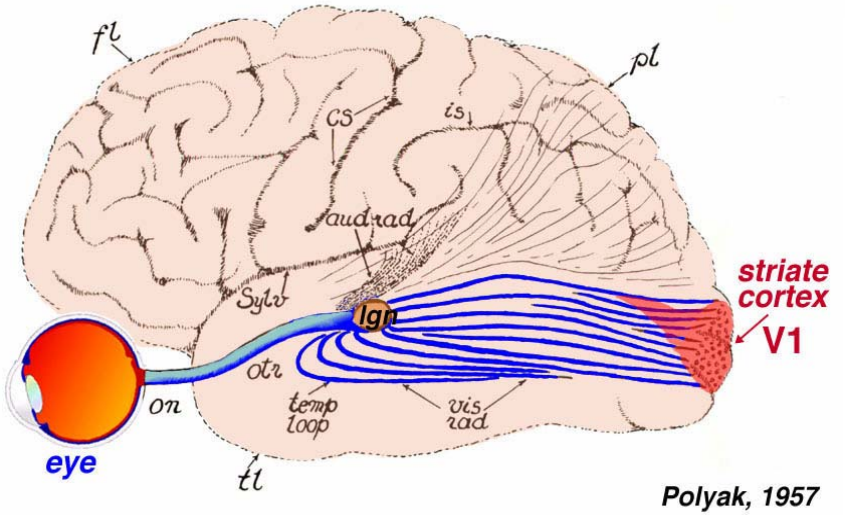
"Higher" Visual Areas
(V2, V3, V4, Medial Temporal)

Striate Cortex
(V1/area 17)

Lateral Geniculate Nucleus

Retina

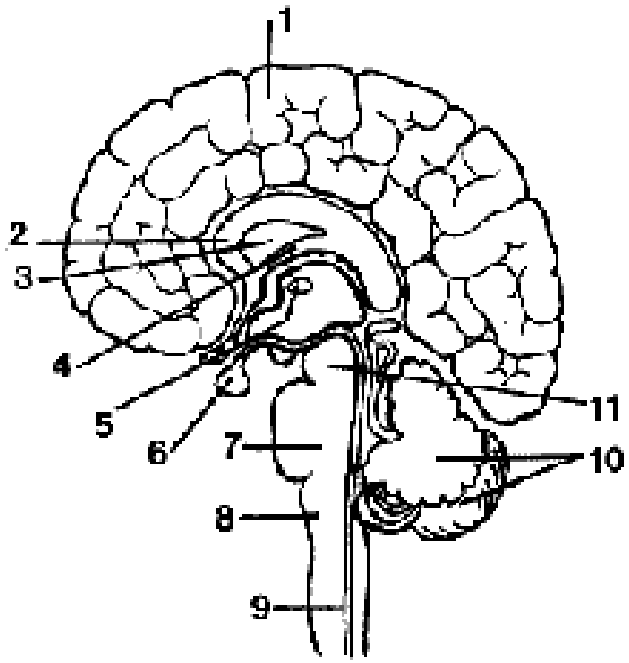


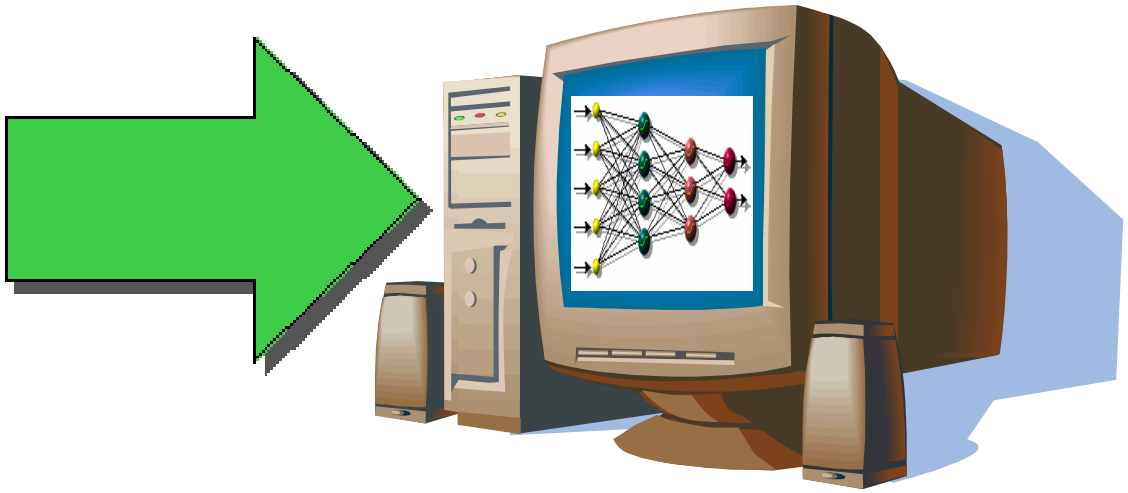


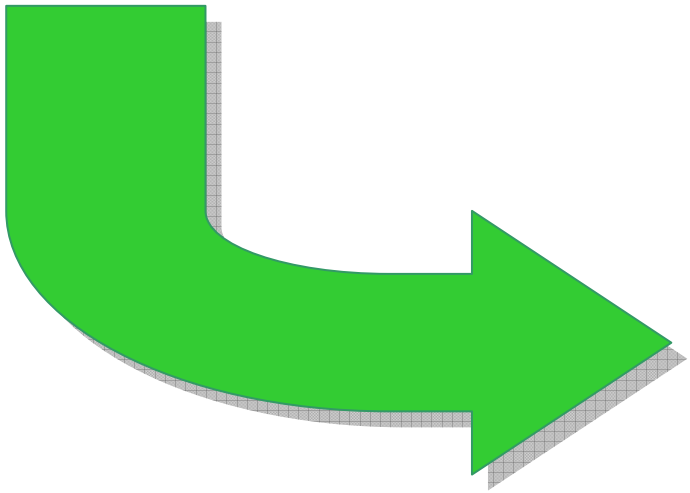
Polyak, 1957

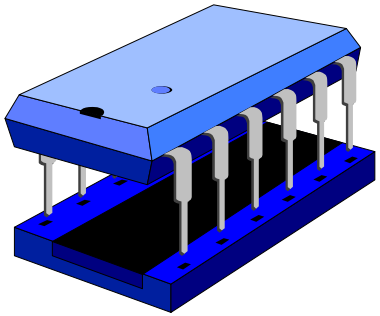
IMITATION DU CERVEAU : SOFT & HARD

Réseaux Neuro-mimétiques ; Réseaux de Neurones Artificiels (RNA)









COMPARAISON

Réseaux de neurones/ Clalcul standard (Computer)

