

MEMOIRE

présenté en vue d'obtenir

LE DIPLOME D'INGENIEUR I.I.E.

Arnaud KNIPPEL

UTILISATION DES MODELES CONNEXIONNISTES  
POUR LA CLASSIFICATION AUTOMATIQUE  
DE DONNEES SATELLITAIRES

Directeur de Recherche : M. Laurent DUFOUR, enseignant-chercheur à l'I.S.A.B.

remis le 16 mars 1995

Conservatoire National des Arts et Métiers

INSTITUT D'INFORMATIQUE D'ENTREPRISE

FICHE SIGNALÉTIQUE

Mémoire d'Ingénieur I.I.E.

UTILISATION DES MODELES CONNEXIONNISTES  
POUR LA CLASSIFICATION AUTOMATIQUE  
DE DONNEES SATELLITAIRES

**Auteur :** Arnaud KNIPPEL

**Directeur de recherche :** M. Laurent DUFOUR, enseignant-chercheur à l'ISAB.

**Centres d'intérêt :** analyse numérique, réseaux neuronaux, classification supervisée, télédétection.

**Descriptif :** Mettre au point un procédé de classification supervisée des pixels d'une image satellite à l'aide des réseaux neuronaux.

## Résumé

Cette recherche s'inscrit dans un projet de mise au point d'un procédé de classification automatique supervisée des pixels d'une image satellitaire à l'aide de réseaux neuronaux, en l'occurrence des perceptrons multicouches. Ces réseaux sont en effet reconnus comme de bons classifieurs et n'imposent pas d'hypothèses quant à la distribution des données, mais ils rebutent souvent par leur caractère irrégulier et la lenteur de leur apprentissage.

Après avoir étudié et sélectionné quelques méthodes connexionnistes parmi les plus significatives, je me suis attaché à développer un outil, en C++, qui permette de faciliter l'implémentation de ces méthodes et leur test. Parmi les méthodes de rétro-propagation que j'ai pu explorer, certaines ont manifesté de bonnes qualités pour résoudre ce type de problèmes, notamment la rétro-propagation avec inertie, pourtant la plus ancienne, mais aussi la méthode de Barnard et la méthode SCG de Möller avec calcul exact du produit du hessien par un vecteur.

Les expériences ont été réalisées à partir des données d'une image LANDSAT-TM de la région de Ciney en Wallonie. Sur ces données, certaines méthodes connexionnistes ont montré pouvoir faire au moins aussi bien en qualité que des méthodes statistiques aussi prisées que les k-plus proches voisins ou des méthodes de maximum de vraisemblance.

[Barnard 92] BARNARD E. - *Optimization for Training Neural Nets*, I.E.E.E. Transactions on Neural Networks Vol. 3, n°. 2, 1992, p. 232-240.

[Girard Girard 89] GIRARD M.C. et GIRARD C.M. - *TELEDETECTION APPLIQUEE zones tempérées et intertropicales*, Masson, Paris, 1989, 260p.

[Möller 93b] MOLLER M. - *A scaled conjugate gradient algorithm for fast supervised learning*. Neural Networks, vol 6, 1993, p. 525-533.

[Pearlmutter 94] PEARLMUTTER B. A. - *Fast Exact Multiplication by the Hessian*, Neural Computation, Vol. 6, 1994, p. 147-160.

[Rumelhart et al. 86] RUMELHART D.E., HINTON G.E., WILLIAMS R.J. - *Learning Internal Representations by Error Propagation*, Parallel Distributed Processing : Explorations in the Microstructures of Cognition. - M.I.T. Press. - Vol. 1, 1986, p. 318-362.

## Table des matières

0.1. Présentation de l'ISAB .....	6
0.2. La télédétection .....	6
1. Exploitation des images satellites.....	8
1.1. Nature des données .....	8
1.2. Classification.....	9
1.2.1. Classification non supervisée.....	10
1.2.2. Classification supervisée .....	10
1.3. Post-traitements .....	10
1.4. Utilisation des réseaux neuronaux .....	11
2. Présentation de la rétro-propagation.....	11
2.1. Perceptron multicouches .....	12
2.2. Principe de la rétro-propagation .....	13
2.3. Possibilités d'accélération de l'apprentissage .....	14
2.3.1. Modification de la rétro-propagation.....	14
2.3.2. Fonctions d'activation .....	15
2.3.3. Codage des entrées et des sorties .....	16
2.3.4. Topologie et généralisation .....	18
3. Accélérer la rétro-propagation par la voie algorithmique.....	20
3.1. Méthodes numériques .....	20
3.1.1. Algorithme des gradients conjugués .....	20
3.1.2. Procédures de reprise de l'algorithme .....	22
3.1.3. Méthode de Newton .....	23
3.1.4. Méthodes de sécantes .....	24
3.2. Méthodes connexionnistes.....	25
3.2.1. Terme d'inertie.....	25
3.2.2. Adaptation du coefficient d'apprentissage.....	25
3.2.2.1. Méthode de [Vogl et al. 88].....	26
3.2.2.2. Vecteur d'apprentissage .....	26
3.2.2.3. Adaptation du terme d'inertie .....	27
3.2.2.4. Méthode stochastique de Barnard .....	27
3.2.2.5. Estimation de la valeur propre principale du hessien.....	28
3.2.2.6. Approximation diagonale [Le Cun 88] .....	28
3.2.2.7. Algorithme quickprop [Fahlman 88] .....	29
3.2.2.8. Méthode OSS (One Step Secant) [Battiti 90] [Battiti 92].....	29
3.2.2.9. Méthode SCG (Scale Conjugate Gradient) [Möller 93b] 30	30
3.2.3. Multiplication du hessien par un vecteur.....	31
4. Description des données d'expérimentation .....	33
4.1. Origine des données .....	33
4.2. Nature des données .....	33
4.3. Construction des bases .....	34
5. Choix et implémentation des méthodes connexionnistes.....	34
5.1. Organisation des objets .....	35

5.2. Choix et description des algorithmes .....	35
6. Comparaison des méthodes .....	37
6.1. Complexité .....	37
6.2. Comportement des méthodes .....	37
6.2.1. ReseauRPG .....	38
6.2.2. ReseauHd .....	38
6.2.3. Apprentissage sur toutes les classes.....	38
6.2.4. Comparaison avec des méthodes statistiques .....	39
Conclusion .....	40
Bibliographie.....	41
Annexes .....	45

## 0.1. Présentation de l'ISAB

L'Institut Supérieur Agricole de Beauvais a été fondé en 1854 à Beauvais par des Frères des communautés chrétiennes, un scientifique et un professionnel. Depuis 1991, l'ISAB est aussi installé à Cergy-Pontoise et forme avec 8 autres écoles supérieures l'Institut Polytechnique Saint-Louis qui constitue l'un des pôles d'enseignement supérieur de l'Institut Catholique de Paris.

L'ISAB forme en 5 ans des ingénieurs généralistes pour les entreprises de l'agro-industrie, de l'agriculture et de l'environnement.

Le site de Beauvais, sur un campus boisé de 10 ha jouxtant une ferme d'application et des parcelles d'expérimentation, accueille les 2<sup>ème</sup>, 3<sup>ème</sup> et 5<sup>ème</sup> années d'étude à dominante agronomique. Le site de Cergy accueille quant à lui les 1<sup>ère</sup> et 4<sup>ème</sup> années d'étude à dominantes sciences fondamentales, gestion et industrie.

Compte tenu du caractère essentiellement utilitaire de l'informatique pour les ingénieurs en agriculture ou en agronomie industrielle, l'ISAB a fait le choix du PC. Les nombreux PC sont reliés sur chaque site par un réseau Novell, et l'ISAB a prévu de se connecter prochainement au réseau Internet.

## 0.2. La télédétection

Gérard Guyot définit la télédétection comme *l'ensemble des techniques qui sont utilisées pour la détermination à distance des propriétés des surfaces naturelles (sols, cultures, roches, surfaces d'eau...), à partir des rayonnements qu'elles réfléchissent ou émettent dans différents domaines de longueurs d'ondes.*[Guyot 93]

Les rapides progrès sur les matériels de détection et sur les méthodes de traitement et d'analyse des images contribuent à l'importance croissante des images satellites, qui remplacent encore très imparfaitement les photographies aériennes ou les relevés sur le terrain, mais permettent de substantielles économies (on peut parler d'économies d'échelle).

Les applications concernent bien sûr la météorologie, l'océanographie et la recherche de gisements pétroliers et miniers, domaines utilisant systématiquement les données de télédétection. Elles concernent en fait tous les domaines, civils ou militaires, qui nécessitent des informations sur de vastes territoires dont les éléments varient rapidement. C'est notamment le cas de l'agriculture, secteur d'importance stratégique s'il en est : estimation de la production, surveillance des cultures pour le contrôle de l'attribution des subventions, l'évaluation des dégâts causés par les catastrophes naturelles, ou encore la surveillance de l'activité agricole des pays étrangers (programme américain LACIE de 1977).

**Téledétection et rétro-propagation :**

**présentation**

# **1. Exploitation des images satellites**

Entre l'observation de la Terre par un satellite artificiel et la création d'une carte, toute une chaîne d'opérations a lieu.

Les données brutes fournies par le satellite, habituellement enregistrées sous forme numérique, doivent notamment être traitées pour tenir compte des déformations optiques et géométriques dues aux conditions d'observation. On parle de prétraitements radiométriques (correction des effets instrumentaux) et de prétraitements géométriques (effet panoramique, rotation et courbure de la Terre, variation d'altitude du satellite par rapport à l'ellipsoïde de référence). La phase d'analyse peut ensuite avoir lieu. Elle consiste essentiellement en une classification des pixels de l'image et est accompagnée de prétraitements et de post-traitements destinés à faciliter l'analyse (réduction de l'information par une ACP par exemple) ou à améliorer la lisibilité de la carte produite (lissage, filtrage...).

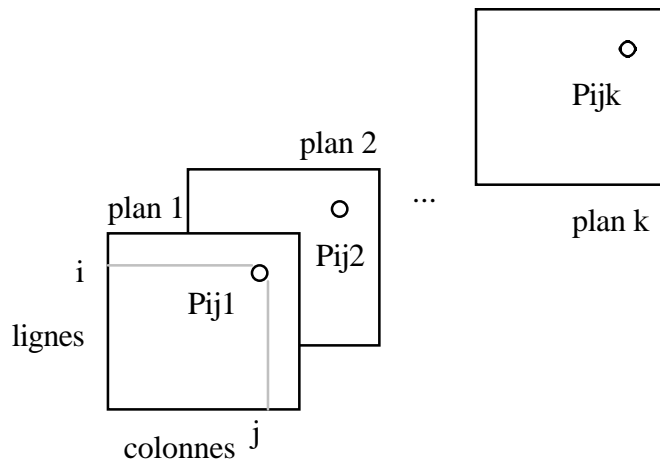
Dans la suite, on s'intéresse plus particulièrement au cas d'images multispectrales produites à partir du satellite LANDSAT-TM.

## **1.1. Nature des données**

Dans le cas où les capteurs sont des radiomètres (SPOT, LANDSAT TM), chacun de ces capteurs fournit une mesure de la luminance associée à un élément de terrain appelé pixel (c'est en fait la valeur moyenne des luminances des points du sol contenus dans ce pixel). Cette mesure est numérisée sur 8 bit : il y a donc 256 niveaux de luminance, le niveau 0 signifiant que l'intégralité du rayonnement a été absorbé (luminance nulle).

