Méthodes constructives pour l'apprentissage à partir d'exemples : les arbres neuronaux hybrides et leur comportement asymptotique

Florence d'Alché-Buc * et Jean-Pierre Nadal †

Laboratoires d'Electronique Philips S.A.S.

B.P. 15, 22 avenue Descartes

94453 Limeil-Brévannes Cedex France

1 Introduction

L'apprentissage supervisé d'un réseau de neurones formels consiste à déterminer son architecture et ses paramètres à partir d'une base finie d'exemples de la tâche qu'il doit réaliser. Qu'il s'agisse d'une tâche de classification ou de prédiction, le problème peut se formuler comme celui de l'approximation d'une fonction inconnue f_* (discrète pour une classification, au moins continue pour une approximation de fonction) à partir d'un nombre fini d'exemples, à l'aide d'une famille F de fonctions paramétrées, à savoir l'ensemble des fonctions représentées et calculées par un réseau de neurones d'un type d'architecture donnée.

Il s'agit donc de déterminer le réseau $\hat{f}$, autrement dit la fonction $\hat{f}$ de la famille F, définie par ses paramètres $\hat{w}$, qui soit la meilleure approximation possible selon un critère fixé au départ [1]. Pour un choix du critère, par exemple la minimisation d'une distance entre entre f_* et $\hat{f}$, l'objectif n'est pas tant d'avoir de bonnes performances sur les points où f_* est connue, que d'obtenir de bonnes propriétés en "généralisation" : on souhaite que le réseau se comporte de manière satisfaisante sur de nouveaux exemples (dans le cas d'une approximation d'une fonction régulière, il s'agit en somme d'obtenir une bonne interpolation entre les points connus). Les performances se mesurent alors par l'erreur commise par le réseau lorsqu'il traite de nouveaux exemples.

*Adresse actuelle : LAFORIA - Institut Blaise Pascal, Université Pierre et Marie Curie, 4 Place Jussieu, 75252 Paris cedex 05 France, e-mail : dalche@laforia.ibp.fr

†Adresse permanente : Laboratoire de Physique Statistique, Ecole Normale Supérieure, 24, rue Lhomond, 75231 Paris Cedex 05 France

Les méthodes neuronales regroupent des classes de fonctions paramétrées qui sont des combinaisons linéaires ou non linéaires de fonctions simples qui opérent un calcul distribué sur leurs entrées. Les systèmes correspondants sont appelés *réseaux neuronaux*. Le développement d'un système neuronal, dans l'approche la plus courante, comprend :

- le choix des fonctions qui vont être utilisées
- le choix d'un type d'architecture (généralement une structure en couches)
- le dimensionnement *a priori* de cette architecture (choix du nombre de couches, du nombre de neurones dans chacune,...)
- l'apprentissage (l'optimisation) des paramètres appelés poids synaptiques

Pour s'affranchir de la phase initiale de dimensionnement du réseau, des algorithmes constructifs ont été proposés. Parmi eux, les approches hiérarchiques [15, 11, 19, 7, 10, 9, 8, 16] présentent différents avantages et aspects originaux. Elles permettent notamment de bien contrôler l'apprentissage à chaque étape de la construction. En outre, elles fournissent à l'issue de la construction une structure qui favorise l'interprétation des décisions. Nous nous intéressons en particulier aux approches que l'on peut regrouper sous le terme d'*arbres neuronaux*. Leur principe, commun aux arbres de classification et de régression introduits par Breiman et collègues, Friedman, Quinlan [3, 12, 21], consiste à décomposer hiérarchiquement une décision complexe en plusieurs décisions simples. Le système obtenu a une structure d'arbre, chaque nœud de l'arbre fournissant une décision et chaque branche aboutissant à l'identification d'une région de décision.

Dans cet article, nous présentons l'algorithme des arbres hybrides comme un exemple caractéristique et probant de l'application d'une stratégie constructive dans le cas particulier de la classification supervisée. Il reste toutefois valable pour d'autres applications telles que la régression de fonctions, l'estimation de densités, l'estimation de distributions a posteriori [10]. La partie **2** est consacrée à la description des algorithmes de construction et d'élagage et à leur mise en oeuvre. Dans la partie **3**, nous montrons l'efficacité et la validité de notre méthode constructive par une étude du comportement asymptotique du système [5, 6, 24]. La méthode est illustrée sur une tâche réelle de grande dimension, à savoir une discrimination de caractères manuscrits. Dans la partie **4**, nous rappelons les résultats présentés et concluons.

