1- les différents types de réseaux de neurones

1.1 la dynamique neuronale

Les réseaux de neurones artificiels (ANN : artificial neural network) constituent une représentation simplifiée d'un système nerveux réel contenant un ensemble de cellules communiquant entre elles.

L'histologiste Hebb fut le premier à postuler que la connectivité dans le cerveau est continuellement en évolution d'une façon dépendante de l'apprentissage.

Des systèmes artificiels implémentent la règle de Hebb dans leurs algorithmes de modifications des connexions intercellulaires pour une tâche de l'information et le traitement.

On peut estimer que le cerveau humain contient environ 10^{11} neurones et 10^{14} synapses (jonctions inter-neuronales). Sur chaque neurone, on peut estimer à plus de 1000 le nombre de synapses.

Le temps temporel d'un potentiel d'action produit par un neurone est de l'ordre de quelques millisecondes, c'est-à-dire un temps 10^6 fois plus grand que le temps caractéristique de modification de bit dans les ordinateurs. La connectivité neuronale est cependant de milliers de fois plus parallèle que dans un ordinateur.

Un neurone artificiel va collecter les signaux et en réaliser une sommation. Si le résultat atteint dépasse un certain seuil, il va envoyer un signal aux autres neurones environnants.

Pour toutes les notations, on appelle généralement **champ local** la somme des valeurs de sortie x_j variant sur un neurone donné pondérée par les valeurs de efficacité des connexions synaptiques. Par exemple, pour un neurone i, dont les efficacités sont W_{i1} W_{i2} ... W_{in}

$$G_i = \sum_j W_{ij} x_j$$

Souvent noté net$_i$ en Champ local en$_i$

La valeur de sortie du neurone est obtenue à partir de cette somme par action de la fonction d'activation f

$$x_i = f(G_i)$$

f peut prendre plusieurs formes

— N —

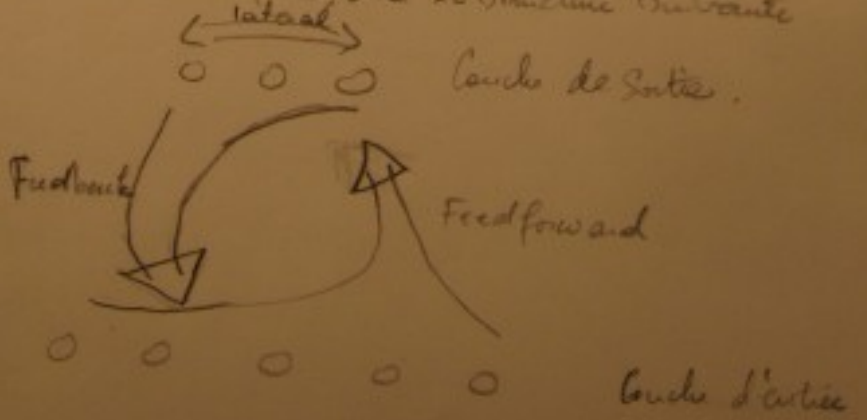
1.2 Les architectures : feedforward, récurrent

Concernant leur structure, les réseaux neuronaux sont divisés en 2 types : réseaux feedforward (propagation vers l'avant) et réseaux récurrents (comportant des boucles de rétrocontrôle)* ou feedback

Réseaux Feedforward : les neurones sont groupés en couches. Les signaux se propagent de la couche d'entrée vers la couche de sortie par des connexions unidirectionnelles, les neurones étant groupés dans une couche mais pas à l'intérieur d'une couche. Les exemples sont

- le perceptron multicouches (MLP : multilayer Perceptron Rumelhart et McClelland 1986)
- le quantificateur vectoriel (LVQ : linear vector quantization Kohonen 1989)
- le réseau de groupement de données (GMDH : group method of data handling Hecht Nielsen 1990)

* De manière schématique, on a la structure suivante



Ces réseaux généralement réalisent des transformations statiques entre un espace d'entrée et un espace de sortie : la sortie à un instant est une fonction seulement de l'entrée à cet instant.

Réseaux récurrents : Dans ce cas, les sorties de certains neurones sont renvoyées vers ces mêmes neurones ou vers des neurones de couches inférieures - les signaux évoluent vers l'avant et vers l'arrière.

Les exemples sont :

- Réseau de Hopfield (1982)
- Le réseau de Elman (1990), Jordan (1986).

La mémoire de ce système peut être considérée comme dynamique : la sortie à un instant donné peut refléter les valeurs d'entrée à cet instant ou bien les valeurs à des instants antérieurs.

—— N ——

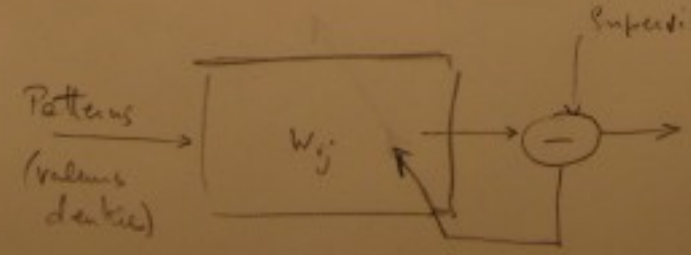
1.3 Les algorithmes d'apprentissage

Les réseaux sont entraînés par 2 type d'algorithme supervisé et non supervisé. Il existe un 3ème type, l'apprentissage par renforcement, qui peut être vu comme une forme spéciale d'apprentissage supervisé.

- Apprentissage supervisé : Les algorithmes d'apprentissage supervisé ajuste le poids ou efficacités des connections intercellulaires suivant la différence entre les valeurs désirées pour les valeurs de sortie du réseau et les valeurs effectivement obtenus par le réseau, ceci pour une donnée des valeurs d'entrée. -

L'apprentissage supervisé nécessite un superviseur pour indiquer les valeurs désirées en sortie.

Les exemples sont : la règle delta (Widrow et Hoff 1960), l'algorithme de retro propagation du gradient ou règle delta généralisée (Rumelhart, McClelland 1986)



Apprentissage non supervisé

Il ne dispose pas de connaissance de valeurs de sortie - Durant l'apprentissage, seuls les patterns sont présentés au réseau qui s'adapte automatiquement afin de réaliser une tâche, celle de catégoriser les patterns en groupes ayant la même caractéristique.
Les exemples d'algorithmes sont ceux de:

- Kohonen (1984)
- Carpenter et Grossberg : ART Adaptive Resonance Theory (1988)

Ce sont des règles d'apprentissage dites compétitives.

Apprentissage par renforcement.

Au lieu d'utiliser un superviseur, l'algorithme emploie une critère de sélection pour évaluer la qualité de la sortie.

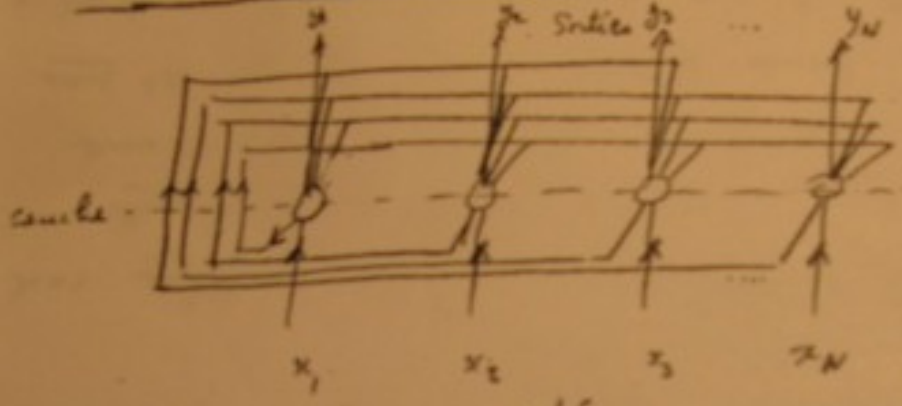
L'exemple typique est

- l'algorithme génétique Holland 1975
  Goldberg 1989.

——N——

1.4 le réseau de Hopfield :

1.4.1 Structure et fonction d'énergie



le réseau de Hopfield est composé de N neurones binaires c-à-d qui peuvent ne prendre que deux valeurs $X_i \in \{0,1\}$ $i=1,2,\dots N$

Appelons $G_i(x) = \sum_{j=1}^{N} w_{ij} x_j$ la somme pondérée par les poids des valeurs w_{ij} de $X_i(X_1 X_2 \dots X_N)$. C'est le "champ local" le neurone i, à l'instant $t+1$, va changer son état X_i de la façon suivante

$$(1.1)\quad \begin{cases} X_i(t+1) = 1 & \text{si } G_i(X(t)) > \theta_i \\ X_i(t+1) = 0 & \text{si } G_i(X(t)) < \theta_i \\ X_i(t+1) = X_i(t) & \text{si } G_i(X(t)) = \theta_i \end{cases}$$

Seuil (ne peut varier au cours du calcul à priori)

Il peut exister plusieurs variantes.
La mise à jour (updating) peut se faire de 2 façons, suivant le nombre de cellules considérées.
Dans le cas du schéma d'itération parallèle, toutes les cellules sont mises à jour à chaque cycle suivant la règle (A.1)
Dans le schéma d'itération asynchrone, seule une cellule est mise à jour à chaque pas d'itération. Dans ce cas, la suite d'indices cellulaire est soit prédéfinie, soit choisie aléatoirement. De fait, des mises à jour par blocs peuvent également être faites.