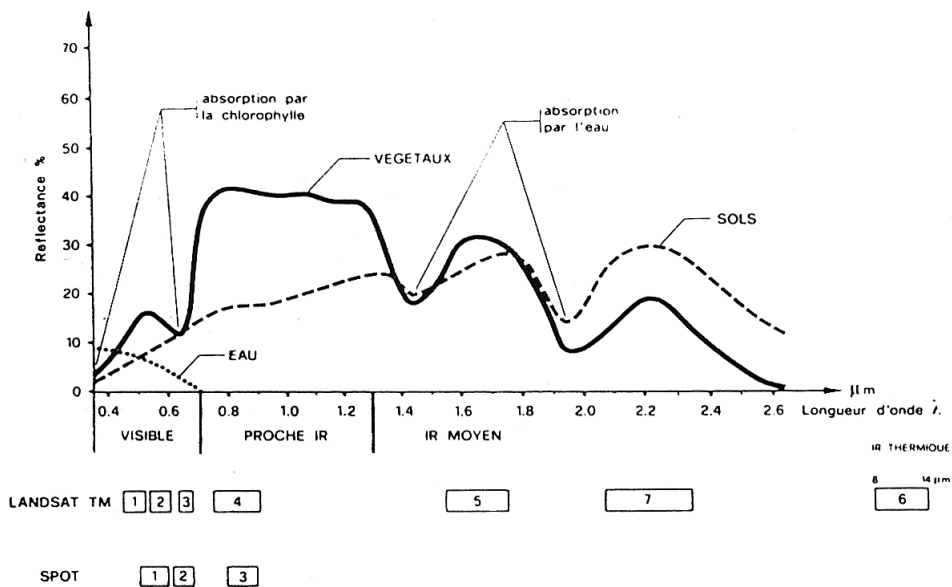
Cette mesure est étroitement dépendante de la nature du sol et de la longueur d'onde associée au radiomètre, de sorte que l'on peut théoriquement retrouver des caractéristiques d'un élément de terrain à partir des données numérisées provenant d'un nombre suffisant de canaux associés à des longueurs d'onde bien choisies.

Une image multispectrale est un ensemble de matrices ou plans (un plan par longueur d'onde) où sont enregistrées les mesures associées à chaque pixel (figure 1).



**figure 1** : image multispectrale

En fait, les capteurs sont sensibles à des longueurs d'onde comprises dans des bandes étroites (figure 2) et les données peuvent être entachées de variations sensibles liées à la couverture nuageuse, au taux d'humidité, à la saison (aspect des végétaux), à l'heure d'observation et autres facteurs incontrôlables.



**figure 2** : Réponse spectrale caractéristique de la végétation verte et localisation des canaux des satellites SPOT et LANDSAT TM sur le spectre électromagnétique. D'après [Caloz 87].

## 1.2. Classification

L'image fournie par LANDSAT-TM peut contenir  $(256)^7$  types de réponses radiométriques alors que seulement de quelques classes thématiques à quelques dizaines sont habituellement utilisées en cartographie. La classification consiste à réduire le nombre

de classes des pixels. On distingue deux grands types de méthodes : supervisées ou non supervisées.

### **1.2.1. Classification non supervisée**

On utilise ces méthodes pour répartir les pixels entre un nombre fixé de classes sans aucune référence préalable à la nature de ces classes. Celles-ci sont déterminées par l'algorithme sans l'aide de l'utilisateur, ce qui peut rendre l'interprétation des classes délicate. Les méthodes les plus répandues sont celles des k-moyennes et des nuées dynamiques.

### **1.2.2. Classification supervisée**

Avec ces méthodes, les pixels sont rattachés à des classes thématiques définies à l'avance. Le rattachement d'un pixel à une classe se fait par comparaison avec un certain nombre d'autres pixels, dits pixels de référence, qui définissent les classes. Il s'agit donc plutôt d'un classement que d'une classification.

La précision de ces classements repose beaucoup sur le choix des pixels de référence, donc sur les moyens dont on dispose pour repérer la nature des sols correspondants. La précision maximale est atteinte si l'on va repérer sur le terrain la nature de chaque pixel, mais alors on ne peut plus parler de classification automatique! Inversement, un nombre de pixels de référence trop réduit renforce la probabilité d'un mauvais classement du aux variations radiométriques mais aussi à un choix plus ou moins subjectif des classes thématiques (zones d'habitation dense/ zone d'habitation moyenne, forêt de conifères/ forêt de feuillus). La proportion de pixels pris comme référence est en général de quelques pour-cent.

L'interprétation des données multispectrales s'effectue le plus souvent par une méthode statistique de classement par maximum de vraisemblance, méthode généralement considérée comme la plus heureuse. Il est cependant difficile en pratique de dépasser une proportion de 65% de pixels bien classés [Porchier 93].

## **1.3. Post-traitements**

Après la classification, la carte a un aspect "poivre et sel" plus ou moins marqué : de nombreux pixels isolés affectent la lisibilité de la carte. Ce sont soit des pixels mal classés soit des pixels dont la classe thématique n'a pu être déterminée et que l'on met dans une classe à part. Certains de ces pixels correspondent à un mélange de plusieurs types de terrain, ou sont affectés par un élément temporaire non prévu (activité humaine) ou plus petit que la précision des capteurs ( par exemple les éléments linéaires comme les routes et les rivières).

C'est pourquoi des traitements s'avèrent nécessaires. Ils peuvent inclure un filtrage (pour éliminer les points isolés et ainsi améliorer la lisibilité), une fusion de classes considérées distinctement lors de la classification en raison de comportements trop distincts.

A ce stade, on peut ajouter les éléments linéaires existants (routes, chemins de fer...) et imaginaires (limites administratives) ou divers codes cartographiques.

#### **1.4. Utilisation des réseaux neuronaux**

Dans des configurations complexes fortement bruitées (milieux humides par exemple), les méthodes statistiques ne permettent pas d'obtenir des résultats satisfaisants parce que l'on y observe d'importantes perturbations radiométriques, donc des confusions thématiques [Girard Girard 89]. Dès la fin des années 80, quelques auteurs discutent de l'utilisation des réseaux neuronaux dans la classification de pixels satellitaires [Decatur 89] [Mac Clellan 89]. Les réseaux appliqués à la classification sont autant d'alternatives attrayantes des méthodes statistiques classiques [Gorman Sejnowski 88][Benediktsson et al. 90]. En effet, ils font mieux que les méthodes statistiques si les fonctions de distribution des entrées ne sont pas connues ou approchées (voir la démonstration dans [Decatur 89]) ou si le nombre d'exemples est faible [Benediktsson et al. 93].

Ils n'ont cependant pas fait l'unanimité pour deux raisons essentiellement. La première est que l'apprentissage est conventionnellement associé à un choix heuristique de paramètres ; si ces paramètres sont choisis de manière incorrecte, les performances du réseau décroissent rapidement. La deuxième est que les réseaux neuronaux sont gourmands en ressources de calcul et de stockage pendant l'apprentissage.

La plupart des travaux s'appuient sur l'utilisation de l'algorithme de rétro-propagation de l'erreur. Decatur compara les résultats obtenus avec d'autres réseaux avec ceux obtenus avec l'algorithme précédemment cité [Decatur 89]. Benediktsson a démontré par comparaison avec les méthodes statistiques classiques qu'un réseau à rétro-propagation à trois couches s'avérait être un très bon classifieur [Benediktsson et al. 90]. L'inconvénient majeur de cette méthode neuronale est la lenteur de sa phase d'apprentissage. Cependant, de nombreuses améliorations à l'algorithme de rétro-propagation ont été proposées qui, à notre connaissance, n'ont pas encore été essayées au problème de la classification de données satellitaires.

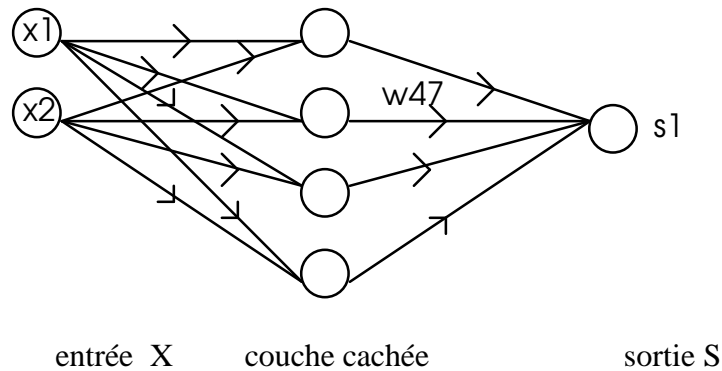
## **2. Présentation de la rétro-propagation**

Rappelons que, par analogie avec le système nerveux, un neurone formel est une cellule (un élément autonome) capable:

- de détecter les activités d'autres neurones;
- d'en calculer une somme pondérée;
- de comparer cette somme avec un seuil dit seuil d'activation;
- et, si ce seuil s'avère dépassé, de transmettre un signal dépendant des signaux reçus.

## 2.1. Perceptron multicouches

Dans ce document, on s'intéresse uniquement aux réseaux de neurones de type perceptron multicouches (PMC), appropriés à l'apprentissage supervisé. Chaque neurone  $j$  d'une couche autre que la première (dite couche d'entrée ou plus simplement entrée) reçoit une impulsion de la part de chaque neurone  $i$  de la couche précédente par le biais d'une connexion synaptique à laquelle on associe un coefficient (ou poids)  $w_{ij}$ . Le réseau comprend  $n$  neurones et  $N$  connexions.



**figure 3** : réseau à 3 couches (une couche cachée)

L'impulsion  $x_i$  donnée par un neurone  $i$  (son activité) dépend de la somme pondérée des activités que détecte ce neurone, notée  $net_i$ , de sa fonction d'activation  $f$  (souvent la même pour tous les neurones) et de son seuil d'activation  $\theta_i$  suivant la formule  $x_i = f(net_i)$  où  $net_i = \sum_j w_{ij} x_j - \theta_i$

D'un point de vue algorithmique, il est intéressant de considérer le seuil d'activation d'un neurone comme l'opposé du poids d'une connexion issue d'un neurone que l'on ajoute à la couche précédente et dont l'activité est fixée à 1. Cette astuce classique, adoptée par la suite, permet d'écrire:

$$net_i = \sum_j w_{ij} x_j$$

La base d'apprentissage est constituée de  $P$  exemples (entrée, sortie désirée  $d_p$ ) qu'il s'agit de faire apprendre au réseau. Les entrées des exemples sont successivement présentées au réseau et les sorties calculées par le réseau sont comparées aux sorties désirées. Les poids sont modifiés en fonction des mauvaises réactions du réseau en vue d'améliorer ses performances.

Les différentes méthodes présentées ici s'appuient toutes sur l'algorithme de rétro-propagation expliqué au chapitre suivant et reposent souvent sur des méthodes d'optimisation plus générales comme les gradients conjugués (chapitre 3.1.1.) ou des méthodes du second ordre dérivant de la méthode de Newton (chapitre 3.1.3.), mais elles sont toujours complétées par des astuces propres au type de problème concerné: réglage de

paramètres, choix de fonctions d'activation appropriées, centrage des entrées... Aucune méthode ne peut d'ailleurs prétendre être la meilleure dans tous les cas: les méthodes du second ordre permettent une diminution drastique du nombre d'itérations mais au prix d'une telle augmentation du nombre d'opérations par itération (et du nombre de variables à stocker) qu'elle semblent peu adaptées aux réseaux de très grande taille.

## 2.2. Principe de la rétro-propagation

Le but est de minimiser une fonction objective, dite erreur, mesurant l'écart entre les sorties effectives  $s$  et les sorties désirées  $d$ . L'erreur la plus couramment employée est l'erreur quadratique :

$$E_p = \frac{1}{2} \cdot \sum_k (d_k - s_k)^2 \text{ erreur sur l'exemple } p$$

$$E = \sum_{p=1}^P E_p \text{ erreur totale sur la base d'apprentissage}$$

Les poids sont ajustés au fur et à mesure des présentations d'exemples au réseau, proportionnellement à l'erreur qu'ils entraînent.

Le problème, pour un réseau à une ou plusieurs couches cachées, est d'évaluer l'erreur imputable au poids d'une connexion vers un neurone caché. La solution, proposée indépendamment par plusieurs auteurs ([Werbos 74], [Parker 85], [Rumelhart et al. 86], [Le Cun 87]) est de "rétro-propager" l'erreur proportionnellement aux poids. La rétro-propagation comprend donc deux phases :

- présentation d'un exemple et propagation des activations vers la sortie;
- calcul de l'erreur en sortie et propagation vers l'entrée.

