Notes de cours

LES RESEAUX DE NEURONES ET APPRENTISSAGE AUTOMATIQUE

Volume horaire hebdomadaire : 3h cours et 1h TD

Enseignant : Dr AMROUN Kamal

Programme

Chapitre 1 : Introduction
Chapitre 2 : Les réseaux de neurones
Chapitre 3 : Les modèles mathématiques
Chapitre 4 : Apprentissage
Chapitre 5. Mémoires associatives
Chapitre 6 : Carte auto-organisatrice
Chapitre 7 : Un réseau à architecture évolutive, ART
Chapitre 8 : Apprentissage par pénalité / récompense (renforcement)
Chapitre 9 : Réseaux multicouches
Chapitre 10 : Connexionnisme et applications

Bibliographie

1. « Réseaux de neurones, Méthodologies et applications », Gérard Dreyfus, Manuel Samuelides, Jean-Marc Martinez, Mirta B. Gordon, Fouad Badran, Sylvie Thiria, Laurent Hérault, 2e édition Eyrolles (29 avril 2004)

2. Neural networks, S. Haykain, prentice hall, 1998

3. Les réseaux de neurones artificiels, introduction au connexionnisme, cours, exercices et travaux pratiques, Claude Touzet, Juillet 1992 (Disponible sur le net : http://www.touzet.org/Claude/Web-Fac-Claude/Les_reseaux_de_neurones_artificiels.pdf)

Chapitre 1 : Introduction

Il existe trois approches fondamentales pour résoudre un problème :

- L'approche algorithmique classique
- L'approche basée sur les connaissances
- Les réseaux de neurones artificiels

Dans l'approche algorithmique, nous avons besoin d'écrire un algorithme décrivant toutes les étapes à suivre pour résoudre le problème considéré. Lorsque tous les cas sont connus et prévus par l'algorithmicien, cette approche est efficace. Malheureusement, ceci n'est pas toujours possible dans la réalité car les problèmes sont souvent complexes et difficiles. Notons qu'un cas non prévu dans cette approche peut conduire à une catastrophe.

La seconde approche est celle basée sur l'Intelligence artificielle (IA), dont la méthode la plus connue est les systèmes experts. Mais là aussi toutes les règles être bien exprimées par l'expert du domaine traité (mécanique, physique, etc.), les cas non prévue par l'expert ne sont pas traités ici aussi. Cette approche se limite aux domaines où les connaissances peuvent être exprimées sous forme de règles. Il est très difficile d'exprimer les connaissances empiriques (Médecine, psychologie, etc.).

Pour remédier aux inconvénients des deux approches précédentes, les chercheurs ont proposé une troisième approche qui consiste à s'inspirer du traitement de l'information effectué par le cerveau humain. C'est l'approche basée sur les réseaux de neurones artificiels.

Cette troisième approche fait l'objet de ce cours. Nous commençons par présenter quelques notions biologiques relatives au cerveau humain, et à ses constituants. L'organisation en réseaux des neurones permet d'illustrer les notions d'apprentissage et de mémorisation (modification des connexions). Le deuxième chapitre montre le passage des modèles de réseaux de neurones biologiques à des modèles mathématiques : les réseaux de neurones artificiels. Nous établissons un tableau des correspondances entre un neurone et un neurone artificiel. Il existe de nombreux modèles de réseaux de neurones artificiels, nous en présentons quelques uns de ces modèles (le Perceptron, les cartes auto-organisatrices). Nous allons ensuite

introduire la notion d'apprentissage avec ses deux variantes supervisée et non supervisée. Puis nous allons étudier les mémoires associatives, le réseau ART et le perceptron multicouche. A la fin nous allons présenter le développement d'applications.

Chapitre 2 : Les réseaux de neurones

Dans ce chapitre, nous allons présenter brièvement cette notion de réseaux de neurones artificiels.

1 Définition

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

2 Historique

- 1943 : J. Mc Culloch et W. Pitts ont défini la notion de neurone formel. Ils ont montré que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes.

- 1949 : D. Hebb a expliqué le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à

- 1957 : F. Rosenblatt a développé le modèle du Perceptron. Il a construit le premier neuroordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes.

- 1960 : B. Widrow a développé le modèle Adaline (Adaptative Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente. Celle-ci est à l'origine de l'algorithme de rétropropagation de gradient très utilisé aujourd'hui avec les Perceptrons multicouches.

- 1969 : M. Minsky et S. Papert ont publié un ouvrage qui montre l'impossibilité pour le perceptron de traiter des problèmes non linéaires..

- 1967-1982 : C'est une phase déclin. Cependant, les recherches se poursuivaient sous d'autres noms : la reconnaissance de formes, la modélisation en neurobiologie, etc.

- 1983 : La Machine de Boltzmann est le premier modèle connu apte à traiter de manière satisfaisante les limitations recensées dans le cas du perceptron. Mais l'utilisation pratique s'avère difficile, la convergence de l'algorithme étant extrêmement longue (les temps de calcul sont considérables).

- 1985 : La rétro propagation de gradient apparaît. C'est un algorithme d'apprentissage adapté aux réseaux de neurones multicouches (aussi appelés Perceptrons multicouches).

Actuellement, les réseaux de neurones sont très largement utilisés surtout avec l'apparition de l'apprentissage profond, très utilisé dans le traitement d'images, pour la classification, pour l'analyse des données, etc.

3 Applications

Les réseaux de neurones peuvent s'appliquer dans plusieurs domaines, nous pouvons citer :

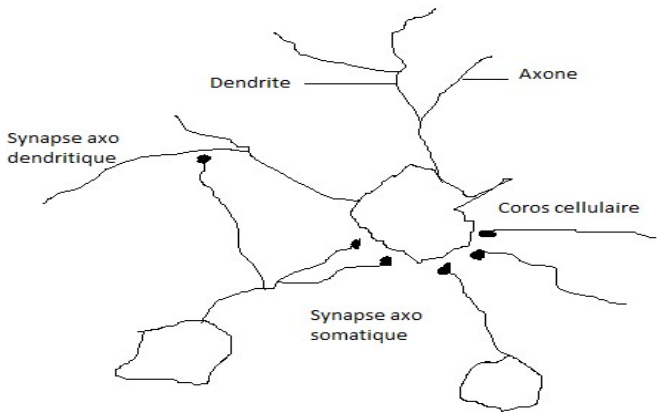
• statistiques : analyse de données / prévision / classification, etc.
• robotique : contrôle et guidage de robots ou de véhicules autonomes
• imagerie / reconnaissance de formes
• traitement du signal
• simulation de l'apprentissage

Chapitre 3 : Les modèles mathématiques

Le cerveau contient approximativement 10^{12} neurones ; chaque neurone a un nombre de connexions allant de 1000 à 10000.

