PUBLICATIONS MATHÉMATIQUES ET INFORMATIQUES DE RENNES

DANIEL COLLOBERT

Où l'on verra comment les dérivées partielles réhabilitèrent les réseaux de neurones

Publications de l'Institut de recherche mathématiques de Rennes, 1991, fascicule S6
« Vième école d'été de didactique des mathématiques et de l'informatique », , p. 136-145

<http://www.numdam.org/item?id=PSMIR_1991___S6_136_0>

Exposé : "*Où l'on verra comment les dérivées partielles réhabilitèrent les réseaux de neurones*"

par Daniel COLLOBERT

C.N.E.T., Département T.E.P. 22301 LANNION Cédex

1/ INTRODUCTION

Lorsque Monsieur Gras m'a demandé de faire un exposé sur les Mathématiques employées dans notre travail de tous les jours, je me suis interrogé pendant quelques instants. Non pas que notre vie quotidienne ne fasse pas intervenir de mathématiques, au contraire elles y sont omniprésentes, mais n'étant pas mathématicien, que pouvais-je bien vous raconter qui puisse offrir de l'intérêt, pour vous qui l'êtes ?

Cet exposé porte sur les réseaux de neurones, plus exactement sur l'apprentissage dans les réseaux de neurones; or vous vous intéressez à la didactique, vous vous sentirez peut-être impliqué.

Les réseaux de neurones sont des systèmes cognitifs comme les autres, qui se sont développés au cours des cinquantes dernières années. Mais pendant cette période ils disparurent quasiment des thèmes de recherches durant une vingtaine d'années. Leur renouveau dépendit d'une remarque mathématique toute simple relative aux dérivées partielles, remarque faites en 1985. C'est le trajet de cette découverte que je vais vous relater.

Neurones, réseaux de neurones et biologie

Je serai très bref sur ce chapitre. Un neurone est une cellule du système nerveux. (cf Fig.1)

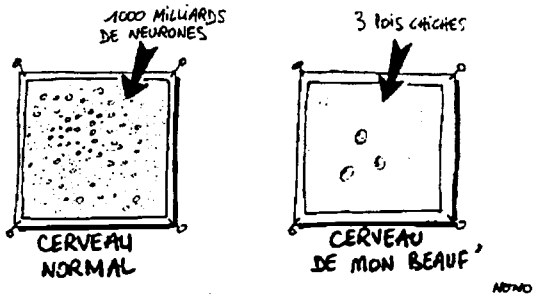


Fig. 1: le neurone est un constituant élémentaire du cerveau

Fig 2: le cerveau présente certaines facultés

Les neurones sont assemblés en une structure plus ou moins ordonnée. La plus grosse structure que l'on connaisse est le cerveau. Le cerveau possède plusieurs propriétés intéressantes (Cf Fig.2)

Applications des réseaux de neurones dans les télécommunications

Je n'en citerai qu'un certain nombre:

- celles qui fonctionnent bien : reconnaissance de la parole, compression d'image, reconnaissance de caractères.
- celles sur lesquelles certains laboratoires travaillent: traitement et compréhension de documents, authentification de signatures ou d'empreintes, filtrage d'antennes, supervision de réseaux de télécommunications.

2/ PRÉLIMINAIRES

Formalisme et notations

Pour nous, tout au long de cet exposé, un neurone sera un opérateur non linéaire possèdant plusieurs entrées et une sortie (Cf Fig 3). Nous oublierons dorénavant toute référence à la biologie.

Les entrées sont notées e_i, la sortie s. Chaque entrée est munie d'un coefficient de pondération W_i (de l'anglais weight) appelé un poids synaptique (faisant référence à la synapse du neurone biologique).

Le neurone effectue la somme de ses entrées pondérées par les poids. Cette somme pondérée sert d'argument à une fonction non linéaire f, qui peut être la fonction signe ou une fonction plus "douce" analogue qu'on appelle alors une "sigmoïde".

