Les réseaux de neurones artificiels

Pierre Borne

p.borne@ec-lille.fr

Les Réseaux de Neurones Artificiels

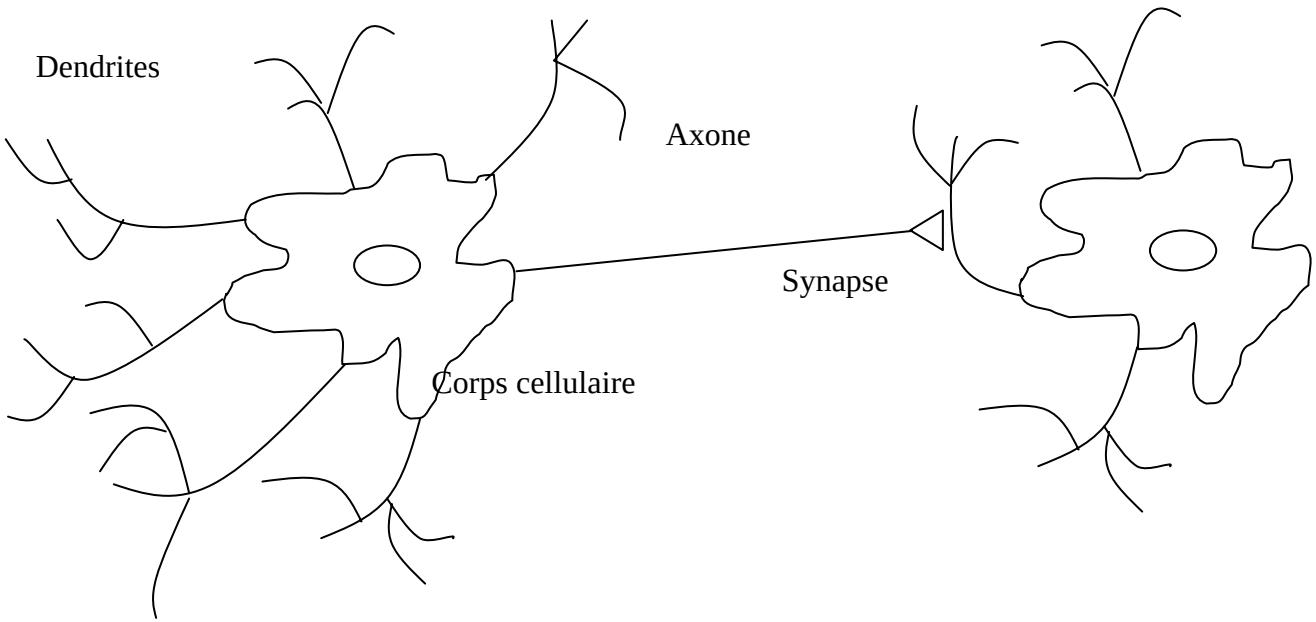
- 1 Présentation
- 2 Apprentissage
- 3 Analyse en composantes principales
- 4 Classification
- 5 Réseaux de Hopfield
- 6 Modélisation et commande des processus

Les réseaux de neurones artificiels

Présentation

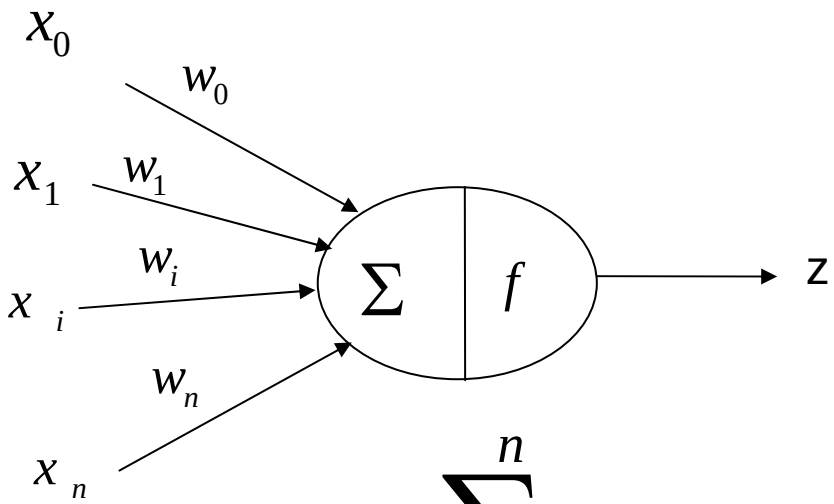
Présentation

Neurone biologique



Présentation

Neurone artificiel

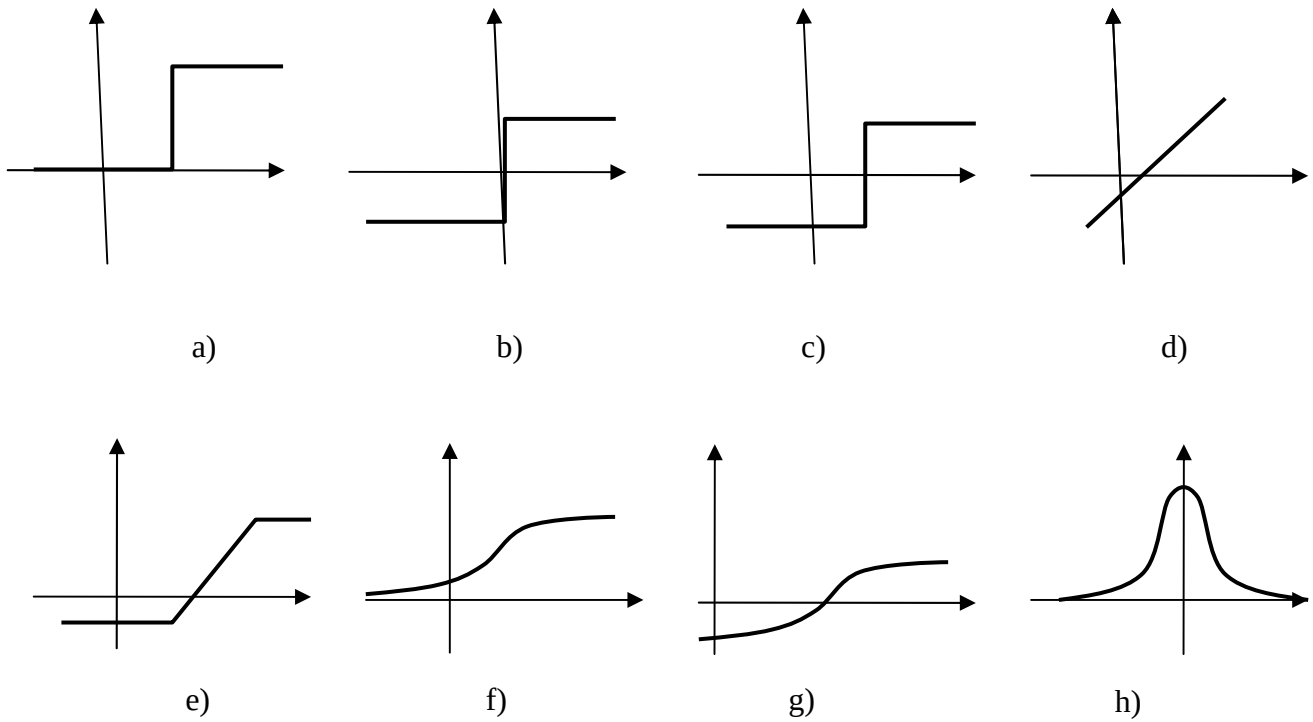


$$y = \sum_{i=0}^{n} w_i x_i$$

$$z = f(y)$$

Présentation

Fonctions d’activation



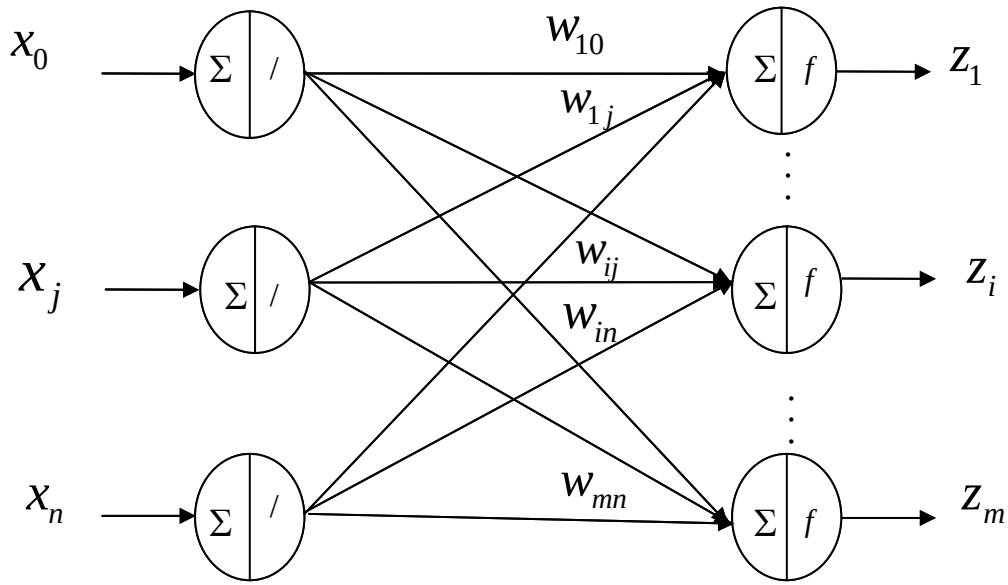
Présentation

Le perceptron

Le perceptron comporte une couche d'entrée constituée de neurones élémentaires dont la fonction d'activation est linéaire et une couche de sortie constituée d'un ou de plusieurs neurones dont la fonction d'activation est en général du type plus ou moins, ou tout ou rien