Le neurone

Le neurone est une cellule cellulaire composée d'un noyau et d'un corps cellulaire (dendrites). C'est à travers les dendrites que l'information est acheminée de l'extérieur vers le corps du neurone. L'information traitée est alors transmise vers les autres neurones via les axones. Notons que l'information la transmission de l'information n'est pas directe entre deux neurones, mais traverse par un espace intercellulaire entre l'axone et les dendrites. Cette jonction est appelé la synapse. Selon le type du neurone, la longueur de l'axone peut varier de quelques microns à 1,5 mètres.

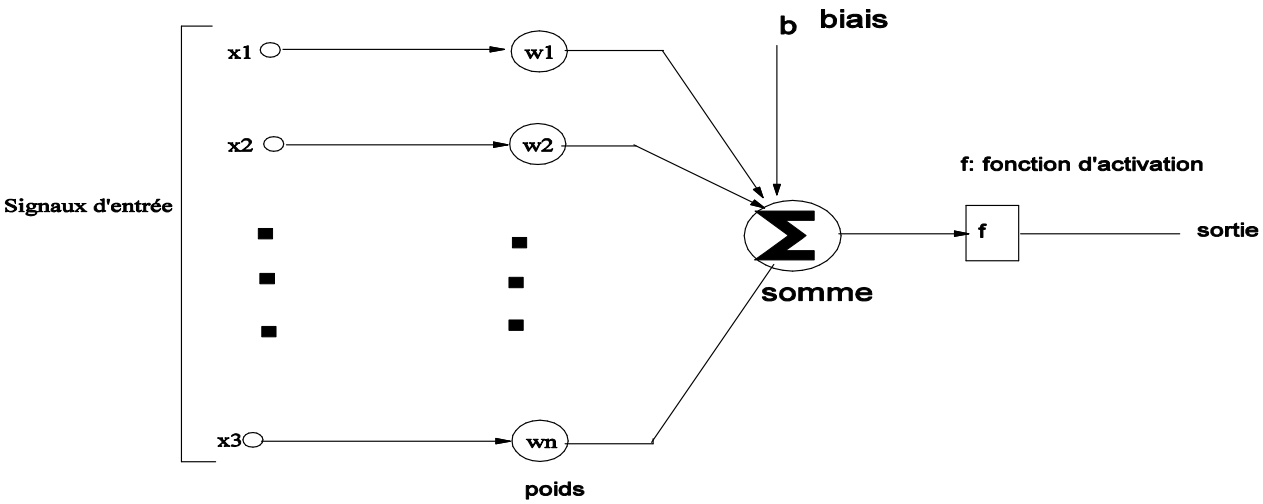


Réseau de neurones

Un réseau de neurones est un ensemble de neurones fortement connectés entre eux. Les neurones sont comme des processeurs élémentaires fonctionnant en parallèle. Une structure hiérarchique de réseaux de neurones est aussi un réseau de neurones.

Modèles de neurones

La figure suivante montre un modèle de neurone qui sert de base pour le développement de plusieurs familles de réseaux de neurones.



Modèle de neurone.

Nous distinguons les entrées qui sont x1, x2, … xn multipliées par des poids w1n, w2, …, wn. Puis une fonction somme sur ces entrées pondérées et finalement une fonction d'activation. Nous remarquons aussi une entrée particulière appelée le biais.

Mathématiquement, on aura :

La sortie Y est donnée par la formule suivante :

$$Y = \sum x_i w_i + b$$

Fonctions d'activation

Il existe plusieurs fonctions d'activation. Nous pouvons citer les types suivants ;

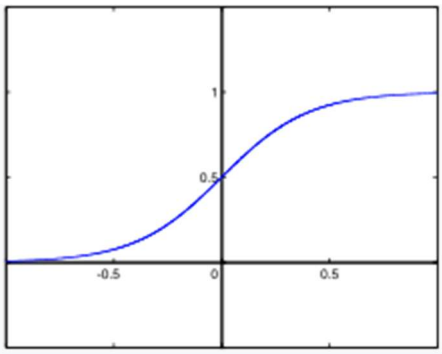
1- Fonction sigmoïde
2- Fonction linéaire
3- Fonction seuil
4- Etc.

Fonction sigmoïde

La fonction sigmoïde est très utilisée dans les réseaux de neurones, strictement croissante. Un exemple de fonction sigmoïde est définie par l'équation suivante :

$\frac{1}{1+e^{-x}}$ ou plus généralement $\frac{1}{1+e^{-\lambda x}}$ avec λ paramètre de la fonction.

La courbe sigmoïde possède pour asymptotes les droites d'équation y = 0 et y = 1. Elle a pour centre de symétrie le point I de coordonnée (0;1/2). Sa courbe est donnée par la figure suivante :



La fonction sigmoïde avec $\lambda = 5$. (Courbe reprise de wikipedia)

Fonction seuil

Elle est définie par la fonction suivante

$$f(x) = \begin{cases} 1 & si \quad x \geq 0 \\ 0 & si \quad x \prec 0 \end{cases}$$

Topologies des réseaux de neurones

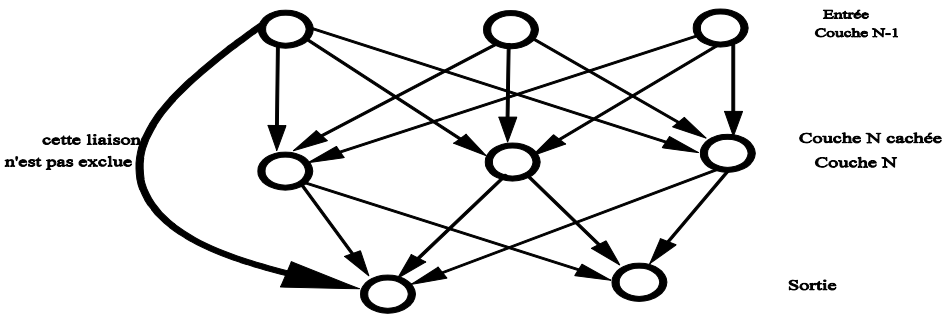
Les réseaux de neurones sont définis par leurs topologies