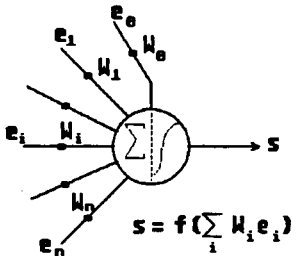


Fig. 3: le neurone élémentaire fait une somme pondérée de ses entrées.

On peut combiner plusieurs neurones dans des structures topologiques différentes.

Par exemple en couches (Cf Fig 4), où un certain nombre de neurones recoivent les entrées, envoient leur sortie sur tous les neurones de la couche située au-dessus d'eux, etc... Deux neurones de deux couches consécutives sont reliés par un poids qui dépend de ces deux neurones. Il n'y a pas de liaison entre deux neurones d'une même couche. Les neurones qui sont ni des entrées ni des sorties son appelés "cachés".

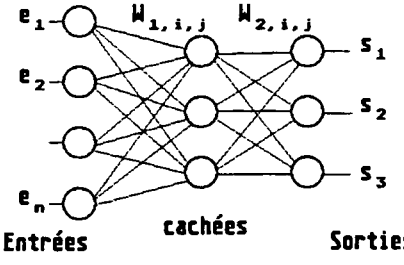


Fig. 4: une structure en couche.

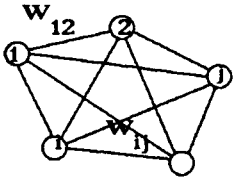


Fig. 5: un réseau totalement connecté

Un autre exemple canonique est le réseau où tous les neurones sont connectés entre eux (Cf Fig. 5). Là aussi les neurones sont reliés par des poids; généralement le poids de la liaison du neurone 1 vers le neurone 2 est posé égal

au poids de la liaison du neurone 2 vers le neurone 1 ($W_{ij} = W_{ji}$).

Apprentissage

On appellera apprentissage du réseau, la détermination des poids de façon à ce que le réseau fonctionne de la manière recherchée.

C'est un terme que nous sommes en droit d'utiliser. E. Kandel dans "Cellular basis of behaviour" donne la caractérisation suivante:

"L'apprentissage, c'est à dire la capacité d'acquérir de nouveaux comportements ou d'en modifier d'anciens par expérience, est une propriété de presque tous les animaux, y compris ceux que l'on peut difficilement qualifier d'intelligents".

Daniel Andler de Paris VII, que vous connaissez bien, en donne une autre définition:

"On appelle apprentissage toute transformation d'un système cognitif ayant pour effet de conférer à ce système une nouvelle disposition".

La circularité de cette phrase dénote la difficulté de définir précisement ce qu'est l'apprentissage; néammoins, sans dire que les réseaux de neurones sont intelligents et sans affirmer qu'il s'agit de systèmes cognitifs nous sommes en droit de parler d'apprentissage lorsque un réseau de neurones modifie ses comportements ou acquièrt de nouvelles dispositions.

L'apprentissage n'a de sens que si une **généralisation** est possible. On comprendra: la capacité à étendre une compétence à des exemples non appris. Pour citer encore D. Andler, la généralisation:

"C'est la transformation d'une présentation partielle en une représentation complète..."
"... C'est un processus d'inférence déductive"

Le problème que nous allons discuter est de savoir comment faire pour apprendre quelque chose à un réseau de neurones de façon à ce qu'il puisse généraliser ses connaissances.

Il existe plusieurs façons d'atteindre ce but. La plus simple consisterait à utiliser une des théories plus ou moins formelles, qui existent actuellement. Les procédures d'apprentissage d'un réseau s'en dérivent facilement. Il se peut que dans quelques années, la connaissance savante qu'elles impliquent actuellement fasse tellement partie du fond scientifique de tout honnête homme qu'elles en seront évidentes. Ce n'est pas encore le cas, aussi, pour montrer où se placent les problèmes, je vais vous faire suivre l'histoire des réseaux de neurones de 1943 à 1985.