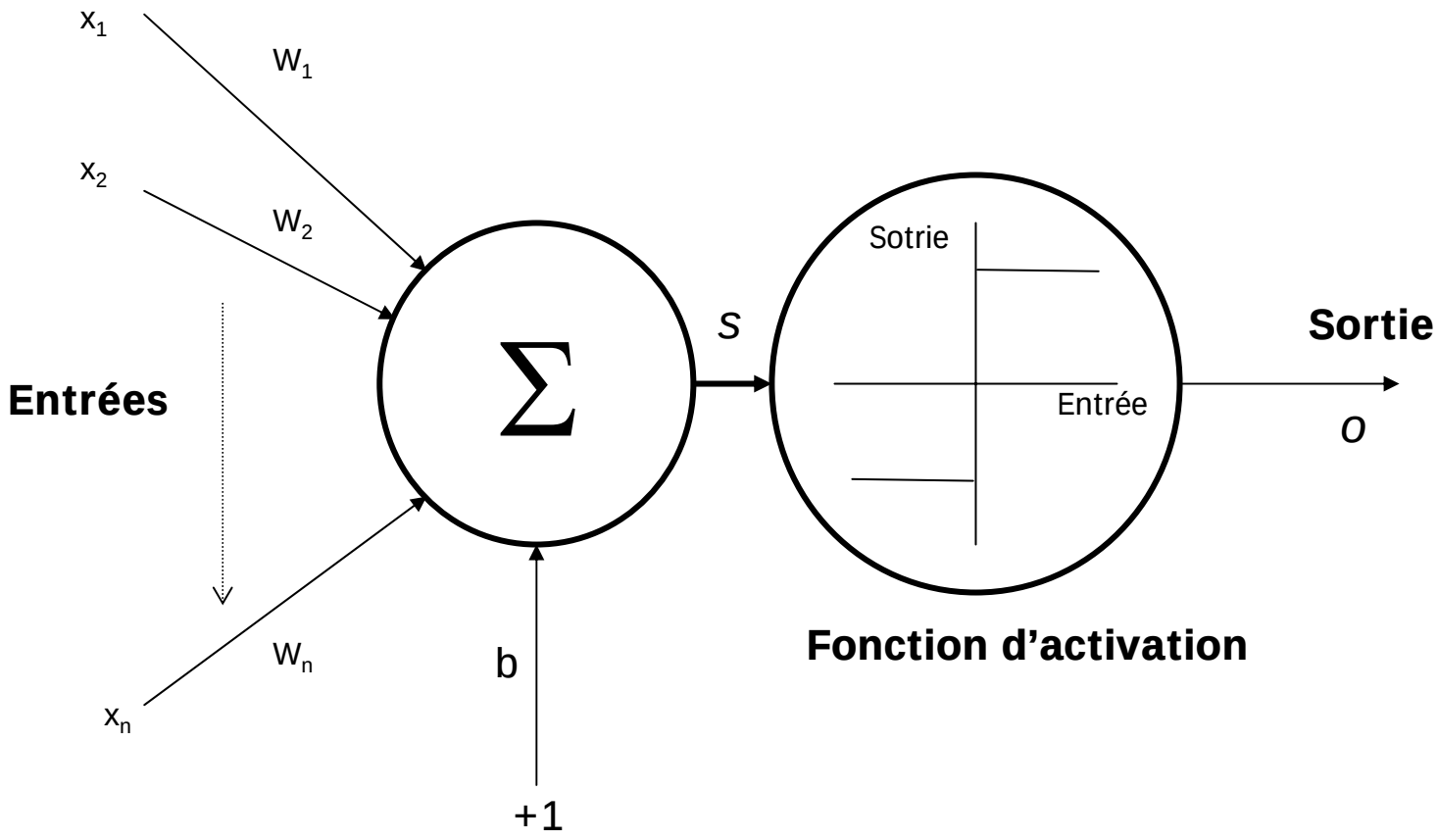
RNA	Calcul “Standard”
• Calcul collectif, Parallèle	• Calcul centralisé, Séquentiel
• Non algorithmique	• Algorithmique
• Raisonnement inductif : Entrées-Sorties $\Rightarrow$ Réseau	• Raisonnement déductif : Entrées, règle $\Rightarrow$ sortie.
• Capacité de généraliser	• Pas de généralisation
• Tolérance aux défauts	• Sensibilité aux défauts
• Apprentissage, adaptation	• Programmé
• Mémoire distribuée	• Mémoire locale
• Applicable aux Pb mal connus, compliquées, ou si données bruités	• Applicable si tout est bien défini, règles, données.

HISTORIQUE

- **Av. 1940: von Hemholtz, Mach, Pavlov, etc**.
  - Théories générales d'apprentissage, vision, traitement;
  - Pas de modèle mathématique spécifique pour le fonctionnement du neurone;
- **1940s: Hebb, McCulloch & Pitts**
  - Mécanisme pour l'apprentissage des neurones biologique;
  - Neurone formel (le **Perceptron** : fonctions logiques);
- **1950s: Rosenblatt, Widrow and Hoff**
  - Premiers réseaux pratiques et règles d'apprentissage;
- **1960s: Minsky and Papert**
  - Démonstration des limites des Réseaux existants & pas de nouveaux algorithmes d'apprentissage, Recherches suspendues;
- **1970s: Amari, Anderson, Fukushima, Grossberg, Kohonen**
  - Progrès continus, bien que à pas lents;
- **1980s: Grossberg, Hopfield, Kohonen, Rumelhart, etc.**
  - Nouveaux développements importants causent une résurgence (Algorithme de la rétro propagation en 1986).

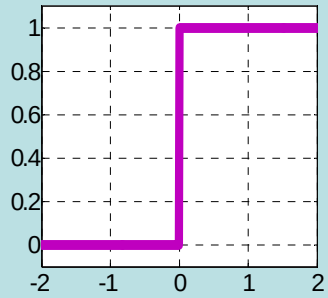
LE NEURONE FORMEL

LE PERCEPTRON

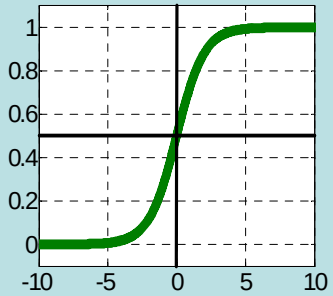


$$o = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = f([w_1 \dots w_n \; b].[x_1 x_2 \dots x_n \; 1]^{\mathrm{T}})$$

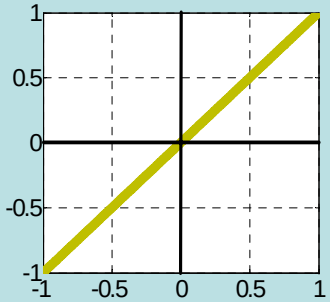
FONCTIONS D'ACTIVATION



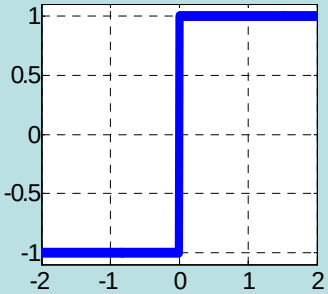
y=hardlim(x)



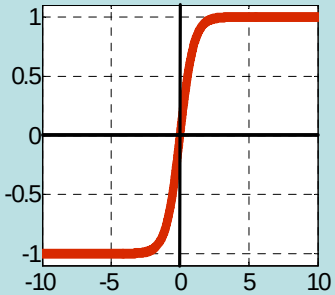
y=logsig(x)



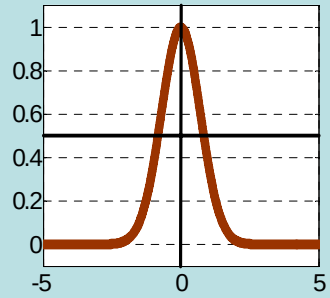
y=purelin(x)



y=hardlims(x)