Feed forward

Il existe principalement deux types de cette topologie

1- Réseau multicouche

Dans cette topologie, un neurone de la couche (N-1) est connecté à tous les neurones de la couche N (Voir la figure suivante par exemple)



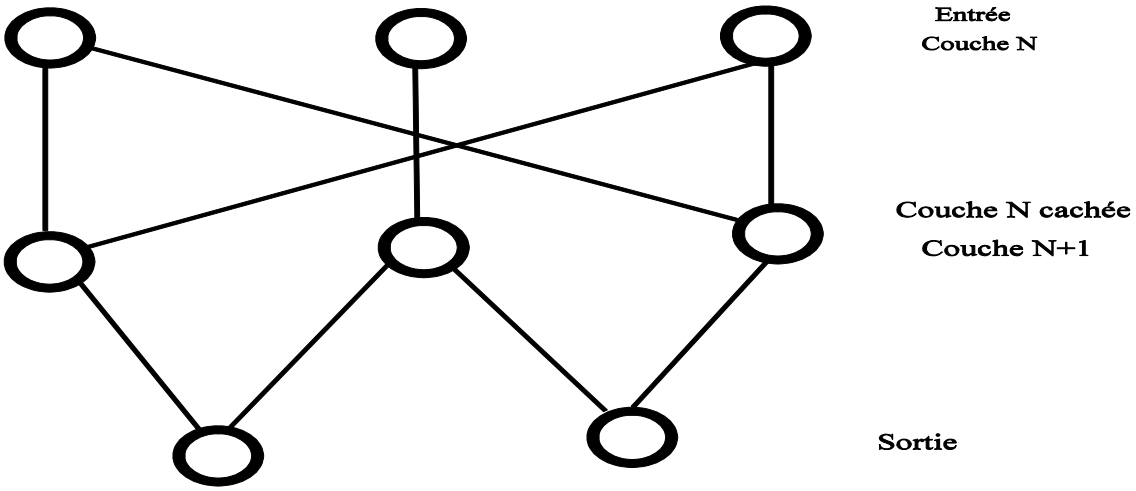
Réseau de neurones multicouche

Le sens de l'activation est bien défini.

Remarque : dans un réseau de neurones multicouche, il n'y a pas de connexion entre les neurones d'une même couche.

2 Réseau à connexions locales

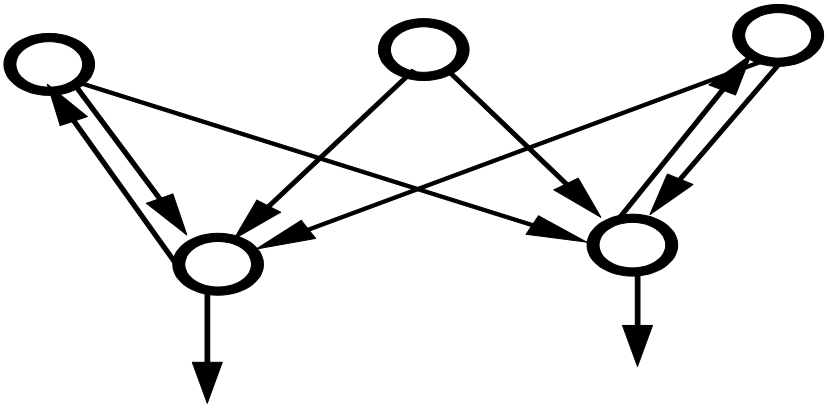
Dans cette topologie, un neurone de la couche N n'est pas forcément connecté à tous les neurones de la couche N+1 (Voir la figure suivante par exemple)



Réseau de neurones à connexions locales

Réseaux récurrents (bouclés)

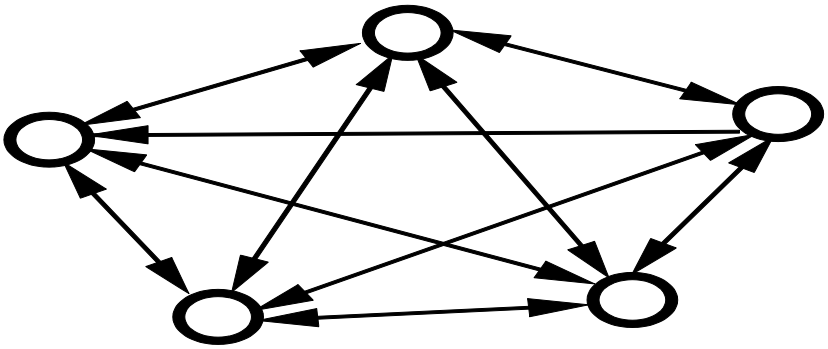
Dans cette topologie, l'information d'activation est ramenée en arrière (couche N vers couche N-1. Les connexions sont le plus souvent locales.



Réseau récurrent

Réseaux à connexions complètes

Dans cette topologie, chaque neurone est connecté à tous les neurones du réseau (y compris lui-même). C'est un graphe complet.



Réseau complet

Chapitre 4 : **Apprentissage**

Déterminer un réseau de neurones consiste à trouver les coefficients synaptiques. La phase d'apprentissage est la phase où les caractéristiques du réseau sont modifiées jusqu'à ce que le comportement désiré soit obtenu. La base d'apprentissage contient des exemples représentatifs du comportement ou de la fonction à modéliser. Ces exemples sont sous la forme de couples (entrée ; sortie) connus. La base d'essai consiste à calculer pour une entrée quelconque (bruitée ou incomplète) la sortie correspondante. On peut alors évaluer la performance du réseau

Il existe 3 types d'apprentissage :

-apprentissage supervisée

- apprentissage non supervisé

- apprentissage par renforcement

• **apprentissage supervisé** : les coefficients synaptiques sont évalués en minimisant l'erreur (entre sortie souhaitée et sortie obtenue) sur une base d'apprentissage.

• **apprentissage non-supervisé** : on ne dispose pas de base d'apprentissage. Les coefficients synaptiques sont déterminés par rapport à des critères de conformité : spécifications générales.