Quelques dates qui vont titrer les paragraphes suivants:
1943: Mc Culloch et Pitts proposèrent un modèle de neurone, qu'ils appelèrent formel, et montrèrent que ce modèle induisait des fonctions intéressantes. 1960: B. Widrow invente le terme de Adaline (ADAptative LINear Element) qui dénomme un neurone muni d'une procédure d'apprentissage qui sera connu par la suite comme procédure de Widrow-Hoff. 1961: Rosenblatt invente une machine de reconnaissance des formes qu'il appelle "Perceptron". Pendant les 7 années qui vont suivre la cybernétique se développe jusqu'à ce qu'en 1968 Minsky et Papert publient leur livre "Perceptrons" démontrant de façon irréfutable les limites de

ceux-ci. Les réseaux de neurones ne s'en relevèrent pas. D'autant plus qu'à la même époque le développement des ordinateurs et l'apparition de techniques symboliques de traitement de l'information (appelées Intelligence Artificielle) autorisèrent d'autres espoirs.

Il fallut attendre 1982 pour entendre à nouveau parler de réseaux de neurones, grâce à un article de J. Hopfield qui utilisa ses outils de physicien statistique pour étudier ces objets. Il est surprenant de savoir qu'il ne levait pas les limites du Perceptron, et que pourtant il marqua le commencement d'une nouvelle ère que l'on appelle maintenant connexionniste.

Ce renouveau d'intérêt conduisit en 1985 à la publication d'algorithmes qui, eux, levaient les limitations du Perceptron et de théorèmes de mathématiciens qui autorisaient les espoirs les plus grands.

Nous sommes maintenant en 1991, trente ans après le Perceptron, cet exposé marque donc un anniversaire.

3/ Mc CULLOCH ET PITTS 1943

Ces auteurs proposèrent une version très simplifiée du neurone biologique qu'ils appelèrent neurone formel. Il s'agit d'un objet qui possède des entrées et une sortie et qui effectue une somme pondérée de ses entrées. Si cette somme est supérieure à un certain seuil Θ, alors la sortie prend la valeur + 1.

$s = +1 \quad si \quad \sum_{i=1} W_i \, e_i > \Theta$

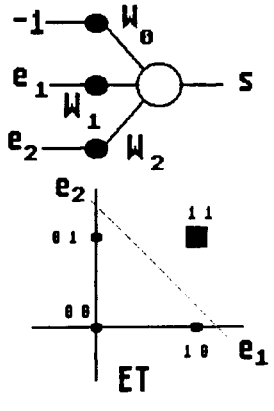
$s = 0 \quad sinon$

Afin de ne pas avoir à manipuler le seuil, nous allons créer une entrée particulière e_o qui aura toujours la valeur -1, associée à un poids W_o égal au seuil (Cf Fig. 3). On appellera "a" l'entrée totale définie par :

$$s = +1 \quad ssi \quad f(a) = f\left(\sum_{i=0} W_i \, e_i \right) > 0$$

On appellera vecteur d'entrée le vecteur **E** à n+1 composantes $\{e_o \ldots e_n\}$ et **W** le vecteur de poids $\{W_o \ldots W_n\}$.

Il est facile de voir qu'un tel neurone est un séparateur linéaire. Il réalise une partition de son espace d'entrées en deux classes $\mathcal{E}^+$ et $\mathcal{E}^-$ telles que si le vecteur d'entrée $\in \mathcal{E}^+$ la sortie vaut +1 et si le vecteur d'entrée $\in \mathcal{E}^-$ la sortie vaut -1 (Cf Fig. 6). La frontière est un hyperplan.



$$a = W_1 e_1 + W_2 e_2 - W_o$$

$$s = Sign(a)$$

$$e_2 = - \frac{W_1}{W_2} e_1 + \frac{W_0}{W_2}$$

Fig 6. Le ET logique est linéairement séparable.

C'est la raison pour laquelle un neurone ne peut calculer que les fonctions linéairement séparables.

Exemples : Comment réaliser les opérations logiques ET, OU et XOR (OU exclusif)?