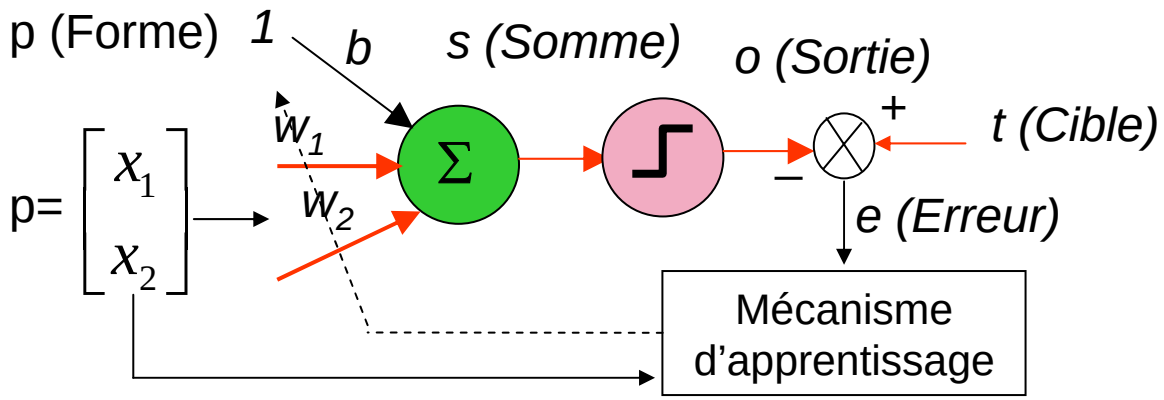
y=tansig(x)



y=radbas(x)

Apprentissage du Perceptron

REGLE DU PERCEPTRON



Apprentissage : $w_i(k+1) = w_i(k) + \Delta w_i$

où: $\Delta w_i = \eta (t - o) x_i$

Prg. Matlab

avec: η le coefficient d'apprentissage

REGLE DU Widrow-Hoff

$$\Delta w_i = \eta (t - s) x_i$$

Poids et biais sont initialisés à des valeurs aléatoires faibles

Apprentissage des RNA

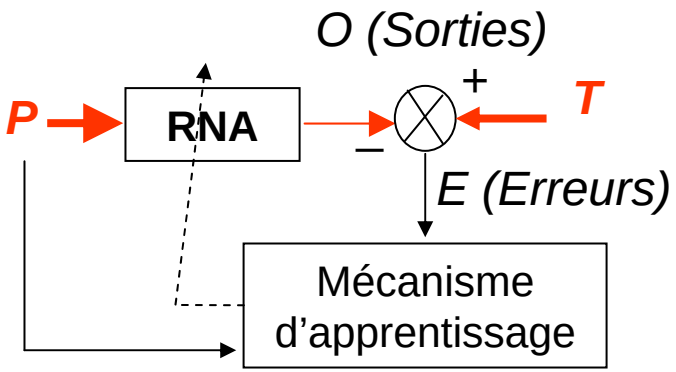
L'apprentissage pour les RNA (et cerveau) est une étape inéluctable.

Ce processus requiert des exemples (généralement de l'expérience) à présenter au réseau (étape cruciale). Ces exemples peuvent être :

Entrées-Sorties désirées : (P, T) ;

où P matrice pattern et T Matrice target

App. Supervisé



Entrées : (P) ;

App. Non Supervisé

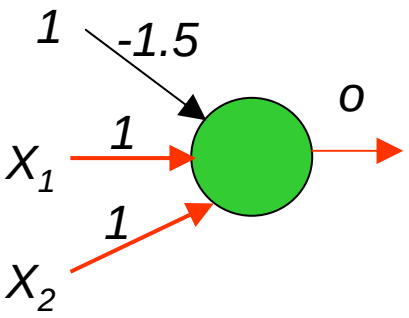
En fonction des différentes excitations, le réseau organise les P en catégories.

Après apprentissage, lorsque une entrée est présentée au réseau, une sortie est activée indiquant son appartenance à une classe, sinon une nouvelle classe est Générée.

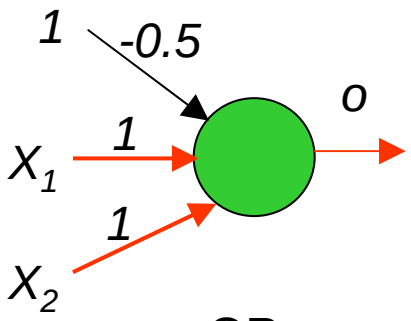
L'apprentissage peut être en ligne (On-line) ou hors ligne (Off-line)

REALISATION DES UNITES BOOLEENES SIMPLES

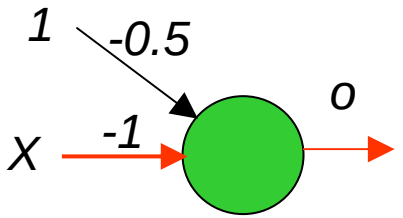
Il est possible de construire des neurones réalisant les opérations AND, OR et NOT