La règle régissant la mise à jour des poids  $w_{ij}$  est la règle du delta généralisé:

$$\Delta w_{ij} = \eta \delta_j x_i \quad \text{avec } \delta_j = f'(\text{net}_j) \cdot (d_j - x_j) \text{ pour la sortie } (x_j = s_j)$$

$$\text{et } \delta_j = f'(\text{net}_j) \cdot \sum_k \delta_k w_{jk} \text{ pour les neurones cachés}$$

La puissance de cette règle provient du fait qu'elle équivaut à effectuer un algorithme de descente du gradient sur la fonction objective  $E$  dans l'espace des poids ([Rumelhart et al. 86]) :

*En effet, on peut montrer que  $\Delta w_{ij}$  est proportionnel à  $\partial E_p / \partial w_{ij}$ .*

*Pour cela, on utilise les propriétés des dérivées croisées :*

$$\left( \frac{\partial E_p}{\partial w_{ij}} \right) = \left( \frac{\partial E_p}{\partial x_j} \right) \cdot \left( \frac{\partial x_j}{\partial \text{net}_j} \right) \cdot \left( \frac{\partial \text{net}_j}{\partial w_{ij}} \right) = -\zeta_j \cdot \left( \frac{\partial \text{net}_j}{\partial w_{ij}} \right) \text{ en posant } \zeta_j = -\frac{\partial E_p}{\partial \text{net}_j}$$

La semi-linéarité des neurones donne :  $\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left( \sum_i w_{ij} x_i \right) = x_i$

De plus, pour une fonction d'activation  $f$  dérivable :  $\frac{\partial x_j}{\partial net_j} = \frac{\partial}{\partial net_j} (f(net_j)) = f'(net_j)$

Donc pour une erreur quadratique et un neurone de sortie,  $\frac{\partial Ep}{\partial x_j} = -(d_j - x_j)$

et pour un neurone  $j$  caché,  $\frac{\partial Ep}{\partial x_j} = \sum_k \frac{\partial Ep}{\partial net_k} \frac{\partial net_k}{\partial x_j} = \sum_k -\zeta_k w_{jk}$

On retrouve la règle du delta généralisé avec  $\delta = \zeta$  et donc  $\Delta w_{ij} = -\eta (\partial Ep / \partial w_{ij})$

L'algorithme de rétro-propagation peut prendre deux formes. La première, dite rétro-propagation continue ou stochastique (*on line*), consiste à modifier les poids après chaque présentation. Au contraire, la rétro-propagation périodique ou par lots (*batch backpropagation*) n'ajuste les poids qu'au bout d'un certain nombre de présentations (une époque), généralement égal au nombre d'exemples de la base d'apprentissage. Sous cette forme, on évite l'interférence entre les modifications successives des poids : en calculant le gradient sur la somme des erreurs obtenues pour chaque exemple du lot, on effectue une sorte de moyenne. Cependant, si la base d'apprentissage est très importante, la redondance des informations rend la procédure périodique moins efficace que la procédure continue.

La limite principale de la descente du gradient est sa lenteur. En effet, dans le cas où  $E$  est une forme quadratique ( $E(w) = 1/2 w^t H w + b^t w + c$ ), la descente du gradient  $\Delta w = -\eta dE/dw$  converge linéairement avec une constante de temps au mieux égale à  $\lambda_{\max}/4\lambda_{\min}$ , où  $\lambda_{\max}$  et  $\lambda_{\min}$  sont les plus grande et plus petites valeurs propres du hessien  $H$  de  $E$  (matrice des dérivées secondes). Accélérer la rétro-propagation est un problème important, et de nombreuses méthodes ont été proposées, empiriques ou empruntées à l'analyse numérique non linéaire.

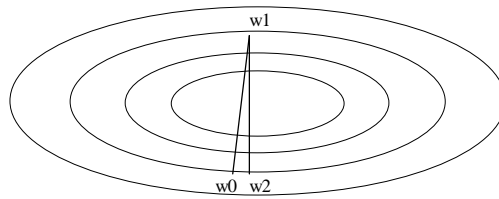
Un autre problème est celui des minima locaux : la convergence vers le minimum global n'est pas assurée. Ce défaut, inhérent à toute méthode à base de descente de gradient, est moins grave dans le cas d'une classification que lorsqu'il s'agit de résoudre des équations complexes (comme en robotique). L'important est en effet d'aboutir non pas à la solution globale, mais plutôt à une solution suffisamment bonne pour permettre une séparation correcte des différentes classes à reconnaître et ainsi obtenir un bon pourcentage de classification sur la base de test.

## 2.3. Possibilités d'accélération de l'apprentissage

### 2.3.1. Modification de la rétro-propagation

Le coefficient d'apprentissage  $\eta$  fixé permet de régler la rapidité de l'apprentissage mais limite la précision du résultat. En effet, près d'une solution  $w^*$ , le pas fixe force la suite des vecteurs  $w$  à osciller autour de cette solution (figure ?). Il y a donc un compromis à trouver en fonction de l'erreur acceptable, c'est à dire en fonction du problème. Pour la

classification, ce compromis dépend du choix des classes et ne peut être déterminé une fois pour toutes.



**figure 4** : oscillations dues à un pas trop grand

Une approche pour accélérer la rétro-propagation est d'adapter la valeur de  $\eta$  en cours d'apprentissage en fonction de l'erreur. Cette approche poussée à l'extrême conduit à la résolution du sous-problème de la minimisation de l'erreur suivant la direction du gradient.

Une autre approche consiste à choisir une meilleure direction que celle du gradient en tenant compte de plus d'information. Cette approche, sous-jacente à l'utilisation d'un terme d'inertie, conduit aux méthodes du second ordre: gradients conjugués, méthodes quasi-Newton ou de sécantes.

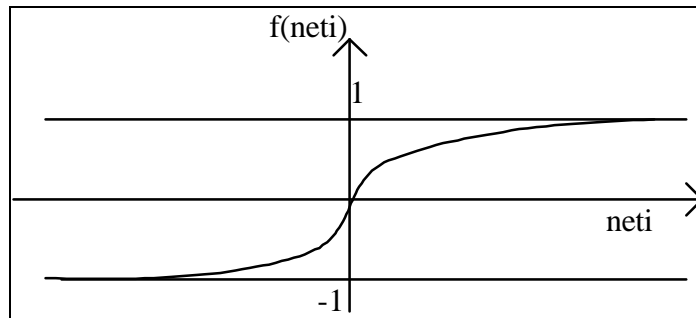
On verra que les nombreuses méthodes ont des avantages et des inconvénients et que les méthodes demandant le moins d'itérations nécessitent en contrepartie beaucoup plus de calculs. Cela explique que le choix des algorithmes, quoique d'une importance primordiale, dépend étroitement de l'application concernée et demeure principalement expérimental.

### 2.3.2. Fonctions d'activation

Les qualités recherchées pour une fonction d'activation sont la monotonie, la continuité, un ensemble de valeurs bornées (de -1 à 1) et la dérivabilité (pour la rétro-propagation). La fonction d'activation choisie par la plupart des chercheurs est une fonction sigmoïde, qui, outre les caractéristiques précédentes, présente l'avantage d'avoir une zone de comportement approximativement linéaire :

$$f(\text{net}_i) = 1 / [ 1 + \exp(- \text{net}_i) ] \quad (\text{valeurs dans } [0,1] )$$

ou  $f(\text{net}_i) = [ 1 - \exp(-\text{net}_i) ] / [ 1 + \exp(-\text{net}_i) ] \quad (\text{valeurs dans } [-1,1] )$



**figure 5** : fonction d'activation sigmoïdale

D'autres fonctions sont possibles:

- fonction sinusoïdale  $f(\text{net}_i) = \sin(\pi \cdot \text{net}_i)$
- fonction gaussienne  $f(\text{net}_i) = \exp(-\text{net}_i^2)$

Hara et Nakayama ont testé ces trois types de fonction sur un réseau recevant en entrée des signaux sinusoïdaux à fréquences multiples bruités et ont constaté que la fonction sinusoïdale donnait de meilleurs résultats, montrant ainsi qu'il n'y a pas de règle absolue [Hara et Nakayama 94].

Le danger avec toutes les fonctions d'activation est qu'elles ont une zone de saturation (due au fait qu'elles sont bornées) dans laquelle la dérivée est quasiment nulle, ce qui peut ralentir énormément la convergence. En effet, la variation des poids est proportionnelle à cette dérivée, de sorte que dès que la somme des entrées d'un neurone est importante, les poids des connexions en entrée de ce neurone sont peu ajustées ou pas du tout (avec l'arrondi). Ces poids sont alors pratiquement constants alors qu'il faudrait les modifier pour sortir de la zone de saturation. Pour y remédier, Fahlman préconise l'ajout d'une constante (il prend 0.1) à la dérivée de la sigmoïde et observe une accélération de 25% sur des problèmes de reconnaissance de la parité d'un entier entré sous forme binaire [Fahlman 88].

### 2.3.3. Codage des entrées et des sorties

Le codage de l'information traitée a également une forte influence. On s'en convainc aisément en examinant le cas où les données s'étalent sur un intervalle de grande taille (par exemple  $[0,255]$ ). Dans ce cas, les fonctions d'activation des neurones sont souvent dans leur zone de saturation et l'apprentissage s'en trouve considérablement ralenti. Il convient donc de ramener les entrées dans un intervalle plus modeste (comme  $[0,1]$ ).

Il y a aussi le choix du type d'information que l'on veut traiter. Dans le cas de la classification satellitaire, on peut se limiter aux canaux du pixel à classer, mais on peut également prendre en compte les données relatives aux pixels voisins. On traite ainsi une "fenêtre" de pixels ( $3 \times 3, 5 \times 5 \dots$ ). On est alors en droit d'atteindre des résultats plus précis, mais cela complique le choix des pixels de référence et l'apprentissage est plus lent car le réseau doit être de plus grande taille.

Une fois déterminées les informations à traiter, les possibilités les plus simples sont:

- pas de codage : un canal = une entrée ;
- un codage binaire : un canal = 8 entrées (256 valeurs).

Ce dernier codage n'est pas très efficace pour une classification, car des valeurs radiométriques très proches peuvent donner des entrées très différentes. [Benediktsson et al. 90] ont prouvé expérimentalement la supériorité du codage de Gray, avec lequel deux entiers adjacents ne diffèrent que par un ou deux bits, sur le codage binaire. Ce codage permet de transformer un mot  $(b_1, b_2, \dots, b_n)$  à  $n$  bits en un mot  $(g_1, g_2, \dots, g_n)$  par addition modulo 2 :

$$g_1 = b_1$$

$$g_k = b_k \oplus b_{k-1} \text{ avec } 2 \leq k \leq n$$

Par contre, il ne s'est avéré plus performant qu'un codage continu que dans le cas d'un problème à faible dimensionnalité, plus précisément dans le cas d'un problème à moins de 10 variables [Benediktsson et al. 93].

D'autres codages sont envisageables (figure 6).

Le codage grossier est un codage intermédiaire entre un codage continu et un codage discret. On choisit un nombre  $n$  de neurones pour coder un nombre  $m$  dans un intervalle  $[A, B]$ . Chaque neurone  $i$  est donc caractérisé par un emplacement  $z_i$  dans l'intervalle précédent. La sortie de ce neurone est calculée à l'aide d'une fonction de répartition gaussienne :

$$o_i(m) = e^{-\frac{(m-z_i)^2}{\sigma^2}}$$

Ce type de codage a déjà été utilisé dans [Bischof et al. 92] où les auteurs choisissent  $n$  à 13,  $A$  à 0,  $B$  à 255 et  $\sigma$  à 23. Les auteurs ne donnent hélas aucun élément de comparaison avec d'autres types de codage.

type de codage	réponse d'un neurone	codage de 2 dans l'intervalle [0,7]	codage de 5 dans l'intervalle [0,7]
codage "grossier"			
codage discret simple			
thermomètre discret			
thermomètre continu			
interpolation			

figure 6 : Exemples de codages . D'après [Hancock 88]

Hancock a observé la supériorité des codages continus sur les codages discrets pour des problèmes de résolution d'équation [Hancock 88], mais rien n'est prouvé pour la classification.

#### **2.3.4. Topologie et généralisation**

Il ne faut pas perdre de vue que le but d'un bon apprentissage n'est pas d'obtenir un excellent résultat sur la base d'apprentissage, mais de permettre au réseau de traiter correctement l'ensemble de la base de test. Un entraînement trop intensif du réseau tend à lui faire perdre ses capacités de généralisation, car il finit par se spécialiser sur les données de la base d'apprentissage : il les apprend par coeur.

Le problème de la généralisation est fortement dépendant de la topologie du réseau. Lorsqu'on augmente le nombre de neurones cachés, le nombre d'itérations nécessaire pour obtenir une précision donnée diminue, mais la généralisation se fait moins bien. Cela tient au fait que le réseau est surdimensionné : les paramètres (les poids) en surnombre évoluent plus facilement vers une des multiples solutions, mais le réseau apprend moins bien les caractéristiques des données qui lui permettraient de généraliser correctement.