2 Une approche hiérarchique constructive : les arbres hybrides de neurones

2.1 Architecture hybride

Dans le cas d'une classification binaire, un arbre neuronal simple [15, 11] consiste en un arbre de décision dont chaque nœud est un neurone formel; autre-

ment dit chaque nœud opère une dichotomie en plaçant dans l'espace des données un hyperplan. Cette architecture d'arbre de neurones se généralise en un *arbre hybride* [10, 9, 16] de la manière suivante: on distingue le rôle des nœuds intermédiaires de celui des nœuds terminaux de l'arbre; chaque nœud intermédiaire segmente l'espace d'entrée en deux domaines tandis que chaque nœud terminal calcule effectivement la sortie de l'arbre dans la région auquel il est associé. On voit immédiatement l'intérêt potentiel de ce type d'architecture: les nœuds intermédiaires calculent des fonctions qui permettent une segmentation de manière à focaliser "rapidement" l'attention sur un domaine limité pertinent; dans chaque domaine, un nœud terminal réalise la tâche par le calcul d'une fonction "experte" pour ce domaine. Ce principe peut s'adapter à différentes fonctionnalités de l'arbre: la discrimination ou classification avec un nombre arbitraire de classes, la régression, l'estimation de densités.... Dans la suite, nous nous limiterons au cas de la classification supervisée à deux classes.

2.2 Apprentissage par trio

Un arbre neuronal simple, appris localement, nœud par nœud, peut conduire à des architectures trop grandes du fait de mauvais choix lors des premières dichotomies. Une manière de corriger cette absence d'optimisation globale est d'appliquer un algorithme d'élagage, puissant mais coûteux, tel que celui proposé par Breiman *et al.*: après la construction par apprentissage, on détermine le meilleur sous-arbre d'une famille d'arbres dérivés de l'arbre construit. Cependant, cette méthode n'ôte pas le caractère local de l'apprentissage de chacun des nœuds. L'algorithme d'apprentissage par trio, présenté en détail dans [7, 10], a été proposé afin d'améliorer la constructiondes arbres de décision. Il s'agit de tenir compte, durant l'apprentissage d'un nœud-séparateur, du fait qu'il sera suivi par deux nœuds fils susceptibles de résoudre (localement) la tâche considérée. Ce principe se traduit par un apprentissage de l'arbre, trio de nœuds par trio de nœuds, au lieu de construire l'arbre nœud par nœud. Cet apprentissage par trio,naturel pour un arbre hybride, offre un compromis intéressant entre une optimisation globale très coûteuse sur tous les niveaux de l'arbre, et une construction séquentielle de l'arbre nœud par nœud qui conduit à des arbres de grande profondeur.

2.3 Principe de l'algorithme du trio pour l'arbre hybride

Dans le cas de la classification, un trio de nœuds est défini de la manière suivante (voir figure 1):

On associe à chaque nœud une fonction de l'espace d'entrée vers l'intervalle $[0, 1]$. Soit x un vecteur de l'espace d'entrée. Considérons un trio. On note $f_{\text{père}}$ la fonction associée au nœud père du trio, g_{gauche} et g_{droite} les fonctions associées aux nœuds fils. La fonction de sortie du trio h est alors définie par:

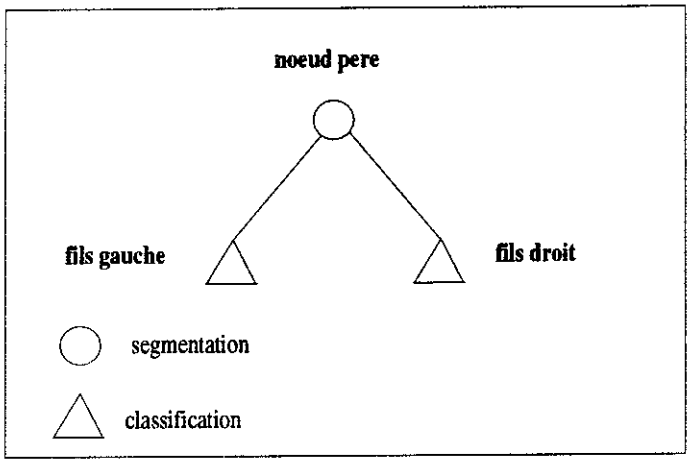

FIG. 1 - Architecture d'un trio hybride

$$\forall x, h(x) = f_{\text{père}}(x)\, g_{\text{gauche}}(x) + (1 - f_{\text{père}}(x))\, g_{\text{droite}}(x) \qquad (1)$$