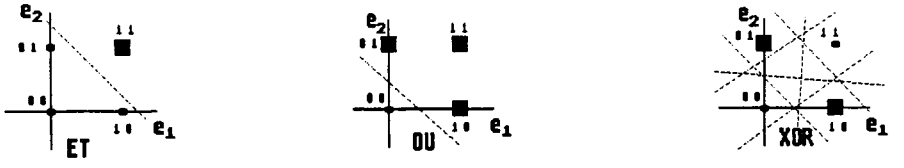


Fig. 7 Séparabilité linéaire du ET, du OU et du XOR

Pour trouver les valeurs des poids l'intuition suffit pour un ET: il suffit de prendre $W_1 = 1$, $W_2 = 1$, $W_0 = 2$. Pour réaliser le OU, le lecteur pourra essayer $W_1 = 1$, $W_2 = 1$, $W_0 = 1$. Mais quelles sont les valeurs des poids qui réalisent le XOR? Il n'y en a pas. Pourquoi? Parce que cette opération n'est pas linéairement séparable. Sur la Fig. 7 j'ai dessiné en pointillé diverses séparatrices, et on y voit qu'il est impossible de placer les 2 gros carrés d'un côté d'une droite et les 2 petits de l'autre côté.

4/ LE PERCEPTRON, ROSENBLATT, 1957

Les différents éléments d'un Perceptron sont (cf Fig. 8):

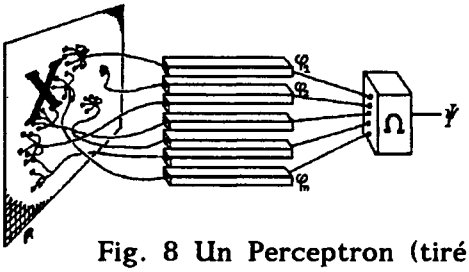


Fig. 8 Un Perceptron (tiré de Minsky et Papert)

- une rétine constituée de capteurs binaires (+/- 1).
- de plusieurs cellules d'association qu'on peut appeler des extracteurs de primitives et qui réalisent des fonctions booléennes quelconques sur les pixels de la rétine. Les prédicats peuvent prendre les valeurs +/- 1.
- une seule couche de poids modifiables.
- une cellule de décision (dans un Perceptron simple), notée ici Ω et fournissant une sortie Ψ valant +/- 1.

Minsky et Papert ont trouvé le talon d'Achille du Perceptron en formulant le théorème:

"Le prédicat de connexité n'est pas calculable par un Perceptron de diamètre limité".

Ce qui m'oblige à définir le diamètre d'un Perceptron: chaque prédicat ne prend ses valeurs que sur un disque de diamètre D de la rétine (les tentacules de la Fig. 8). Si l'image d'entrée est connexe (en un seul morceau) le Perceptron doit répondre 1. Il doit répondre 0 si il y a plusieurs composantes.

Pour avoir une idée de la démonstration regardons la Fig. 9 où 2 prédicats travaillent, l'un sur le côté gauche de l'image, l'autre sur le côté droit.
Pour la fig 1, la sortie du premier prédicat est 1, la sortie du deuxième est 1 et le Perceptron doit répondre 0. Pour la figure 2 les valeurs des prédicats sont 1

et 0 et le Perceptron doit répondre 1 etc ... Le lecteur se convaincra que le Perceptron doit réaliser, en fait le XOR ! qui n'est pas calculable par un élément linéaire à seuil comme nous l'avons vu plus haut.

D'où la déchéance du Perceptron en 1968.

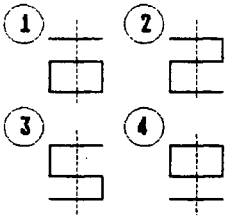


Fig 9. Deux figures connexes (la 2 et la 3) et deux qui ne le sont pas (la 1 et la 4).

Si le groupe de gauche voit  la sortie du prédicat est "1" par ex.

Si le groupe de droite voit  la sortie vaut "1", sinon "0".