Présentation

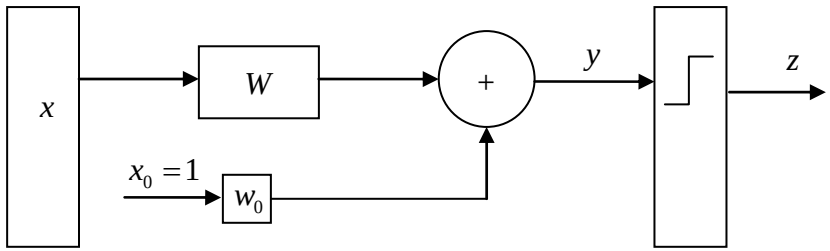
Schéma du perceptron



$$y = Wx + w_0 x_0$$

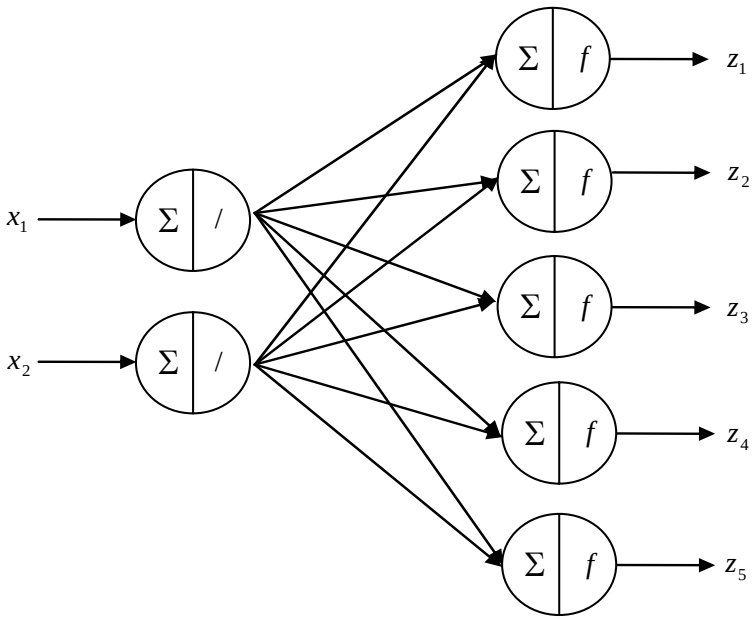
PERCEPTRON

représentation matricielle



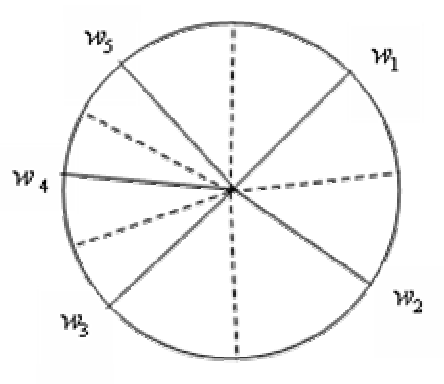
Perceptron

Séparation linéaire



$$\left.\begin{aligned} w_i^T &= [w_{i1}, w_{i2}] \\ w_{i0} &= 0 \\ y_i &= \sum_{j=1}^{2} w_{ij} x_j \\ &= w_i^T x \end{aligned}\right\} \quad \forall i = 1, 2, \ldots, 5$$

Classification à partir du Perceptron



Présentation des réseaux multicouches

Réseaux multicouches

Dans ce cas, le réseau comporte en général au moins trois couches : une couche d’entrée, une ou plusieurs couche(s) cachée(s) et une couche de sortie l’information circulant de l’entrée vers la sortie à travers la (les) couche(s) cachée(s).

Réseaux multicouches, notations

$z_i^{(l)}$: sortie du $i^{ième}$ neurone de la couche *l* à n_l neurones

$$z^{(l)} = \left[z_1^{(l)}, z_2^{(l)}, \ldots, z_{n_l}^{(l)} \right]^T$$

$w_{ij}^{(l)}$: poids liant le $j^{ième}$ neurone de la couche *l -1* au $i^{ième}$ neurone de la couche *l*

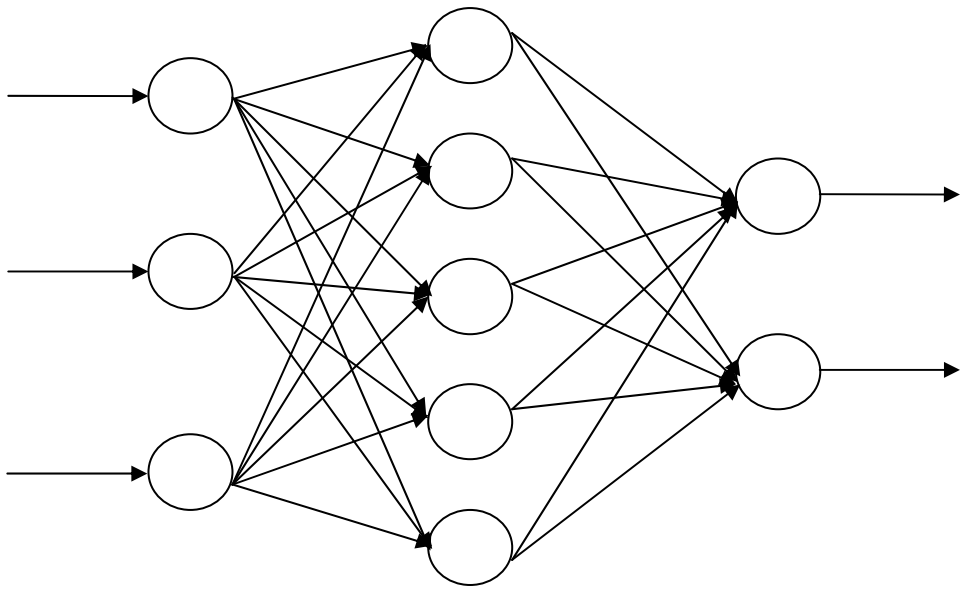
$f_i^{(l)}(\cdot)$: fonction d'activation du $i^{ième}$ neurone de la couche *l*

$$z_i^{(1)} = x_i \qquad x = \left[x_1, x_2, \ldots, x_{n_1} \right]^T$$

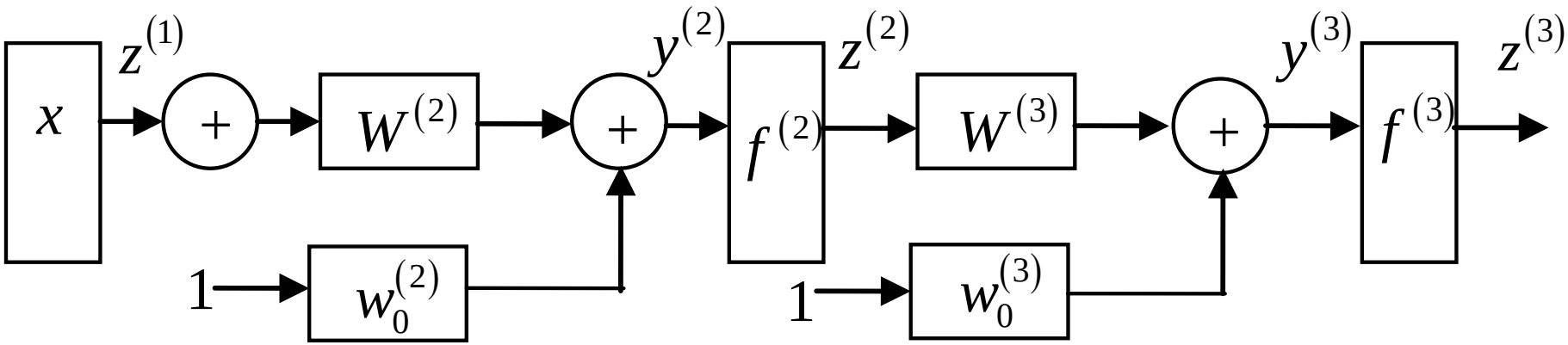
$$y_i^{(l+1)} = \sum_{j=1}^{n_l} w_{ij}^{(l+1)} z_j^{(l)} + w_{i0}^{(l+1)}$$

Représentation

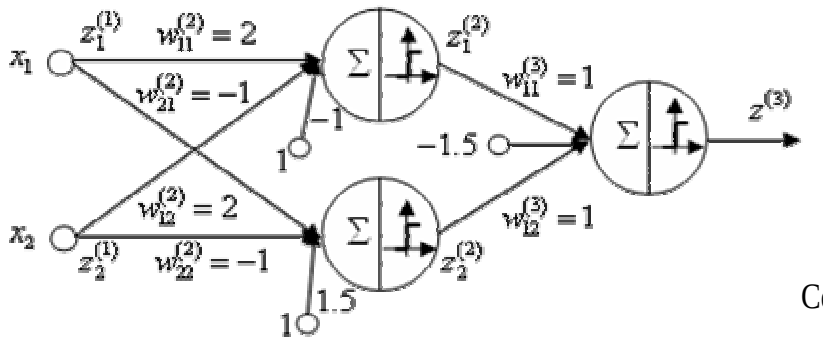
Schéma du réseau multicouches



Représentation matricielle



Réalisation du OU exclusif



Il apparaît que $z^{(3)}$ est égal à 1 pour $x_1 \neq x_2$ et à 0 dans le cas contraire.

Il vient en effet :

$2x_1 + 2x_2 - 1 > 0$ si x_1 et/ou $x_2 = 1 \Rightarrow z_1^{(2)} = 1$

$2x_1 + 2x_2 - 1 < 0$ si $x_1 = x_2 = 0 \Rightarrow z_1^{(2)} = 0$

$-x_1 - x_2 + 1.5 < 0$ si $x_1 = x_2 = 1 \Rightarrow z_2^{(2)} = 0$

$-x_1 - x_2 + 1.5 > 0$ si x_1 et/ou $x_2 = 0 \Rightarrow z_2^{(2)} = 1$

Pour avoir $z^{(3)} = 1$, il faut avoir :