On peut donner à cette définition une interprétation probabiliste en considérant les sorties des différents nœuds comme des probabilités conditionnelles. Soient A et B les deux classes considérées. La sortie du trio $h(x)$ pour l'entrée x est interprétée comme la probabilité *a posteriori* de la classe A : $h(x) = P(A|x) = 1 - P(B|x)$

$f_{\text{père}}(x)$ (resp. $1 - f_{\text{père}}(x)$) est la probabilité que x doive être traité par le fils gauche (resp. droit) ; $g_{\text{gauche}}(x)$ (resp. $g_{\text{droite}}(x)$) est la probabilité d'être de la classe A sachant que x a été placé à gauche (resp. droite) par le neurone père.

Lors de l'apprentissage d'un trio de nœuds, l'ensemble des exemples d'apprentissage, accessibles au niveau du nœud père, est utilisé pour l'apprentissage des paramètres des 3 nœuds du trio : le nœud père, le nœud fils gauche et le nœud fils droit.

Voici les étapes de l'algorithme d'apprentissage par trio pour un problème de discrimination à deux classes, illustrées sur la figure 2.

- **Etape initiale** : On commence par construire le premier nœud comme un simple classifieur. Après initialisation, les paramètres de la fonction g_1 qu'il calcule sont appris, puis ses performances sont mesurées sur l'ensemble d'apprentissage (étape 0 sur la figure 1). Si ces performances satisfont le critère de qualité choisi, alors on arrête la construction : le problème ne nécessite pas la construction d'un arbre, un simple classifieur suffit. Dans le cas contraire, on remplace ce premier nœud par un trio de nœuds $(1, 2, 3$ sur la figure 2), et on passe à l'Etape 1.

- **Etape 1** : On considère le trio de nœuds $1, 2, 3$ qui doit traiter l'ensemble d'exemples X(1). On applique une méthode de minimisation du coût choisi

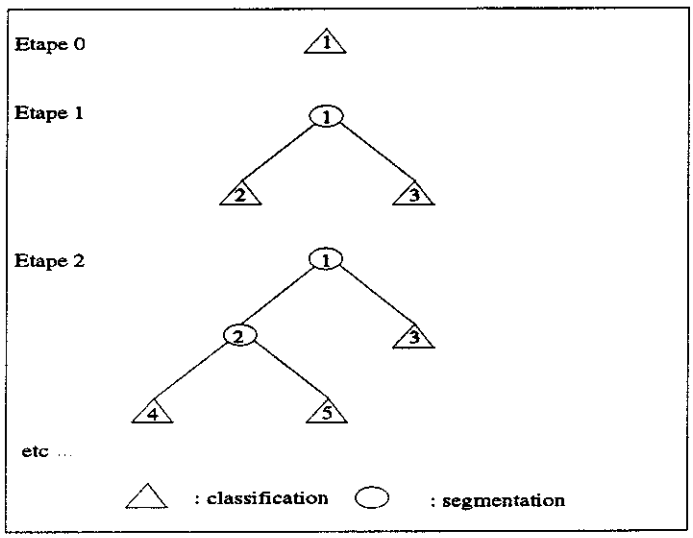

FIG. 2 - Premières étapes de la stratégie d'apprentissage par trio

pour déterminer les paramètres du trio de manière à classifier le mieux possible les exemples d'apprentissage. Cet apprentissage est arrêté dès que le critère d'arrêt est satisfait. On doit ensuite évaluer les performances du trio appris. Pour cela, les réponses continues (entre 0 et 1) des fonctions sont quantifiées et prennent comme valeur 0 ou à 1. On utilise alors le nœud 1, père du trio, pour séparer l'ensemble X(1) des exemples accessibles à ce nœud en deux sous-ensembles, X(2) et X(3), qui seront traités respectivement par le nœud fils 2 et le nœud fils 3.

- **Etape 2**: On évalue ensuite la qualité des classifications obtenues par chacun des nœuds fils sur l'ensemble d'exemples d'apprentissage correspondant. On considère chaque fils séparément. Si la classification obtenue est satisfaisante pour le nœud fils considéré, alors ce nœud (par exemple le noeud 3 sur notre figure) devient un nœud terminal: on lui adjoint deux feuilles correspondant aux deux régions de décision qu'il sépare.

  Si en revanche, la classification au niveau du nœud fils considéré n'est pas satisfaisante, ce dernier est remplacé par un nouveau trio de nœuds (sur la figure 2, c'est le cas du nœud 2 qui est remplacé par le trio $2, 4, 5$). Pour ce nouveau trio, on applique l'Etape 1.