AND



OR



NOT

X1	X2	S	O
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

O=1 si S>=0; sinon *O*=0

X1	X2	S	O
0	0	-0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

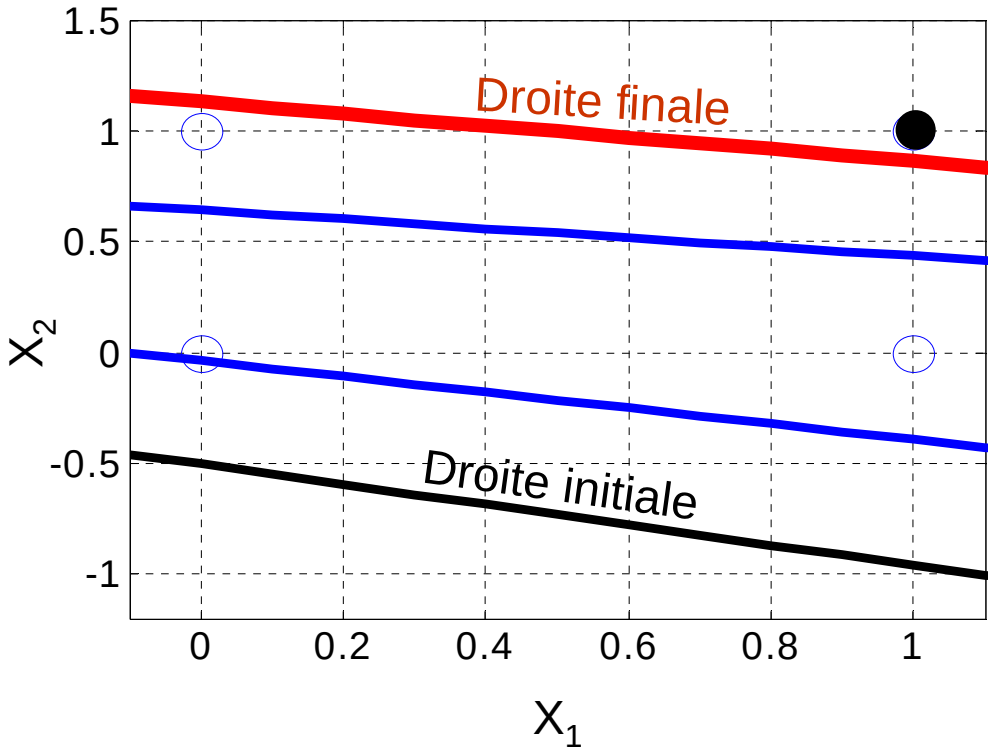
X	S	O
0	0.5	1
1	-0.5	0

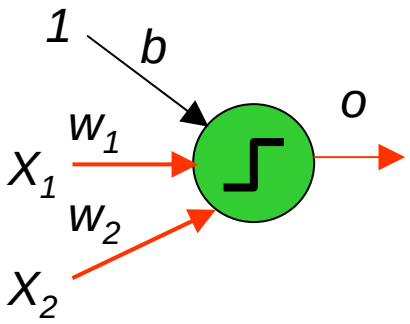
A vérifier : Matlab

N.B. La solution n'est pas unique

Séparabilité linéaire

Exemple du AND



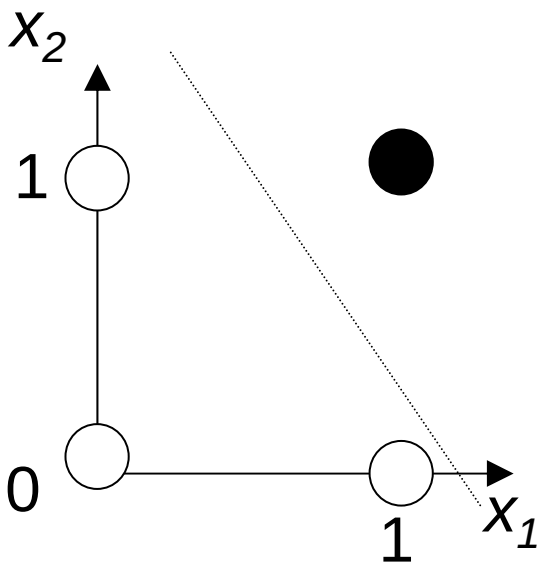


Equation de la droite

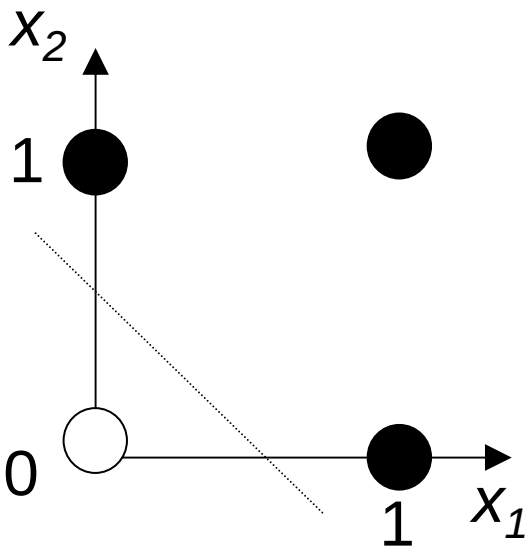
$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

Problème du OUEX

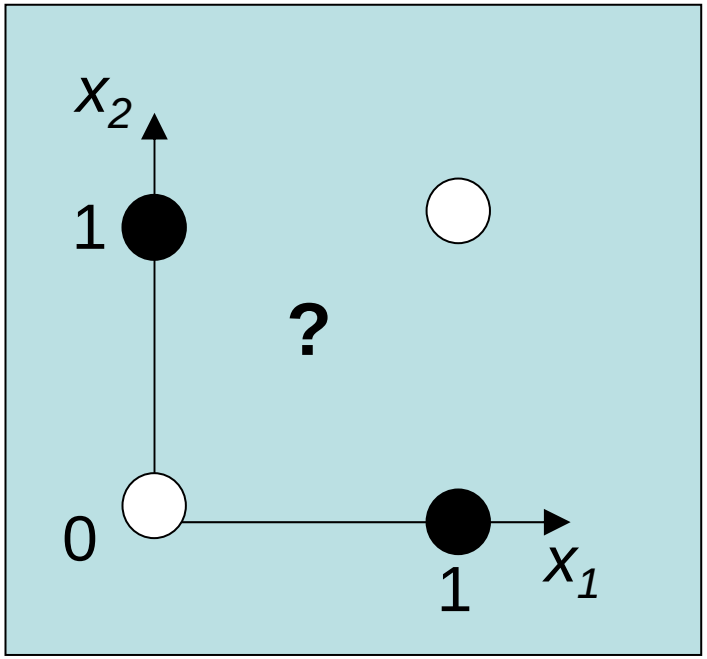
AND



OR



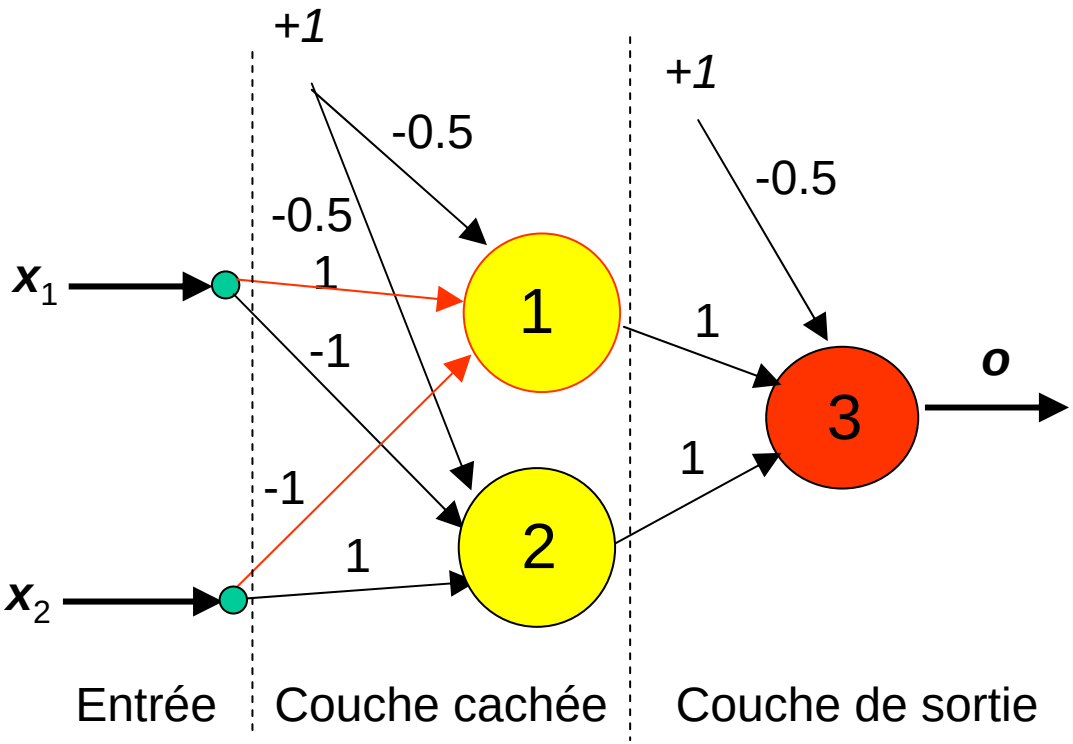
XOR



Une solution pour le OU-Ex (XOR)

Le réseau multicouche

A vérifier : Matlab



Conclusion (I)

1- Classification

2- Le perceptron monocouche résout seulement les problèmes linéairement séparable