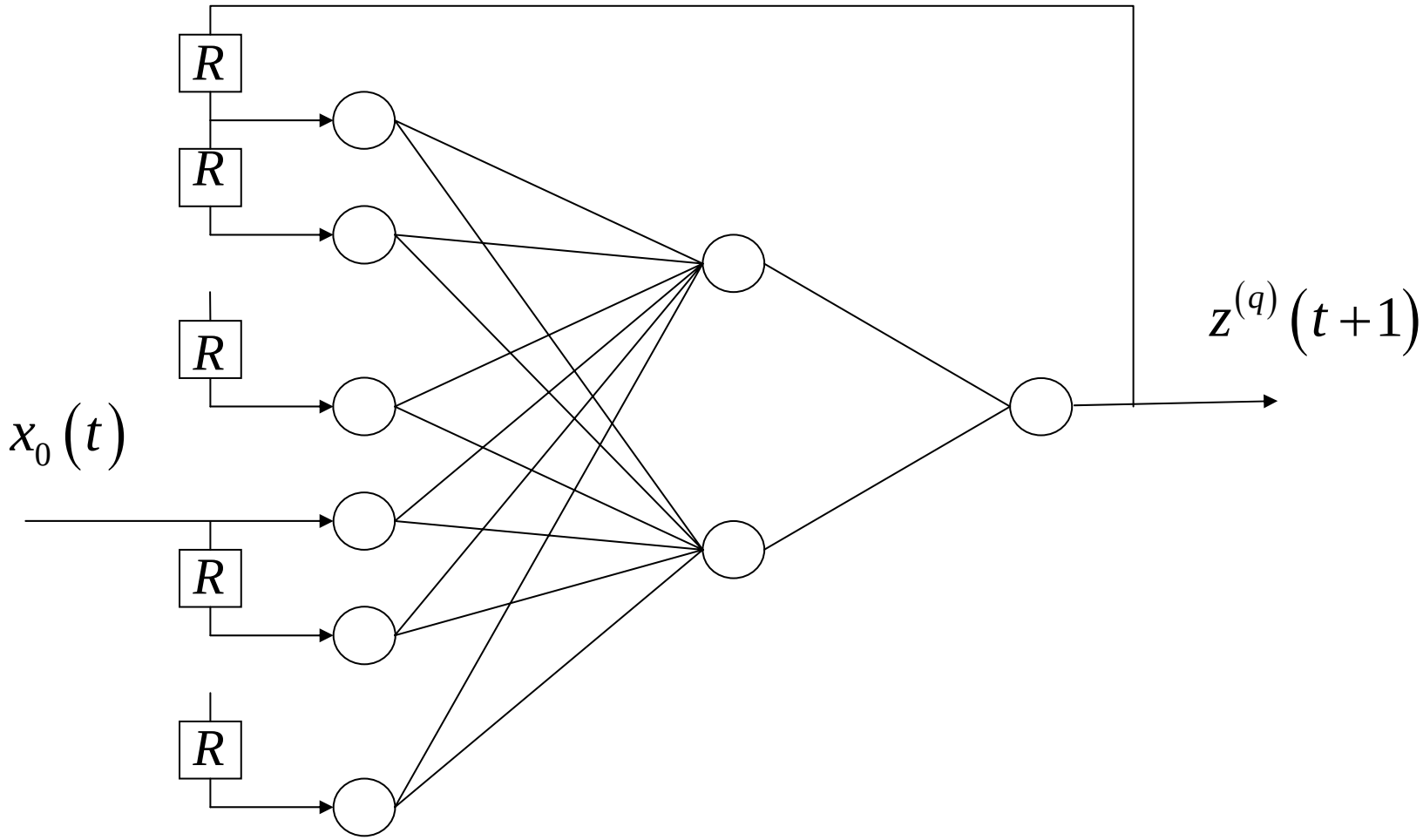
$z_1^{(2)} = z_2^{(2)} = 1$

Ce qui exclut le cas :

$x_1 = x_2$.

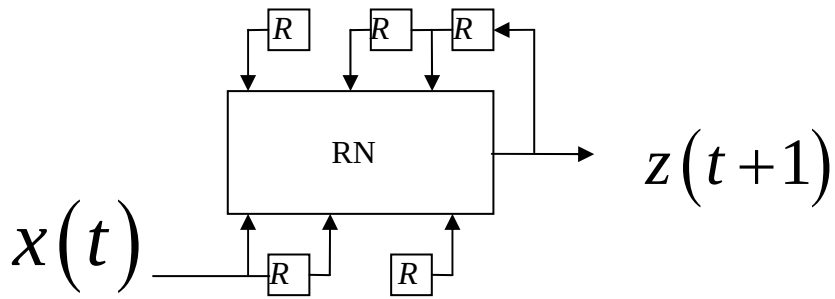
Présentation

Réseaux de neurones récurrents



Présentation

Schémas simplifiés



Les Réseaux de neurones artificiels

APPRENTISSAGE

Apprentissage non supervisé

Règle de Hebb

$w_{ij}(t)$: valeur à l'instant du poids liant le neurone *j* au neurone *i*

$z_i(t)$: sortie à l'instant *t* du $i^{\text{ième}}$ neurone

$f(v)$: fonction d'activation du type tout ou rien

$$w_{ij}(t+1) = w_{ij}(t) + \mu\, z_i(t)\, z_j(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \mu\, z_i(t)\, z_j(t) - \gamma w_{ij}(t)$$

Apprentissage non supervisé

Règle de Kohonen

Cette règle s'écrit sous la forme :

$$w_{ij}(t+1) = w_{ij}(t) + \mu\left(z_i(t) - w_{ij}(t)\right)$$

avec : $i \in I(t)$.

Elle permet au neurone d'apprendre le vecteur présenté en entrée et donc d'être utilisé en reconnaissance des formes.

L'apprentissage a lieu lorsque l'indice du neurone appartient à un ensemble $I(t)$, défini en fonction des objectifs.

Apprentissage supervisé

Règle du perceptron

L'objectif est d'apprendre K patrons p_k

Notons $p(t)$ le patron présenté à l'instant t, $s(t)$ la sortie désirée et $z(t)$ la sortie réelle

Erreur pour la sortie du $i^{\text{ème}}$ neurone $e_i(t) = s_i(t) - z_i(t)$

$$w_i(t+1) = w_i(t) + e_i(t)\, p(t)$$

$$w_{i0}(t+1) = w_{i0}(t) + e_i(t)$$

Variante de la méthode

$$w_i(t+1) = w_i(t) + \eta(t) e_i(t)\, p(t)$$

Apprentissage compétitif standard

$$z_j^{(1)}(t) = \left\| w_j(t) - x(t) \right\|$$

$$w_j = \left[w_{j1}, w_{j2}, \ldots, w_{jn} \right]^T$$

$$z^{(2)}(t) = \arg\min_j \left\| x(t) - w_j(t) \right\|$$

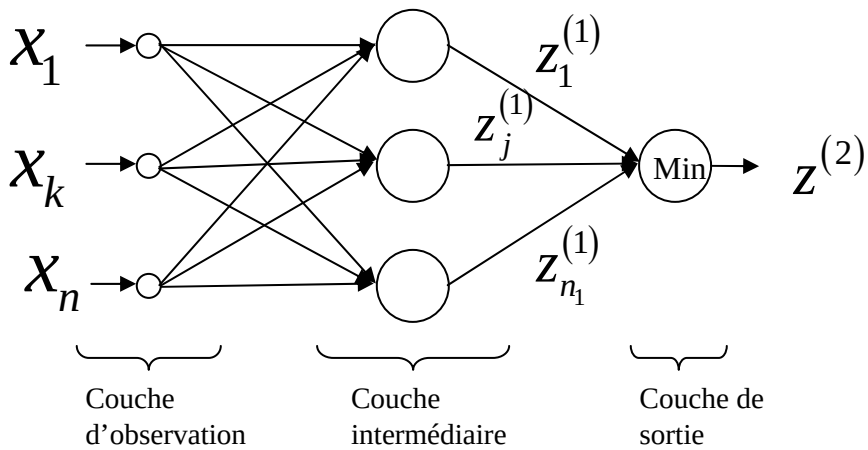
$$E(t) = \frac{1}{2} \sum_{j=1}^{n_1} \alpha_j(t) \left\| x(t) - w_j(t) \right\|^2$$

$$\frac{\partial E(t)}{\partial w_j} = -\alpha_j(t) \left(x(t) - w_j(t) \right)$$

$$w_j(t+1) = w_j(t) - \eta(t) \frac{\partial E(t)}{\partial w_j}$$

Apprentissage compétitif standard

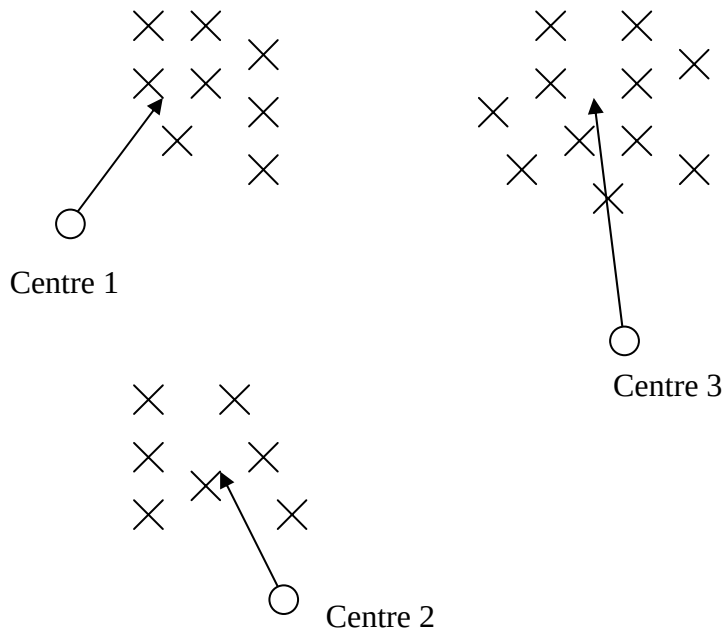
Réseau



Apprentissage compétitif

Algorithme des K-moyennes

C’est l’algorithme le plus simple et le plus répandu en classification automatique.



$\mathcal{C}_j$

Algorithme des K-moyennes

Classe $\mathcal{C}_j$

$$x(p) \in \mathcal{C}_j \Leftrightarrow \left\| x(p) - w_j \right\| = \min_k \left\| x(p) - w_k \right\|$$

$$E = \frac{1}{2} \sum_{p=1}^{N} \sum_{k=1}^{K} \alpha_k(p) \left\| x(p) - w_k \right\|^2$$

$$\frac{\partial E_k}{\partial w_j} = -\sum_{p=1}^{N} \alpha_j(p) \left(x(p) - w_j \right)$$

$$w_j = \frac{\sum_{p=1}^{N} \alpha_j(p) x(p)}{\sum_{p=1}^{N} \alpha_j(p)}$$

Algorithme des K-moyennes

Cet algorithme converge en un nombre fini d'itérations, mais la solution obtenue peut varier suivant l'initialisation.

Si les observations arrivent séquentiellement, il est possible d'adopter une forme itérative de l'algorithme.