On poursuit ainsi cette construction récursive jusqu'à obtenir, dans chaque branche, un taux de succès en classification satisfaisant. Nous donnons en Annexe des précisions sur la mise en œuvre de l'algorithme.

2.4 Algorithme d'élagage

L'élagage d'une architecture permet de réduire sa complexité après l'apprentissage de ses paramètres[26]. Différentes techniques ont été utilisées pour des architectures en couches et des architectures hiérarchiques [3]. Nous proposons une méthode simple et moins coûteuse que celle de [3]. Le but de cette méthode est de trouver le meilleur sous-arbre de l'arbre construit selon un critère choisi. Un critère simple consiste à évaluer les performances du sous arbre courant sur un ensemble d'exemples *indépendant* de l'ensemble d'apprentissage, que nous appellerons *ensemble de test* T. Il peut s'agir du même ensemble T qui a pu être utilisé pour arrêter l'apprentissage d'une branche (cf annexe).

L'algorithme d'élagage est basé sur l'idée suivante : il s'agit de parcourir l'arbre récursivement en déterminant le meilleur sous-arbre de chaque nœud courant. Nous l'exposons toujours dans le cas d'une tâche de classification. L'algorithme d'élagage nécessite de garder en mémoire, au cours de la construction par apprentissage de l'arbre, les paramètres de chaque nœud classifieur (racine ou fils d'un trio à l'issu de l'apprentissage du trio).

Précisons l'algorithme.

Chaque nœud intermédiaire de l'arbre a été, au cours de l'apprentissage, un nœud-classifieur (fils d'un trio), avant de devenir, au cours d'une Etape 2, un nœud séparateur. On garde en mémoire les paramètres de ce nœud-classifieur ainsi que le nombre d'exemples bien classés. Appelons ce dernier le *score* du nœud : c'est donc le score du nœud dans son ancien état de nœud-classifieur. Ce score est la performance réalisée par ce nœud si on avait arrêté la construction à ce niveau (si on avait gardé ce nœud comme nœud terminal). Le score d'un sous-arbre donné est alors égal à la somme des scores réalisés par tous les nœuds terminaux. Pour déterminer le meilleur sous-arbre de l'arbre entier, on compare la somme des scores des deux meilleurs sous-arbres à droite et à gauche au score obtenu par l'ancienne racine (i. e. avant l'apprentissage trio ayant remplacé ce nœud par un séparateur, racine commune aux deux sous-arbres). Ces scores à droite et à gauche de la racine sont eux-mêmes obtenus récursivement en descendant jusqu'aux nœuds les plus profonds. De façon plus précise :

- pour chaque trio terminal, c'est à dire chaque trio dont les deux fils sont des nœuds terminaux, on compare la somme des scores réalisés par ces fils à l'ancien score réalisé par le père. On supprime les nœuds fils s'ils n'améliorent pas les anciennes performances du nœud père dans la région considérée. Le nœud père du trio est remplacé par l'ancien nœud père. Dans le cas contraire, on garde le trio et on lui associe comme score la somme des scores des fils.

- à un nœud interne de l'arbre, on compare de la même manière la somme des scores obtenus par le meilleur sous-arbre à gauche et le meilleur sous-arbre à

droite au score de l'ancien père. Lorsque le score total des deux sous-arbres est moins bon que celui de l'ancien nœud père, les deux sous-arbres sont supprimés et remplacés par l'ancien nœud père. Dans le cas contraire, on associe au sous-arbre constitué du nœud interne étudié et des deux sous arbres droit et gauche, la somme des scores réalisés à droite et à gauche.

Notons que cet algorithme permet de réduire la complexité du système trouvé en gardant les paramètres trouvés en cours d'apprentissage à chaque nœud. Ceci est différent de la méthode plus puissante, mais bien plus coûteuse, de Breiman *et al.*, qui évalue une famille d'arbres paramétrée. Notons aussi que Gelfand et Guo [13] ont proposé un algorithme d'élagage assez proche, spécifique des arbres dont tous les nœuds sont de même nature, et pour lequel, lors de la construction les rôles de A et de T sont échangés à chaque nœud.

3 Evaluation des performances par l'étude du comportement asymptotique de l'algorithme

3.1 Motivations

De nombreux résultats théoriques sur l'ordre de grandeur de l'erreur en généralisation ont été obtenus, principalement dans la limite asymptotique [25, 23, 14, 2, 20], c'est à dire pour des réseaux de grandes tailles, avec un nombre d'exemples au moins aussi grand que le nombre de paramètres du réseau. Ces résultats, pour la plupart obtenus pour des réseaux de neurones d'architecture fixée *a priori*, les réseaux multicouches, donnent la décroissance du taux d'erreur en généralisation en fonction du nombre d'exemples appris et de la taille du réseau (le nombre de neurones utilisés...).