• **apprentissage par renforcement** : cet algorithme d'apprentissage par renforcement est de type qualitatif par opposition aux apprentissages supervisé ou non supervisé. Il n'est pas nécessaire de disposer pour les exemples d'apprentissage des sorties désirées, seulement d'une appréciation "globale" du comportement du réseau pour chacun des exemples traités. Cet algorithme s'applique à toutes les structures de réseaux. La seule condition est de disposer de neurones de sortie stochastiques (binaires). La réponse du réseau de neurones est ainsi fonction des entrées et, aussi, des neurones de sortie.

Pour ces trois types d'apprentissage, il y a également un choix traditionnel entre :

1- *l'apprentissage << off-line >>* : toutes les données sont dans une base d'exemples d'apprentissage qui sont traités simultanément ;

2- *l'apprentissage << on-line >>* : Les exemples sont présentés les uns après les autres au fur et à mesure de leur disponibilité.

Loi de Hebb

Considérons le réseau de neurones suivant :

- n entrées e1, …, en
- m neurones N1, …, Nm.
- wij le coefficient synaptique de la liaison entre les neurones Ni et Nj
- une sortie y
- un seuil S
- Fonction de transfert : fonction Signe

si x > 0 : Signe(x) = +1

si x ≤ 0 : Signe(x) = –1

Algorithme d'apprentissage de HEBB

1/ Initialisation des poids et du seuil S au hasard.

2/ Considérer une entrée $E_l = (e_1, \ldots e_n)$ de la base d'apprentissage.

3/ Calcul de la sortie obtenue x pour cette entrée :

$a = \sum (w_i . e_i) - S$ (la valeur de seuil est introduite ici dans le calcul de la somme pondérée)

x = signe (a) (si a > 0 alors x = +1 sinon a ≤ 0 alors x = -1)

4/ Si la sortie x est différente de la sortie désirée d_l pour E_l alors

// modification des poids (μ est une constante positive, qui spécifie le pas
// de modification des poids) :

$w_{ij}(t+1) = w_{ij}(t) + \mu.(x_i . x_j)$

5/ Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement alors modification des poids puis retour à l'étape 2.

Exemple d'apprentissage par l'algorithme de HEBB

Soit la base d'apprentissage suivante :

e1	e2	x	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

Conditions initiales : $\mu = +1$, les poids et le seuil sont nuls.

Calcul de la valeur de x pour (1) :

$a = w_1.e_1 + w_2.e_2 - S = 0.0 \cdot 1 + 0.0 \cdot 1 - 0.0 = 0$ $a \leq 0 \Rightarrow x = -1$

La sortie est fausse, il faut donc modifier les poids en appliquant :

$w_1 = w_1 + e_1.x = 0.0 + 1.1 = 1$

$w_2 = w_2 + e_2.x = 0.0 + 1.1 = 1$

On passe à (2) :

$a = 1.1 + 1.-1 - 0.0 = 0$ $a \leq 0 \Rightarrow x = -1$

La sortie est fausse, il faut donc modifier les poids en appliquant :

$w_1 = 1 + 1.1 = 2$

$w_2 = 1 + 1.-1 = 0$

L'exemple suivant (3) est correctement traité : $a = -2$ et $x = -1$ (la sortie est bonne).

On passe directement, sans modification des poids à l'exemple (4). Celui-ci aussi est correctement traité.

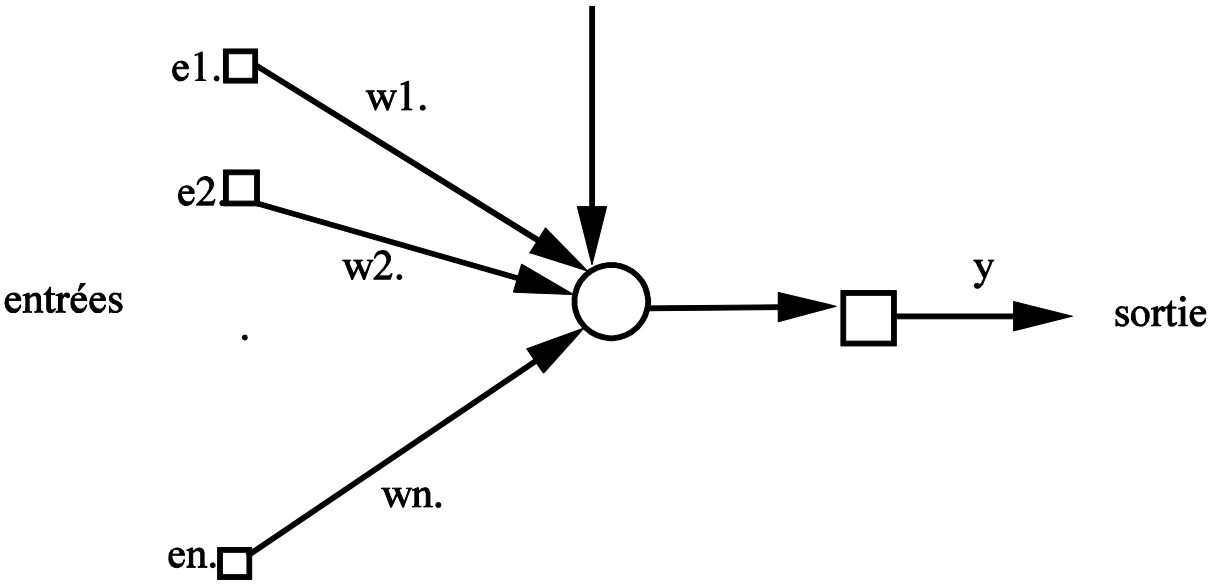
On revient alors au début de la base d'apprentissage : l'exemple (1). Il est correctement traité, ainsi que le second (2) donc l'algorithme d'apprentissage est terminé.

Remarque il faut tester tous les exemples de la base d'apprentissage

Le perceptron

Le perceptron est un modèle de réseau de neurones avec algorithme d'apprentissage créé par Frank Rosenblatt en 1958. La version ci-dessous est simplifiée par rapport à l'originale.

• n entrées e1, …, en
• n coefficients synaptiques w1, …, wn
• une sortie y
• un seuil S



Perceptron

La fonction d'activation est définie par un seuil S.
La sortie est fonction du seuil S.
Le perseptron est un classifier.