Notons $x(t)$ l'observation disponible à l'instant t et $w_j(t)$ le vecteur définissant le centre de la classe $\mathcal{C}_j$ au même instant. Il vient :

$$w_j(t+1) = w_j(t) + \frac{\alpha_j(t+1)}{N_j(t+1)}\left(x(t+1) - w_j(t)\right)$$

avec : $N_j(t)$ le nombre d'éléments de la classe $\mathcal{C}_j$ à l'instant t :

$$N_j(t+1) = N_j(t) + \alpha_j(t+1)$$

A l'instant $t+1$, seul le centre w_j tel que :

$$\alpha_j(t+1) = 1$$

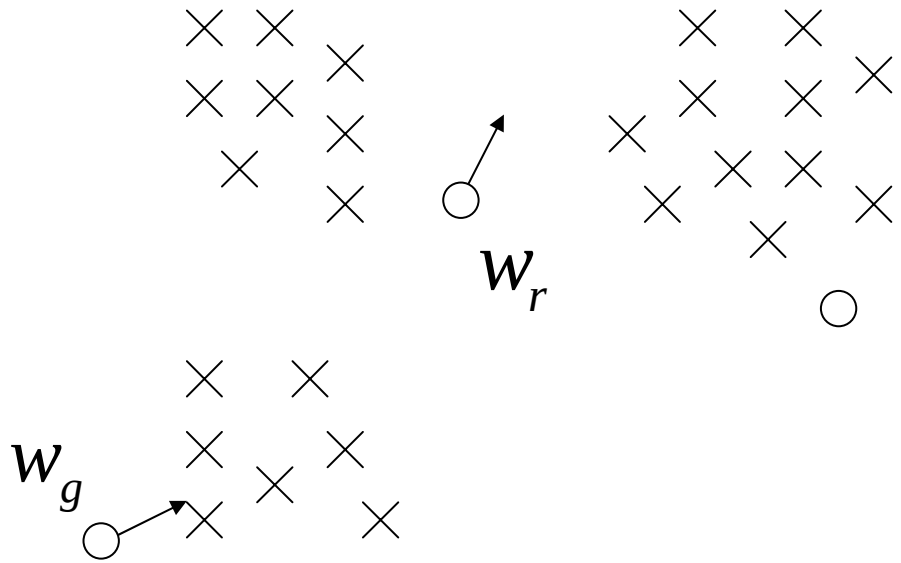
est actualisé, les autres restant inchangés.

Apprentissage compétitif

Apprentissage pénalisant le rival

$$g = \arg\min_{k} \left\| x(t) - w_k(t) \right\|$$

$$r = \arg\min_{k \neq g} \left\| x(t) - w_k(t) \right\|$$



Apprentissage compétitif

Apprentissage pénalisant le rival

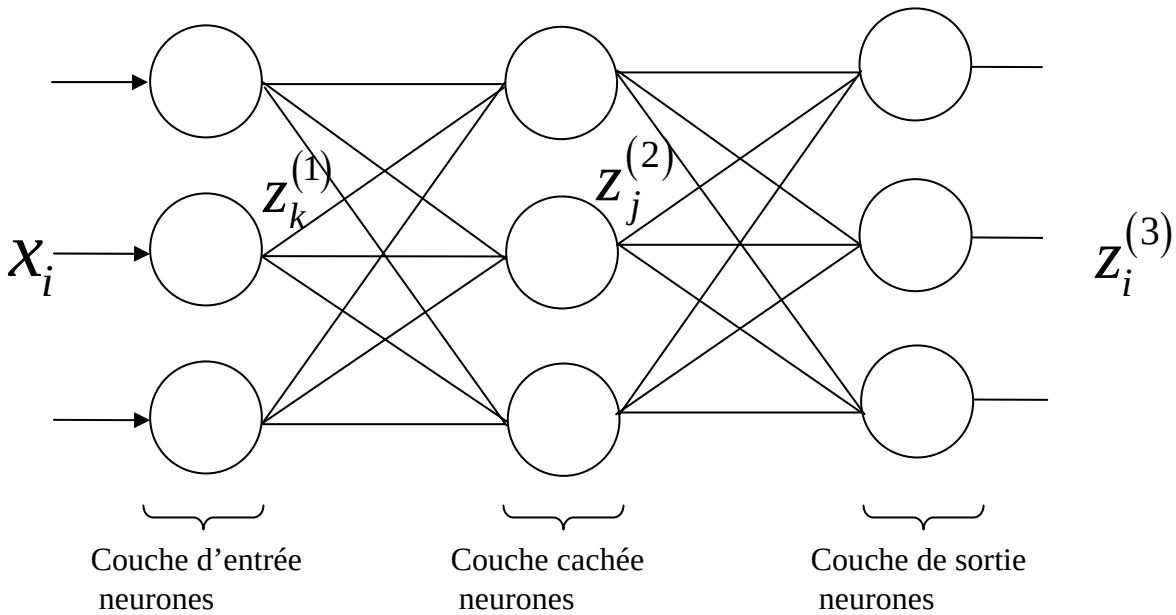
$$w_g(t+1) = w_g(t) + \eta(t)\left(x(t) - w_g(t)\right)$$
$$w_r(t+1) = w_r(t) - \gamma(t)\left(x(t) - w_r(t)\right)$$
$$w_k(t+1) = w_k(t) \;\forall k \notin \{g, r\}$$

Algoritme compétitif pénalisant le rival

- si le nombre K de neurones est supérieur au nombre réel de classes en présence, les neurones gagnants évoluent vers les centres des classes et les autres s'éloignent de l'ensemble des observations ;
- si le nombre K de neurones est inférieur au nombre de classes en présence, il y a oscillation des vecteurs poids entre les classes durant l'apprentissage ; ce qui indique la nécessité d'ajouter une ou plusieurs classes.

Apprentissage supervisé

- Réseau à 3 couches



Apprentissage supervisé

- Rétropropagation du gradient

$$z_i^{(3)}(t) = f\left[\sum_{j=1}^{n_2} w_{ij}^{(3)}(t)\, f\left(\sum_{k=1}^{n_1} w_{jk}^{(2)} z_k^{(1)}(t) + w_{j0}^{(2)}(t)\right) + w_{i0}^{(3)}(t)\right],\ \forall i = 1,\ldots,n_3$$

$$E(t) = \frac{1}{2}\sum_{i=1}^{n_3}\left(z_i^{(3)}(t) - s_i(t)\right)^2$$

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) - \eta(t)\frac{\partial E(t)}{\partial w_{ij}^{(l)}}$$

Apprentissage supervisé du réseau multicouche

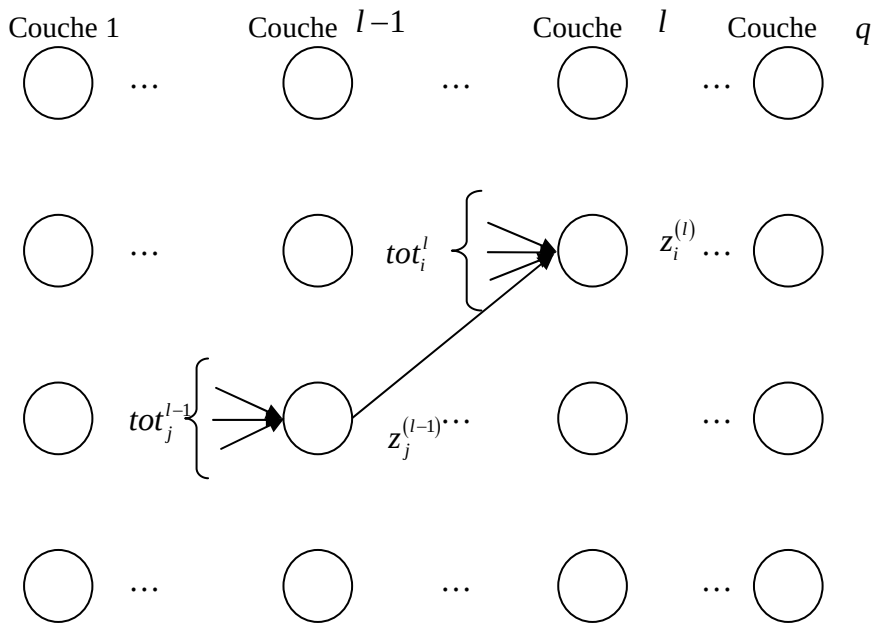
Méthode de rétropropagation du gradient

$$E(t) = \frac{1}{2} \sum_{i=1}^{n_q} \left(z_i^{(q)}(t) - s_i(t) \right)^2$$

$$W^{(l)}(t+1) = W^{(l)}(t) - \eta(t) \frac{\partial E(t)}{\partial w^{(l)}}$$

Apprentissage supervisé

Rétropropagation du gradient



Algorithme de rétropropagation du gradient

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) - \eta \left(\frac{\partial E(t)}{\partial z_i^{(l)}(t)} \right) \left(\frac{\partial z_i^{(l)}(t)}{\partial tot_i^{l}(t)} \right) \left(\frac{\partial tot_i^{l}(t)}{\partial w_{ij}^{(l)}} \right)$$

$$w_{ij}^{(q)}(t+1) = w_{ij}^{(q)}(t) - \eta(t)\,\delta_i^{(q)}(t)\,z_j^{(q)}(t)$$

$$\delta_i^{(q)}(t) = \left(z_i^{(q)}(t) - s_i(t) \right) \frac{\partial f\left(tot_i^{q}(t) \right)}{\partial tot_i^{q}(t)}$$

Les réseaux de neurones artificiels

Analyse en composantes principales

Analyse en composantes principales

Ensemble des observations:

$$X = \left\{ x_1, x_2, \ldots, x_p, \ldots, x_P \right\}$$

Ensemble de leurs projections :

$$Y = \left\{ y_1, y_2, \ldots, y_p, \ldots, y_P \right\}$$

$$x_k = \left[x_{k1}, x_{k2}, \ldots, x_{kn} \right]^T$$

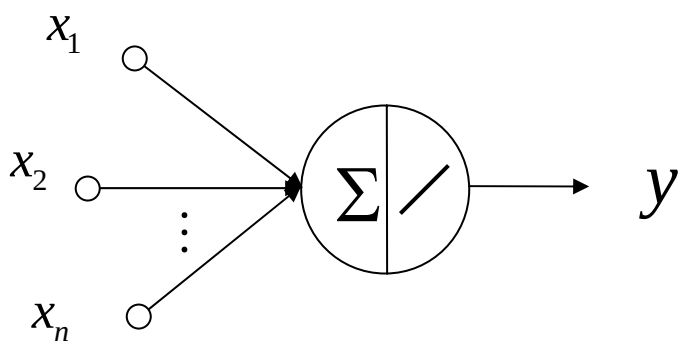
$$y_k = \left[y_{k1}, y_{k2}, \ldots, y_{km} \right]^T$$

$$Y = \varphi(X)$$

$$x_p \in \mathbf{R}^n \rightarrow y_p \in \mathbf{R}^m$$

Analyse en composantes principales

Extraction de la composante principale



$$\Sigma = \frac{1}{P-1}\sum_{p=1}^{P}\left(x_p - \mu\right)\left(x_p - \mu\right)^T$$