Soit une mise à jour asynchrone, avec la condition que toutes les cellules ont été mises à jour dans un temps fini. Le système va alors converger vers des points fixes (points dans l'espace des états tels que $(X_0^0 X_0^0 \dots X_{N0}^0)$) quand les fonctions ont certaines formes.

Fonction d'énergie
Supposons les connexions de forme symétrique c.à.d $W_{ij} = W_{ji}$, le système admet une fonction de type Lyapunov (ou fonction d'énergie) qui est une fonction bornée inférieurement et qui est non croissante lors de l'évolution du système $t \to t+1$

Cette fonction est

$$E(X) = -\frac{1}{2} \sum_{\substack{i,j=1 \\ j \neq i}}^{N} w_{ij} X_i X_j + \sum_{i=1}^{N} \theta_i X_i \qquad (1.2)$$

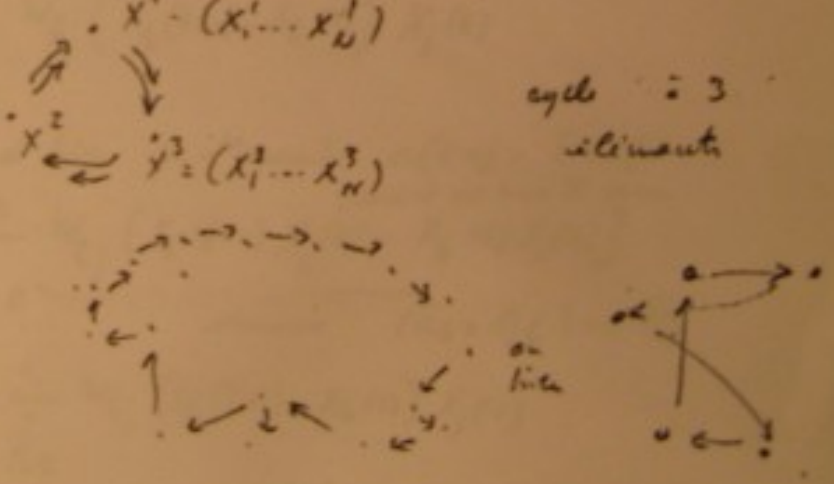
(1.3) En fait, on a $E(X(t+1)) \leq E(X(t))$

De plus, comme $X(t+1)$ et $X(t)$ ne diffèrent que par un élément au plus, on a

$$E(X(t+1)) = E(X(t))$$

Cela implique $X(t+1) = X(t)$. On en déduit que dans le cas d'une dynamique asynchrone, les seuls attracteurs sont des points fixes. Les attracteurs sont le domaine de l'espace des états vers lesquels tendent asymptotiquement les états.

Des attracteurs qui ne seraient pas des points fixes pourraient être des cycles, des attracteurs "étranges" (de structure chaotique)



Pendant (N.3) - Supposons que le réseau a été laissé pour la mise à jour à l'instant t, et soit $\delta X_k(t)$ sa variation alors que $\delta E(t)$ désigne la variation de la fonction d'énergie. Donc :

(N.4) $\delta E(t) = E(t+1) - E(t)$

$$= -\frac{1}{2} \sum_{\substack{i,j \\ j \neq i}} W_{ij} \left[X_i(t+1) X_j(t+1) - X_i(t) X_j(t) \right] (\delta_{ik} + \delta_{jk})$$

$$+ \sum_i \theta_i \left(X_i(t+1) - X_i(t) \right) \delta_{ik}$$

Pour la 1ère somme, on a

$$\sum_i \theta_i \left(X_i(t+1) - X_i(t) \right)$$

avec $X_i(t+1) - X_i(t) = 0$ si $i \neq k$

$\neq 0$ si $i = k$

Donc on reste bien $\sum_i \theta_i (X_i(t+1) - X_i(t)) \delta_{ik}$

De la même façon

$$-\frac{1}{2} \sum_{\substack{i,j \\ i \neq j}} W_{ij} \left[\underbrace{X_i(t+1) X_j(t+1) - X_i(t)}_{A} X_j(t) \right]$$

donc $A = 0$ si $i \neq k$ et $j \neq k$

si $i \neq k$ et $j = k$

$$W_{ik} \left(\underbrace{X_i(t+1) X_k(t+1) - X_i(t) X_k(t)}_{\text{écrire}} \right) = W_{ik} \left(X_i(t) X_k(t+1) - X_i(t) X_k(t) \right)$$

$$= W_{ik} \left(X_i(t) \left(X_k(t+1) - X_k(t) \right) \right) \neq 0$$

De même pour $i \neq k$ $j \neq k$

A présent, si $i = k$ et $j = k$ ceci est impossible car on a vu comme on $i \neq j$ (on peut dire aussi $W_{ii} = 0$)

En effet, on a

$$(H.5)\quad \delta E(t) = -\delta X_k \left(G_k(X(t)) - \theta_k \right)$$

avec (rappel) $G_k(X(t)) = \sum_{\substack{j=1 \\ j \neq k}}^{N} W_{kj} X_j(t)$

le 2^e terme donne bien

$$(H.6)\quad \sum_i \theta_i \left(X_i(t+1) - X_i(t) \right) \delta_{ik} = \theta_k \, \delta X_k(t)$$

(2^e membre dans H.5

D'autre part

$$-\delta X_k \, G_k(X(t)) = \left(X_k(t+1) - X_k(t) \right) \sum_{\substack{j=1 \\ j \neq k}}^{N} W_{kj} X_j(t)$$

$$(H.7)\quad = -\sum_{\substack{j=1 \\ j \neq k}}^{N} W_{kj} \left(X_k(t+1) - X_k(t) \right) X_j(t)$$

Ça correspond bien au 1er membre de (H.4). En effet

→ n'est pas mis à jour

$$(H.4) \Rightarrow -\frac{1}{2} \sum_{\substack{j \\ j \neq k}} W_{kj} \left[X_k(t+1) X_j(t) - X_k(t) X_j(t) \right]$$

somme $(\delta_{ik} + \delta_{jk}) \rightarrow \delta_k$

$$(H.8)\quad = -\frac{1}{2} \sum_{\substack{j \\ j \neq k}} W_{kj} \left(X_k(t+1) - X_k(t) \right) X_j(t)$$

2ème cas $(\delta_{ik} + \delta_{jk}) \to \delta_{jk}$

$$(4.4) \to -\frac{1}{2} \sum_{i \neq k} W_{ik} \underbrace{\left[X_i(t+1) X_k(t+1) - X_i(t) X_k(t) \right]}_{X_i''(t)}$$

$$= -\frac{1}{2} \sum_{i \neq k} W_{ik} \left[X_k(t+1) - X_k(t) \right] X_i(t)$$

$$= -\frac{1}{2} \sum_{j \neq k} W_{jk} \left[X_k(t+1) - X_k(t) \right] X_j(t)$$

$\left(W_{kj} \right)$

c'est bien (4.4) en faisant le changement

Donc donc

$$(4.5)' \quad \delta E(t) = -\delta X_k \left(G_k(x(t)) - \theta_k \right)$$

Si $\delta X_k(t) \neq 0$, $\delta X_k(t)$ a toujours le même signe que $G_k(x(t)) - \theta_k$.

En effet

Si $X_k(t) = 0 \to X_k(t+1) = 1$ si $G_k(x(t)) - \theta_k > 0$

$\delta(X_k(t)) > 0$ ——— > 0

Si $X_k(t) = 1 \to X_k(t+1) = 0$ si $G_k(x(t)) - \theta_k < 0$

$\delta(X_k(t)) < 0$ ——— < 0

Donc $\delta E(t) < 0$

Le réseau de Hopfield est donc un système dynamique à plusieurs degrés de liberté, c'est non linéaire. Sa dynamique lui permet de converger vers des points fixes qui sont des attracteurs.

Cela veut dire que pour une large classe de conditions initiales, les états asymptotiques sont les points fixes. Cette insensibilité par rapport aux conditions initiales est mise à profit dans la mécanisme de reconnaissance de patterns bruités puisqu'ils convergent vers une condition initiale.