Ces analyses théoriques suggèrent des approches empiriques pour sélectionner le meilleur réseau. L'idée générale est d'étudier expérimentalement la valeur du taux de succès de classification en fonction de la taille de la base d'apprentissage, et d'en déduire par extrapolation les performances qu'on peut attendre de l'apprentissage sur un nombre bien plus grand d'exemples. Cette démarche a initialement été développée par Cortès, Vapnik et collègues en 1993 [5] et 1994 [6]. Elle a été reprise en 1994 dans [4] pour un problème simulé. Notons que la mise en œuvre peut être plus ou moins complexe suivant le type de résultats théoriques sur lequel on s'appuie[5].

De façon très générale, la caractérisation des propriétés d'un système par l'étude de leur dépendance dans la taille du système (nombre de paramètres,...) est une méthode classique aussi bien en Statistiques [20] qu'en Physique[23]. C'est en particulier un outil très utile pour tester les capacités d'un algorithme, même en l'absence de résultats analytiques spécifiques.

Pour notre part, nous nous sommes intéressés en 1993 à cette approche empirique dans le cas des algorithmes constructifs [7, 8] pour lesquels il n'y a pas,

à notre connaissance, de résultat analytique donnant une estimation des performances à attendre. Notre double objectif était l'étude des performances et de la complexité de l'architecture après apprentissage. Nous avons ainsi étudié le comportement de l'algorithme du trio pour arbres hybrides en fonction de la taille de la base d'apprentissage. Cette étude numérique montre comment l'algorithme d'apprentissage de l'arbre (construction et élagage) induit des arbres de complexité réduite, autrement dit une architecture de taille adaptée aux données. En outre, cette méthode permet de choisir entre différents types de fonctions "expertes" pour les nœuds terminaux. Nous présentons ci-après ce travail expérimental réalisé sur une application "réelle".

3.2 Application : un problème de discrimination de caractères manuscrits

La discrimination de caractères manuscrits constitue une application réelle, typique du domaine de la reconnaissance des formes. Il s'agit de construire un système de classification à partir d'exemples (de caractères) qui soit suffisamment général pour reconnaître avec de bonnes performances des caractères manuscrits obtenus en particulier auprès de scripteurs distincts de ceux qui ont contribué à la formation de la base d'apprentissage. Dans l'application que nous considérons, les caractères acquis à l'aide d'une tablette électronique sont codés dynamiquement par un vecteur de dimension 20 dont les coordonnées correspondent à la suite temporelle de 10 vecteurs représentant le tracé du caractère.

Dans ce cadre, nous nous sommes intéressés à la discrimination de caractères 2 à 2 difficilement discernables, c'est-à-dire pour lesquels la réponse d'un système de reconnaissance global est ambigüe. Par exemple, les lettres minuscules l et b, le chiffre 5 et la lettre majuscule S, les lettres majuscules H et M sont des cas qui posent des difficultés. Il s'agit donc de construire des systèmes pour la résolution de tâches de classification supervisée binaire. Ces tâches peuvent se voir comme des sous-tâches à effectuer dans le cadre d'un système global [17]. Une dizaine de problèmes de ce type ont été extraits du problème global de la reconnaissance dynamique de caractères.

L'évaluation des arbres hybrides sur cette application réelle pose le problème de l'estimation des performances d'un système appris ou construit à partir d'un nombre réduit d'échantillons. C'est exactement le cas ici : l'acquisition d'une base de caractères manuscrits est longue et coûteuse puisqu'elle fait intervenir de nombreux scripteurs (ici de l'ordre d'une centaine). Il faut donc pouvoir estimer les performances du système appris en utilisant au mieux les échantillons recueillis.