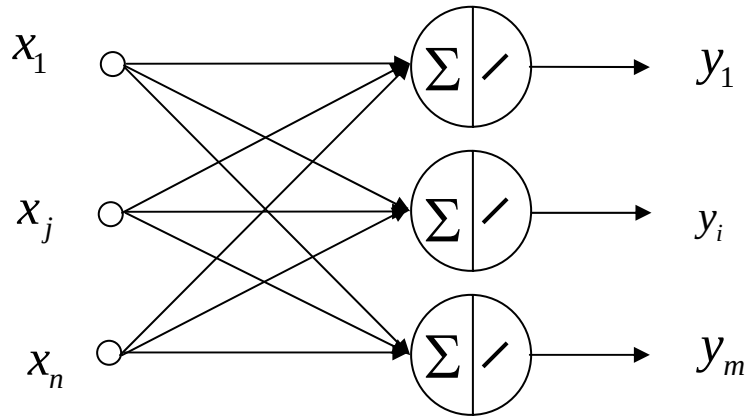
$$\mu = \frac{1}{P}\sum_{p=1}^{P} x_p$$

$$y(t) = \sum_{i=1}^{n} w_i(t)\, x_i(t)$$

$$w_i(t+1) = w_i(t) + \eta(t)\, y(t)\left(x_i(t) - y(t)\, w_i(t)\right)$$

Analyse en composantes principales

Réseaux de neurones hiérarchiques



$$y_i(t) = w_i^T(t)\,x(t)\,;\quad i = 1, 2, \ldots, m$$

$$w_i(t+1) = w_i(t) + \eta(t)\,y_i(t)\left[x(t) - \sum_{l=1}^{i} y_l(t)\,w_l(t)\right]$$

Analyse en composantes principales

Réseaux de neurones hiérarchiques

L'algorithme de Hebb généralisé est le suivant :

1) Choisir $\eta(t)$ petit paramètre positif et choisir le nombre maximum d'itérations T .

2) Initialiser les poids du réseau à des valeurs aléatoires proches de zéro.

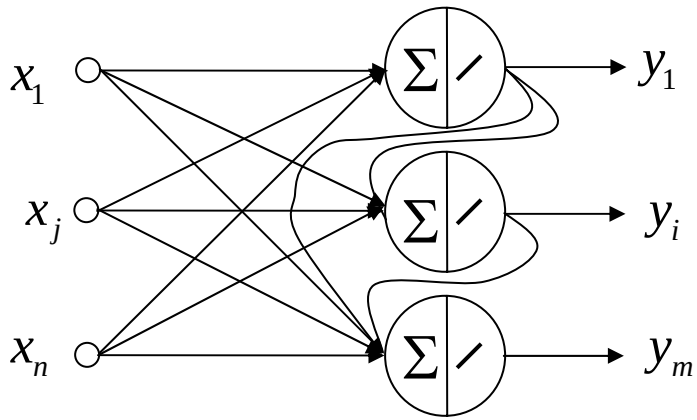
3) Présenter au réseau une observation $x(t)$ de la base d'apprentissage.

4) Calculer les équations du réseau et actualiser les poids conformément à la règle précédemment définie.

5) Tant que $t < T$, faire $t = t + 1$ et aller en 3.

Analyse en composantes principales

Réseaux de neurones adaptatifs



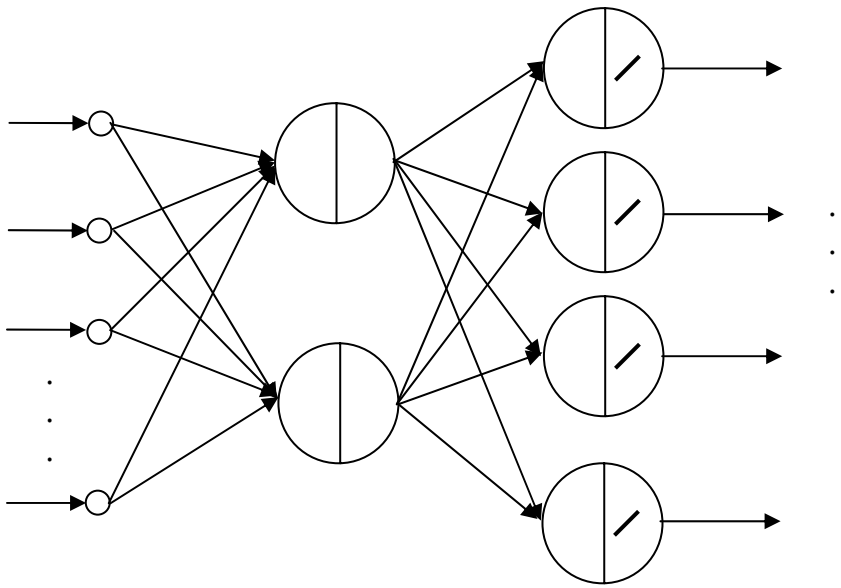
$$y_i(t) = w_i^T(t)\,x(t) + \sum_{l=1}^{i-1} v_{il}(t)\,y_l(t) \qquad j = 1, 2, \ldots, m$$

$$w_i(t+1) = w_i(t) + \eta(t)\left(y_i(t)\,x(t) - y_i^2(t)\,w_i(t)\right)$$

$$v_i(t+1) = v_i(t) - \eta(t)\left(y_i(t)\,Y_{i-1}(t) + y_i^2(t)\,v_i(t)\right)$$

Analyse en composantes principales

- Perceptron avec projection dans la couche cachée



Analyse en composantes principales

Méthodes de projection non linéaire

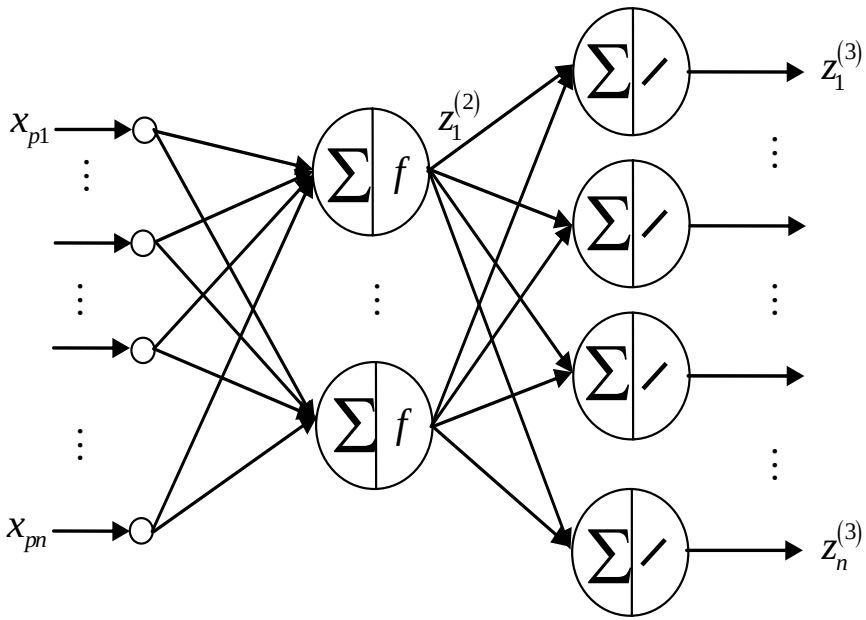
L'algorithme de Sammon consiste à déterminer la projection qui minimise la fonction d'erreur : E_{PNL}

$$E_{PNL} = \left(\sum_{1 \le u,v \le P} \frac{\left(d_{uv}^{x} - d_{uv}^{y} \right)^2}{d_{uv}^{x}} \right) \Bigg/ \left(\sum_{1 \le u,v \le P} d_{uv}^{x} \right)$$

$$y_p(t+1) = y_p(t) - \eta(t) \left[\frac{\partial^2 E_{PNL}}{\partial y_p^2} \right]^{-1} \frac{\partial E}{\partial y_p}$$