5/ APPRENTISSAGE DES POIDS D'UN NEURONE FORMEL. PRÉSENTATION INTUITIVE ET INFORMELLE

Même si le Perceptron (le neurone) souffre de limitations, il est intéressant de trouver une méthode qui calculerait les valeurs des poids. L'intuition ne suffit pas toujours, surtout si il a beaucoup d'entrées.

Soit un ensemble de p formes $\{ f_1 \ldots f_p \}$, chaque forme $f_k \in \mathcal{E}^+$ ou $\mathcal{E}^-$ et est représentée par un vecteur $\mathbf{E_k}$.

Il s'agit de trouver le vecteur **W** tel que le neurone répondra +1 si la forme $\in \mathcal{E}^+$, -1 autrement. La sortie désirée, d_k valant +1 ou -1.

On peut procéder par itération en présentant une forme d'entrée k et en calculant la sortie effective s_k. On compare cette sortie à la sortie désirée d_k. Eventuellement on modifie le vecteur de poids de façon à ce que la sortie effective se rapproche de la sortie désirée. On recommence avec une nouvelle forme d'entrée. C'est ce processus itératif qui tient lieu d'apprentissage.

Il y a 4 cas possibles:

$d_k = +1 \quad s_k = +1$ } bon
$d_k = -1 \quad s_k = -1$ }

$d_k = +1 \quad s_k = -1$ } mauvais
$d_k = -1 \quad s_k = +1$ }

Si, par exemple, la sortie effective vaut -1 alors que la sortie désirée vaut +1, il faut augmenter les poids tels que (e>0 et w>0) ou (e<0 et w<0). L'entrée totale sera ainsi augmentée et on se rapprochera de la sortie désirée.

La règle de modification des poids pour le Perceptron, telle qu'elle a été proposée en 1961 est la suivante:

$$W_i^{t+1} = W_i^t + \lambda (d_k - s_k) e^k_i \qquad \lambda \text{ "petit" } > 0$$

k relatif à la forme présentée, i étant la ième composante du poids ou de l'entrée (Cf Fig. 3). Si d = s, W n'est pas modifié; si d = +1 et s = -1, W est

modifié de $+2\lambda e$; si d = -1 et s = +1, W est modifié de $-2\lambda e$. La modification est du même signe que d et tend à rendre a du même signe que d. (d-s) est appelé le signal d'erreur; l'algorithme d'apprentissage est appelé à correction d'erreur.

6/ APPRENTISSAGE DES POIDS D'UN NEURONE FORMEL. PRÉSENTATION PLUS FORMELLE: MINIMISATION D'UNE FONCTION DE COÛT

Le problème peut se transformer en un problème d'optimisation consistant à trouver les poids qui minimisent une certaine fonction de coût.

Fonction de coût

Ex1: nombre d'exemples mal classés:

$$C(W) = \sum_k (d_k - s_k)$$

Cette fonction n'a pas de bonnes propriétés: pas continue, pas dérivable.

Ex2: Perceptron:

$$C_p = \sum_k (s_k - d_k)\, a_k$$

positive, ou nulle si il n'y a pas d'erreur; cette fonction de coût est continue et dérivable par morceaux.

Ex3: moindres carrés:

$$C_c = \sum_k (d_k - a_k)^2$$

Même si elle ne garantit pas la minimisation du nombre d'erreurs cette fonction de coût à de bonnes propriétés.

Procédure de minimisation

Une fois la fonction de coût fixée il faut définir une méthode de minimisation. Il en existe plusieurs, je ne parlerai que de la méthode de descente de gradient qui est une méthode itérative simple dont le fonctionnement est schématisé sur la Fig. 10.

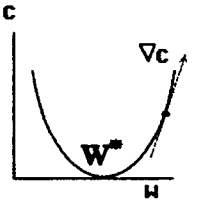


Fig. 10: Descente de gradient pour une fonction de coût ne dépendant que d'un seul paramètre W.