Il s'agit à présent d'identifier les points fixes de la dynamique avec les patterns que l'on désire stocker dans le système neuronal. Pour cela, Hopfield a proposé la prescription suivante

1.4.2 Enregistrement et identification

Supposons que l'on veuille stocker M patterns différents $A^1 \dots A^M$, chaque pattern étant un vecteur à N dimensions (N neurones au total)

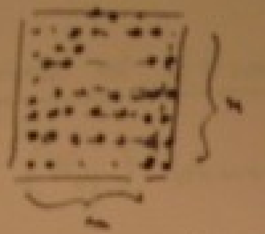
la règle pour construire W_{ij} et alors la nouvelle

$$W_{ij} = (1-\delta_{ij}) \sum_{m=1}^{M} (2A_i^m - 1)(2A_j^m - 1)$$

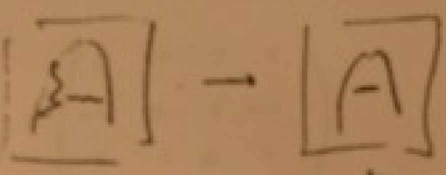
si $A_i^k = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ - on a ainsi $W_{ij} = 0$ par $(1-\delta_{ij}=0) = 0$

Comme $(2A_i^k - 1) = \begin{cases} -1 \text{ si } A_i^k = 0 \\ +1 \text{ si } A_i^k = 1 \end{cases}$

on voit que W_{ij} est renforcé par le pattern A^m si A_i^m et A_j^m prennent la même valeur (0 ou 1)
Soit $A_i^m = 0$ et $A_j^m = 0$ Soit $A_i^m = 1$ et $A_j^m = 1$
Cette concomittance des valeurs de patterns change les connexions. Sinon, si $A_i^m = 0$ et $A_j^m = 1$, la connexion W_{ij} est diminuée pour ce qui concerne le pattern A^m. - Les patterns A^m sont des pistes frées pour la dynamique neuronale
l'algorithme est implémenté dans une fonction en C qui est distribuée.
l'exemple traité porte sur des patterns ayant la forme d'images.



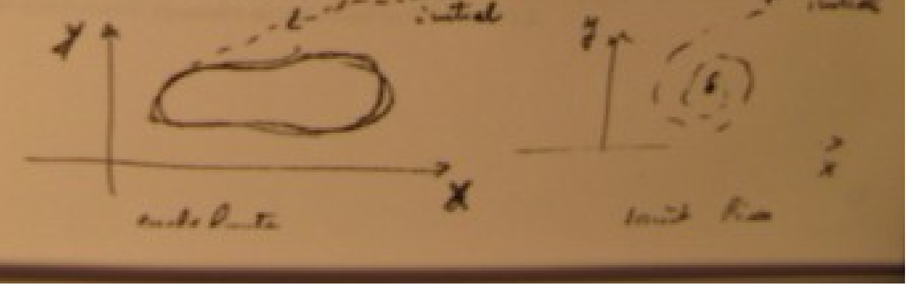
le nombre de cellules N en entrée (sortie) du réseau est $N = m \times n$ – les patterns ont certaine forme qui peuvent être des lettres ou bien avoir d'autre structure.



* point fixe

L'existence de la fonction d'énergie associée à l'existence de point fixe permet d'envisager de présenter (comme conditions initiales de la dynamique) des patterns brouillé situés dans le bassin d'attraction d'un point fixe donné et d'effectuer un parcours vers ce point fixe établissant ainsi une forme de reconnaissance.–

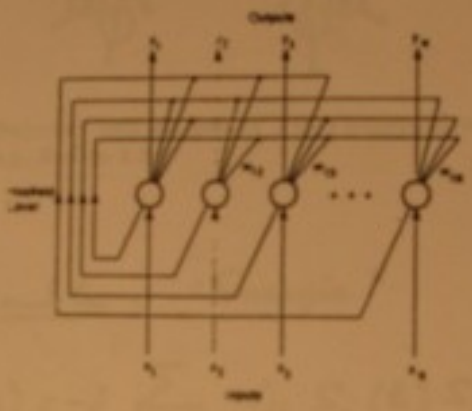
(*) Ceci peut être visualisé sur des systèmes plus simple bidimensionnels. –



Réseaux d'attracteurs (complément sur le modèle d'Hopfield

Traj:

Modèles de mémoire associative,
récurrents à une couche (Hopfield 82 → 95.



Connections : fixées en fonction des patterns que l'on désire enregistrer.

Variables dynamiques cellulaires : $\sigma_i = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
suivant émission ou non émission d'un "spike"

Relation d'entrée sortie

$$S_i(k+1) = Sgn\left(\sum_j W_{ij} S_j(k) - \theta_i\right)$$

$S_i(k+1)$: temps discret ; $\sum_j W_{ij} S_j(k)$: règle de somme ; θ_i : seuil

$S_i = 2\sigma_i - 1 = \begin{pmatrix} +1 \\ -1 \end{pmatrix}$

$Sgn(u) = +1$ si $u > 0$

Sorties du réseau $y_i = x_i(\pi)$ ... y

Trajectoires et Attracteurs.

(W_{ij}) donnée → Trajectoires dans l'espace de tous les états

Exemple : 4 cellules, 2^4 états possibles

(-1 -1 -1 -1) (-1 1 -1 1) ----- (1 1 1 1)

0 1 2 3 4 5 6 7 8 9 A B C D E F

$$W_{ij} = \begin{pmatrix} 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & -1 \\ 1 & 1 & 0 & 1 \\ 1 & -1 & 1 & 0 \end{pmatrix}$$

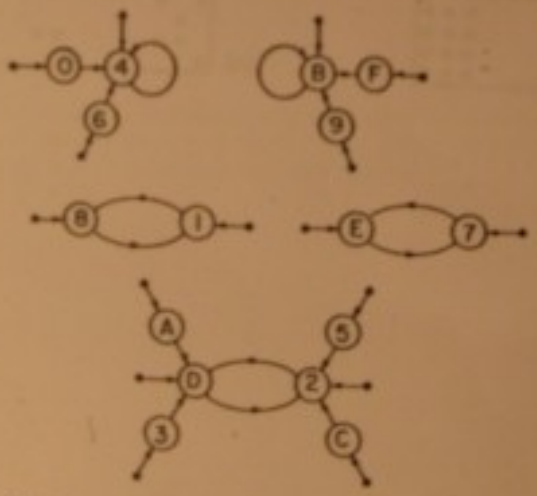


Figure 2.7: Trajectory map of 16 initial states in network 2 (J_{ij} of Eq. 2.25) with synchronous dynamics as described in Section 2.2.2.

13

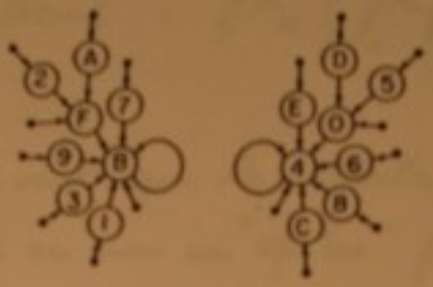


Figure 2.8: Trajectory map for network 2 with sequential, asynchronous dynamics. See e.g., Section 2.2.3.

Variable d'ensemble. "Energie"

$$E(k) = -\frac{1}{2} \sum_{i \neq j} W_{ij} S_i(k) S_j(k)$$

$$\Rightarrow \quad E(k+1) \leq E(k)$$

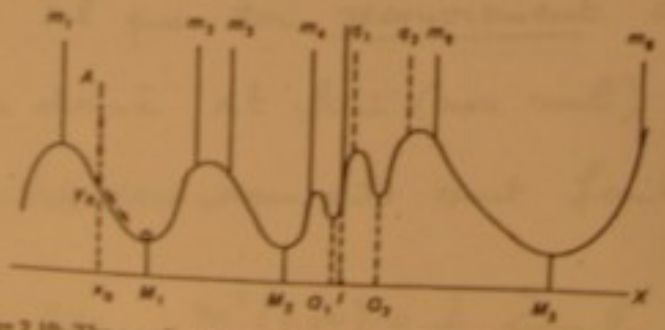
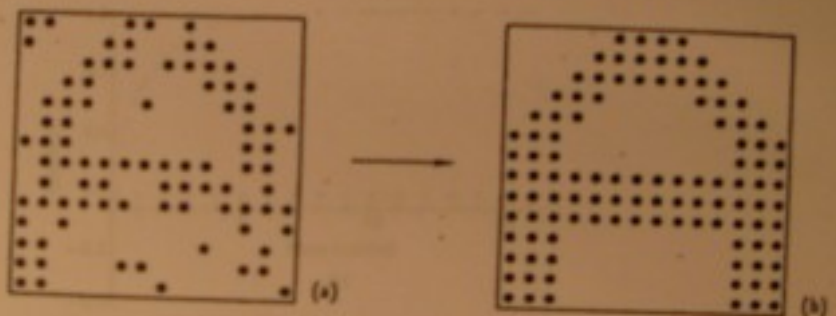


Figure 2.10: The one-dimensional landscape metaphor for associative, readout addressable memory. M_1, M_2 are memories, Q_1, Q_2 are spurious states, $m_1, \ldots, m_9$ are maxima delimiting basins of attraction.


(a)
(b)

Apprentissage

- Stocker p patterns : $\xi^\mu = \{\xi_1^\mu \xi_2^\mu \dots \xi_N^\mu\}$

$\mu = 1, 2, \dots p$

- $\xi_i^\mu = \pm 1$ suivant que la cellule d'input a émis ou non un "spike" pour le pattern μ

Prescription de Hopfield.

$$\Delta W_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu \qquad \sim \text{Règle de Hebb}$$

Reconnaissance

Expérience numérique.