Analyse en composantes principales

Perceptron avec projection dans la couche cachée



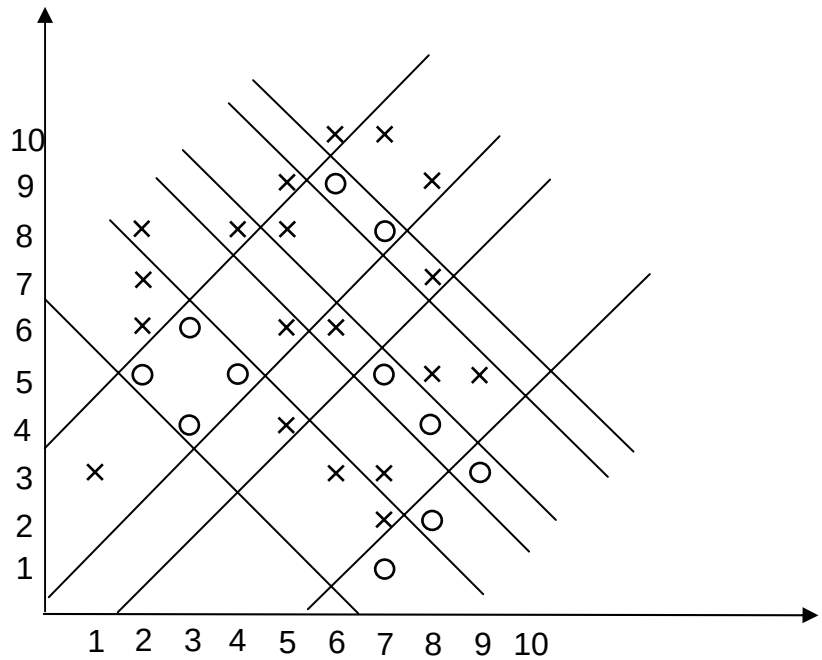
Réseaux de neurones artificiels

Classification

Classification

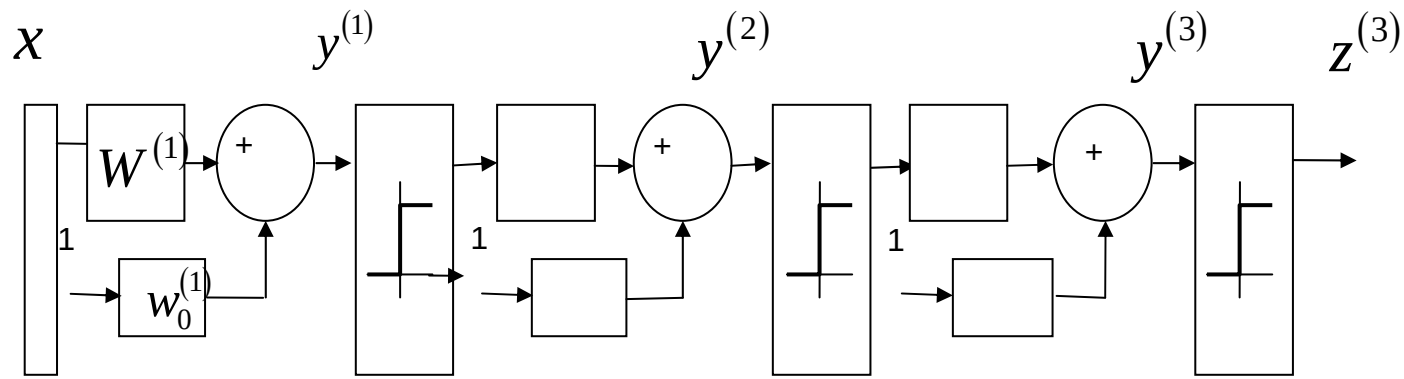
Classification automatique supervisée

Exemple :



Classification

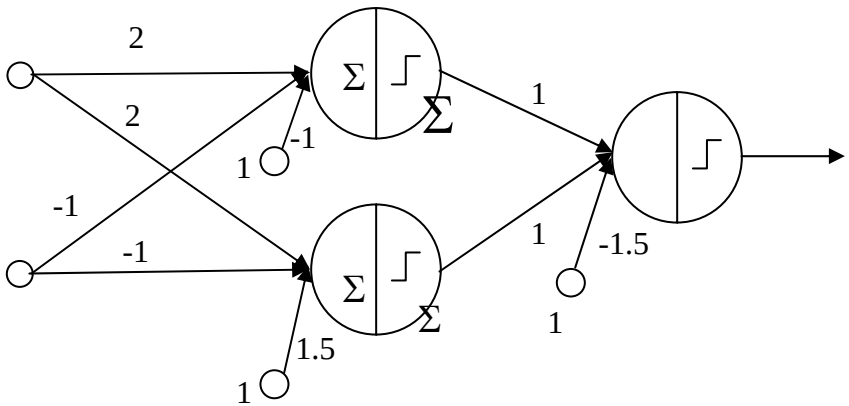
Réseaux multicouches



La première couche de neurones permet de définir des séparateurs linéaires afin de distinguer les classes de vecteurs. La deuxième couche réalise la fonction logique ET permettant de définir des zones et la troisième couche va servir à regrouper les zones de même nature en réalisant la fonction OU.

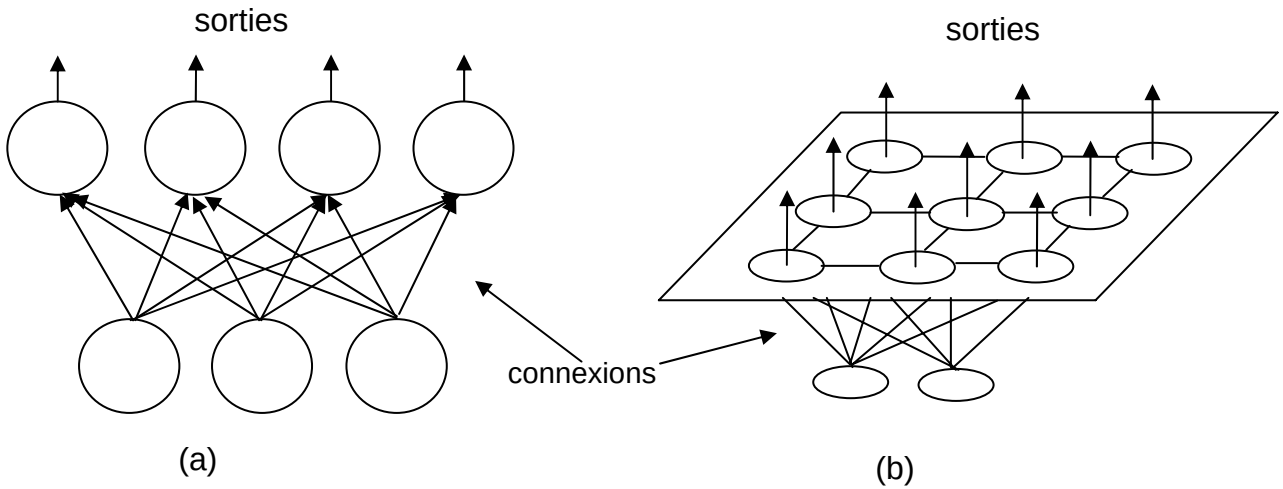
Classification

- Réalisation du « ou » exclusif



Classification

Cartes topologiques de Kohonen



$$f(\sigma) = \frac{1}{1+\mathrm{e}^{-\sigma}}$$

$$\|w_v - x\| = \min_r \|w_r - x\|$$

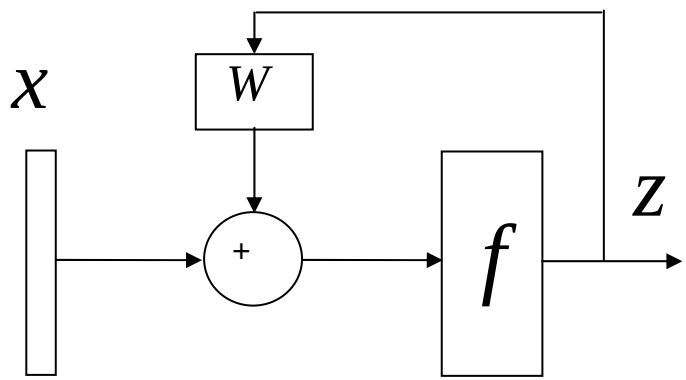
$$w_{v'}(t+1) = w_{v'}(t) + \alpha(v,v')\eta(t)\left(x(t) - w_{v'}(t)\right) \qquad \forall v' \in V$$

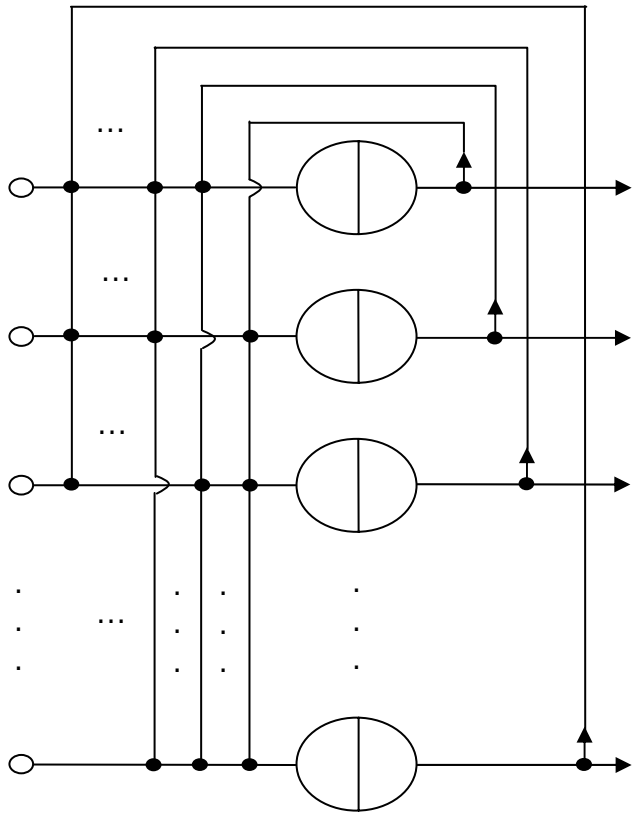
Réseaux de neurones artificiels

Réseaux de Hopfield

Réseaux de Hopfield

Réseaux totalement connectés





$$f\left(y_i\right) = \text{signe}\left(y_i\right)$$

Réseaux de Hopfield discrets

Reconnaissance de patrons

$$E_H = -\frac{1}{2}\sum_i \sum_j w_{ij} z_i z_j - \sum_i x_i z_i$$

$$W = \frac{1}{n}\sum_{k=1}^{p} p_k p_k^T - \frac{P}{n} I$$

Il vient si le $i^{\text{ième}}$ neurone est choisi pour l'actualisation l'instant t+1 :

$$z_j(t+1) = z_j(t) \text{ si } j \neq i$$

$$z_i(t+1) = \text{signe}\left(\sum_{j \neq i} w_{ij} z_j + x_i \right)$$

Initialisation du réseau : $z_i = x_i \quad \forall i$

Réseaux de Hopfield continus

Optimisation

Critère : $E_1 = \frac{1}{2} x^T R x + s^T x + \delta$ avec $x = \left[x_1, x_2, \ldots, x_n \right]^T$