Le choix du nombre de neurones cachés est donc d'une grande importance. Plusieurs auteurs ont proposé des procédures pour préserver les capacité de généralisation, soit en repérant le moment où l'apprentissage doit être arrêté, soit en détruisant régulièrement les connexions jugées les moins utiles (méthodes Optimal Brain Damage de [Le Cun Denker Solla 90] et Optimal Brain Surgeon de [Hassibi Stork 93], soit encore en modifiant la fonction de l'erreur pour qu'elle tende à faire baisser les poids ([Hanson Pratt 89]- ce qui revient à détruire les plus insignifiants).

Ces procédures sont sans doute trop lourdes à mettre en place en pratique dans le cas de la classification satellitaire, car l'apprentissage doit être refait pour chaque nouvelle image ou pour tout changement des classes thématiques. Cependant, il peut être intéressant d'appliquer un de ces algorithmes un nombre limité de fois, comme aide à la décision d'une topologie efficace qui serait ensuite prise pour tous les apprentissages.

**Méthodes d'accélération**  
**de la rétro-propagation**

### 3. Accélérer la rétro-propagation par la voie algorithmique

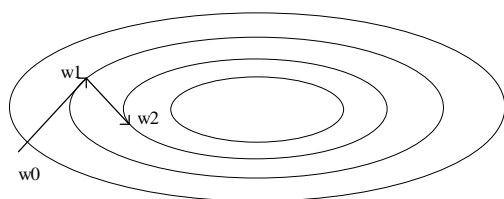
Ce chapitre décrit les différentes techniques algorithmiques qui ont été utilisées avec un certain succès ou promettent de bons résultats. Des comparaisons de certaines de ces méthodes ont parfois été réalisées, concluant la plupart du temps à la supériorité de la méthode proposée par l'auteur de la comparaison. Mais ces études ne donnent en fait qu'une indication, car elles portent souvent sur des problèmes très particuliers (comme les problèmes de parité) ou de nature fondamentalement différente de celle du problème de classification supervisée (résolution d'équations, modélisation, prédiction). Il faut donc considérer ces méthodes d'un oeil neutre et se souvenir qu'elles influenceront probablement des choix de codage des entrées et des sorties et des topologies différentes.

#### 3.1. Méthodes numériques

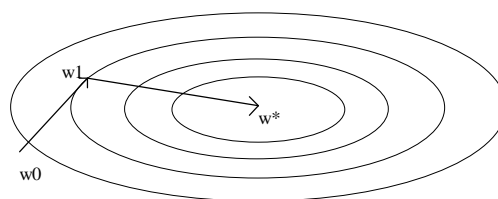
Les progrès les plus importants réalisés ces dernières années autour de la rétro-propagation ont été réalisés grâce à l'utilisation d'algorithmes issus de l'optimisation non linéaire. L'algorithme de la rétro-propagation lui-même était déjà proposé en 1969 dans le domaine du contrôle optimal avant d'être redécouvert plusieurs fois indépendamment [Bryson et Ho 69]. Que de temps perdu !

##### 3.1.1. Algorithme des gradients conjugués

Le principe est de prendre comme directions d'ajustement des poids des directions qui tiennent compte au mieux de l'information contenue dans les directions précédentes. L'idéal serait que  $N$  directions successives forment une base de l'espace des poids de dimension  $N$ . C'est ce que tente de réaliser l'algorithme des gradients conjugués.



**figure 6 :** algorithme de descente du gradient avec minimisation sur la direction du gradient (fonction quadratique)



**figure 7 :** algorithme des gradients conjugués :  $N$  itérations

L'algorithme des gradients conjugués recouvre plusieurs algorithmes dont le premier a été appliqué au problème général de minimisation sans contraintes par Fletcher et Reeves

en 1964 [Fletcher Reeves 84]. Leur algorithme pour minimiser une fonction  $F : x \in \mathfrak{R}^N \rightarrow \mathfrak{R}$  est le suivant :

**début**

$k := 0;$

$g_0 := \nabla F(x_0);$

$d_0 := -g_0;$

**tant que** ( $\|g_k\|^2 > \varepsilon$  ou  $k < K$ ) //  $K$  et  $\varepsilon$  sont fixés

Calculer le minimum  $\lambda_k$  de la fonction réelle  $\lambda \rightarrow F(x_k + \lambda d_k)$

$x_{k+1} := x_k + \lambda_k d_k;$

$g_{k+1} := \nabla F(x_{k+1});$

$\beta_{k+1} := \|g_{k+1}\|^2 / \|g_k\|^2;$  // formule de Fletcher-Reeves

$d_{k+1} := -g_{k+1} + \beta_{k+1} d_k;$

$k := k+1;$

**finq;**

**fin.**

Pour une fonction quadratique, la solution est obtenue en au plus  $n$  itérations en notant  $n$  la dimension du problème. En effet, si  $F(x) = 1/2 x^t H x + b^t x + c$ , alors on montre que les directions de recherche  $d_k$  sont mutuellement  $H$ -conjuguées, ce qui s'écrit  $d_i^t H d_j = 0$  pour  $i \neq j$  et  $i, j \leq N$ . Il s'ensuit que  $x_{k+1}$  donne le minimum de  $F(x)$  dans l'espace affine de dimension  $k$  engendré par  $x_0$  et  $d_1, d_2, \dots, d_k$ .

En fait, pour une fonction quadratique,  $\beta_k = \frac{g_k^t H d_{k-1}}{d_{k-1}^t H d_{k-1}}$  et  $\lambda_k = -\frac{d_k^t g_k}{d_k^t H d_k}$ .

Pour une fonction quelconque, on ne souhaite pas avoir à effectuer le calcul du hessien (en  $O(N^2)$ ). C'est pourquoi on remplace l'expression précédente de  $\beta_k$  par une expression approchée et on estime  $\lambda_k$  en minimisant  $F$  sur la direction  $d_k$ . L'important est de choisir une expression de  $\beta_k$  qui assure autant que possible la  $H$ -conjugaison des directions successives, ce qui donne lieu à plusieurs formules dont aucune n'est jugée meilleure dans tous les cas. Le choix dépend donc de la fonction  $F$ . La formule de Polak-Ribiere est cependant souvent préférée à celle de Fletcher-Reeves.

Formule de Polak-Ribière :  $\beta_k = g_k^t (g_k - g_{k-1}) / \|g_{k-1}\|^2$  [Polak Ribiere 69]

Formule de Hestenes-Stiefel :  $\beta_k = g_k^t (g_k - g_{k-1}) / d_{k-1}^t (g_k - g_{k-1})$  [Hestenes Stiefel 52]

La stabilité numérique de ces algorithmes est liée à la précision de l'évaluation de  $\lambda_k$  obtenu par minimisation de  $F$  sur une droite (méthodes de dichotomie, méthode de la section dorée, méthodes itératives, de Newton, des sécantes, interpolation polynomiale, ...). Si cette évaluation est trop peu précise, les méthodes précédentes peuvent mener à prendre des directions qui ne sont pas des directions de "descente". Shanno propose une méthode assurant une direction de descente [Shanno 78]:

$$d_{k+1} = -g_{k+1} - \lambda_k \left[ \left( 1 + \frac{\|g_{k+1} - g_k\|^2}{\lambda_k d_k^t (g_{k+1} - g_k)} \right) \frac{d_k^t g_k}{d_k^t (g_{k+1} - g_k)} - \frac{(g_{k+1} - g_k)^t g_k}{g_{k+1} - g_k} \right] d_k + \frac{d_k^t g_k}{d_k^t (g_{k+1} - g_k)} (g_{k+1} - g_k)$$

### 3.1.2. Procédures de reprise de l'algorithme

Comme la H-conjugaison des directions se détériore au fur et à mesure du déroulement de l'algorithme, le comportement de celui-ci tend à devenir linéaire. C'est pourquoi tous les auteurs conseillent de reprendre l'algorithme des gradients conjugués au plus toutes les N ou N+1 itérations.

La procédure la plus simple consiste à réinitialiser le vecteur direction à l'opposé du gradient toutes les N itérations :  $d_{kN} = -g_{kN}$ . C'est ce qu'on appelle une reprise (*restart*). L'inconvénient de cette formule est "l'oubli" des informations contenues dans  $d_{k-1}$ .

Pour pallier ce défaut, Beale s'intéresse au cas quadratique et trouve une formule pour rendre mutuellement conjuguées les directions  $d_t, d_{t+1}, d_{t+2}, \dots, d_k$ , où  $d_t$  est une direction de reprise:  $d_k = -g_k + \beta_k d_{k-1} + \gamma_k d_t$ .  $\beta_k$  est toujours calculé pour rendre  $d_k$  et  $d_{k-1}$  conjugués et  $\gamma_k$  permet la conjugaison avec  $d_t$ . Il obtient les expressions suivantes :

$$\beta_k = g_k^t (g_k - g_{k-1}) / d_{k-1}^t (g_k - g_{k-1})$$

$$\gamma_k = g_k^t (g_{t+1} - g_t) / d_t^t (g_{t+1} - g_t) \quad \text{sauf pour } k=t+1 \text{ où } \gamma_k=0.$$

Powell propose un algorithme directement inspiré de la méthode de Beale et appliqué à une fonction quelconque, en prenant comme critère de reprise  $g_k^t g_{k+1} / \|g_{k+1}\|^2 \gg 0$  [Powell 77]. La reprise a donc lieu lorsque deux gradients successifs sont jugés insuffisamment proches de l'orthogonalité. Ce critère permet d'éviter que  $g_k$  ne tende vers une limite non nulle. Il y ajoute un deuxième critère pour que  $d_k$  soit toujours une assez bonne direction de descente : l'algorithme est redémarré si  $(d_k^t g_k)$  n'est pas suffisamment proche de  $(-\|g_k\|^2)$ . En effet, l'égalité entre ces deux termes a lieu lorsque la conjugaison est numériquement respectée.

Cet algorithme est le suivant :

$x_0$  est donné.

**début**

$g_0 := \nabla F(x_0)$ ;

$d_0 := -g_0$ ;

Calculer le minimum  $\lambda_0$  de la fonction réelle  $\lambda \rightarrow F(x_0 + \lambda d_0)$

$x_1 := x_0 + \lambda_0 d_0$  ;

$t := 0$ ;  $k := 1$ ;

**tant que**  $(\|g_k\|^2 > \varepsilon \text{ ou } k < K)$  // K et  $\varepsilon$  sont fixés

$g_k := \nabla F(x_k)$ ;

**si**  $(|g_{k-1}^t g_k| > 0.2 * \|g_{k-1}\|^2 \text{ ou } (k-t \geq n))$

**alors**

//reprise

```

t := k-1;
γk := 0;
sinon
γk := gkt(gt+1-gt)/dtt(gt+1-gt);
fsi;
βk := gkt(gk-gk-1)/dk-1t(gk-gk-1)
dk := -gk + βkdk-1 + γkdt;
si ((dktgk < -1.2*||gk||2) ou (dktgk > -0.8*||gk||2))
alors //reprise
t := k-1;
dk := -gk + βkdk-1;
fsi;

Calculer le minimum λk de la fonction réelle λ → F(xk + λdk)
xk+1 := xk + λkdk;
k := k+1;
fintq
fin.

```

### 3.1.3. Méthode de Newton

Elle provient d'une approximation quadratique de la fonction F à minimiser :

$$F(x_0 + \Delta x) \approx F(x_0) + g(x)^t \cdot \Delta x + \frac{1}{2} \Delta x^t \cdot H(x) \cdot \Delta x$$

En choisissant bien  $\Delta x$ ,  $x + \Delta x$  est un minimum :  $\nabla F(x_0 + \Delta x) = 0$ . On obtient alors  $0 = g + H \cdot \Delta x$ .

Si H est inversible,  $\Delta x = -H^{-1} \cdot g$ .

La solution est trouvée en une seule itération lorsque F est quadratique, mais l'inversion de H demande un nombre d'opérations en  $O(N^3)$  (les meilleures méthodes d'inversion en  $O(N^{\log_2 7})$  s'inspirent des méthodes de dichotomie [Press et al. 88]), ce qui peut s'avérer excessivement élevé pour des valeurs importantes de N. La nécessité de stocker  $N^2$  variables pour la matrice est également une contrainte de taille.