Présentation au réseau d'un stimulus $\{S_i^{in}\}$ tel que son recouvrement avec un pattern donné soit fini (non nul) alors que les autres recouvrements sont faibles

Ex: $m^1(t=0) = \frac{1}{N} \sum_i \xi_i^1 S_i^{in} \neq 0$

$m^\mu(t=0) \simeq 0 \quad \mu \neq 1$

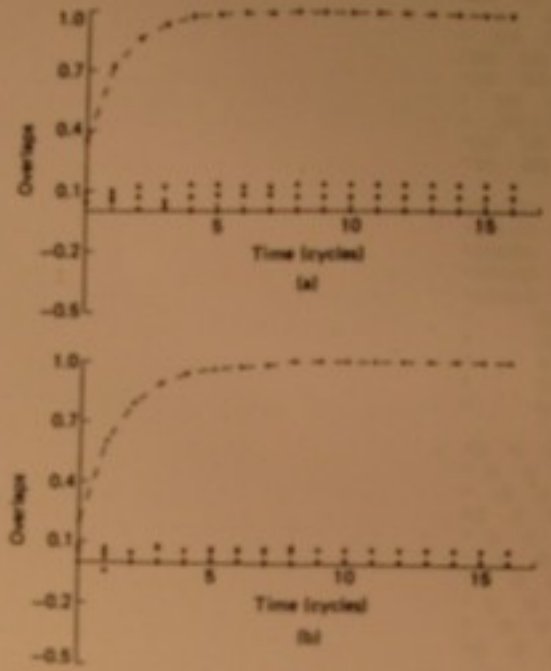


Figure 4.4: Time evolution of overlaps of a network state with the stored memories for two initial configurations. (a) m^1=0.35; (b) m^1=0.2. Simulation with N=400, p=7, and noiseless asynchronous dynamics.

1.5 Systèmes autoorganisant

Von der Malsburg (73) Grossberg (76) Fukushima (7
Kohonen (84)
Obermayer (92) : Applications au système visuel.

Classifieurs de grandes quantités de données.
Mécanisme de plasticité synaptique : compétitif
Amplifier les réponses des cellules ayant initialement une forte activité

Kohonen : Cellules avec interactions latérales de portée finie

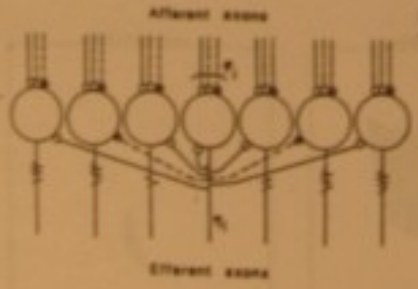


Fig. 5.1. Laterally interconnected neurons

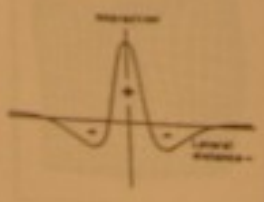


Fig. 5.2. The "Mexican-hat function" of lateral interaction

Conséquence : Amplification des réponses de certaines cellules dites "vainqueurs"

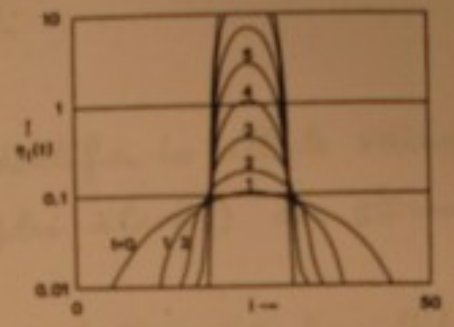
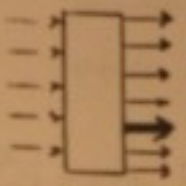


Fig. 5.4. Clustering of activity in a one-dimensional array



Apprentissage (non supervisé) pour l'autoorganisation

But : modifier les connections de façon dynamique afin de classifier les patterns (E_1 .... E_m)

→ Transformations somatosensorielles.

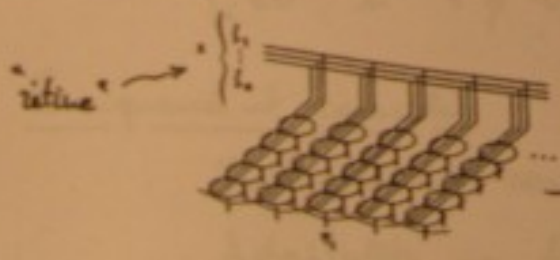
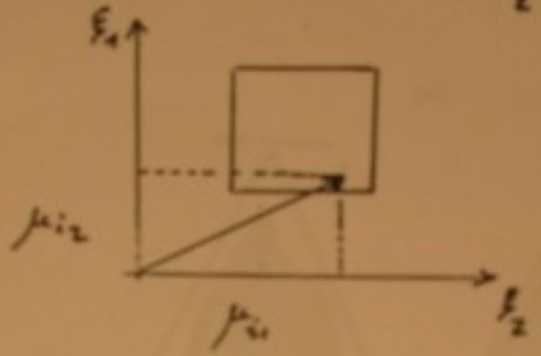


Fig. 5.18. Two-dimensional self-organizing system

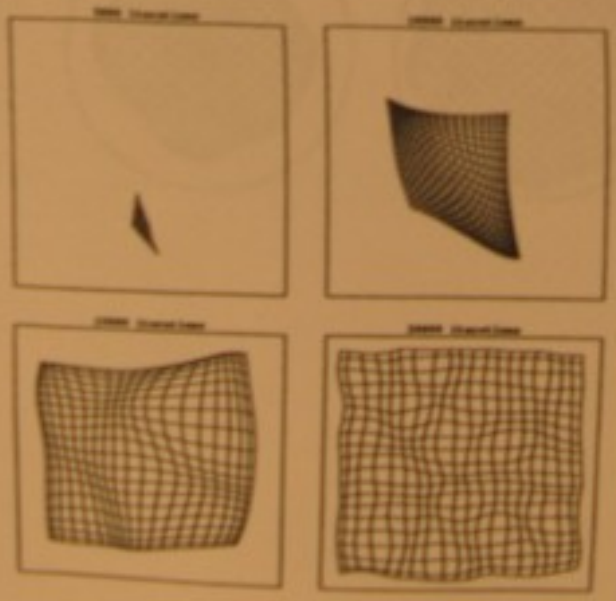
Câblage : Représentation des connexions dans le même espace que celui des patterns

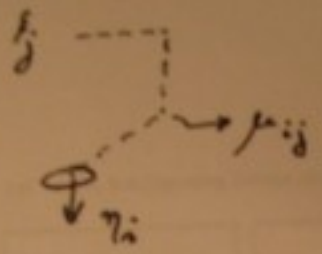
Exemple Rétine = Carré

$$\vec{\xi} = \begin{cases} \xi_1 \\ \xi_2 \end{cases}$$



On connecte 2 extremités de vecteurs $\vec{\mu_i}$ et $\vec{\mu_j}$ si i et j labellent 2 cellules voisines sur la couche corticale





Algorithme

1) Identifier la cellule vainqueur i^* (η_{i^*} le plus élevé) au stimulus présenté

2) Adaptation (de type Hebbien) dans un voisinage de la cellule vainqueur, itération après présentation de chaque pattern.

$$\frac{d\mu_{ij}}{dt} = \alpha\left(\eta_i \xi_j - \gamma(\eta_i)\mu_{ij}\right) \qquad 0 \leq \alpha \leq 1$$