Pour trouver W^* tel que $C(W^*)$ soit minimum, on part d'un W^0 et on chemine sur la surface C en sens inverse du gradient. On calcule itérativement

$$W^{t+1} = W^t - \lambda(t)\ \mathrm{Grad}\{ C(W^t) \}$$ λ est un pas d'itération "petit"

Ex1: pour le Perceptron

$$C_p = \sum_k (s_k - d_k)\, a_k \quad \text{avec } a_k = W_1 e_1 + ... + W_n e_n$$

Il vient facilement:

$$\frac{\partial C_p}{\partial W_i} = \sum_k (s_k - d_k)\, e^k{}_i \qquad \text{car } \frac{\partial a_k}{dW_i} = e^k{}_i$$

Il s'ensuit que la procédure de convergence s'écrit:

$$\mathbf{W}_i^{t+1} = \mathbf{W}_i^t + \lambda \sum_k (d_k - s_k)\, \mathbf{E}_k$$

Le théorème de convergence existe.

Ex2: Pour les moindres carrés

$$C_c = \sum_k (d_k - a_k)^2$$

le gradient par rapport aux poids sera:

$$\frac{\partial C_c}{\partial W_i} = \sum_k (a_k - d_k)\, e^k{}_i$$

Il s'ensuit que la procédure de convergence s'écrira ici:

$$\mathbf{W}_i^{t+1} = \mathbf{W}_i^t + \lambda \sum_k (d_k - a_k)\, \mathbf{E}_k$$

S'il n'existe pas de théorème de convergence pour cette procédure, c'est pourtant celle-ci qui est utilisée car elle est généralement plus rapide. C'est Widrow et Hoff qui l'ont publiée en 1960 pour leur neurone formel qu'ils appellent Adaline.

Remarque: la seule différence avec la règle du Perceptron est que la valeur de (d-s) est non binaire.

7/ LES LIMITATIONS ET LE PROBLÈME DE FOND

Les neurones et perceptrons sont donc limités aux problèmes linéairement séparables. Néammoins, comme un théorème de logique affirme que toute proposition booléenne peut se mettre sous la forme d'une conjonction de disjonctions (un OU de ET's) on peut penser qu'il suffit de réaliser un réseau à deux couches de poids modifiables pour tout résoudre. Et donc qu'il suffit de modifier les algorithmes que nous venons de voir en tenant compte de la structure en couche.

Le problème de fond est de trouver le bouc émissaire (Cf Fig. 11)

Comment une erreur trouvée sur la sortie 1 doit être répercutée sur les valeurs des poids des couches intermédiaires alors que toutes les connexions participent à cette erreur? Autrement dit, quelles sorties désirées doivent être assignées aux cellules cachées?

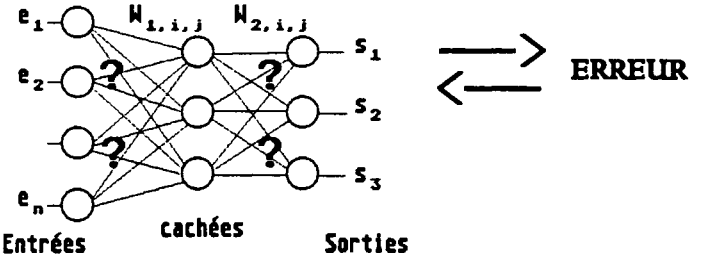


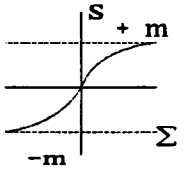
Fig. 11: comment modifier les poids internes alors que la seule chose connue est l'erreur de sortie?

C'est pour ne pas avoir su trouver une solution à ce problème de fond que les réseaux de neurones subirent un arrêt de leur développement. Et cela malgré des travaux comme ceux de Widrow qui proposa des modèles multi-couches (appelés Madaline pour Multiple Adaline) mais où les poids de la dernière couche sont tous identiques et fixes.