Contraintes : $r_i^T x - s_i = 0 \qquad i = 1, 2, \ldots, l$

$$E' = \frac{1}{2} x^T T x + b^T x + c$$

$$E = \frac{1}{2} x^T T x + b^T x$$

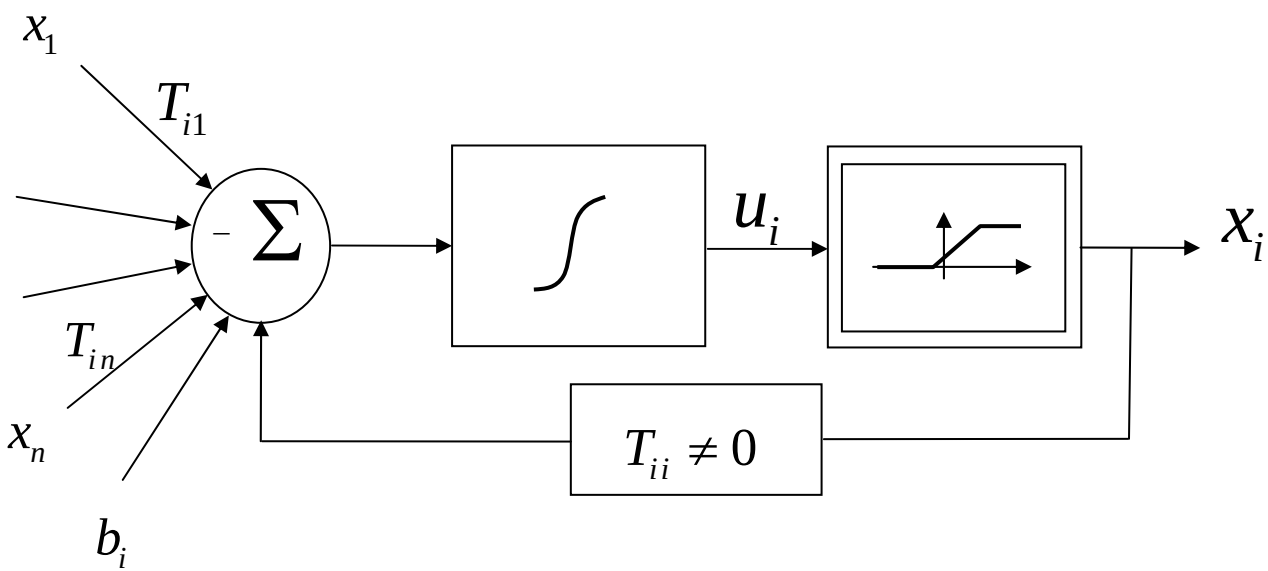
$$f(u_i) = \frac{1}{1 + \exp\left(-\frac{u_i}{\tau} \right)} = x_i$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{\partial E}{\partial x} = -(Tx + b)$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \left(\frac{\partial E}{\partial x} \right)^T \frac{\partial x}{\partial u} \frac{\mathrm{d}u}{\mathrm{d}t} = -(Tx + b)^T \operatorname{diag}\left\{ \frac{x_i}{\tau}(1 - x_i) \right\} (Tx + b)$$

Réseaux de Hopfield continus

Schéma du réseau



Réseaux de Hopfield continus

Problème du voyageur de commerce

Visiter n villes. V_{xi} grandeur égale à 1 si le voyageur visite la ville *x* à l'étape *i*

d_{xy} représente la distance pour aller de la ville *x* à la ville *y*

$$E_1 = \frac{1}{2}\sum_i \sum_x \sum_{y \neq x} \left(d_{xy} V_{xi} V_{y,i+1} + d_{yx} V_{xi} V_{y,i-1} \right)$$

$$\sum_x V_{xi} = 1 \quad \forall i \qquad\qquad \sum_i V_{xi} = 1 \quad \forall x$$

$$E_2 = \sum_x \left(\sum_i V_{xi} - 1 \right)^2 + \sum_i \left(\sum_x V_{xi} - 1 \right)^2 = 0$$

Le réseau de Hopfield est constitué de n^2 neurones

Réseaux de neurones artificiels

Modélisation et commande des processus

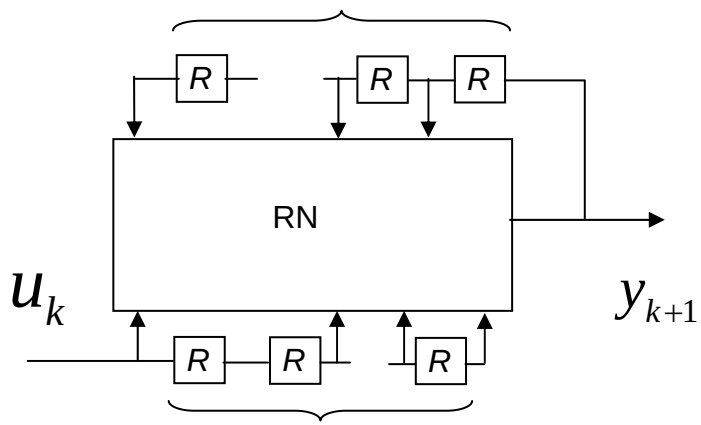
Modélisation et commande des processus

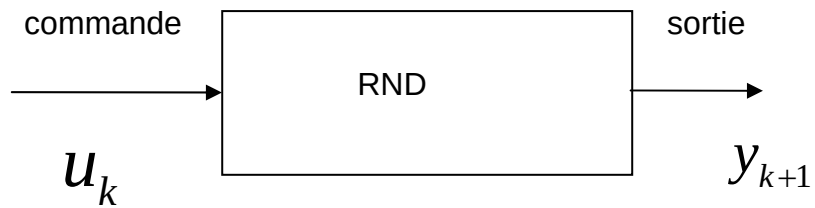
La faculté de recopie par apprentissage des Réseaux de Neurones (RN) permet une utilisation particulièrement intéressante pour la modélisation et la commande des processus, en particulier en utilisant des réseaux de neurones dynamiques.

$$y_{k+1} = f\left(u_k, u_{k-1}, \ldots, u_{k-n_N}, y_k, y_{k-1}, \ldots, y_{k-n_D}\right)$$