η_i : valeur de sortie de la cellule i

ξ_j : valeur d'entrée sur cellule j

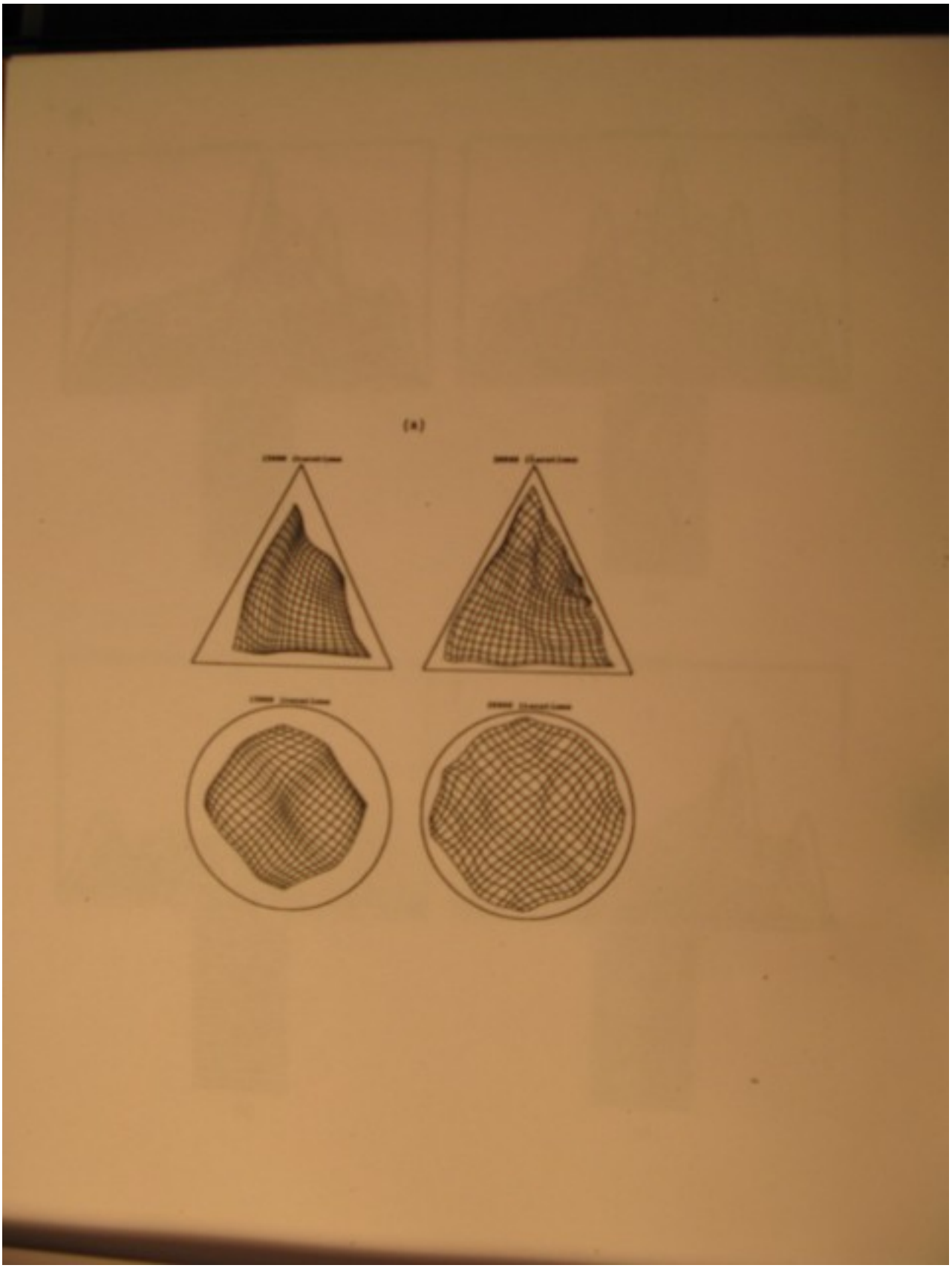
$\eta_i = 1$ si $i \in V_{i^*}$ $\qquad \gamma(0) = 0$

$\eta_i = 0$ si $i \notin V_{i^*}$ $\qquad \gamma(1) = 1$

Conséquences et exemples

Initialement connections arbitraires

Adaptation → Etablissement d'un ordre



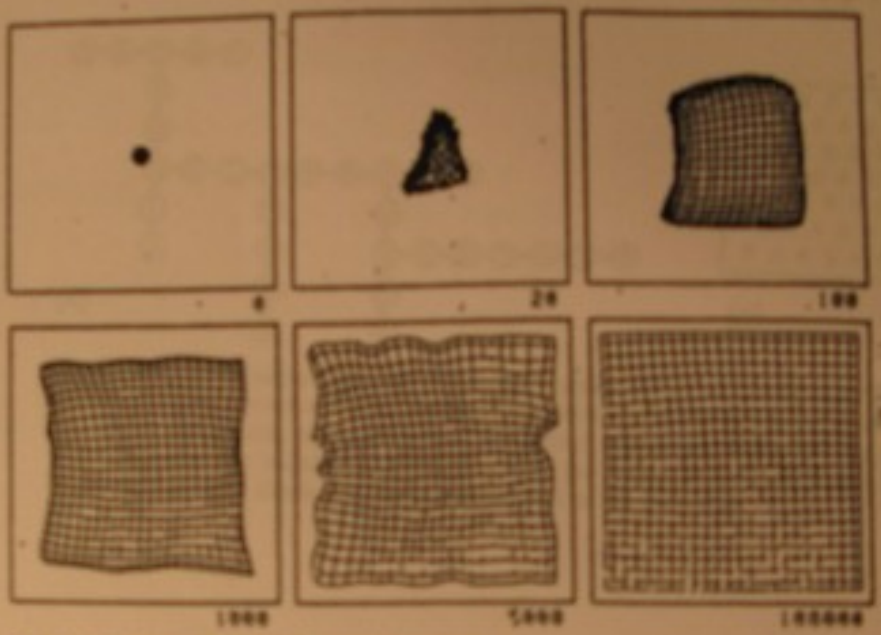


Fig. 3.36. Weight vectors during the ordering process

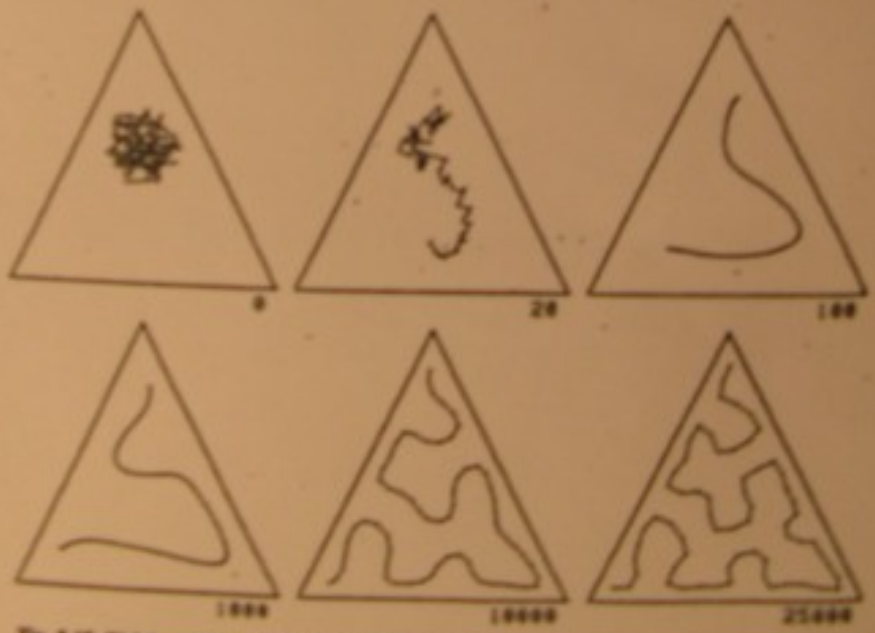


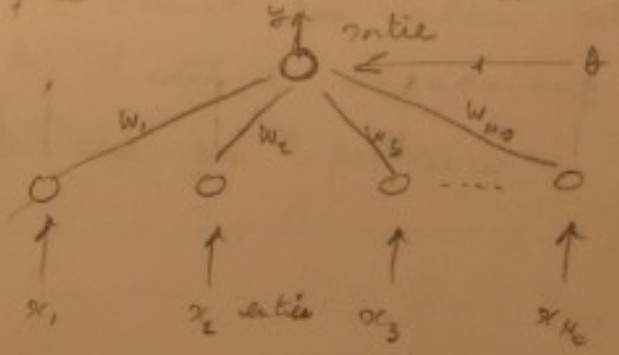
Fig. 3.37. Weight vectors during the ordering process

2. La règle de Widrow-Hoff et sa généralisation
(apprentissage supervisé)

2.1 Le réseau Adaline. (adaptative linéaire)

Le réseau Adaline est le système le plus simple incluant un algorithme supervisé apprenant. Le but ici est de minimiser une fonction d'erreur calculée on l'écart entre valeurs de sortie et valeur calculée par un réseau constitué d'une couche d'entrée et d'un neurone de sortie.

la fonction d'activation est linéaire



La valeur de sortie est donc

$$y = \sum_{j=1}^{N_0} w_j x_j + \theta \qquad \text{Soit } \theta = w_{N_0+1}$$

On peut écrire

$$y = w^T z \quad \text{où}$$

$$w^T = (w_1\ w_2 \cdots w_{N_0}\ \theta)^T = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N_0} \\ \theta \end{pmatrix}$$

et $z = (x_1, x_2 \cdots x_{N_0}\ 1)^T = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_0} \\ 1 \end{pmatrix}$

w_1

Il s'agit d'obtenir la meilleure valeur de W telle que elle minimise l'écart

$$\text{Erreur } E = \frac{1}{2} \sum_{m=1}^{M} \left(y^{(m)} - t^{(m)} \right)^2$$

entre la valeur désirée en sortie $t^{(m)}$ pour M patterns (ce sont des valeurs connues - N_y = 1 seul neurone en sortie) et la valeur $y^{(m)}$ calculée par le réseau.

Il s'agit de trouver la valeur des poids w_j telle que E admet une valeur minimale. Soit $w_j^{(m)}$ l'état des connexions après la présentation du $m^{ième}$ pattern.