3.3 Performance et complexité asymptotiques

Nous illustrons l'étude du comportement asymptotique de l'algorithme des arbres hybrides sur le problème de discrimination de caractères manuscrits "H" et "M". Des résultats du même ordre ont été obtenus pour d'autres problèmes

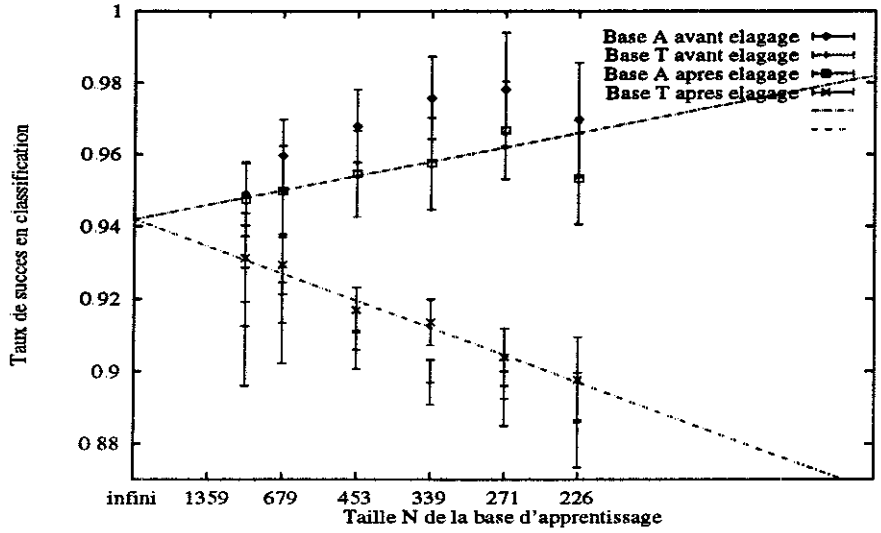

FIG. 3 - Moyennes et écarts-types des valeurs du taux de succès en classification pour le problème 'H' et 'M'

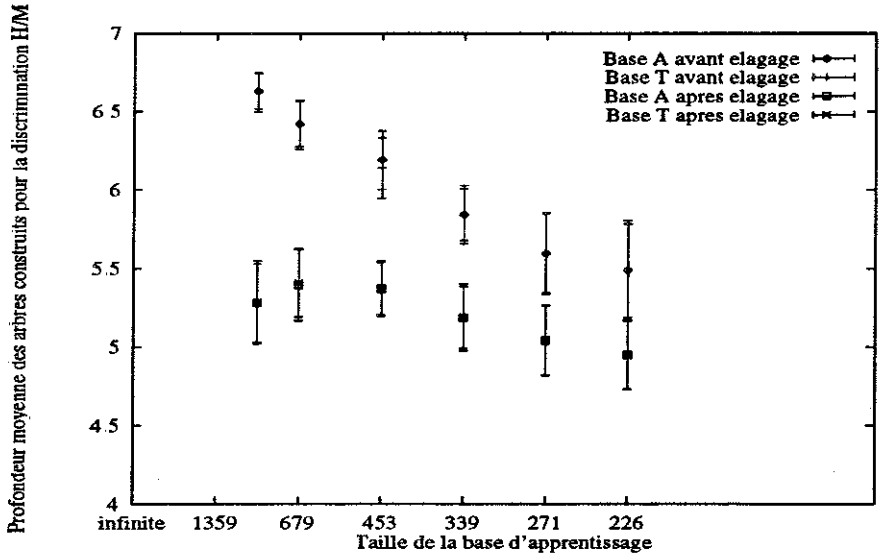

FIG. 4 - Moyennes et écarts-types des profondeurs des arbres pour le problème 'H' et 'M'

de discrimination de deux caractères [7]. Nous faisons tourner l'algorithme avec des ensembles d'apprentissage (base A) de tailles n = 271, 339, 453, 679, 1018 exemples en utilisant les 1359 -n exemples restant pour la base de test (notée base T). Comme nous ne disposions pas d'une base totale suffisamment grande, nous avons utilisé l'ensemble T pour l'élagage de l'arbre. A chaque expérience, nous limitons arbitrairement la profondeur de l'arbre à une profondeur maximale égale à 7. Pour une taille de la base d'apprentissage fixée, nous effectuons 10 tirages aléatoires de la base d'apprentissage de manière à fournir un résultat moyen (donc pour 10 arbres construits) avec son écart-type.

Nous présentons sur la figure 3 le taux moyen de succès en classification en fonction de la taille de la base d'apprentissage; les valeurs ont été obtenues pour des arbres hybrides dont les nœuds intermédiaires calculent des séparateurs linéaires (neurones formels) et les nœuds terminaux des fonctions radiales (des gaussiennes). Nous avons représenté les performances pour la base d'apprentissage et la base de test avant et après l'élagage.

L'analyse de ces résultats se traduit par les remarques suivantes: On vérifie tout d'abord que:

- lorsque la taille de la base d'apprentissage augmente, il est de plus en plus difficile pour le système d'apprendre mais en revanche, le système généralise de mieux en mieux.