Un autre problème important est naturellement l'inversibilité de H et son comportement lorsque F n'est pas quadratique. Dans le cas général, si on est assez près d'un minimum et si H est symétrique définie positive, la convergence est q-quadratique :

$$|F(x_{k+1}) - F(x_{\min})| \leq \text{Constante} \times |F(x_k) - F(x_{\min})|^2 \quad [\text{Dennis et Schnabel 83}]$$

Cependant, avec le cas du perceptron multicouches, il arrive fréquemment que H ne soit pas définie positive, auquel cas le comportement de l'algorithme peut se dégrader considérablement. Cela se produit souvent dans des régions où il y a saturation de la

fonction d'activation sigmoïde et où les dérivées premières et secondes de l'erreur sont très faibles et donc particulièrement sensibles à la précision des calculs.

### 3.1.4. Méthodes de sécantes

Pour réduire la complexité élevée qu'implique l'inversion, on peut réaliser une approximation itérative de l'inverse du hessien (notée A) qui demande un nombre d'opérations en  $O(N^2)$  pour chaque itération. C'est le principe des méthodes de sécantes dont la démarche se résume en trois points:

- déterminer une direction de recherche  $d = -Ag$ ;
- minimiser F sur d;
- mettre à jour la matrice A.

Les différentes formules envisageables pour la mise à jour de A doivent toutes remplir la condition:

$$\Delta x = Ay \quad \text{avec} \quad y = g(x+\Delta x) - g(x) \quad (\text{formule de la sécante})$$

La méthode DFP - pour Davidson/Fletcher/Powell [Davidson 59] [Fletcher Powell 63] - utilise la formule suivante :

$$A = A + \frac{\Delta x \Delta x^t}{\Delta x^t y} - \frac{A y y^t A}{y^t A y}$$

La méthode BFGS - Broyden Fletcher Goldfarb Shanno [Broyden 70] [Fletcher 70] - utilise une autre formule:

$$A = A + \left(1 + \frac{y^t A y}{\Delta x^t y}\right) \frac{\Delta x \Delta x^t}{\Delta x^t y} - \frac{\Delta x y^t A + A y \Delta x^t}{\Delta x^t y}$$

Remarquons que ces deux transformations laissent A symétrique et définie positive, ce qui rend l'algorithme numériquement stable : en prenant une matrice initiale simple et définie positive (par exemple la matrice identité I), on est assuré d'arriver à un minimum. Fletcher et Powell [Fletcher Powell 63] ont démontré la convergence en N itérations dans le cas quadratique.

Ces deux formules sont duales: la matrice  $H=A^{-1}$  de la méthode BFGS vérifie la formule DFP en échangeant y et  $\Delta x$ . La formule BFGS est souvent préférée à la formule DFP car elle se comporte mieux lorsque la minimisation sur une direction  $d=-Ag$  n'est pas très précise.

La complexité de ces algorithmes est encore prohibitive par rapport à la rétro-propagation simple (  $O(N^2)$  contre  $O(N)$ ), de sorte que ces méthodes seront appropriées pour des réseaux de taille petite à moyenne (pas beaucoup plus de 100 connections).

## 3.2. Méthodes connexionnistes

### 3.2.1. Terme d'inertie

Une des améliorations les plus anciennes est l'ajout d'un terme  $\alpha$  de momentum (ou d'inertie) :

$$\Delta w(t) = \alpha \Delta w(t-1) - \eta (dE/dw)(t) \quad [\text{Rumelhart 86}]$$

On tient ainsi compte des modifications précédentes de façon exponentielle.

En effet, une suite  $(x_n)_{n \geq 0}$  définie par  $x_{n+1} = a_n x_n + b_n$  s'exprime de façon équivalente par :

$$x_{n+1} = \left( \prod_{i=0}^n a_i \right) x_0 + \sum_{i=0}^n \left( \prod_{j=i+1}^n a_j \right) b_i + b_n$$

En appliquant cette formule à  $\Delta w_n = \Delta w(t)$  puis à  $(w_n)$ , on obtient :

$$\Delta w_{n+1} = \eta \sum_{i=0}^n \alpha^{n-i} \left( -\frac{dE}{dw} \right) \text{ et } w_{n+1} = w_0 + \eta \sum_{i=0}^n \frac{1 - \alpha^{n-i+1}}{1 - \alpha} \left( -\frac{dE}{dw} \right)$$

$$\text{Lorsque le gradient est constant, } \Delta w_{n+1} = \eta \frac{1 - \alpha^{n+1}}{1 - \alpha} \left( -\frac{dE}{dw} \right) \xrightarrow{n \rightarrow \infty} \frac{\eta}{1 - \alpha} \left( -\frac{dE}{dw} \right).$$

Le terme de momentum permet donc d'accélérer la convergence d'un facteur  $1/(1-\alpha)$  lorsque la direction prise par le gradient change peu, et d'éviter les instabilités dues à des oscillations en tenant compte des mouvements précédents. L'algorithme se comporte ainsi comme une approximation sommaire des gradients conjugués. Shynk et Roy ont montré que ce terme de momentum n'améliore pas la convergence de la rétro-propagation continue ([Shink Roy 88]). Dans le cas périodique par contre, la convergence est améliorée, la constante de temps passant au mieux à  $(1/4)\sqrt{\lambda_{\max}/\lambda_{\min}}$  [Tugay Tanik 89].

Par ailleurs, Pearlmutter a montré dans [Pearlmutter 92] que l'ajout d'un terme de momentum du deuxième ordre ( $\Delta w(t) = \alpha \Delta w(t-1) + \beta \Delta w(t-2) - \eta (dE/dw)(t)$ ) ne donne qu'une amélioration minime, l'écart avec le momentum du premier ordre tendant vers 0 lorsque  $\lambda_{\min}/\lambda_{\max}$  tend vers 0.

### 3.2.2. Adaptation du coefficient d'apprentissage

La méthode du momentum permet certes une convergence plus rapide, mais elle accroît le problème du réglage des paramètres (deux paramètres au lieu d'un). Ce réglage est important car la convergence en dépend étroitement, et il est fait empiriquement. Or le réglage de paramètres par l'utilisateur est un inconvénient majeur pour bon nombre

d'applications (et notamment pour le temps réel). Pour résoudre ce problème, plusieurs approches ont été utilisées.

L'idée commune à ces approches est que le coefficient d'apprentissage doit pouvoir se rapprocher de 0 au fur et à mesure des progrès de l'apprentissage mais qu'il doit être relativement important lorsqu'on est loin de la solution afin que la convergence soit rapide. Cette idée a été justifiée par Luo dans [Luo 91] où il démontre la convergence de l'algorithme des moindres carrés pour un perceptron multicouches linéaire avec une suite de coefficients d'apprentissage tendant vers 0 ( $\sum_{k=1}^{\infty} \eta_k^2 < \infty$ ) mais pas trop vite ( $\sum_{k=1}^{\infty} \eta_k = \infty$ ).

### 3.2.2.1. Méthode de [Vogl et al. 88]

Cette méthode empirique utilise la rétro-propagation périodique avec momentum mais avec un coefficient d'apprentissage variable  $\eta$ . Après une période, si l'erreur totale diminue, on multiplie  $\eta$  par une constante  $\phi$  supérieure à 1. Si par contre l'erreur totale augmente de plus de quelques pour-cent (1 à 5), on multiplie  $\eta$  par une constante  $\beta$  inférieure à 1 et on omet le terme de momentum pour la modification des poids (car la direction précédente n'est pas intéressante).

### 3.2.2.2. Vecteur d'apprentissage

Jacob propose d'ajuster chaque poids  $w_{ij}$  avec un coefficient d'apprentissage  $\eta_{ij} : \Delta w_{ij}(t) = -\eta_{ij} (dE/dw)(t)$  ([Jacobs 88]).

Jacob montre que pour une erreur quadratique,  $\frac{\partial E}{\partial \eta_{ij}}(t) = -\frac{\partial E}{\partial w_{ij}}(t) \cdot \frac{\partial E}{\partial w_{ij}}(t-1)$ .

Il en tire la règle delta - delta :  $\Delta \eta_{ij} = \gamma \frac{\partial E}{\partial w_{ij}}(t) \cdot \frac{\partial E}{\partial w_{ij}}(t-1)$ . Elle est jugée peu intéressante

du fait de la petitesse de  $\Delta \eta_{ij}$  près d'un optimum de E, mais elle mène à la règle delta barre delta mise au point par Jacob avec l'aide de Sutton :

$$\Delta \eta_{ij}(t) = \begin{cases} K & \text{si } \bar{\delta}_{ij}(t-1)\delta_{ij}(t) > 0 \\ -\Phi \eta_{ij}(t) & \text{si } \bar{\delta}_{ij}(t-1)\delta_{ij}(t) < 0 \\ 0 & \text{sinon} \end{cases}$$

$$\text{en notant } \delta_{ij}(t) = (dE/dw_{ij})(t) \\ \text{et } \bar{\delta}_{ij}(t) = (1-\theta)\delta_{ij}(t) + \theta\bar{\delta}_{ij}(t-1), 0 < \theta < 1$$

Un coefficient d'apprentissage est ainsi augmenté si la dérivée de l'erreur par rapport au poids correspondant est du même signe que la moyenne exponentielle des dérivées précédentes.

### 3.2.2.3. Adaptation du terme d'inertie

Dans [Chan et Fallside 87], le coefficient d'apprentissage et le terme d'inertie sont ajustés en prenant en compte l'angle  $\theta(t)$  entre le pas précédent et le gradient et en évitant la prédominance du terme d'inertie:

$$\Delta w(t) = \eta(t) (-\nabla E(t) + \lambda(t)\Delta w(t-1) )$$

$$\begin{aligned} \text{avec } \lambda(t) &= \lambda(0) \cdot \|\nabla E(t)\| / \|\Delta w(t-1)\| \\ \eta(t) &= \eta(t-1) (1 + \frac{1}{2} \cos \theta(t) ) \\ 0 < \lambda(0) < 1 \end{aligned}$$

On comprend aisément comment ces formules affectent la recherche de la solution. Lorsque la direction empruntée est colinéaire et de même sens que le nouveau gradient, le coefficient d'apprentissage est augmenté de moitié (on est dans la bonne direction mais vraisemblablement encore loin). Il est au contraire divisé par deux lorsque le sens est opposé, pour tenir compte du fait qu'on a "dépassé" la solution à cause d'un pas trop grand, et ainsi éviter des oscillations.

### 3.2.2.4. Méthode stochastique de Barnard

Dans [Barnard 92], un algorithme stochastique est proposé pour estimer le coefficient d'apprentissage  $\eta$  servant à ajuster tous les poids par  $w \leftarrow w - \eta(dE/dw)$ , avec une complexité et un nombre de variables en  $O(N)$ .

Partant de l'hypothèse qu'il existe une grandeur  $L$  caractéristique décroissant faiblement qui représente la distance parcourue à chaque itération dans l'espace des poids, on se propose d'appliquer la rétro-propagation continue en marquant une pause à intervalles réguliers pour calculer  $\eta$  à partir de  $L$ .

Pour estimer  $L$ , on prend un certain nombre d'exemples, on prend comme fonction objective  $E$  la somme des fonctions objectives individuelles, et on minimise cette fonction sur la direction donnée par le gradient.  $L$  est alors la distance parcourue dans l'espace des poids pour cette minimisation unidimensionnelle (c'est la norme du  $\Delta w$  qui serait obtenu par rétro-propagation périodique sur ces exemples).  $l$  est la longueur moyenne des gradients des erreurs de chaque exemple à la position courante de l'espace des poids.  $\eta$  est alors calculé suivant la formule :  $\eta = L / l$ .

Barnard estime  $L$  et  $l$  sur tous les exemples de la base et met à jour  $\eta$  tous les 20 passages (c'est un compromis déterminé expérimentalement). La méthode devient moins performante lorsqu'on est très proche d'une solution; il convient alors d'utiliser une méthode qui est très efficace près d'une solution, comme les gradients conjugués ou l'algorithme BFGS. Barnard annonce des résultats supérieurs à ces deux dernières méthodes.

### 3.2.2.5. Estimation de la valeur propre principale du hessien

La méthode de [Le Cun & al. 93], applicable à la rétro-propagation continue sur de larges réseaux, permet de choisir le meilleur pas pour une descente du gradient en estimant la valeur propre principale  $\lambda_{\max}$  et un vecteur propre associé, sans pour autant devoir calculer le hessien H.

Le principe est d'utiliser la méthode des puissances pour estimer  $\lambda_{\max}$ , en itérant  $\Psi \leftarrow H\Psi/\|\Psi\|$  à partir d'un vecteur  $\Psi$  non orthogonal à l'espace propre associé à  $\lambda_{\max}$  :  $\Psi$  converge alors vers un vecteur colinéaire à  $V_{\max}$  et de norme  $|\lambda_{\max}|$ .