Exemple d'évolution de l'erreur

Neurone linéaire

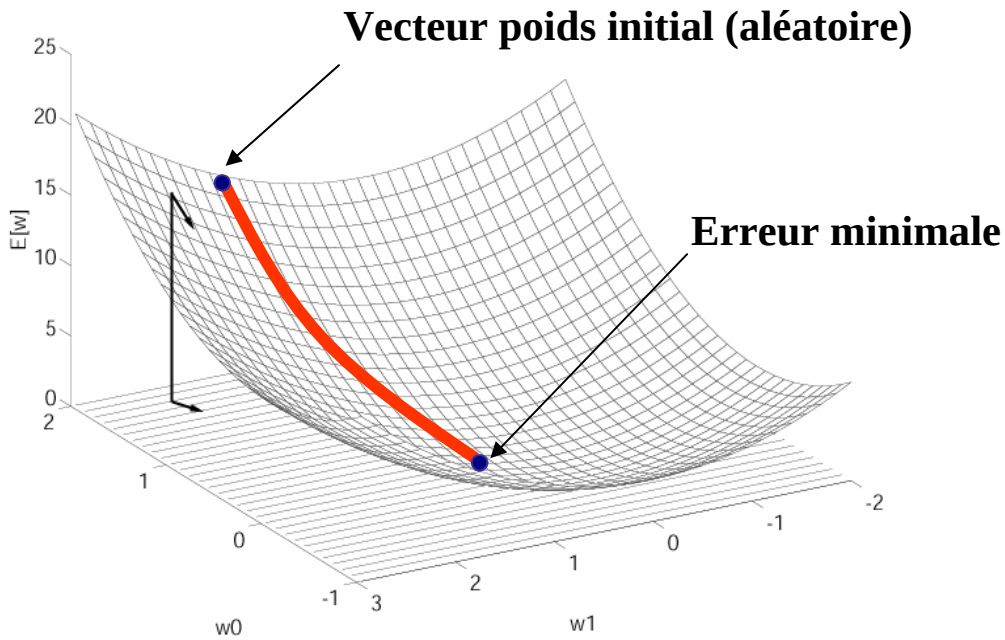
Règle d'apprentissage *Delta* pour un perceptron à fonction d'activation continue;
Pour une unité linéaire, la sortie *o* est donnée par :

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

Pour obtenir la règle d'apprentissage pour ce neurone (linéaire), l'expression de l'erreur suivante, relative aux exemples d'apprentissage, est évaluée, :

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

d, indice pattern



Règle Delta généralisée

Le vecteur dérivé $\nabla E(\vec{w})$ appelé *gradient* de *E* par rapport à $\vec{w}$, s'écrit :

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n}\right]$$

Il spécifie la direction que produit la dimunition de *E* indiquée par le signe négatif. La règle d'apprentissage est alors :

$$w_i \leftarrow w_i + \Delta w_i \qquad \text{où,} \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial s}\frac{\partial s}{\partial w_i} \rightarrow$$

$$\frac{\partial E}{\partial o} = -(t-o)$$

$$\frac{\partial o}{\partial s} = f'(s)$$

$$\frac{\partial s}{\partial w_i} = x_i$$

soit $\Delta w_i = \eta(t-o)x_i f'(s)$

Prg. Matlab

Règle Delta : Amélioration de la convergence

1- Adaptation de η :

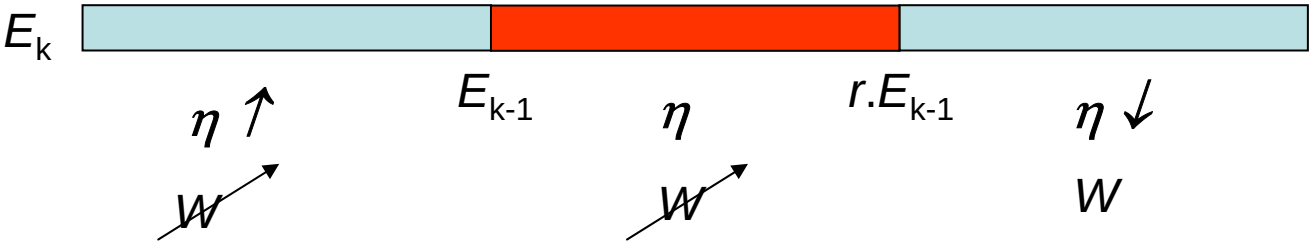
$$\Delta w_i = \eta(t-o)x_i f'(s)$$

Si $E_k > r.E_{k-1}$ alors réduire η et ne pas modifier poids ($r > 1$)

Si $E_k < E_{k-1}$ alors augmenter η et modifier poids

Sinon η reste inchangé et modifier poids

Prg. Matlab



2- Terme Moment :

$$\Delta w_i(k) = \eta(t-o)x_i f'(s) + m.\Delta w_i(k-1)$$

Prg. Matlab

Dérivées des fonctions sigmoïdes

1- Sigmoïde logistique

$$f_l(s) = \frac{1}{1+\exp(-s)}$$ (Matlab : logsig)

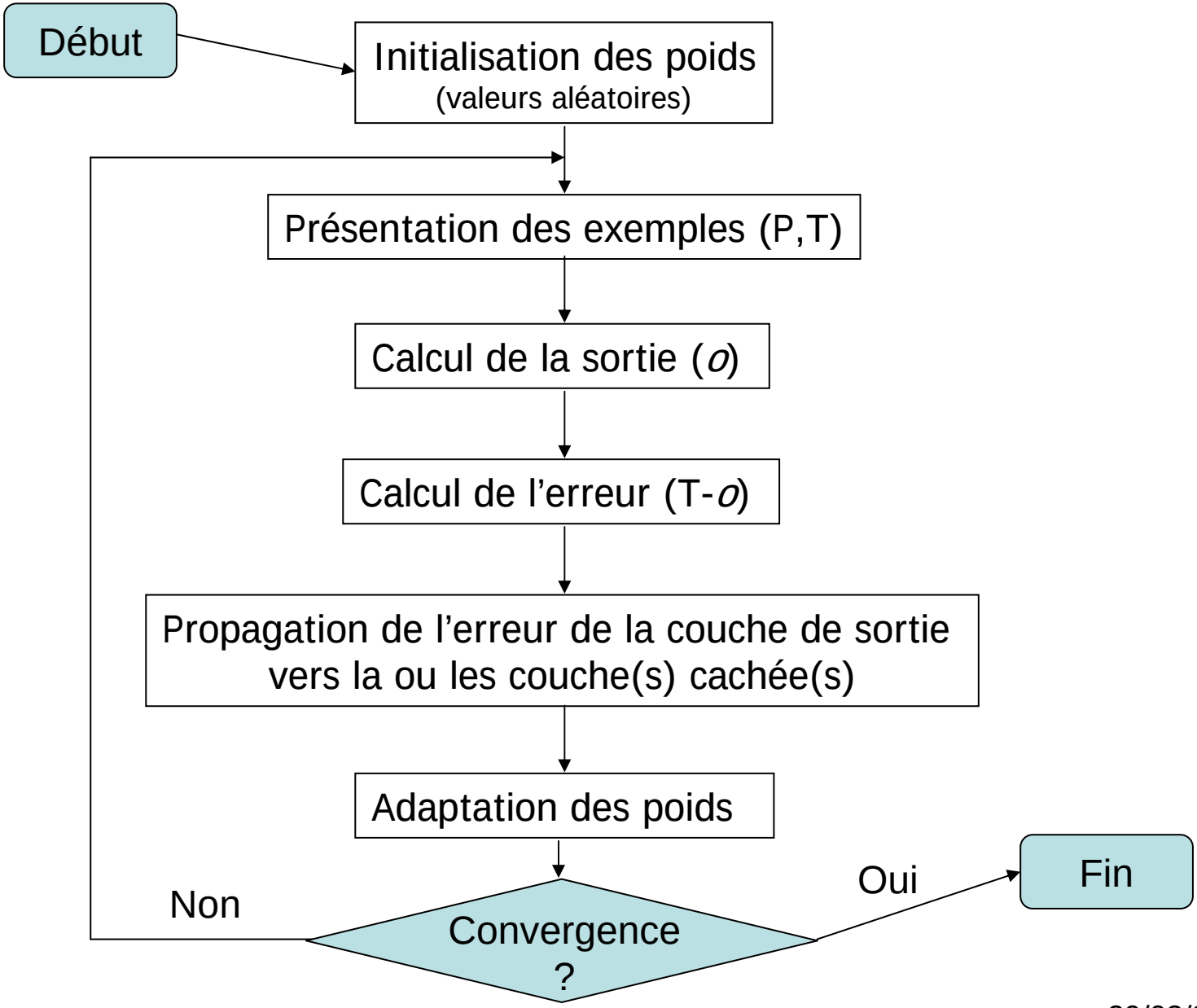
Dérivée : $$f_l'(s) = f_l(s)(1 - f_l(s))$$