L'erreur E est une fonction de $\{w_j^{(m)}\}_m$ par $\{y^{(m)}\}_m$ et la fonction (linéaire) d'activation

$$y^{(m)} = \sum_{j=1}^{N_0} w_j^{(m)} x_j^{(m)} + \theta^{(m)}$$

D'où la dérivée partielle

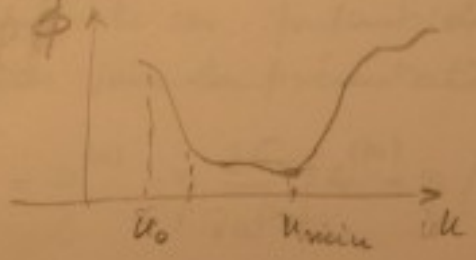
$$\frac{\partial E}{\partial w_j^{(m)}} = \frac{\partial E}{\partial y^{(m)}} \frac{\partial y^{(m)}}{\partial w_j^{(m)}}$$

$= \left(y^{(k)} - t^{(k)}\right) x_j^{(k)}$

Sous forme vectorielle

$$\frac{\partial E}{\partial w^{(k)}} = \left(y^{(k)} - t^{(k)}\right) z^{(k)} \qquad z = (x_1, x_2 \dots x_{N_0}, 1)$$

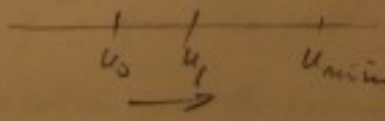
A présent, comment modifier $w^{(k)}$ pour x diriger vers le minimum. Prenons l'exemple de la recherche du minimum d'une fonction ϕ de u - Nous aimerons à trouver $E \equiv E(w^{(k)})$ pour chaque u



On doit trouver u_{min}, de manière **itérative**. On part de u_0 et on fait

$$u_1 = u_0 - \eta \left(\frac{d\phi}{du}\right)_{u_0} \qquad \eta > 0$$

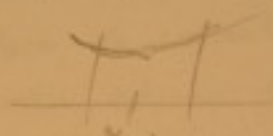
Comme $\left(\frac{d\phi}{du}\right)_{u_0} < 0$ alors $u_1 > u_0$



Arrivé en u_{min}, l'itération donne $u_k = u_{k-1} - \eta \left(\frac{d\phi}{du}\right)_k$ si $\left(\frac{d\phi}{du}\right)_k = 0$ on obtient alors $u_{k+1} = u_k$ et le processus s'arrête

w3

6- Sinon, on part de l'autre côté de u_{min}



Alors $u_{k+2} = u_{k+1} - \eta \left(\frac{\partial \phi}{\partial u}\right)_{k+1}$ avec $\left(\frac{\partial \phi}{\partial u}\right)_{k+1} < 0$

et $u_{k+2} < u_{k+1}$

on s'approche vers u_{min}.

C'est le principe de la méthode.

Dans notre cas, partant de $w^{(m)}$ on modifie les poids pour la présentation du pattern suivant par -

$$w^{(m+1)} = w^{(m)} - \eta \frac{\partial E}{\partial w^{(m)}} = w^{(m)} - \eta \left(y^{(m)} - t^{(m)}\right) z^{(m)}$$

L'algorithme est donc le suivant :

1) - on initialise les poids à des valeurs aléatoires petites

2) - on présente un pattern $\left(x_1^m \dots x_{N_0}^m \cdot 1\right) = z^{(m)}$ et on désigne une sortie désirée $t^{(m)}$ pour ce pattern -

3) - on calcule la sortie $y^{(m)}$ pour ce pattern

4) - la nouvelle valeur des poids pour la présentation du pattern suivant est

$$w_{ij}^{(k+1)} = w_{ij}^{(m)} - \eta \left(y^{(m)} - t^{(m)}\right) z^{(m)}$$

où η est un paramètre $0 < \eta < 1$

5) on calcule l'erreur E. si l'erreur E est trop grande - on retourne à 2) si

2.2 La rétropropagation du gradient

En fait, le caractère linéaire de la fonction d'activation est une limitation très importante. Le réseau Adaline ne pourra pas fonctionner si les patterns sont bruités par exemple, comme dans le cas du réseau de Hopfield. Dans le passé au réseau linéaire

D'autre part, le passage à des structure multi-couches va augmenter les potentialités.

la rétro-propagation du gradient

1 Architecture du réseau

Dans un réseau en couches, toute cellule d'entrée est suivie d'une succession de couches dites "cachées", pour finalement aboutir sur une couche de sortie.

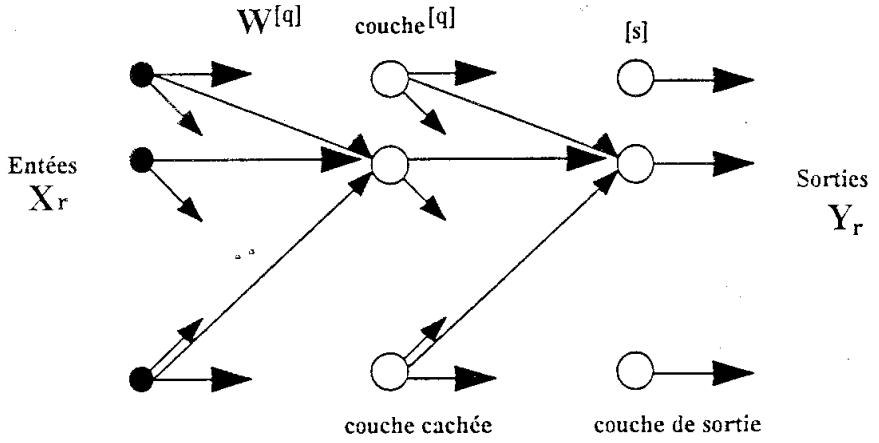


Fig . 1 -Architecture générale d'un réseau multi-couches

I présence de neurones des couches intermédiaires apporte une richesse a la structure qui doit accroître les capacités du réseau. Il faut insister sur le fait que les cellules internes n'ont aucune connexion prédéfinie. elles ne servent qu'à contribuer à l'obtention du résultat souhaité en sortie. Une autre particularité réside dans la fonction de transfert des unités. Ce sont des éléments semi-linéaires, qui compressent le potentiel du neurone dans l'intervalle [-1,+1]. Les fonctions qui réalisent cette tâche doivent être continues et dérivables .
Pour l'unité *i* de la couche *[q]*, en considérant que les $x_j^{[q]}$ sont les entrées de cette couche, le potentiel $p_j^{[q]}$ vaudra:

$$p_j^{[q]} = \sum_j W_{ij} x_j^{[q]}$$

et la sortie $y_j^{[q]}$ sera déterminée, à partir du potentiel, comme:

$y_j^{[q]} = \sigma (p_j^{[q]})$

la fonction de transfert σ peut être, par exemple, égale à:

$$\sigma(v) = th(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}}$$

où th dénote la fonction tangente hyperbolique dont la dérivé s'exprime par:

$th(v)' = 1 - th^2(v)$

Sur cette architecture, nous allons voir comment modifier les poids entre unités, c'est à dire comment faire l'apprentissage. L'architecture que nous venons de décrire comporte plusieurs paramètres, tels que la forme de la fonction de transfert, le nombre de couches, le nombre d'unités par couche. Actuellement, il n'existe pas de méthode analytique pour déterminer ces paramètres de structure. Il convient donc d'acquérir un savoir faire pour être capable d'exploiter au mieux ce modèle. C'est d'ailleurs ce qui le rend physiologiquement peu plausible et difficilement exploitable pour de gros problèmes.

2 Algorithme de retro-propagation du gradient

Comme nous venons de le voir , le réseau est structuré en couches successives, traversées par le signal d'entrée qui se propage vers la couche de sortie. Il n'est pas difficile de trouver un terme d'erreur à minimiser. Il suffira, en considérant qu'un superviseur est capable d'engendrer un objectif à atteindre, de calculer l'écart entre cet objectif et la réponse fournie par le réseau fig. 2 .