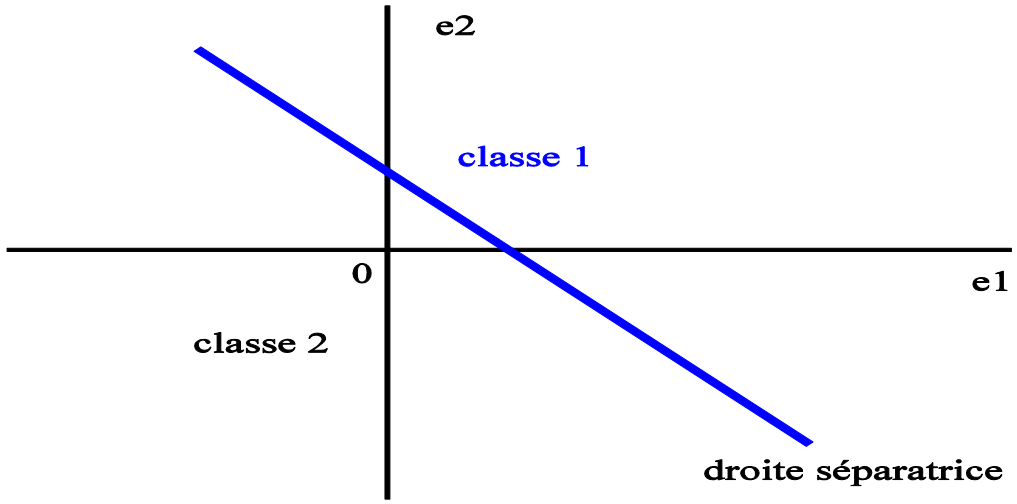
Théorème
Un perceptron linéaire à seuil à n entrées divise l'espace des entrées R^n en deux sous-espaces délimités par un hyperplan. Réciproquement, tout ensemble linéairement séparable peut être discriminé par un perceptron.

Preuve

Il suffit pour s'en convaincre de se rappeler que l'équation d'un hyperplan dans un espace de dimension n est de la forme : $e_1w_1 + e_2w_e + \cdots + e_nw_n = \beta$

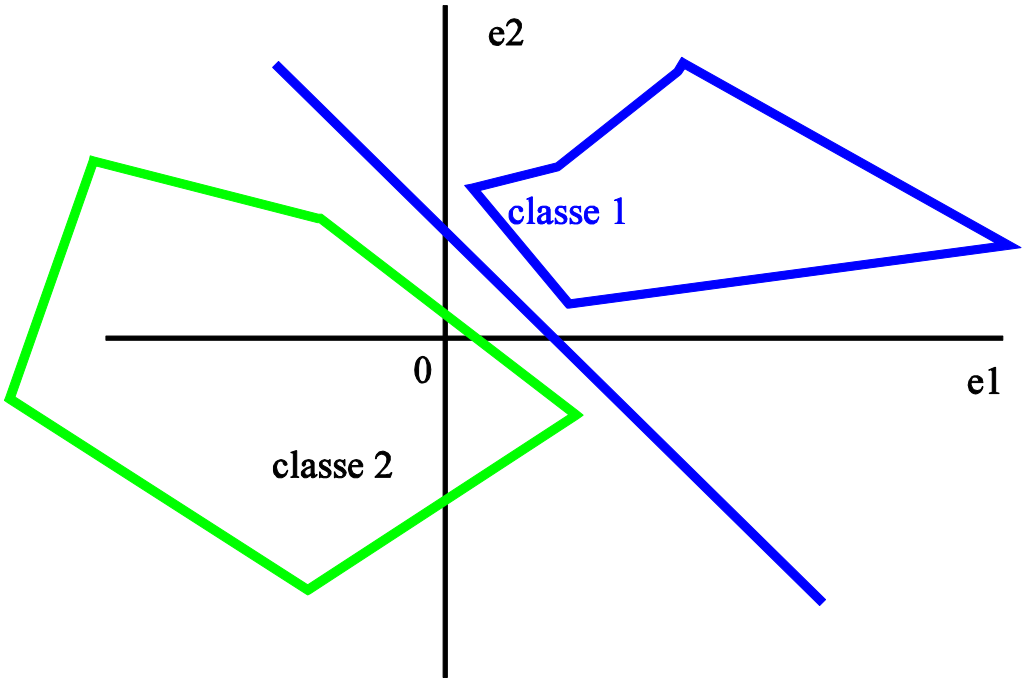
Un perceptron est donc un discriminant linéaire.

La figure suivante est un exemple de classification binaire (2 classes) classe 1 et classe 2 . La droite séparatrice a pour fonction $e_1w_1 + e_2w_e + \cdots + e_nw_n = \beta$

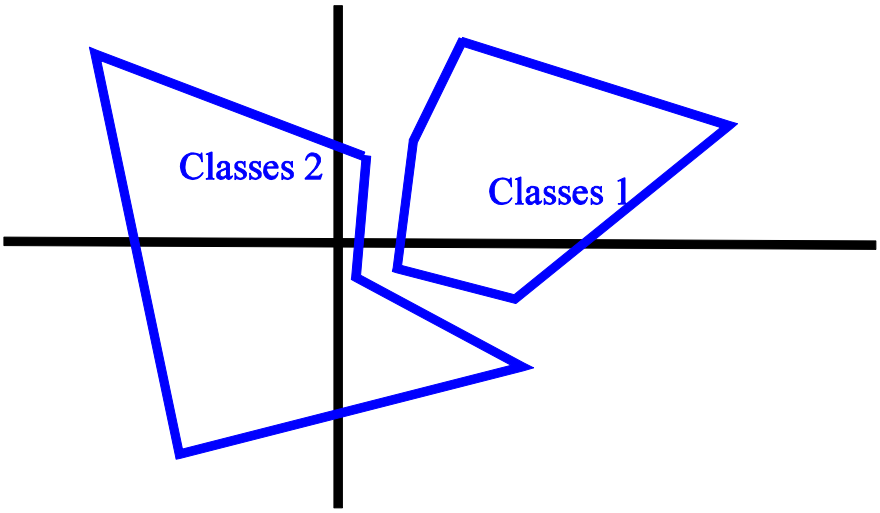


Remarque :

Ce n'est pas tous les espaces de R^n *qui sont linéairement séparables. La figure suivante donne un exemple*



Deux classes séparables



Deux Classes non séparables par un perceptron

Algorithme d'apprentissage du perceptron

1- Initialisation des poids et du seuil S à des valeurs (petites) choisies au hasard.

2- Considérer une entrée $E_l = (e_1, \ldots e_n)$ de la base d'apprentissage.