2- Sigmoïde tanh

$$f_t(s) = \frac{1-\exp(-2s)}{1+\exp(-2s)} = 2\,f_l(2s) - 1$$ (Matlab : tansig)

Dérivée : $$f_t'(s) = 1 - f_t(s)^2$$

Algorithme de la rétropropagation (Organi)



Algorithme de la rétropropagation (Erreur)

Pour un réseau multicouche (MLP) donné, l'algorithme de la rétropropagation d'erreur permet l'adaptation des poids (apprentissage) en utilisant la descente du gradient. L'erreur quadratique entre cibles et sorties est à minimiser.

L'erreur est évaluée uniquement sur la couche de sortie :

✓ Adaptation des poids de la couche de sortie (Delta)

? Comment faire pour les couches cachées ?

Fonction Erreur

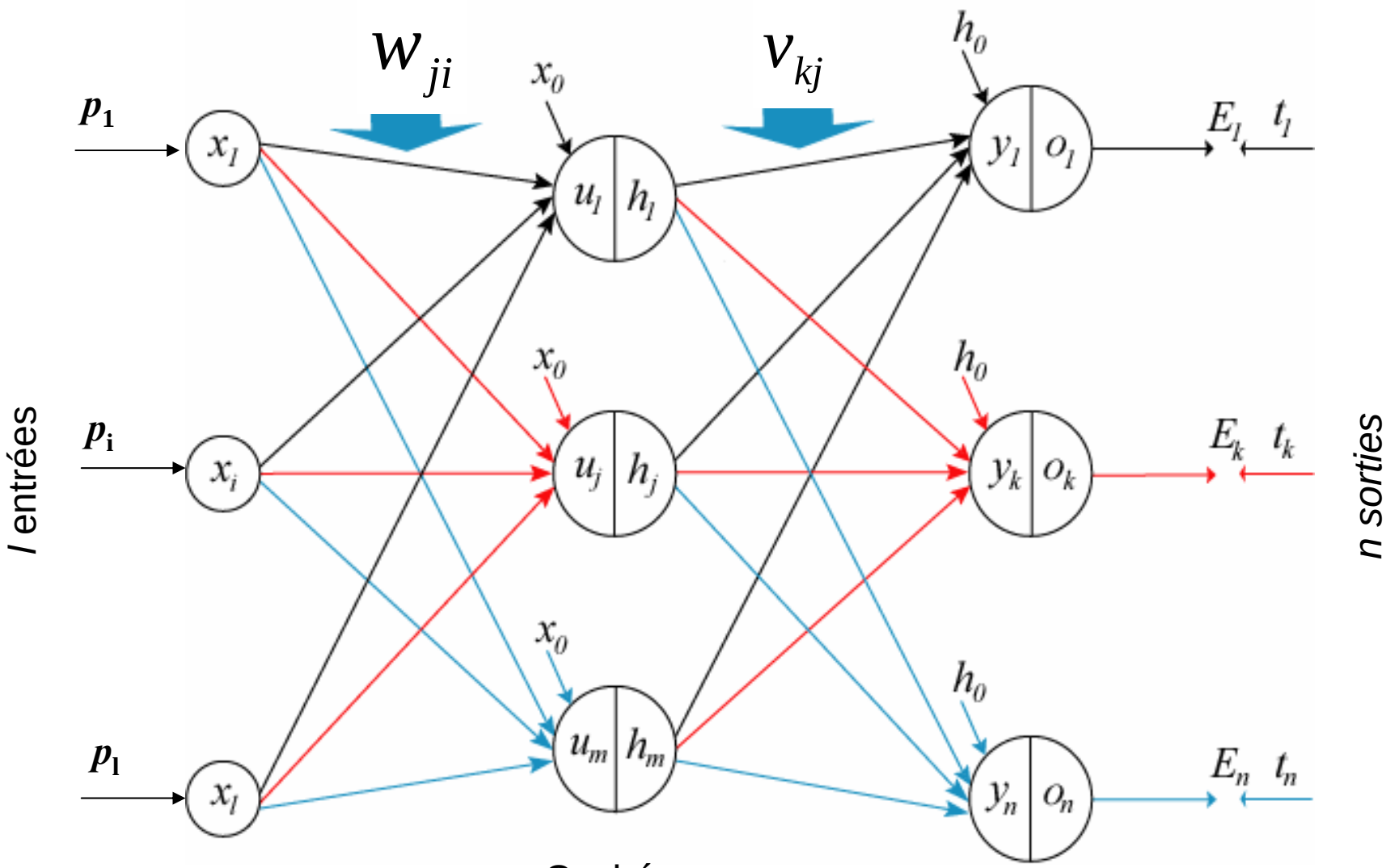
$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

où, *outputs* : Nbre de sorties du RNA (neurones de sortie) ;

t_{kd} : Cible associée à la $k^{ième}$ sortie;

o_{kd}: $k^{ième}$ Sortie associée à l'exemple *d*.

Algorithme de la rétropropagation (Architecture du réseau)



Cachée

Sortie

Pour un exemple (*d*=1)

Algorithme de la rétropropagation (Calculs)

Phase de présentation (activations logsig)

Couche de sortie : $y_k = \sum_{j=0}^{m} v_{kj}\ h_j, \quad o_k = \frac{1}{1+e^{-y_k}}, 1 \le k \le n$

Couche cachée : $u_j = \sum_{i=0}^{\iota} w_{ji}\ x_i, \quad h_j = \frac{1}{1+e^{-u_j}}, 1 \le j \le m$

Adaptation des poids

Couche de sortie : $v_{kj}^{(new)} = v_{kj}^{(old)} + \eta \cdot \delta o_k \cdot h_j, 0 \le j \le m, 1 \le k \le n$

Couche cachée : $w_{ji}^{(new)} = w_{ji}^{(old)} + \eta \cdot \delta h_j \cdot x_i, 0 \le i \le l, 1 \le j \le m$

où : $\delta o_k = (t_k - o_k) o_k (1 - o_k), 1 \le k \le n$

$$\delta h_j = h_j (1 - h_j) \sum_{k=1}^{n} \delta o_k \cdot v_{kj}\ 1 \le j \le m$$

Prg. Matlab

Algorithme de la rétropropagation

Difficultés & limites

Pas de résultats théoriques (ni même règles empiriques satisfaisantes) pour dimensionner RNA correctement pour un problème donné.