- on s'attend à ce que le système ait les mêmes performances en apprentissage et en test lorsque la taille de la base d'apprentissage devient infinie; ce qui semble être vérifié, si on extrapole les courbes d'apprentissage et de test.

En outre, ces résultats expérimentaux montrent que l'on observe un comportement linéaire des deux types de taux de succès (apprentissage et test). Une régression linéaire jointe a été calculée sur les cinq premiers points: les deux droites obtenues ont pour pente respective $-a$ et b:

$$t_A = t_\infty - \frac{a}{N} \tag{2}$$

$$t_T = t_\infty + \frac{b}{N} \tag{3}$$

On ne remarque pas ici $b = a$ comme dans Cortès et al. pour les perceptrons mais plutôt $b \simeq 2a$. Il est cependant remarquable qu'un comportement simple soit obtenu, qualitativement semblable à ceux observés et prédits analytiquement pour les réseaux d'architecture fixée à l'avance. Il est important de noter que, dans notre cas, ce comportement n'est obtenu que *si* la complexité de l'arbre est bien contrôlée, c'est-à-dire si on effectue l'élagage comme indiqué dans la section précédente.

On vérifie d'ailleurs ce rôle de l'élagage en calculant la profondeur moyenne des arbres hybrides pour différentes tailles de la base d'apprentissage. La profondeur moyenne d'un arbre selon un ensemble d'exemples est la moyenne des longueurs de ses branches, chacune des branches étant pondérée par le nombre d'exemples associés. Les résultats expérimentaux montrent que lorsque la taille de l'ensemble

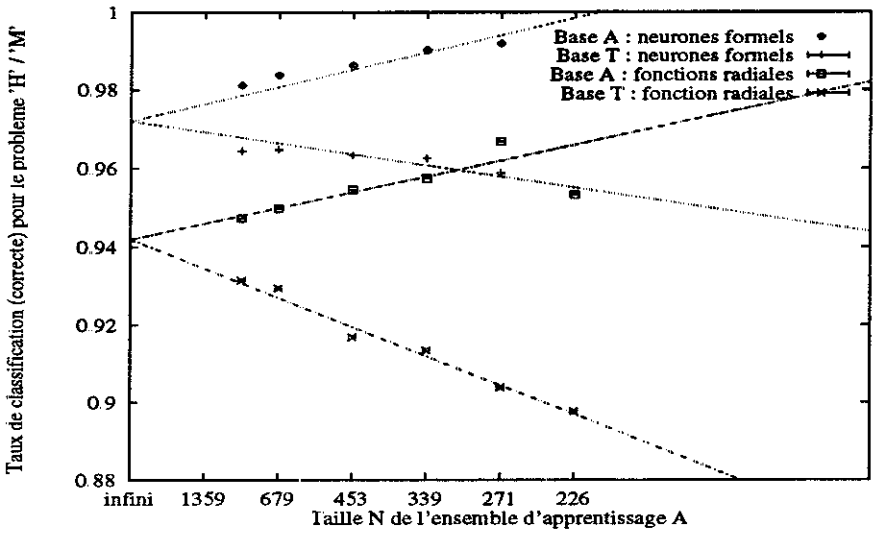

FIG. 5 - Comparaison des performances asymptotiques des arbres de neurones formels et des arbres de fonctions radiales pour le problème "H"/"M"

d'apprentissage augmente, la profondeur se stabilise: la complexité de l'architecture semble donc s'adapter à celle des données. La figure 4 qui représente la valeur de la profondeur moyenne en fonction de l'inverse de la taille de la base d'apprentissage illustre cette propriété des arbres hybrides pour l'exemple de la discrimination "H" / "M".

3.4 Application à la sélection de l'architecture

Nous discutons maintenant l'application de notre approche à la sélection de l'architecture la mieux adaptée pour un problème donné. L'approche constructive que nous avons présentée peut être utilisée pour apprendre différents types de réseaux: on peut notamment choisir différents types de fonctions pour calculer les décisions prises dans les nœuds terminaux. Pour de nombreux problèmes, on ne sait pas *a priori* quel est le type de fonction qui est le plus adapté à la résolution du problème. Il semble raisonnable que la qualité des systèmes construits se traduise par de bonnes performances en généralisation et une complexité réduite. L'étude expérimentale du comportement asymptotique des différents systèmes apporte une réponse. Nous avons comparé les comportement des arbres de neurones formels et des arbres hybrides ayant aux noeuds terminaux des fonctions radiales. Sur la figure 5, sont représentés les taux moyens de succès pour le problème de la discrimination des caractères "H" et "M", en fonction de la taille de la base d'apprentissage pour les arbres de neurones formels et les arbres avec fonctions radiales. On observe le même comportement qualitatif dans les deux cas. Cependant, les performances sont systématiquement meilleures avec le choix des neurones formels comme classifieurs. En particulier, la comparaison des valeurs asymptotiques (calculées à l'aide d'une régression jointe) est en faveur des