3- Calcul de la sortie obtenue x pour cette entrée :

$a = \sum (w_i . e_i) - S$

x = signe (a) (si $a > 0$ alors $x = +1$ sinon $a \leq 0$ alors $x = -1$)

4- Si la sortie x du Perceptron est différente de la sortie désirée d_l pour E_l alors modification des poids comme suit:

$$w_i(t+1) = w_i(t) + \mu.((d_l - x).e_i)$$

μ le pas de modification

Rappel : $d_l = +1$ si E est de la classe 1, $d_l = -1$ si E est de la classe 2 et $(d_l - x)$ est une estimation de l'erreur.

5- Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. Modification des poids), retour à l'étape 2.

Exemple

On considère la base d'exemples d'apprentissage suivante :

e_1	e_2	d	
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

Conditions initiales : $w_1 = -0.2$, $w_2 = +0.1$, $S = 0.2$, $(\mu = +0.1)$

$a(1) = -0.2 + 0.1 - 0.2 = -0.3$

$x(1) = -1$ la sortie désirée $d(1) = +1$, d'où la modification des poids

$w_1 = -0.2 + 0.1 . (1 + 1) . (+1) = 0$

$w_2 = +0.1 + 0.1 . (1 + 1) . (+1) = +0.3$

$a(2) = +0.3 - 0.2 = +0.1$

$x(2) = +1$ la sortie désirée $d(2) = -1$, d'où la modification des poids

$w_1 = 0 + 0.1 . (-1 - 1) . (-1) = +0.2$

$w_2 = +0.3 + 0.1 . (-1 - 1) . (+1) = +0.1$

$a(3) = -0.2 - 0.1 - 0.2 = -0.5$ la sortie désirée $d(3) = -1$, donc avancer

$a(4) = +0.2 - 0.1 - 0.2 = -0.1$ la sortie désirée $d(4) = -1$, donc avancer

$a(1) = +0.2 + 0.1 - 0.2 = +0.1$ la sortie désirée $d(1) = 1$, donc avancer

a(2) = -0.2 + 0.1 - 0.2 = -0.1. La sortie désirée d(2) = -1, donc avancer

Tous les exemples de la base ont été correctement traités, l'apprentissage est terminé.

Le Perceptron réalise une partition de son espace d'entrée en 2 classes (1 et 2) selon la valeur de sa sortie (+1 ou -1). La séparation de ces deux zones est effectuée par un hyperplan
L'équation de la droite séparatrice $0.2e_1 + 0.1e_2 = 0.2$

Remarques

Si l 'échantillon n 'est pas linéairement séparable, l 'algorithme ne converge pas.

L'algorithme peut converger vers plusieurs solutions (selon les valeurs initiales des coefficients, la valeur de ε , l 'ordre de présentation des exemples).

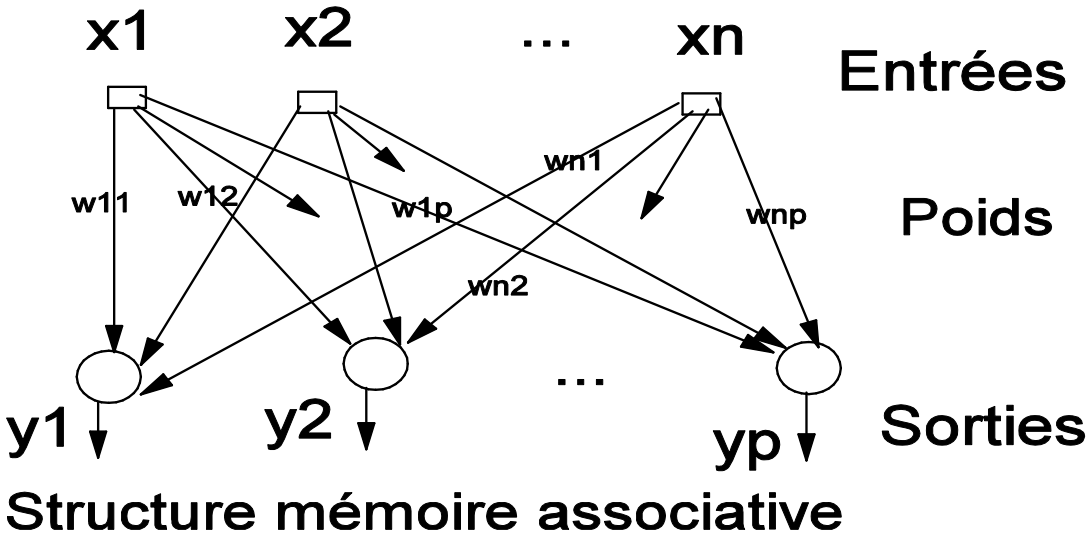
La solution n'est pas robuste : un nouvel exemple peut remettre en cause le perceptron appris.

Chapitre 5. Mémoires associatives

Les mémoires associatives sont proposées par Kohonen. Le terme "*mémoire*" fait référence à la fonction de stockage de l'information et le terme "*associative*" au mode d'adressage. L'expression "*mémoire adressable par son contenu*" est aussi souvent employée. Le système prend en entrée une information incomplète ou bruitée et retourne une information complétée ou corrigée. Par exemple, si la clef d'entrée est une image de visage, le système répond par le nom de la personne correspondante et donne l'image du visage complète.

Structure

Dans les mémoires associatives, les connexions sont totales, directes et adaptables. La figure suivante montre l'architecture d'une mémoire associative où chaque entrée est connectée par des poids modifiables à toutes les sorties. La dimension de la couche d'entrée est de n neurones, celle de sortie de p. Il y a donc n . p poids dans ce réseau.



Fonctionnement

Le principe de fonctionnement d'une mémoire associative se résume ainsi. Soit (X_1, X_2, ..., X_l, ...) un ensemble de vecteurs de R^n. A chaque vecteur X_l appelé "prototype" de l'espace d'entrée est associé un vecteur de sortie Y_l. La relation d'association entre X_l et Y_l est linéaire. Elle est

donnée par l'équation $Y_l = W . X_l$ où W est la matrice des poids de dimension (p.n). C'est une matrice rectangulaire de p lignes et n colonnes.

L'objectif est de faire réaliser à ce réseau des associations entre les vecteurs d'entrées et les vecteurs de sortie désirés. Ce qui nécessite une étape d'apprentissage.