- Nbre de couches cachées;
- Nbre de neurones par couche cachée;
- Relation entre exemples & classes (classification);
- Difficultés de régler les paramètres (ex. pas du gradient);
- Temps de calcul (pendant apprentissage);
- Manière de présenter les exemples.

Overfitting (sur apprentissage)

RNA apprend bien les exemples d'apprentissage et répond mal aux données de test (différentes des exemples d'apprentissage).

Une tendance du RNA à trop appréhender les données de la base d'apprentissage.

- Donc, un problème primaire est comment prévoir l'overfitting lorsqu'on conçoit un RNA

Généralisation

Les problèmes réels ne suivent pas des règles spécifiques.

Représentation d'un problème par des exemples n'incluant pas tous les cas possibles.

- Il est important d'entraîner le RNA pour la plupart des cas:
  - Les cas sélectionnés devraient être représentatifs;
  - Être attentif de ne pas surentraîner le RNA (Overfitting). Cela pourrait résulter de mauvaises performances avec d'autres exemples.
- L'apprentissage est réalisé sur un sous ensemble des données disponibles, le reste des données est réservé au test pour la généralisation

Réussite d’une bonne application

1- Règles inconnues ou difficiles à expliciter ou à formaliser (pour un problème donné), alors qu’on dispose d’exemples (données);

2- Problème faisant intervenir des données bruitées;

3- Évolution du problème avec ses conditions initiales;

4- Nécessité d’une grande rapidité de traitement (temps réel);

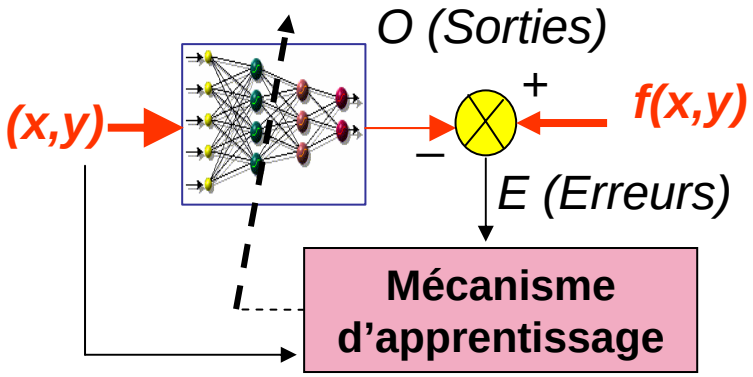
5-Pas de solutions courantes.

… Après :

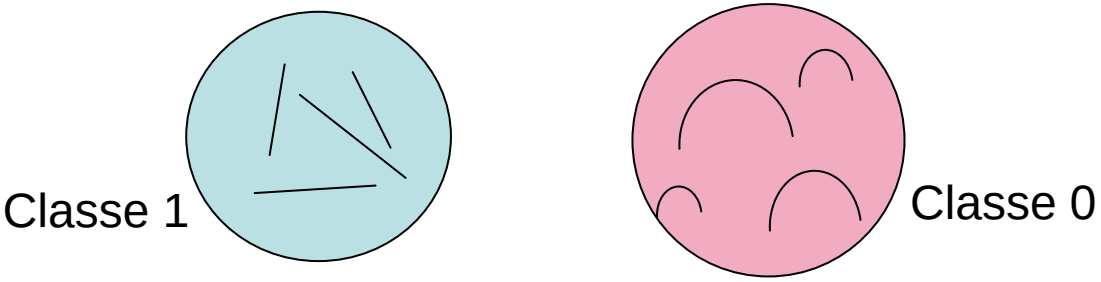
- Collecte d’une base d’apprentissage représentative;
- Bien choisir le réseau : type, dimensions, activation, …, en fonction de la nature et de la complexité du problème;