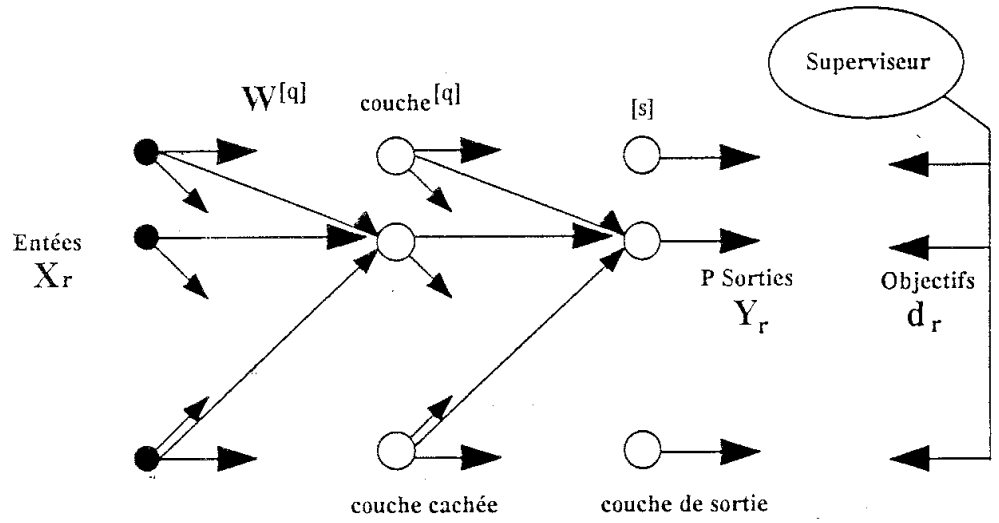


Fig . 2 -Calcul du terme d'erreur

L'algorithme d'ajustement des poids considère le terme d'erreur comme une grandeur à minimiser afin d'obtenir une configuration qui copie au mieux la fonction de transfert définie par les présentations successives des couples (x_r,d_r). Or si nous voulons utiliser une technique de minimisation à base de gradient stochastique, seule la couche d'interconnexions reliée aux neurones de sortie a la connaissance de la grandeur à minimiser. Les couches internes, reliées aux neurones cachés, n'ont pas de terme à minimiser. Il n'y a ainsi aucune façon de connaître directement la variation des poids à appliquer. Le principe de l'algorithme de rétro-propagation du gradient réside dans la possibilité de donner un objectif aux neurones des couches internes et, par conséquent, une mesure de la variation des poids à appliquer pour le satisfaire. Un neurone placé dans une couche interne peut avoir l'erreur calculée en sortie du réseau à travers les poids qui le relient à la sortie. L'algorithme pourra ainsi moduler les poids qui arrivent sur un neurone afin de réduire sa participation à l'erreur calculée en sortie du réseau.

3 Définition du terme d'erreur

Nous définissons un terme d'erreur qu'il faut minimiser afin de trouver un ensemble de poids suceptibles de satisfaire le problème posé.
Pour un prototype r (r = 1... R) , avec K_S le nombre d'éléments de la couche de sortie le terme d'erreur instantanée, qui est un scalaire, s'écrit:

$$\xi_r = \frac{1}{2}\sum_{k=1}^{Ks} (d_{kr} - y_{kr})^2$$

Le terme d'erreur doit être défini sur l'ensemble des erreurs instantanées mesurées à chaque présentation d'un couple (sortie désirée d_r, sortie obtenue y_r).

$$\xi = \sum_{r=1}^{R} \xi_r$$

Mais, dans la suite de la démonstration, nous ne considérerons que le terme d'erreur instantanée, sans nuire en cela à la généralité du calcul.

4 Détermination du gradient de l'erreur par rapport aux poids

Comme nous l'avons constaté, la fonction d'un réseau est basée sur trois paramètres: les poids, le potentiel du neurone et sa fonction de transfert non linéaire. Nous décomposerons les calcul de façon à faire apparaître toutes les opérations élémentaires ce qui nous permettra de simplifier la présentation de la démonstration. En premier lieu, nous examinons la variation de l'erreur par rapport au poids $W_{ij}[q]$ comme la composition de la variation de l'erreur par rapport à une variation du potentiel du neurone i de la couche [q] et de la variation de ce potentiel en fonction des poids incidents $W_{ij}[q]$. Le potentiel p_i est choisi car il dépend de $W_{ij}[q]$.

$$\frac{\partial \xi_r}{\partial W_{ij}^{[q]}} = \frac{\partial \xi_r}{\partial p_{ir}^{[q]}} \frac{\partial p_{ir}^{[q]}}{\partial W_{ij}^{[q]}}$$

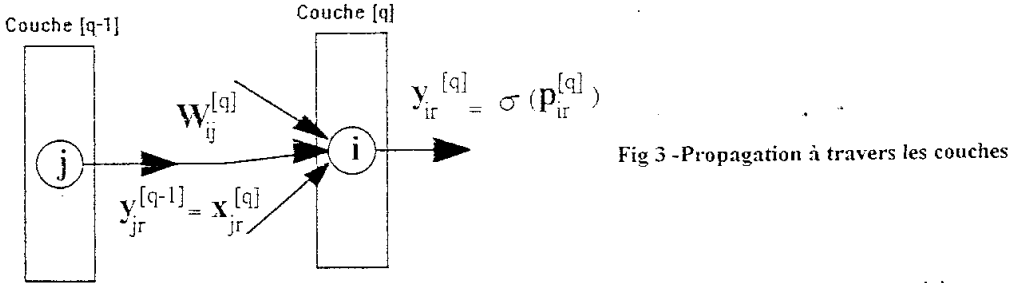


Fig 3 -Propagation à travers les couches

Nous appelons signal d'erreur instantanée le premier terme de cette décomposition, et nous le notons de la façons suivante:

$$\delta_{ir}^{[q]} = \frac{-\partial \xi_r}{\partial p_{ir}^{[q]}}$$

Ce terme sera développé plus loin, en fonction de la couche [q], et nous permettra de faire apparaître une forme récurrente pour son évaluation sur une couche interne à partir de la

sortie désirée. Considérons maintenant un neurone i placé sur une couche interne [q]. Son potentiel est calculé à partir des sorties de la couche précédente pondérées par les poids incidents. La variation de ce potentiel par rapport aux poids s'écrit:

si Kq est le nombre de neurones sur la couche q

$$\frac{\partial p_{ir}^{[q]}}{\partial W_{ij}^{[q]}} = \frac{\partial}{\partial W_{ij}} \left(\sum_{k=1}^{K_q} W_{ik}^{[q]} x_{kr}^{[q]} \right)$$

Le calcul de ce terme est immédiat et montre naturellement que le potentiel ne dépend que de l'entrée des neurones.

$$\frac{\partial p_{ir}^{[q]}}{\partial W_{ij}^{[q]}} = x_{jr}^{[q]}$$

En regroupant les deux membres de l'équation, nous pouvons dégager une forme simple de l'expression du gradient de l'erreur par rapport aux poids:

$$\boxed{\frac{\partial \xi_r}{\partial W_{ij}^{[q]}} = -\delta_{ir}^{[q]} x_{jr}^{[q]}}$$

5 Expression du signal d'erreur

Il faut maintenant expliciter le terme d'erreur δ_{ir} qui dépendra de la couche sur laquelle il est calculé. Nous décomposons le signal d'erreur comme la variation de l'erreur en fonction de la sortie multipliée par la variation de la sortie par rapport au potentiel:

$$\frac{\partial \xi_r}{\partial p_{ir}^{[q]}} = \frac{\partial \xi_r}{\partial y_{ir}^{[q]}} \frac{\partial y_{ir}^{[q]}}{\partial p_{ir}^{[q]}}$$

Ici encore, nous calculerons facilement le second membre de l'expression:

$$\frac{\partial y_{ir}^{[q]}}{\partial p_{ir}^{[q]}} = \sigma'(p_{ir}^{[q]})$$

c'est à dire que la variation de sortie n'est que la dérivée de la fonction de transfert du neurone appliquée à son potentiel. Il reste enfin à déterminer l'expression de la variation de l'erreur en fonction de celles de la sortie des neurones. Pour cela, nous allons considérer deux cas selon que nous nous trouvons sur la couche de sortie ou sur une couche cachée.

6.1. Couche de sortie: q = s

Pour la couche de sortie (s), qui contient Ks neurones, nous connaissons directement le terme d'erreur, comme il a été défini précédemment. L'expression du gradient de l'erreur par rapport aux sorties des neurones vaut donc:

$$\frac{\partial \xi_r}{\partial y_{ir}^{[s]}} = \frac{\partial}{\partial y_{ir}^{[s]}} \frac{1}{2} \sum_{k=1}^{K_s} (d_{kr} - y_{kr}^{[s]})^2$$

$$\frac{\partial \xi_r}{\partial y_{ir}^{[s]}} = -(d_{ir} - y_{ir}^{[s]})$$

Le signal d'erreur δ_{ir}, pour la couche de sortie vaut donc:

$$\boxed{\delta_{ir}^{[s]} = +(d_{ir} - y_{ir}^{[s]})\sigma'(p_{ir}^{[s]})}$$

6.2. Couches cachées: q < s

Pour les couches cachées, l'expression du terme d'erreur n'es pas directement connue: c'est la difficulté principale de cet algorithme. Nous allons considérer que le terme d'erreur est vu *au travers des interconnexions qui relient la couche interne à la couche de sortie*. Nous évaluerons l'erreur de la façon suivante:

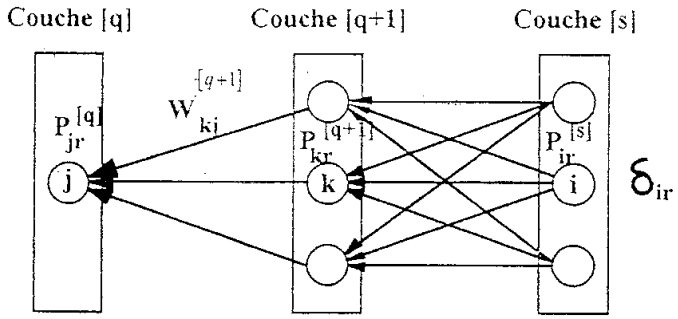


Fig Rétro-propagation du terme d'erreur

$$\frac{\partial \xi_r}{\partial y_{ir}^{[q]}} = \sum_{k=1}^{K^{[q+1]}} \frac{\partial \xi_r}{\partial p_{kr}^{[q+1]}} \frac{\partial p_{kr}^{[q+1]}}{\partial y_{ir}^{[q]}}$$