Apprentissage

L'apprentissage est de type supervisé. La base d'apprentissage est composée de couples de vecteurs d'entrée et des vecteurs de sortie associés. L'algorithme d'apprentissage initial fait appel à la loi de Hebb. Une entrée X_l est appliquée sur les neurones d'entrée du réseau et l'on force dans le même temps les valeurs des neurones de sortie à Y_l. Le poids de chaque connexion est alors modifié selon la coactivité du neurone afférent (entrée) et du neurone efférent (sortie). Cet algorithme est itéré sur tous les exemples de la base d'apprentissage. A la fin du processus d'apprentissage, si la matrice W est initialement nulle (W = 0), on obtient :

$$W = \sum_l Y_l \times X_L^T$$

où X_L^T est la transposée du vecteur X_l (qui transforme un vecteur ligne en un vecteur colonne et réciproquement)

On distingue trois types de mémoires associatives :

- **les mémoires auto-associatives** : X et Y sont de même nature, et la mémoire permet de restaurer des données incomplètes ou bruitées,
- **les mémoires hétéro-associatives** : X et Y sont de natures différentes, par exemple à un caractère E on associe son code ASCII,
- **les classifieurs**, qui sont des mémoires hétéro-associatives particulières : Y est la classe de X

Remarque : les mémoires associatives sont aussi limités au problème séparables..

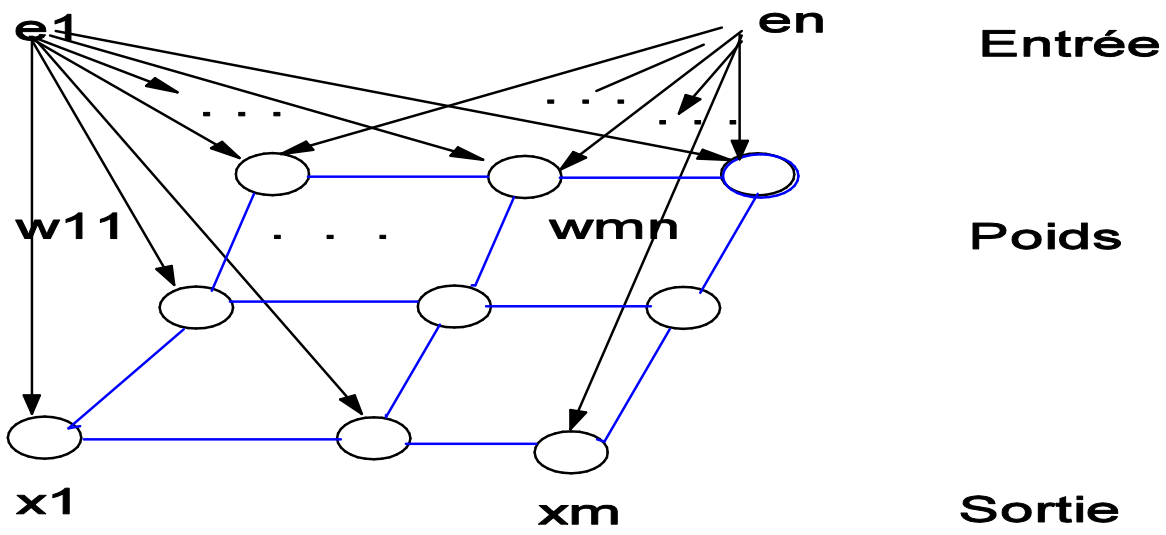
Chapitre 6 : Carte auto-organisatrice (Carte de Kohonen)

Les cartes auto-organisatrices tirent leur origine de l'existence au niveau des *cortex* moteur et sensoriel de cartes *somatotopiques* où chaque partie du corps est représentée : c'est *l'homonculus*. Ce modèle de carte auto-organisatrice appartient à la classe des réseaux à compétition. Un seul neurone de sortie est activé pour une entrée donnée. Les neurones de la couche de sortie entrent en compétition, de telle façon qu'un seul neurone de sortie est activé pour une entrée donnée. Leur principe consiste à mettre en correspondance l'espace d'entrée avec l'espace du réseau. (***Auto-organisation)*** **.** Cette compétition entre les neurones est réalisée grâce à des connexions latérales inhibitrices. Nous présentons deux modèles parmi les plus intéressants : la carte auto-organisatrice et le réseau ART1 (au chapitre suivant).

Ces réseaux utilisent des méthodes d'apprentissage non-supervisées. Ils peuvent être utilisé pour cartographier un espace réel ou encore étudier la répartition de données dans un espace de grandes dimensions comme dans le cas de problème de quantification vectorielle, de clusterisation ou de classification. Les cartes auto-organisatrices peuvent aussi être utilisées dans les cartes phonétique, le diagnostic de pannes, la compression de données, la robotique, les statistiques (méthode semblable à l'ACP), etc.

Structure

La figure suivante montre la structure d'une carte auto organisatrice 2D



• n cellules d'entrée e = (e1, …, en)

• une ***carte*** : réseau de m neurones de sortie x1, …, xm

• connexions latérales (coefficients fixes) entre les neurones de sortie :

un neurone est connecté à ses 4 plus proches voisins

• connexions de coefficient wij entre une cellule d'entrée ei et un neurone de sortie xj

Principes :

Pour une entrée, un seul neurone sur la carte est sélectionné (valeur 1). On encourage le vainqueur : « the winner takes all ». Ce neurone correspond le plus possible à l'entrée : ***minimisation d'une distance***.

Remarques :

En pratique, on normalise les entrées.

<u>Apprentissage</u>

L'algorithme d'apprentissage est dérivé de la loi de HEBB. Il est donné comme suit :

• Initialiser aléatoirement les coefficients wij
• **Répéter**
  • Prendre une entrée e = (e1, …, ei, …, en)
  • Calculer la distance dj de chaque neurone xj par rapport à e

$$d_j = \sqrt{\sum_{i=1}^{n} (e_i - w_{ij})^2}$$

  • Sélectionner le neurone xk le plus proche de e : dk = Min(dj)
  • Modifier les coefficients pour le neurone sélectionné et ses plus proches voisins (4 pour une carte 2D) :
    • **Pour** tout i :
      • wik = wik + μ ∗ (ej − wik)
      • wil = wil + β ∗ (ej − wil) où xl est un voisin de xk
    • **Fin Pour**
• **Fin Répéter**

Chapitre 7 : Un réseau à architecture évolutive, ART

ART (Adaptive Resonance Theory) est un modèle de réseau de neurones à architecture évolutive. Ce modèle se base sur l'idée suivante : c'est l'interaction entre une information issue de l'environnement et la connaissance que l'on a déjà qui nous permet d'identifier et de reconnaître des objets. Cette connaissance est un modèle de référence auquel nous venons confronter les caractéristiques d'un objet perçu. C'est ainsi que l'on arrive à catégoriser un objet.