Modélisation et commande des processus

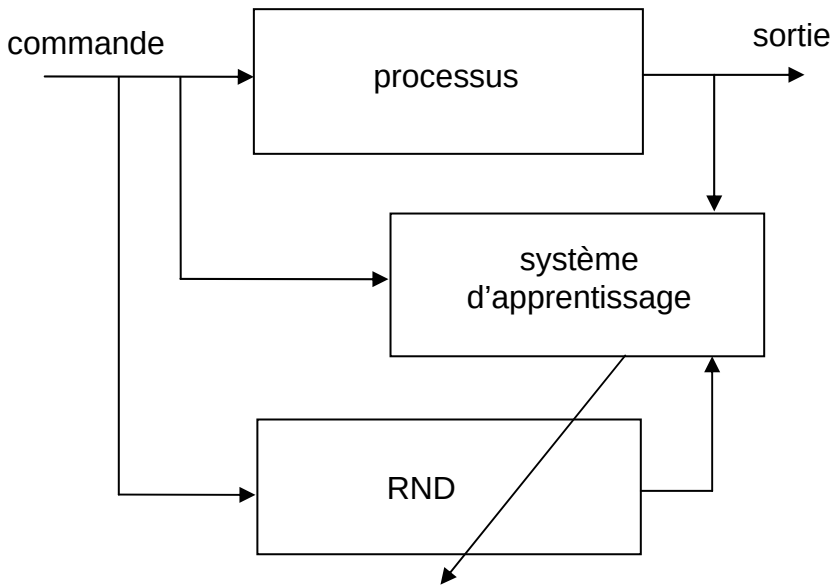
Modélisation





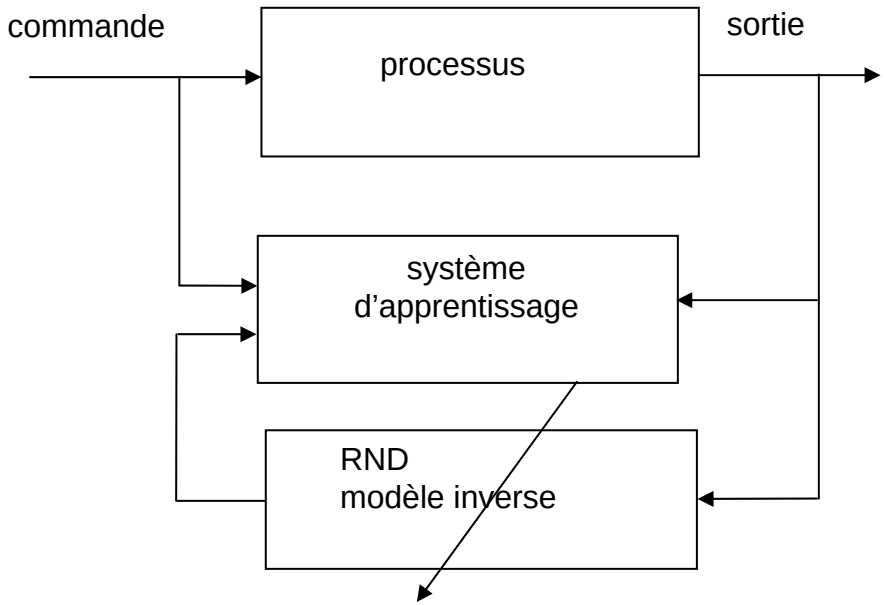
Modélisation

Modèle direct



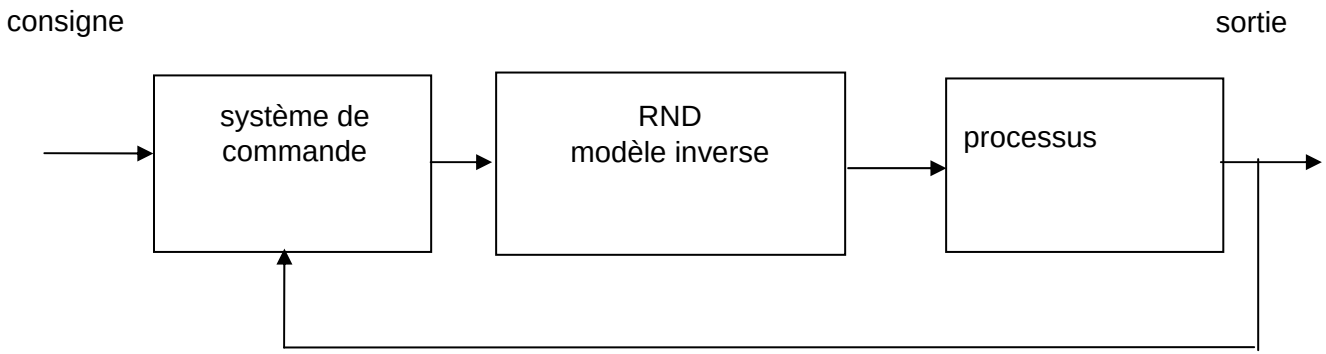
Modélisation

Modèle inverse



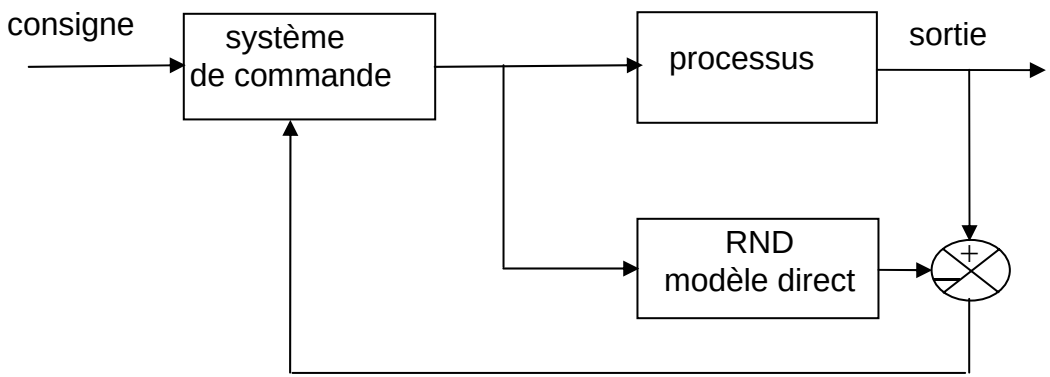
Commande

Utilisation du modèle neuronal inverse



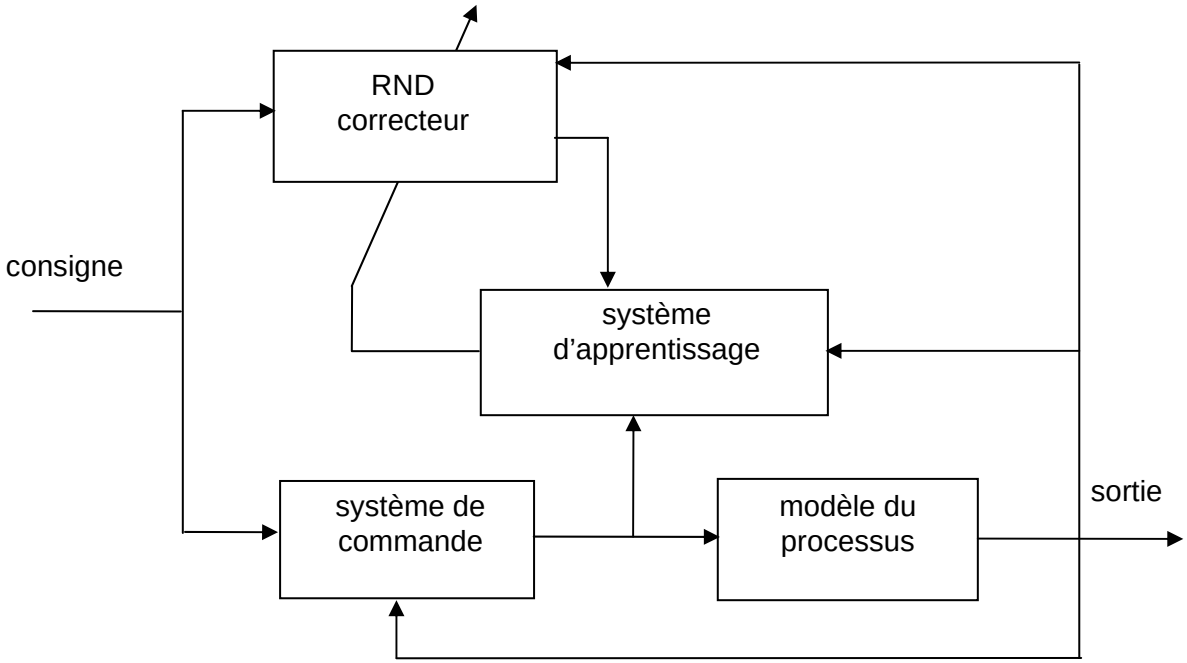
Commande

Commande à Modèle Interne neuronal



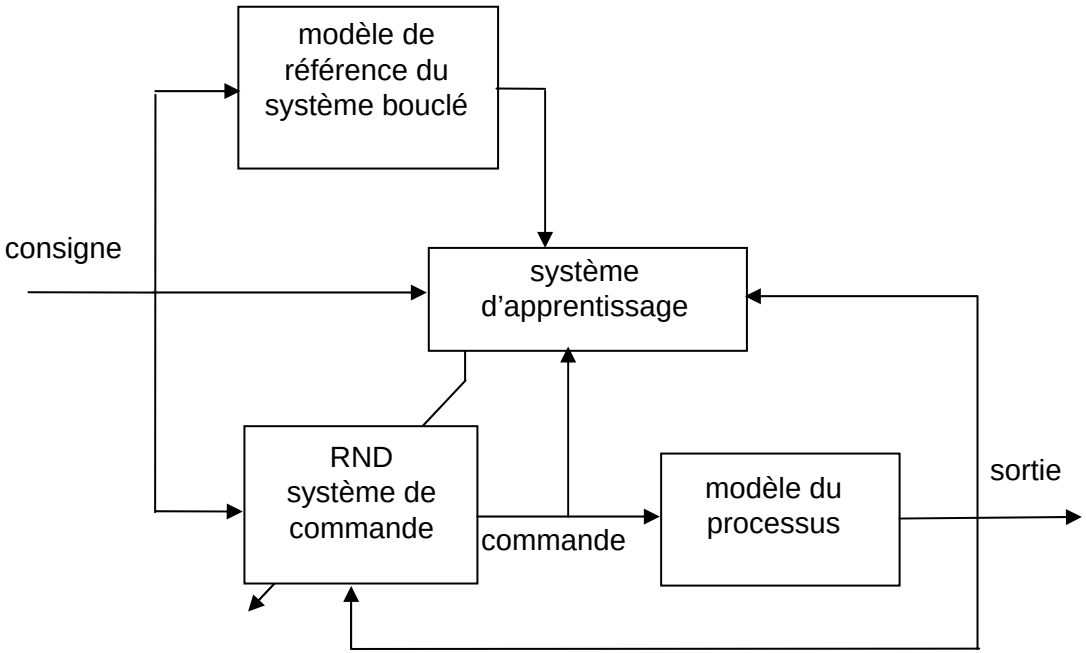
Commande

Recopie d’un système de commande



Commande

Commande à modèle de référence



Réseaux de neurones artificiels

Les réseaux neuro-flous

Réseaux neuro-flous

Le réseau neuro-flou, étudié ici , comporte cinq couches.

La première couche, formée de neurones linéaires, transmet sans changement les variables d'entrée à la deuxième couche qui calcule les degrés d'appartenance des variables d'entrée aux diverses classes des situations préalablement déterminées.

La troisième couche calcule le degré d'appartenance de la partie prémisses en réalisant l'opérateur ET, par exemple en prenant le minimum des appartenances ayant activé la $k^{\text{ième}}$ règle.

En notant I_k l'ensemble des indices des fonctions d'appartenance ayant permis d'activer cette règle, il vient :

$$z_k^{(3)} = \min_{l \in I_k} \left(z_l^{(2)} \right)$$

Les poids liant la troisième à la quatrième couche sont égaux à 1.

Réseaux neuro-flous

La quatrième couche calcule les degrés d'appartenance aux parties conclusions dont les classes sont définies par les fonctions d'activation des neurones de cette couche. Elle réalise la fonction OU, par exemple pour la $j^{\text{ième}}$ variable de la couche 4 :

$$y_j^{(4)} = \max_{k \in I_j} \left\{ z_k^{(3)} \right\}$$

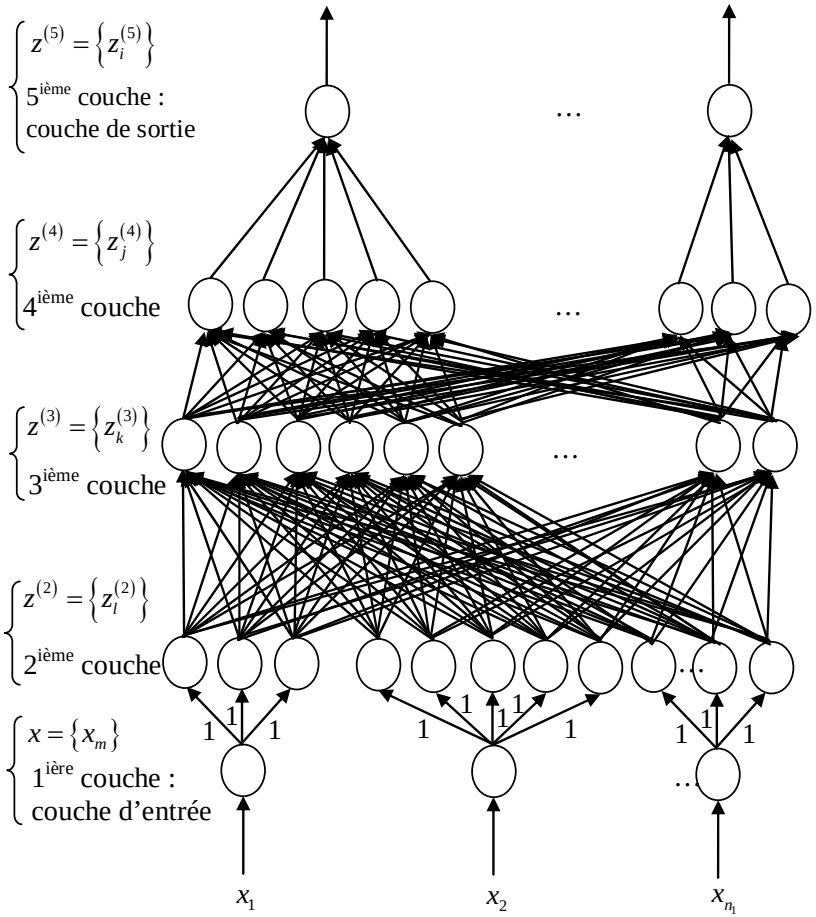
I_j étant l'ensemble des indices des variables de la troisième couche, intervenant dans la détermination de la $j^{\text{ième}}$ variable de la quatrième couche.
La fonction d'appartenance associée à cette variable peut être, là encore, de la forme triangle, trapèze ou fonction gaussienne de la forme :

$$f_j(u) = \exp\left(-\left(\frac{u - c_j^{(4)}}{\sigma_j^{(4)}} \right)^2 \right)$$

La cinquième couche du réseau réalise la défuzzification, par exemple suivant l'expression :

$$z_i^{(5)} = \frac{\sum_{j \in I_i} c_j^{(4)} \sigma_j^{(4)} z_j^{(4)}}{\sum_{j \in I_i} \sigma_j^{(4)} z_j^{(4)}}$$

Réseaux neuro-flous



Réseaux de neurones artificiels

Questions ?