Pour ne pas avoir à calculer H, on utilise la formule de Taylor en supposant E localement quadratique :  $\Psi \leftarrow (\nabla E(w + \alpha\Psi/\|\Psi\|) - \nabla E(w))/\alpha$  avec  $\alpha \ll 1$ .

Une itération demande deux propagations et deux rétro-propagations et la méthode des puissances converge très vite (une dizaine d'itérations suffisent souvent), mais cela fait beaucoup trop pour un traitement par lot, où chaque rétro-propagation doit se faire sur tous les exemples de la base d'apprentissage. La méthode est donc adaptée à la rétro-propagation continue, en remplaçant l'effet de moyenne sur les exemples par une moyenne "sur le tas" (*running average*):

$$\Psi \leftarrow (1-\gamma).\Psi + (\gamma/\alpha).(\nabla E(w + \alpha.\Psi/\|\Psi\|) - \nabla E(w))$$

De façon cocasse, on retrouve le problème du choix de la valeur d'un paramètre. Cependant, une valeur acceptable de  $\eta$  est trouvée en 100 à 200 itérations de cette dernière formule en se contentant de diminuer  $\gamma$  par pallier à partir de la valeur 0.1, ce qui semble un résultat très intéressant..

### 3.2.2.6. Approximation diagonale [Le Cun 88]

L'approximation diagonale du hessien a été utilisée dès 1987 par Le Cun [Le Cun 87] pour améliorer la rétro-propagation continue en conservant une complexité en  $O(N)$  et en stockant le hessien diagonal localement (c'est à dire chaque élément diagonal avec le poids correspondant).

L'approximation permet d'inverser aisément le hessien:  $\Delta w_{ij} = - \frac{\frac{\partial E}{\partial w_{ij}}}{\left| \frac{\partial^2 E}{\partial w_{ij}^2} \right|}$ .

Les termes non diagonaux étant ignorés, on peut poser  $\frac{\partial^2 E}{\partial w_{ij}^2} = \frac{\partial^2 E}{\partial \text{net}_i^2} f(\text{net}_i)^2$

$$\text{avec } \frac{\partial^2 E}{\partial \text{net}_i^2} = f'(\text{net}_i)^2 \sum_k w_{ki}^2 \frac{\partial^2 E}{\partial \text{net}_k^2} + f''(\text{net}_i) \sum_k w_{ki} \frac{\partial E}{\partial \text{net}_k}$$

La formule d'ajustement des poids pose problème pour les régions à faible courbure (plateaux, ravins, inflexions), c'est pourquoi on rajoute au dénominateur une constante  $\mu > 0$ .

$$\Delta w_{ij} = - \frac{\frac{\partial E}{\partial w_{ij}}}{\left| \frac{\partial^2 E}{\partial w_{ij}^2} \right| + \mu}$$

Cet algorithme adopte donc en fonction de la région parcourue un comportement qui se situe entre les deux extrêmes suivants:

- une descente du gradient dans les régions à faible courbure (typiquement loin d'une solution);
- un algorithme quasi-Newton dans les régions à courbure importante, où le modèle quadratique s'applique bien (région de confiance).

Cette méthode donne des résultats seulement un peu meilleures que la rétro-propagation avec inertie, ce que l'on explique par le fait que les termes non diagonaux du hessien ne sont en fait pas négligeables, surtout si le nombre de paramètres est important.

### 3.2.2.7. Algorithme quickprop [Fahlman 88]

L'algorithme quickprop est une sorte d'algorithme de sécantes appliqué localement au niveau de chaque coefficient synaptique. Il repose sur deux hypothèses osées:

- l'erreur en fonction de chaque poids est une parabole convexe;
- les changements de pente ne sont pas affectés par les variations des autres poids.

$$\text{La formule est } \Delta w_{ij}(t) = \frac{\frac{\partial E}{\partial w_{ij}}(t)}{\frac{\partial E}{\partial w_{ij}}(t-1) - \frac{\partial E}{\partial w_{ij}}(t)} \Delta w_{ij}(t-1)$$

On reconnaît l'approximation diagonale du hessien et une formule de sécante pour une fonction réelle. Pour corriger les variations dues à l'influence des autres poids, Fahlman ajoute un terme de descente du gradient,  $-\eta \frac{\partial E}{\partial w_{ij}}(t)$ , sauf si  $-\eta \frac{\partial E}{\partial w_{ij}}(t)$  et  $-\eta \frac{\partial E}{\partial w_{ij}}(t)$  sont de signes contraires, auquel cas la première formule convient parfaitement.

### 3.2.2.8. Méthode OSS (One Step Secant) [Battiti 90] [Battiti 92]

Battiti propose une méthode de complexité  $O(N)$  et demandant une capacité de stockage en  $O(N)$  pour la rétro-propagation par lots.

La matrice  $A=H^{-1}$  est calculée par l'équation BFGS où on prend systématiquement  $H=I$  pour ne pas avoir à stocker de matrice. De la sorte, il obtient comme direction de recherche:

$$d = -Ag = -g + a\Delta x + by \quad \text{avec } y(t) = g(t)-g(t-1)$$

$$\text{en notant } a = -\left(1 + \frac{\|y\|^2}{\Delta x^t y}\right) \frac{\Delta x \Delta x^t}{\Delta x^t y} + \frac{y^t g}{\Delta x^t y} \text{ et } b = \frac{\Delta x^t g}{\Delta x^t y}.$$

De même que pour les gradients conjugués, il est utile de réinitialiser la direction de recherche régulièrement (au plus toutes les  $N$  itérations). On retrouve d'ailleurs des directions mutuellement conjuguées lorsque les minimisations sur ces directions sont faites avec exactitude. La méthode OSS se trouve ainsi à cheval entre les méthodes de gradients conjugués et de sécantes et est suffisamment robuste pour permettre des minimisations unidimensionnelles très approximatives, c'est à dire plus rapides.

### 3.2.2.9. Méthode SCG (Scale Conjugate Gradient) [Möller 93b]

Cette méthode repose également sur les techniques de gradients conjugués et utilise une formule de sécantes pour calculer le pas  $\lambda$ :

$$\lambda = -(d^t \nabla E)/(d^t [Hd]) \quad \text{avec } [Hd] = (\nabla E(w + \alpha d) - \nabla E(w))/\alpha \text{ et } 0 < \alpha < 1$$

Cette formule pose un problème car elle ne permet pas de conserver le hessien défini positif, ce qui est pourtant primordial pour tout algorithme de gradients conjugués. Pour y remédier, Möller introduit un terme variable en fonction de l'environnement (approche de région de confiance du modèle).

$$\lambda = -(d^t \nabla E)/(d^t [Hd] + rd^t d)$$

Le paramètre d'échelle  $r$  sert à rendre  $H$  défini positif (on s'assure que le dénominateur est positif) et est ajusté en fonction de la réduction constatée de l'erreur.

L'avantage de cette méthode est d'éviter les minimisations unidirectionnelles, ce qui permet une réduction assez considérable de la complexité algorithmique par rapport aux autres méthodes à base de gradients conjugués. En effet, une seule de ces minimisations nécessite généralement entre 6 et 20 appels des fonctions  $E(w)$  et  $\nabla E(w)$  d'après [Gill Murray Wright 80], c'est à dire de 6 à 20 cycles de rétro-propagations, alors que la méthode SCG demande seulement environ deux fois plus de calculs que la rétro-propagation de base.

### 3.2.3. Multiplication du hessien par un vecteur

Plusieurs auteurs ont démontré la possibilité d'effectuer le calcul exact du produit du hessien et d'un vecteur par rétro-propagation et avec une complexité en  $O(N)$  : [Werbos 88] [Möller 93a] [Pearlmutter 94].

L'approche de Pearlmutter est intéressante car elle utilise la différenciation automatique. Il définit l'opérateur différentiel  $\mathfrak{R}_v\{f(w)\} = (\partial/\partial r)f(w + rv)|_{r=0}$  et l'applique aux équations utilisées dans la rétro-propagation pour calculer le gradient de l'erreur à partir de  $w$ . On obtient de la sorte les équations permettant d'obtenir le produit  $Hv$  à partir de  $v$  en une seule rétro-propagation.

Ce calcul est exact dans la mesure où il ne suppose pas d'approximation sur le hessien  $H$  (car il ne nécessite pas le calcul de  $H$ ). Son utilisation peut améliorer nombre des méthodes précédentes en facilitant :

- le calcul des valeurs propres et vecteurs propres de  $H$  ;
- la multiplication par  $H^{-1}$ , le calcul de  $x = H^{-1}b$  pouvant se faire en minimisant  $\|Hx-b\|^2$  avec la méthode des gradients conjugués ;
- la détermination de la taille du pas par minimisation du gradient sur une direction.

Ce dernier point est illustré par la méthode Scaled Conjugate Gradient (SCG) de Möller [Möller 93b] qui peut être améliorée en remplaçant la formule de sécante par une valeur exacte sans trop compliquer les calculs.

**Application à la classification  
des pixels d'une image satellite**

## **4. Description des données d'expérimentation**

### **4.1. Origine des données**

Les procédures de classification ont été testées sur les données numérisées de deux images satellites LANDSAT TM, fournies par l'Université Catholique de Louvain, qui se propose de collecter et de comparer les procédures de classifications des différentes équipes européennes de recherche intéressées par l'exercice. Seuls les six capteurs TM d'une résolution de 30m sont pris en compte. Ces images représentent la région belge de Cinney au 16 mai et au 20 août 1989 et ont été corrigées géométriquement.

La carte de référence pour l'évaluation des méthodes est une partie d'une carte réalisée pour le compte de la Région wallonne par deux sociétés wallonnes spécialisées en télédétection, WALPHOT SA et CICADE SA Cette carte à l'échelle 1/50 000<sup>ème</sup> représente l'occupation du sol par 16 classes thématiques issues pour la plupart d'un traitement numérique d'images spatiales multispectrales LANDSAT TM et SPOT en mode multispectral pour les zones à forte densité urbaine. La classification a été complétée par comparaison avec des scènes LANDSAT et SPOT et des cartes topographiques (notamment pour la représentation des routes, cours d'eau et limites administratives). Trois des classes ont été ajoutées après classification.

A titre d'information, la précision globale de 88,8% obtenue a été jugée satisfaisante, compte tenu des catégories représentées, par le comité scientifique chargé du contrôle de la classification.

### **4.2. Nature des données**

Les données qui nous ont été fournies consistent en 16 fichiers contenant les coordonnées et les valeurs radiométriques de pixels d'une des classes pour l'image du 16 mai. Seules 10 classes sont représentées, les autres étant soit non classifiables (routes...) soit jugées trop largement minoritaires. En fait, chaque fichier a été formé à partir d'un polygone de surface très variable de l'image sélectionné par une personne connaissant bien la région. Certaines classes sont représentées par deux fichiers (Infrastructure, Prairie permanente) voire trois (Bois et forêts de résineux, Cultures), chacun constituant une sous-classe.

On trouvera en annexe les résultats d'une étude de normalité réalisée par le logiciel SAS et comprenant les coefficients d'assymétrie (skewness), de Kurtosis ainsi que les histogrammes des résidus, obtenus en sommant les différences entre chaque valeur et la moyenne de la classe d'appartenance. Le test de normalité est celui défini par Kolmogorov. Comme les probabilités pour que chaque variable soit normale sont inférieures à 0.01, l'hypothèse de normalité de chaque variable est rejetée ; l'hypothèse de normalité pour l'ensemble des 6 variables est donc également à rejeter. Il est à noter qu'une deuxième étude de distribution, effectuée classe par classe, a montré les mêmes conclusions que celles

obtenues précédemment. Ce point doit favoriser les méthodes connexionnistes par rapport aux méthodes statistiques, pour lesquelles on fait habituellement l'hypothèse d'une distribution normale des données.

### **4.3. Construction des bases**

Plusieurs jeux de bases d'apprentissage et de généralisation ont été réalisés.

Le choix du nombre de pixels par classe pour l'apprentissage est délicat. Il est clair qu'un nombre important d'exemples ne peut qu'améliorer la qualité de l'apprentissage (au détriment peut-être de la rapidité). Cependant, dans la pratique, la partie la plus difficile de la classification automatique est la détermination de la classe des pixels de référence, la méthode la plus classique- et la plus sûre- étant le repérage sur le terrain de ces pixels. Au contraire, avec un nombre faible d'exemples, on risque de ne pas représenter certaines catégories de pixels, et d'effectuer un apprentissage "lacunaire". C'est pourquoi un choix d'une dizaine d'exemples par sous-classe semble raisonnable.