Ainsi est née l'idée de créer un modèle artificiel évolutif : capable d'adapter les catégories déjà apprises tout en s'adaptant aux nouvelles informations qui génèrent de nouvelles catégories . [Grossberg, 2013]

Ce modèle comporte plusieurs sous modèles (ART-1, ART-2, ART-MAP, etc.). Certains modèles utilisent des méthodes d'apprentissage supervisées et d'autres des méthodes non supervisées. Ils sont appliqués surtout pour résoudre des problèmes de pattern recognition (exemple : reconnaître un chiffre à partir d'une image floue, ou une écriture manuscrite, reconnaissance des formes, etc.).

Structure

Le réseau ART1 est formé d'une couche d'entrée et d'une couche cachée. La couche de sortie est confondue avec la couche d'entrée. Il n'y a pas de connexion entre les neurones d'entrées. La couche cachée est une couche d'activation compétitive, tous les neurones sont reliés les uns aux autres par des connexions inhibitrices de poids fixes. Chaque neurone de la couche d'entrée est relié à tous les neurones de la couche cachée et chaque neurone de la couche cachée est relié à tous les neurones de la couche de sortie aussi. Un poids est associé à chaque connexion.

Fonctionnement d'un réseau ART1

A une entrée donnée E correspond après compétition des neurones de la couche cachée un neurone de la couche cachée J gagnant. J est considéré par le réseau alors comme représentant du vecteur d'entrée E. J génère en retour un vecteur binaire S sur la couche de sortie.
S est alors comparé à E. Si la différence est inférieure à un seuil fixé par le réseau, alors J est considéré comme le représentant de la classe de E. Dans ce cas la modification des poids du neurone gagnant a pour effet de consolider ses d'activation avec l'entrée E. Dans le cas

contraire, le processus reprend avec les neurones de la couche cachée sans les neurones gagnants des étapes précédentes.

Si tous les neurones de la couche cachée sont passés en revue, un nouveau neurone caché est ajouté et initialisé comme représentant de la classe correspondant à l'entrée E.

Algorithme d'apprentissage

1- Initialisation des poids aléatoirement entre 0 et 1 et choix d'un seuil d'unification ß.
2- Présentation d'un vecteur d'entrée E_l appartenant à la base d'apprentissage
3- Calcul du neurone gagnant sur la couche cachée N_j.
4- Génération en retour d'un vecteur de sortie S_j issu de ce seul neurone N_j. S_j a été seuillé afin de le rendre binaire.
5- Tentative d'unification entre S_j et E_l. Soit $|S|$ est la norme de S_j égale au nombre de composantes à 1, par exemple $|(1, 0, 1, 1)| = 3$.
   Si $|S_j| / |E_l| \geq$ ß, l'unification est réalisée. Il faut modifier les poids : étape 7.
6- Sinon $|S_j| / |E_l| <$ ß, le neurone gagnant N_j est inhibé.
   S'il y a encore des neurones non inhibés sur la couche cachée alors retour à l'étape 3.
   Sinon un nouveau neurone cachée est créé, initialisé comme représentant de la classe correspondant à la forme d'entrée E_l en utilisant la loi de modification des poids de l'étape 7.
7- Modification des poids
   Couche des poids montants :
   h neurone de la couche d'entrée, j neurone gagnant de la couche cachée.
   $w_{jh} = 1 / |S_j|$ si le neurone h est actif (valeur 1),
   $w_{jh} = 0$ sinon (valeur 0).
   Couche des poids descendants:
   j neurone gagnant de la couche cachée, k neurone de la couche de sortie.
   $w_{kj} = 1$ si le neurone k est actif,
   $w_{kj} = 0$ sinon.
   Retour à l'étape 2.
8- Quand le passage de tous les exemples de la base d'apprentissage n'occasionne plus aucun ajout de neurone, **il faut mesurer les performances : contrôler le nombre et la qualité des classes construites.**
   Si le nombre **est trop faible**, retour à l'étape 1 avec une augmentation de la valeur de ß.
   Si ce nombre **est trop élevé**, retour à l'étape 1 en diminuant la valeur de ß.

Chapitre 8 : Apprentissage par pénalité / récompense (renforcement)

Un troisième type d'apprentissage est l'apprentissage par pénalité ou par renforcement. Il n'est pas nécessaire de disposer pour les exemples de la base d'apprentissage des sorties désirées, seulement d'une appréciation générale du comportement du système pour chacun des exemples traités. Ce type d'apprentissage à toutes structures de réseaux. La seule condition est de disposer de neurones de sortie stochastiques (binaires).

La réponse du réseau de neurones est ainsi fonction des entrées et, aussi, des neurones de sortie. On introduit donc à ce niveau une part d'aléatoire dans le comportement du système. Si la réponse fournie par le système est considérée comme bonne, l'algorithme tend à favoriser l'apparition de ce comportement en réduisant l'aléatoire. Dans le cas où la réponse du système globale est considérée comme mauvaise, on cherche à éviter l'apparition ultérieure de ce comportement. Ce processus est itéré jusqu'à l'obtention du comportement désiré.

Algorithme

1/ Initialiser les poids à de petites valeurs aléatoires qui placent les probabilités des neurones de sortie autour de 0.5.
2/ Considerer une entrée El = (e1, ... , en) ,
3/ est calculér une sortie correspondante possible xi pour chaque neurone,
4/ analysér la sortie globale produite pour à générer un signal de retour r, positif ou négatif, et une sortie cible (désirée) est choisie :

- di = xi si r = +1 (récompense)
- di = -xi si r = -1 (pénalité)

5/ Modifier les poids e par la méthode du gradient :

¶wij = μ . r . erreur i . xj

En général, μ dépend de r et est pris 10 à 100 fois plus grand (μ^{+}) pour r = +1 que pour r = -1 (μ^{-}).
6/ Tant que la sortie du réseau n'a pas produit une séquence satisfaisante suffisamment longue, retour à 2

Chapitre 9 : Réseaux multicouches

Chapitre 10 : Connexionnisme et applications