8/ LA RÉTRO-PROPAGATION DU GRADIENT D'ERREUR (1985)

Il fallut attendre environ 25 ans pour résoudre ce problème du bouc émissaire, et ce n'est pas parce que les mathématiques nécessaires étaient trop difficiles. En fait, répartir l'erreur de sortie sur les neurones des couches cachées ne fait intervenir que des dérivées partielles comme nous allons le voir.

Soit un réseau comme celui indiqué sur la figure 4. Plutôt que de prendre la fonction Signe comme fonction non linéaire du neurone, prenont une fonction "sigmoïdale" comme celle de la Fig. 12. Une forme canonique de cette fonction peut s'écrire $f(A_k) = m\ th(A_k)$; cette fonction est continue, monotone croissante et dérivable.



$s = m\ th\ (\Sigma)$

Fig. 12: Fonction de transfert du neurone

A la présentation du motif m, la fonction de coût quadratique peut s'écrire:

$C = \sum_I (S_i - D_i)$ où D_i est la i ème sortie désirée

La procédure de descente de gradient s'écrira toujours:

$$W_{ij}^{t+1} = W_{ij}^{t} - \lambda \frac{\partial C}{\partial W_{ij}}$$

L'astuce consiste à poser $\frac{\partial C}{\partial W_{ij}} = \frac{\partial C}{\partial A_i} \frac{\partial A_i}{\partial W_{ij}}$

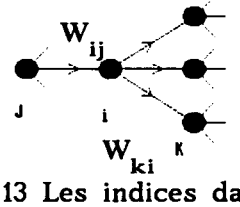
Comme $A_i = \sum_j W_{ij} S_j$ il vient immédiatement: $\frac{\partial C}{\partial W_{ij}} = \frac{\partial C}{\partial A_i} S_j$

On est conduit à calculer le gradient par rapport à l'entrée totale pour les cellules de sortie d'une part et pour les cellules cachées d'autre part.

» Pour les cellules de sortie (pour la présentation d'un motif m):

$$\frac{\partial C}{\partial A_i} = 2\ (S_i^m - D_i^m) \frac{\partial S_i}{\partial A_i} = 2\ (S_i - D_i)\ f'(A_i)$$

» Pour les cellules cachées la dérivation est un peu plus complexe (Fig 13):



$A_k = W_{ki} S_i + W_{kl} S_l + ...$

Fig. 13 Les indices dans les couches cachées

En utilisant une fois encore les differentielles composées:

$$\frac{\partial C}{\partial A_i} = \sum_k \frac{\partial C}{\partial A_k} \frac{\partial A_k}{\partial A_i} = \sum_k \frac{\partial C}{\partial A_k} \frac{\partial A_k}{\partial S_i} \frac{\partial S_i}{\partial A_i} = \sum_k \frac{\partial C}{\partial A_k} W_{ki} \quad f'(A_i)$$

La somme étant effectuée sur la couche supérieure, le gradient par rapport à l'entrée totale a été calculé auparavant; si il y a une seule couche cachée, il est donné par la formule relative aux cellules de sortie donnée plus haut.

On peut ainsi calculer de proche en proche toutes les grandeurs nécessaires au calcul de la descente de gradient. D'où le nom donné à la méthode: rétro-propagation du gradient.

9/ CONCLUSIONS

On a vu comment procéder pour utiliser des réseaux de neurones à plusieurs couches. En principe ils peuvent résoudre toute sorte de problèmes de classification. De fait ils le font (le XOR en fait partie) et certains sont vraiment utilisés (reconnaissance de l'écriture manuscrite par exemple). Mais dès lors que la tâche est très complexe, se posent les problèmes de la détermination du nombre de couches et du nombre de neurones dans chaque couche. Pour l'instant il n'existe pas d'autres guides que l'expérience ou l'intuition du concepteur de réseaux. Et cela dure depuis 1985. S'il a fallu attendre 25 ans pour trouver la rétro-propagation, combien de temps faudra-t-il attendre pour trouver des méthodes analytiques permettant de construire un réseau pour un problème donné ?

Daniel Collobert
Centre National d'Études des Télécommunications
BP 40 22301 Lannion Cedex