D'autre part, contrairement aux méthodes statistiques qui encouragent à utiliser toutes les données disponibles, il n'y a pas de raison, à mon avis, de représenter les sous-classes par des nombres très différents d'exemples sous prétexte qu'on dispose de plus d'exemples d'une classe donnée. Cela serait contraire à la théorie de l'information de Shannon. En effet, si l'on veut que le réseau retire de façon optimale l'information contenue dans les différentes classes, il faut représenter équitablement les groupes de pixels de caractéristiques radiométriques homogènes (les sous-classes). On peut aussi dire que l'information doit être répétée régulièrement pour être assimilée.

De plus, l'ordre de présentation des exemples, qui est sans importance pour les rétro-propagations périodiques où l'information est sommée sur toute la base avant modification des poids, est primordiale pour les rétro-propagations continues. Suivant les mêmes principes que précédemment, il vaut mieux alterner régulièrement les exemples des différentes classes. Ainsi, la rétro-propagation continue appliquée à la base appl.app constituée de 4 classes homogènes et 40 exemples classés classe par classe (10 par classe) n'a donné absolument aucun résultat: le réseau s'entraîne sur une classe pendant 10 itérations puis l'oublie progressivement pendant qu'il s'entraîne sur les autres.

## **5. Choix et implémentation des méthodes connexionnistes**

La comparaison des multiples méthodes algorithmiques, de codage, et autres variantes nécessiterait un temps de programmation démesuré si les développements étaient distincts. C'est pourquoi les caractéristiques du programme recherchées en priorité sont la souplesse et la réutilisabilité du code. Il a donc été décidé d'implémenter les algorithmes dans un langage orienté objet, quitte à négliger la vitesse d'exécution du code.

Comme le matériel informatique de l'ISAB s'organise uniquement autour de l'architecture PC, et afin de pouvoir, à terme, transformer le programme en logiciel de

démonstrations des méthodes les plus significatives, le choix de l'outil de développement s'est arrêté sur Borland C++ 4.5.

## **5.1. Organisation des objets**

Les réseaux et les méthodes connexionnistes sont programmées sous la forme d'objets. Suivant l'organisation d'un réseau du type Perceptron multicouches, un objet réseau comprend des objets couches elles-mêmes constituées d'objets neurones. Ces derniers possèdent des connexions entrantes et sortantes, mais dans le cas des réseaux à couches, où il n'y a pas d'ambiguïté sur le sens de propagation des impulsions à travers les connexions, on peut se contenter des connexions entrantes.

Ces différentes classes d'objets contiennent des données membres et fonctions membres qui reflètent les propriétés locales de tout réseau de neurones à couches, et sont donc réutilisables, ce qui signifie qu'ils peuvent être déclarés et manipulés de la même façon dans les différents algorithmes connexionnistes. Par exemples, les objets connexions possèdent une fonction membre `MiseAJourPoids()` servant à modifier le coefficient synaptique indépendamment des calculs qui ont pu être précédemment effectués. Cela permet de programmer de la même façon rétro-propagations continue et périodique, seul le moment où se fait l'ajustement des poids permettant de faire la différence entre les deux méthodes.

Les algorithmes d'apprentissage sont implémentés dans des classes de réseau dérivant toutes de la classe de base `Reseau`. Ces réseaux sont construits et manipulés via la classe `ControleReseau`. La version actuelle du programme, conçu pour effectuer des tests en série, fonctionne sous DOS. Pour le transformer en programme Windows avec interface graphique, il suffit de modifier la classe `ControleReseau` afin qu'elle hérite des propriétés d'une boîte de dialogue par exemple.

Les objets de base sont décrits dans les fichiers `neurone.h` et `neurone.cpp`, les algorithmes d'apprentissage dans `reseau.h` et `reseau.cpp`, et la manipulation des algorithmes dans `controle.h` et `controle.cpp`, fichiers tous fournis en annexe.

## **5.2. Choix et description des algorithmes**

Sept classes dérivées de la classe de base `Reseau` ont été écrites dans le but de représenter la diversité des idées exposées dans les méthodes jugées significatives et décrites au chapitre 3.

La classe `ReseauRPG` permet d'effectuer des rétro-propagations continues ou périodiques (selon un paramètre) avec inertie.

La classe `ReseauChan` en hérite et permet de plus un ajustement des coefficients d'apprentissage et d'inertie suivant les formules du paragraphe 3.2.2.3.

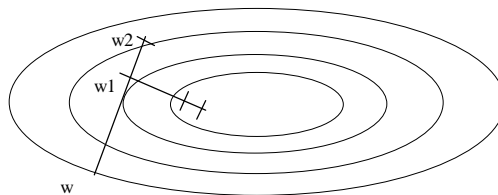
La classe ReseauBarnard reprend la méthode du 3.2.2.4. mais sans minimisation unidirectionnelle. Un cycle sur 20 est consacré à déterminer, à partir d'une simple rétro-propagation périodique, un coefficient d'apprentissage adapté pour effectuer une rétro-propagation continue pendant les 20 cycles suivants.

La classe ReseauGC utilise les gradients conjugués avec une minimisation qui comprend une procédure d'encadrement du minimum et une série d'interpolations quadratiques.

La classe Reseau Hd reprend les gradients conjugués mais sans minimisation, grâce au calcul exact du produit du hessien H par un vecteur, effectué par les fonctions membres RPropagation et RRetro (pour  $\mathfrak{R}$ (Propagation) et  $\mathfrak{R}$ (Retropropagation) avec les notations du 3.2.3.).

La classe ReseauSCG reprend la méthode SCG de Möller du paragraphe 3.2.2.9. en utilisant le calcul exact de Hd plutôt que la formule de sécante.

La classe ReseauKARN est une méthode personnelle qui tente de transposer l'idée de la minimisation unidirectionnelle pour la rétro-propagation continue. A chaque présentation d'un exemple, deux propagations continues sont effectuées en parallèle sur deux jeux de poids distincts. On effectue ensuite une rétro-propagation du gradient de l'erreur totale la plus faible, et pour les poids correspondants. Ces poids servent à générer les deux nouveaux jeux de poids  $w_1$  et  $w_2$  à partir de la direction calculée et de deux coefficients d'apprentissage, coefficients que l'on peut faire varier en fonction de l'erreur comme dans la méthode de [Vogl et al.] du 3.2.2.1. Cependant, pour profiter de l'information supplémentaire que constitue le choix d'un des deux jeux de poids, j'ai choisi d'augmenter ces coefficients si la deuxième propagation a été choisie, de la diminuer sinon. Cette modification se fait en multipliant ou en divisant par une constante que j'ai prise égale à 1.1, et a lieu seulement une fois par cycle, en fonction du jeu de poids qui a été le plus souvent choisi sur une période.



ReseauKARN: ici  $E(w_1) < E(w_2)$   
donc on conserve  $w_1$  pour calculer les nouveaux  $w_1$  et  $w_2$

## **6. Comparaison des méthodes**

### **6.1. Complexité**

La comparaison des méthodes est compliquée par le fait que deux facteurs jouent en sens contraire. Certaines méthodes numériquement plus complexes permettent de gagner en nombre de cycles, mais par contre demandent plus de calculs par cycle, et donc plus de temps. Il faut prendre garde à ce que les calculs supplémentaires ne viennent compenser le gain en nombre de cycles.

Lorsqu'il s'agit de calculer des normes de vecteurs d'après les données rétro-propagées, cela peut être négligeable. Il n'en va pas de même lorsqu'il s'agit d'effectuer des rétro-propagations supplémentaires. C'est le cas dès qu'on veut faire une minimisation unidirectionnelle, dont chaque étape est une rétro-propagation.

### **6.2. Comportement des méthodes**

Au moment de la rédaction de ce mémoire, seule une partie des nombreux tests envisagés ont été réalisés. Les résultats qui se dégagent permettent toutefois des observations intéressantes sur le comportement global de différentes méthodes.

Dans toutes les expériences, le réseau possède 6 neurones en entrée (un par canal) et autant de neurones de sortie que de classes à reconnaître. Les entrées sont ramenées entre -1.25 et 1.2 pour éviter d'être systématiquement dans la zone de saturation des fonctions d'activation dès le début de l'apprentissage. Le codage en sortie est le codage continu classique: pour chaque exemple présenté, le neurone associé à sa classe doit avoir sa sortie à 1 et les autres à -1.

Encore par soucis de normalisation, le coefficient d'apprentissage pour chaque neurone est divisé par le nombre de connexions à l'entrée de ce neurone, ce qui fait qu'il y a en fait un coefficient d'apprentissage par couche de neurone.

La fonction d'activation est une fonction sigmoïdale:

$$f(x) = [\exp(kx)-1]/[\exp(kx)+1]$$

Ses dérivées s'expriment facilement:

$$\begin{aligned} f' &= k(m+f)(m-f)/(2m) \\ f'' &= -k^2f(m+f)(m-f)/(2m^2) \end{aligned}$$

La dérivée de  $f$  est systématiquement augmentée de 0.1, suivant en cela les conseils de Fahlman (cf 2.3.2.), pour éviter les problèmes dus aux zones de saturation.

### **6.2.1. ReseauRPG**

La base `applal.app` a été utilisée pour tester la rétro-propagation continue en faisant varier les valeurs des coefficient d'apprentissage et d'inertie. Les résultats classiques ont été constatés. Notamment, les meilleurs résultats sont obtenus pour un coefficient d'inertie proche de 1.

Il est intéressant de noter qu'erreur totale et taux de généralisation n'évoluent pas symétriquement. Ainsi, dans le fichier `trp01.txt`, la meilleure généralisation est obtenue pour un coefficient d'apprentissage de 0.6 et l'erreur la plus faible pour 3.6.

### **6.2.2. ReseauHd**

La méthode de rétro-propagation avec gradients conjugués et calcul exact du produit Hd a également été testée sur la base `applal.app`: on obtient des résultats un peu moins bon que pour la rétro-propagation continue, avec un taux de généralisation pouvant tourner autour de 92%, ce qui est curieusement assez étonnant et relativement satisfaisant. En effet, Möller a trouvé qu'une méthode de gradients conjugués avec une formule de sécante au lieu de la classique minimisation unidirectionnelle ne fonctionne pas (plus précisément, le réseau converge vers une solution très mauvaise)[Möller 93b], et cela en raison de l'approximation quadratique, abusive loin du minimum. C'est d'ailleurs ce qui l'a amené à concevoir la méthodeSCG.

Se pourrait-il alors que le calcul exact de Hd donne un tel avantage sur la formule de sécante? En fait non. Ce bon comportement est du à l'effet de la constante de Fahlman: en l'absence de cette constante, on observe le comportement décrit par Möller.

Dans la suite, cette méthode est abandonnée au profit de la méthode SCG qui s'est montrée plus efficace.

### **6.2.3. Apprentissage sur toutes les classes**

La base d'apprentissage `3.app` contient 10 classes homogènes représentées chacune par 10 exemples: elle a été constituée à partir d'un seul fichier par classe, ce qui simplifie naturellement l'apprentissage.

Les expériences ont porté sur `ReseauBarn`, `ReseauSCG` et `ReseauKARN` (cf Annexe). Les deux premières se sont bien comportées avec des taux de généralisation atteignant respectivement 96% et 95,3% après 30000 itérations (moyenne sur 20 apprentissages) et le dernier moins bien: 68%.

Ce résultat décevant m'incite à penser qu'une trop grande souplesse du coefficient d'apprentissage dans la rétro-propagation tend à favoriser la convergence vers un minimum local. Lorsqu'il est plus "rigide", comme pour ReseauBarn où il est constant pendant 20 cycles, il évite plus facilement les "trous" de l'erreur dans l'espace des poids.

#### 6.2.4. Comparaison avec des méthodes statistiques

Des méthodes statistiques ont été appliquées aux bases 7.app et 7.gen qui contiennent la totalité des pixels de référence. Il y a toujours 10 neurones de sorties, mais 7.app contient 10 exemples de chaque sous-classe thématique, ce qui fait 10 à 30 exemples par classe.