La variation de l'erreur en sortie du réseau par rapport à la sortie d'une couche interne est décomposée comme la somme de toutes les variations de l'erreur par rapport aux potentiels de la couche suivante multipliée par la variation de ces potentiels par rapport à la sortie du neurone considéré. Le second terme vaut, en écrivant $p_{kr}^{(q+1)}$ en fonction de la sortie de la couche précédente:

$$\frac{\partial \xi_r}{\partial y_{ir}^{[q]}} = \sum_{k=1}^{K_{[q+1]}} \frac{\partial \xi_r}{\partial p_{kr}^{[q+1]}} \frac{\partial}{\partial y_{ir}^{[q]}} \sum_{i=1}^{K_{[q]}} w_{ki}^{[q+1]} y_{ir}^{[q]}$$

donc:

$$\frac{\partial \xi_r}{\partial y_{ir}^{[q]}} = \sum_{k=1}^{K_{[q+1]}} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]}$$

En reprenant la définition du signal d'erreur:

$$\frac{\partial \xi_r}{\partial p_{ir}^{[q]}} = \frac{\partial \xi_r}{\partial y_{ir}^{[q]}} \frac{\partial y_{ir}^{[q]}}{\partial p_{ir}^{[q]}}$$

nous pouvons extraire une formulation récurrente de δ_i. Le signal d'erreur pour une couche interne vaut:

$$\boxed{\delta_{ir}^{[q]} = \sum_{k=1}^{K_{q+1}} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]} . \sigma'(p_{ir}^{[q]})}$$

Cette expression permet de répercuter le signal d'erreur depuis la couche externe, où il est directement calculé à partir des sorties désirées, vers les couches internes où il est estimé.

7. Expression de la variation des poids.

Enfin, en reprenant la règle générale de modification des poids, on modifie les $W_{ij}^{(q)}$ suivant:

$$\boxed{\Delta W_{ij}^{[q]} = \alpha(-\frac{\partial \xi_r}{\partial W_{ij}^{[q]}}) = \alpha(\delta_i^{[q]} x_{jr}^{[q]})}$$

nous trouvons les formules suivantes:

1- Pour les poids entre l'avant dernière couche et la couche de sortie:

$$\boxed{\Delta W_{ij}^{[s]} = \alpha(d_{ir} - y_{ir}) x_{jr}^{[s]} \sigma'(p_{ir}^{[s]})}$$

2- Pour tous les poids entre la couche (q) et la couche(q+1)

$$\boxed{\Delta W_{ij} = \alpha(\sum_{k=1}^{K_{[q+1]}} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]}) x_{jr}^{[q]} \sigma'(p_{ir}^{[q]})}$$

Cette expression signifie que l'erreur en sortie du réseau est propagée en sens rétrograde avec les mêmes poids de connexion que ceux employés en sens direct.

La procédure de modification des poids se résume donc de la manière suivante:

-(1) Initialisation des poids W(q) entre les neurones des couches consécutives q=1,...,s à des petites valeurs aléatoires

-(2) Présentation d'une entrée x_r et de la sortie désirée d_r, r est le numéro de la présentation d'un couple (entrée, sortie désirée), r = 1,..., R

-(3) Calcul de la sortie actuelle par propagation directe à travers toutes les couches:

$$y_{jr}^{[q]} = \sigma\left(\sum_{i=1}^{Kq} W_{ji}^{[q]} y_{ir}^{[q-1]}\right) = \sigma\left(\sum_{i=1}^{Kq} W_{ji}^{[q]} x_{ir}^{[q]}\right)$$

où σ est la fonction neurone.
Pour la couche d'entrée q = 1, $y^{[q-1]}=x$

-(4) Calcul de l'erreur en sortie (pour le tracé de l'évolution du terme d'erreur, mais n'est pas nécessaire au calcul d'adaptation).

$$\xi_r = \frac{1}{2}\sum_{K=1}^{Ks} (d_{kr} - y_{kr}^{[s]})^2$$

où
d_r est la sortie désirée associée au vecteur x_r
$y_r[s]$ est la sortie obtenue sur la dernière couche lors de la présentation du vecteur d'entrée x_r
ξ_r est l'erreur calculée pour le couple (x_r, d_r)

-(5) Rétro-propagation du signal d'erreur (δ_{ir}) depuis la dernière couche vers la première couche :
-pour chaque cellule de sortie:

$$\delta_{ir}^{[s]} = \sigma'(p_{ir}^{[s]})(d_{ir} - y_{ir}^{[s]})$$

- pour les cellules cachées:

$$\delta_{ir}^{[q]} = \sum_{k=1}^{K_{q+1}} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]} \sigma'(p_{ir}^{[q]})$$

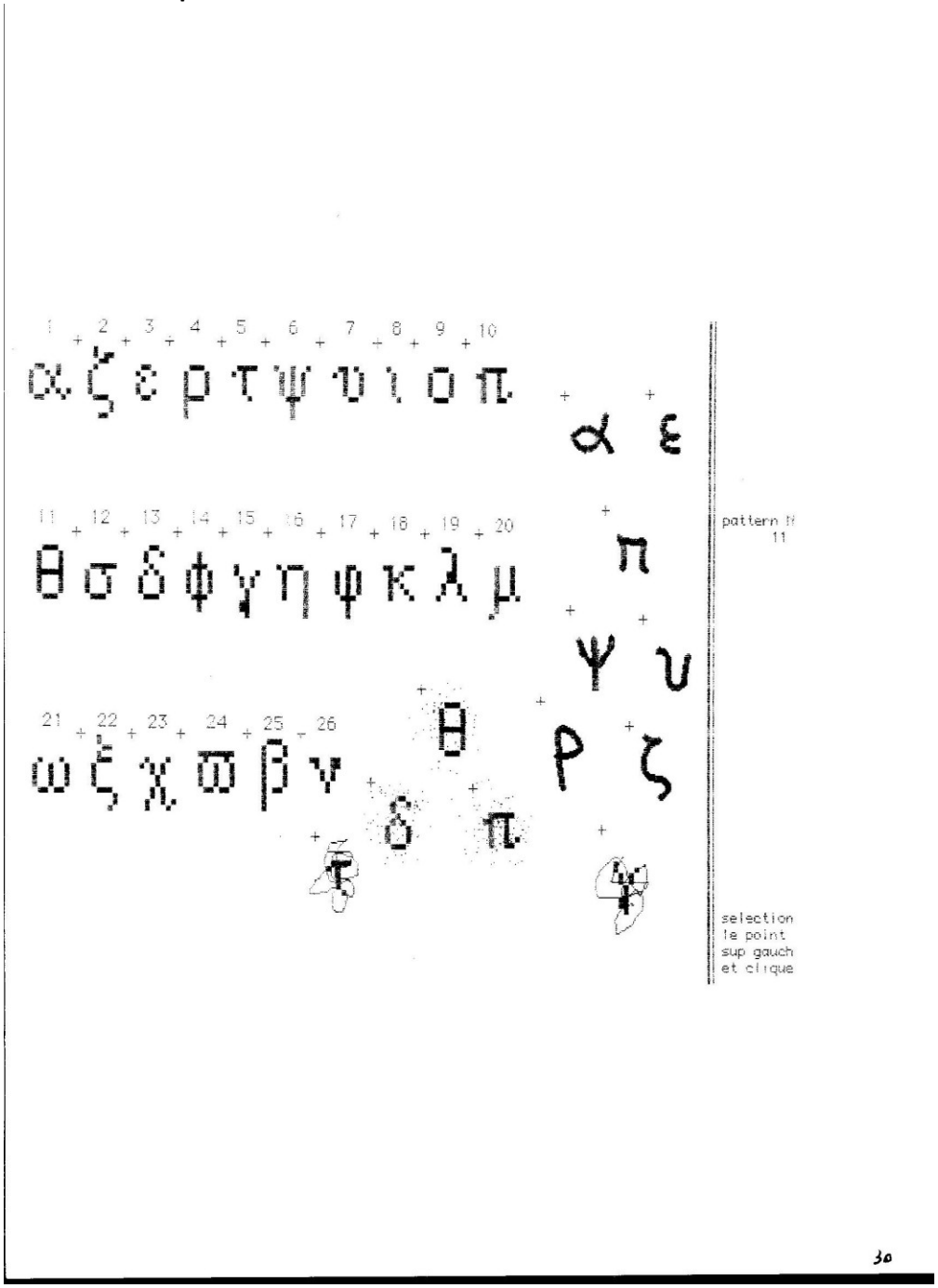
-(6) Mise à jour des poids selon la règle:

$$\Delta W_{ij}^{[q]} = \alpha(\delta_{ir}^{[q]} x_{jr}^{[q]})$$

où α est un paramètre d'apprentissage compris entre 0 et ÷1.

-(7) retour à (2) tant qu'il y a des couples à présenter

<u>Applications</u> : Identification de patterns



Prédiction de séries temporelles