Applications

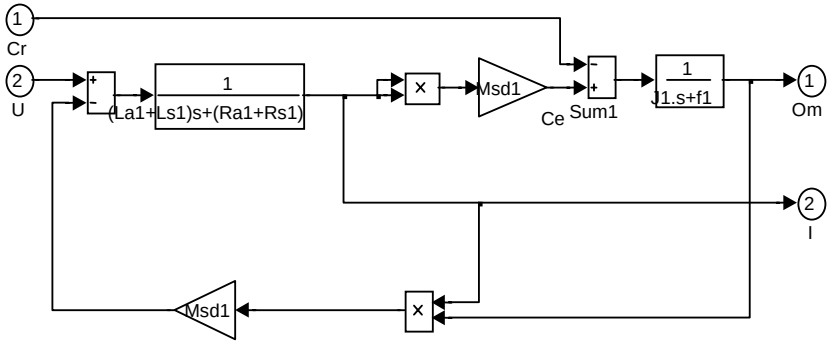
1. Approximation de fonction

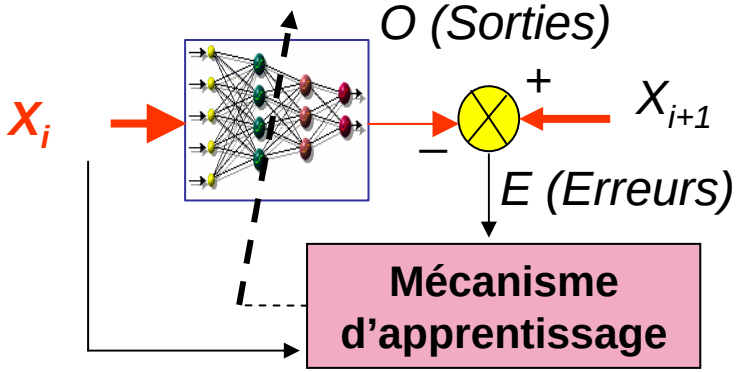


2. Classification d'objets



3. Identification d'un système non linéaire (MCC Série)





End of Presentation

ANNEXES

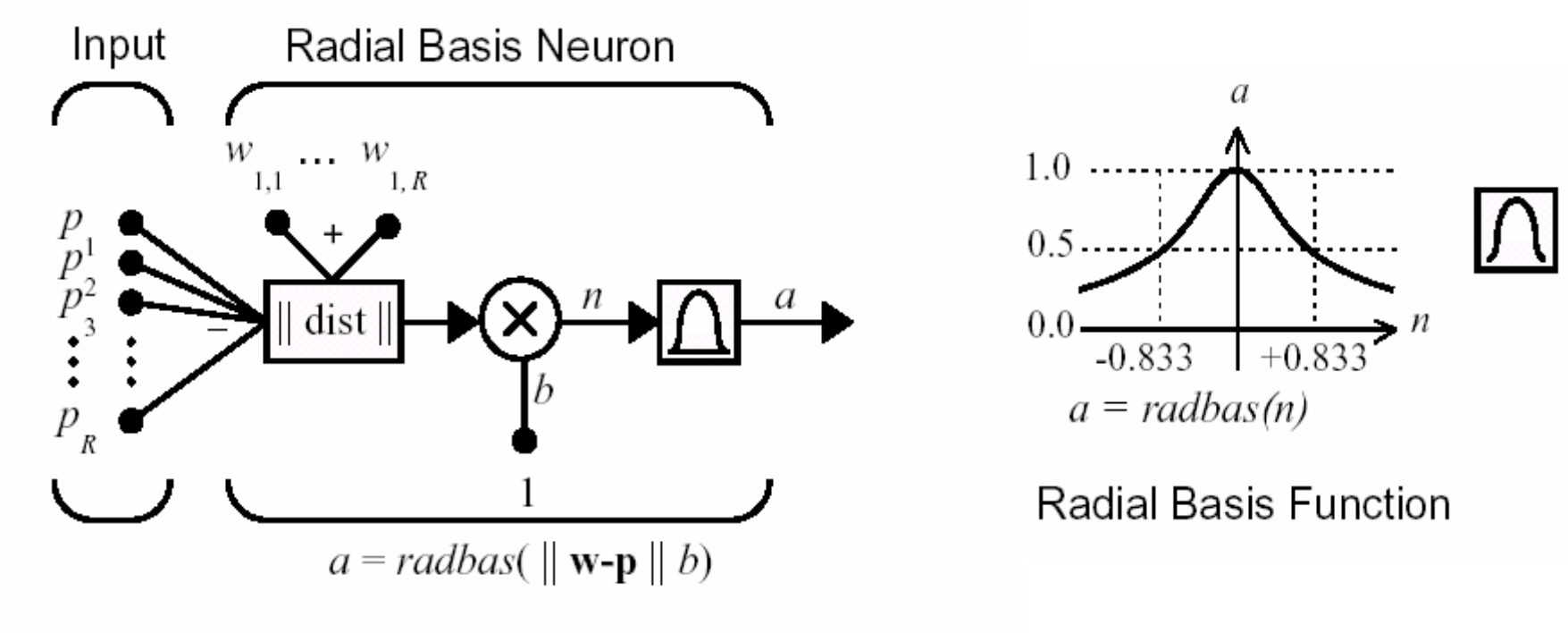
Some other ANN types

- Self-organizing maps
  - Typical use: Classification, clustering
- Radial basis function networks
  - Function approximation
- Recurrent networks
  - Modeling dynamic behavior
- Hopfield networks (variant of recurrent netw.)
  - Optimization, minimizing of energy function

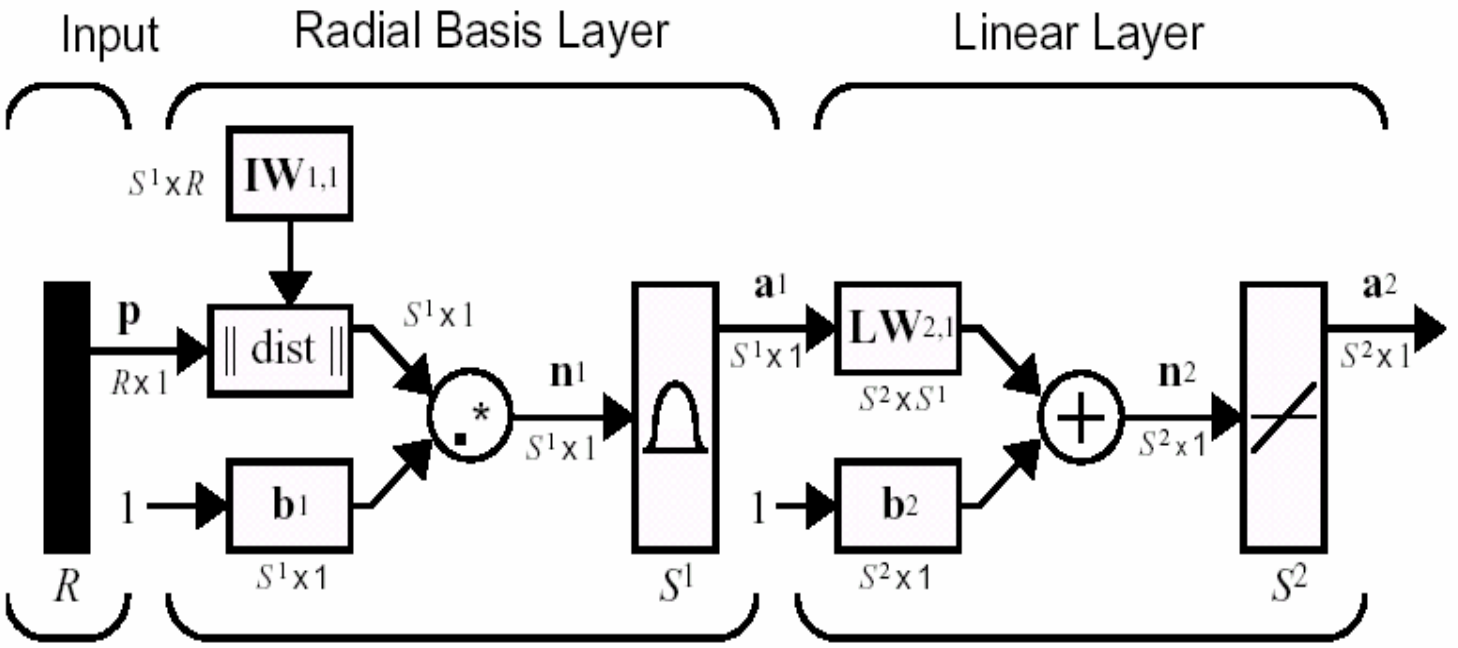
Characteristics of Radial Basis Function Net

- Only one nonlinear layer
  - Input layer is linear, for distribution only
  - Hidden units implements radially symmetric non-linear function (e.g., Gaussain)
  - Output layer forms linear combination of hidden layer
- Training may use several strategies
  - Hidden layer pre-specified, non- adaptive, only output weights are trained
  - One may use cluster analysis to specify function centers and widths in
  - Hidden layer may also be adaptive
- Very efficient for classification, less useful fro function approximation
- computationally intensive (requires large numbers of hidden units).

Radial Basis Neuron Model



Radial Basis Network Architecture



Where...

R = # elements in input vector

S^1 = # Neurons in layer 1

S^2 = # Neurons in layer 2

$$a_i^1 = radbas(\| {}_i\mathrm{IW}^{1,1} - \mathbf{p} \| b_i^1) \qquad \mathbf{a}^2 = purelin(\mathbf{LW}^{2,1}\,\mathbf{a}^1 + \mathbf{b}^2)$$

a_i^1 is ith element of $\mathbf{a}^1$ where ${}_i\mathrm{IW}^{1,1}$ is a vector made of the ith row of $\mathbf{IW}^{1,1}$

Characteristics of the Hopfield Net

- Can be either binary (0 or 1) or analog
- All units receive feedback from all other units, but not themselves
- Connection matrix is symmetric with 0's on the diagonal
- Connection matrix is programmed, not learned
- Vectors are stored as stable states
- Network iterates until it reaches stability, uses energy minimization rule (can go down or stay the same, but not go up)
- Usually finds stable state with closest Hamming distance to starts state
- Suitable for content addressable memory, optimization
- Storage capacity limited to about 15% of dimensionality (e.g., 6 stable vectors for a dimensionality of 40).

Characteristics of Fully Recurrent Networks

- The output of every units is connected to every other unit
- Some units have external inputs, some are designated as outputs, the rest are Hidden Units
- The system is clocked; the current inputs to each unit specify the state each unit will assume on the next clock tick, This is to prevent race conditions due to feedback
- Since the system is time-dependent, it can show dynamic behavior
- Due to limits of accuracy, dynamic properties can only be defined across 30-40 clock ticks or less
- Due to time dependence, training sis computationally intensive. Storage is proportional to the third power of n, time is proportional to the fourth power
- Useful for dynamic control problems, any function requiring delay, state machines requiring internal storage (Turing machine).

Face Detection using Neural Networks