Ces méthodes statistiques, utilisées dans le logiciel SAS, sont les K-plus proches voisins pour K=5 et K=10, et deux variantes des méthodes utilisant le principe du maximum de vraisemblance: l'analyse discriminante linéaire et l'analyse discriminante quadratique. Les taux de généralisation obtenus sont:

Analyse discriminante linéaire:	94,95%
Analyse discriminante quadratique:	92,38%
10 plus proches voisins:	93,76%
5 plus proches voisins:	95,62%

Ces pourcentages sont assez élevés malgré l'absence de normalité des données, mais il faut garder à l'esprit que la généralisation se fait en fait sur une petite partie de l'image satellitaire, et sur des pixels recueillis en bloc.

Les apprentissages connexionnistes qui ont pu être réalisés avant la rédaction de ce mémoire ont donné les résultats suivant (14 neurones cachés, moyenne sur 10 apprentissages):

SCG	90%
RP continue	93,5%
RP périodique	96,24%

Le score plus faible de la méthode SCG est surprenant et est peut-être du à la confusion sur certaines des données et au fait qu'on s'interdit dans cette méthode toute augmentation de l'erreur totale, ce qui peut conduire à un blocage dans une région dépressionnaire suffisamment large.

## Conclusion

Les premiers résultats obtenus sont encourageants dans la mesure où il semble possible de classer les pixels satellitaires au moins aussi bien avec des réseaux neuronaux qu'avec les méthodes statistiques les plus réputées. On peut penser que ces résultats peuvent encore être améliorés, d'autant plus que plusieurs facteurs importants, qui ont été abordés dans ce mémoire, n'ont pu être testés par manque de temps, comme l'influence du codage de sortie ou le choix d'autres fonctions d'activation.

Cet optimisme doit cependant être tempéré par deux remarques. Tout d'abord, ces résultats ne portent que sur une image satellite (et même une petite partie de l'image) et il sera important de les vérifier sur d'autres données satellitaires. De plus, il ne faut pas oublier l'extrême lenteur de l'apprentissage par rapport aux méthodes statistiques -de l'ordre de quelques dizaines de minutes contre quelques dizaines de secondes- surtout si l'on veut faire plusieurs apprentissages pour réduire l'aléa de l'initialisation des poids.

Cette lenteur, dont l'importance dépend de l'amélioration qualitative qui pourra être obtenue, peut également être sensiblement réduite, tant par le choix de méthodes plus rapides que par la réduction du nombre de pixels de références jugé nécessaire à l'apprentissage.

Davantage que des résultats expérimentaux, de portée provisoire par nature, il restera de mon passage à l'ISAB un programme contenant des fonctions et des algorithmes réutilisables et permettant l'élaboration rapide de toute nouvelle méthode connexionniste qu'il semblera intéressant de tester. Il conviendra d'améliorer ce programme, notamment pour faciliter le codage des sorties, afin de poursuivre les tests.

## Bibliographie

[Barnard 92] BARNARD E. - *Optimization for Training Neural Nets*, I.E.E.E. Transactions on Neural Networks Vol. 3, n° 2, 1992, p. 232-240.

[Battiti 92] BATTITI R. - *First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method*, Neural Computation, Vol. 4, 1992, p. 141-166.

[Battiti Masulli 90] BATTITI R. et MASULLI F. - *BFGS Optimization for faster and automated supervised learning*, INCC-90 Paris, 1990, p.757-760.

[Beale 72] BEALE E.M.L.-*A derivation of conjugate gradients*, Numerical methods for nonlinear optimization, F.A. Lootsma,ed.- 1972, p. 39-43.

[Benediktsson et al. 90] BENEDIKTSSON J.A. SWAIN P.H. et ERSOY O.K. - *Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data*, I.E.E.E. Transactions on Geoscience and Remote Sensing. - Vol. 28. - No. 4. -1990 - p. 540-552.

[Benediktsson et al. 93] BENEDIKTSSON J.A. SWAIN P.H. et ERSOY O.K.. - *Conjugate-gradient neural networks in classification of multisource and very-high-dimensional remote sensing data* - International Journal of Remote Sensing. - Vol. 14. - 1993 - No. 15. - 21 p.

[Bischof et al. 92] BISCHOF H. SCHNEIDER et W PINZ A.J. - *Multispectral Classification of Landsat Images Using Neural Networks*. I.E.E.E. Transactions on Geoscience and Remote Sensing, vol. 30, n° 3, 1992, p. 482-490.

[Broyden 70] BROYDEN C.G. - *The convergence of a class of double-rank minimization algorithms*. - J.Inst. Maths Applics, 6:76-90, 1970, p. 222-231.

[Bryson Ho 69] BRYSON A.E. et HO Y.- *Applied Optimal Control*. Blaisdell Publishing Co., 1969.

[Caloz 87] CALOZ R.- *Téledétection appliquée* - notes de cours. Ecole Polytechnique Fédérale de Lausanne, Institut de Génie Rurale Hydrologie et Aménagement, Lausanne, mai 1987, 123 p.

[Chan 90] CHAN L.W.- *Efficacy of different learning algorithms of the backpropagation network*, Proc. IEEE TENCON-90, 1990.

[Chan Fallside 87] CHAN L.W. et FALLSIDE F. - *An adaptive training algorithm for backpropagation networks*, Comput. Speech Language 2, 1987, p205-218.

[Davidson 59] DAVIDSON (W.C.). - *Variable Metric Methods for Minimization*, AEC Research and Development Report ANL-5990 Rev, Argonne National Laboratories, 1959, novembre.

- [Decatur 89] DECATUR S. E. - *Application of Neural Networks to Terrain Classification*. Proceedings of the I.J.C.N.N.'89. - Washington. - Vol. 1. -1989 - p. 283-288.
- [Dennis Schnabel 83] DENNIS J.E. et SCHNABEL R.B. - *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewoods Cliffs,NJ, 1983.
- [Fahlman 88] FAHLMAN S.E. - *Faster-Learning Variations on Back-Propagation: An Empirical Study*, Proceedings of the 1988 Connectionist Models Summer School, Carnegie Mellon University, 1988, p.38-51.
- [Fletcher Reeves 64] R.FLETCHER et C.M.REEVES, *Function minimization by conjugate gradients*, Computer Journal 7 , 1964, p. 149-154.
- [Fletcher Powell 63] FLETCHER R. et POWELL M.J.D. - *A rapidly convergent descent method for minimization*, The Computer Journal, vol. 6, 1963, p. 163-168.
- [Girard Girard 89] GIRARD M.C. et GIRARD C.M. - *TELEDETECTION APPLIQUEE zones tempérées et intertropicales*, Masson, Paris, 1989, 260p.
- [Gill Murray Wright 80] GILL P.E. MURRAY W. WRIGHT M.H. - *Practical Optimization*, Academic Press Inc., New York, 1980.
- [Gorman Sejnowski 88] GORMAN R.P. et SEJNOWSKI T.J. - *Analysis of hidden units in a layered network trained to classify sonar signals* - Neural Networks, vol. 1, n° 1, 1988, p. 75-90.
- [Guyot 93] GUYOT G. - *De la physique de la mesure à l'agriculture*. Ecole Nationale Supérieure Agronomique de Montpellier, mai 1993, 124 p.
- [Hancock 88] HANCOCK P.J.B. - *data representation in neural nets: an empirical study*. Proceedings of the 1988 Connectionist Models Summer School, Carnegie Mellon University, 1988, p.11-20.
- [Hanson Pratt 89] HANSON S.J. et PRATT L.Y. - *Comparing biases for minimal network construction with back-propagation*. Advances in Neural Information Processing Systems 1, Morgan-Kaufmann, 1989, p. 177-185.
- [Hara Nakayama 94] HARA K. et NAKAYAMA K. - *Comparison of Activation Fonctions in Multilayer Neural Network for Pattern Classification*, ICNN-94, 1994, p. 2997-3002.
- [Hassibi Stork 93] HASSIBI B. et STORK D.G. - *Second order derivatives for network pruning: Optimal Brain Surgeon*. Advances in Neural Information Processing Systems 5, Morgan-Kaufman, 1993, p. 164-171.
- [Hestenes Stiefel 52] HESTENES M.R. et STIEFEL E.L. - *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, **49**(6), 1952, p. 409-436.

[Jacobs 88] JACOBS R.A. - *Increased Rates of Convergence Through Learning Rate Adaptation*, Neural Networks, Vol. 1, 1988, p. 295-307.

[Le Cun et al. 93] LE CUN Y., SIMAR P. Y. et PEARLMUTTER B. A. - *Automatic Learning Rate Maximisation by On-Line Estimation of the Hessian's Eigenvectors*, Advances in Neural Information Processing Systems 5, 1993, p. 156-157.

[Le Cun Denker Solla 90] LE CUN Y., DENKER J.S. et SOLLA S.A. - *Optimal Brain Damage*. Proceedings of the Neural Information Processing Systems-2, Morgan-Kaufman, 1990, p. 598-605.

[Le Cun 88] LE CUN Y. - *A theoretical Framework for Back-Propagation*, in Proceedings of the 1988 Connectionist Models Summer School, Carnegie Mellon University.- 1988. - p. 21-28.

[Le Cun 87] LE CUN Y. - *Modèles Connexionnistes de l'Apprentissage*, Thèse de doctorat de l'Université Paris 6, 1987. - 223 p.

[Mac Clellan et al. 89] MAC CLELLAN G. E. DE WITT R. N. HEMMER T. H. MATHESON L. H. MOE G.O. - *Multispectral image-processing with a three-layer backpropagation network*, Proceedings of the I.J.C.N.N.'89. - Washington. - Vol. 1. - 1989 - p. 151-153.

[Möller 93a] MOLLER M. - *Exact calculation of the product of the Hessian matrix of feedforward network error function and a vector in  $O(n)$  time*. Daimi PB-432, Computer Science Department, Aarhus University, 1993.

[Möller 93b] MOLLER M. - *A scaled conjugate gradient algorithm for fast supervised learning*. Neural Networks, vol 6, 1993, p. 525-533.

[Parker 85] PARKER D.B. - *Learning Logic* .- Tech Report TR-47, Center for Computational research in economics and management science, MIT, avril 1985.

[Pearlmutter 92] PEARLMUTTER B. A. - *Gradient Descent: Second-Order Momentum and Saturating Error*, Advances in Neural Information Processing Systems 4, 1992, p. 887-894.

[Pearlmutter 94] PEARLMUTTER B. A. - *Fast Exact Multiplication by the Hessian*, Neural Computation, Vol. 6, 1994, p. 147-160.

[Polak Ribiere 69] POLAK E. et RIBIERE G. - *Note sur la convergence de methodes de directions conjuguées*, Revue Française Information Recherche Opérationnelle **16**, 1969, p. 35-43.

[Porchier 93] PORCHIER J.C. *La télédétection et la statistique agricole*. Bulletin technique d'information du Ministère de l'Agriculture et de la Forêt, n°13, 1993, mai-juin, p 18-27.

[Powell 77] POWELL M.J.D. - *Restart procedures for the conjugate gradient method*, Mathematical Programming 12, 1977, p. 241-254.

[Press et al. 86] PRESS W. H., FLANNERY B. P., TEUKOLSKY S. A., VETTERLING W. T. - *Numerical Recipes: The Art of Scientific Computing*. - Cambridge University Press, 1986, 818 p.

[Rumelhart et al. 86] RUMELHART D.E., HINTON G.E., WILLIAMS R.J. - *Learning Internal Representations by Error Propagation*, Parallel Distributed Processing : Explorations in the Microstructures of Cognition. - M.I.T. Press. - Vol. 1, 1986, p. 318-362.

[Shanno 78] SHANNO D.F. - *Conjugate gradient methods with inexact line searches*, Mathematics of Operations Research 3, 1978, p. 244-256.

[Shynk Roy 88] SHYNK J.J. et ROY S. - *The LMS algorithm with momentum updating*, Proceedings of the IEEE International Symposium on Circuits and Systems, 1988, p. 2651-2654.

[Tugay Tanik 89] TUGAY M.A. et TANIK Y. - *Properties of the momentum LMS algorithm*, Signal Processing, 18(2), 1989, p. 117-127.

[Vogl et al. 88] VOGL T. P., MANGIS J. K., RIGLER A. K., ZINK W. T. et ALKON D. L. - *Accelerating the Convergence of the Back-Propagation Method*, Biological Cybernetics, Vol. 59, 1988, p. 257-263.

[Werbos 74] WERBOS P.J. - *Beyond regression : New tools for prediction and analysis in the behavioral sciences*, Ph.D. thesis, Havard University, 1974.

## **Annexes**