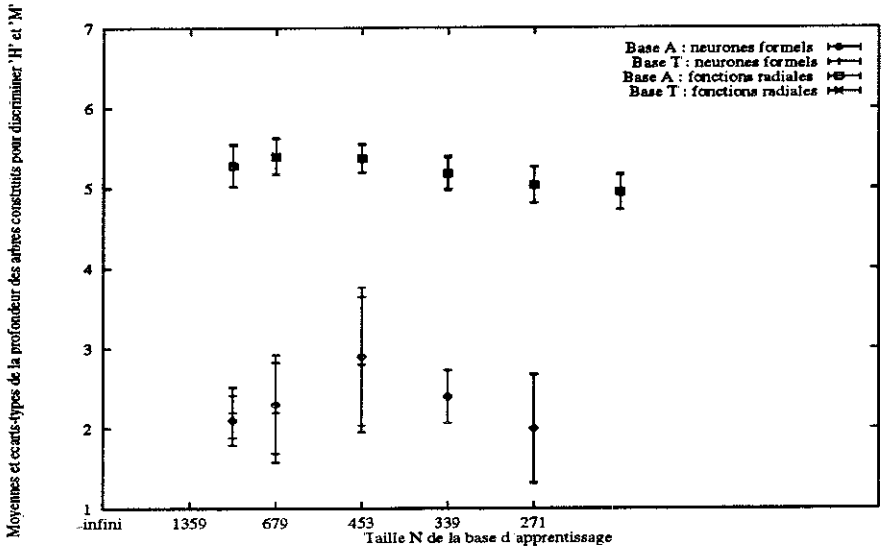

FIG. 6 - Moyennes et écarts-types des profondeurs des arbres pour le problème 'H' et 'M'

arbres de neurones formels qui ont un taux de succès estimé à environ 0.97.

Sur la figure 6 sont représentées les complexités (profondeurs) moyennes des deux types d'arbres après élagage. Les arbres de neurones formels ont une profondeur estimée à 2.5 alors que celle des arbres avec fonctions radiales est estimée à 5.

Pour ce problème, l'utilisation d'hyperplans pour classifier les données permet donc d'obtenir de meilleurs résultats que l'utilisation de boules dont l'appartenance est calculée par une gaussienne en dimension 20. Il est vrai qu'en grande dimension, il faut généralement un nombre important de fonctions radiales pour caractériser une région de décision là où quelques hyperplans sont nécessaires. Rappelons cependant que pour une application où l'interprétabilité des résultats est importante, les fonctions radiales fournissent une représentation lisible en terme de projections sur chaque sous-espace des variables d'entrée.

4 Conclusion

A travers la description de nos travaux sur les arbres hybrides, nous avons montré l'intérêt des approches constructives pour la construction automatique d'une architecture de complexité adaptée à celle des données pour la classification. Plus précisément, les arbres hybrides constituent une méthode hiérarchique pour construire différents types de réseaux avec un algorithme efficace de construction et un algorithme d'élagage simple. L'étude du comportement asymptotique des arbres, c'est-à-dire la mesure des performances et de la complexité de l'architecture construite en fonction de la taille de la base d'apprentissage, permet d'estimer de manière empirique les performances et la complexité des arbres dans

le cas où un plus grand nombre d'exemples serait disponible. En outre, l'algorithme ayant le même type de comportement quel que soit le type de classifieur choisi, il est possible de sélectionner ainsi le type de classifieur à utiliser pour un problème donné. Ceci se fait de manière semblable à ce qui a été proposé pour les réseaux en couches[5, 6] : on effectue des tests sur un nombre réduit d'exemples suivant la procédure présentée dans la section précédente ; en se basant sur les extrapolations aux tailles plus grandes, on choisit le classifieur conduisant aux meilleurs performances asymptotiques.

Notre étude, effectuée sur un algorithme constructif particulier, est clairement adaptable à toute approche constructive. Il serait intéressant de disposer d'autres résultats numériques, et surtout de résultats analytiques, afin de savoir si les comportements simples obtenus en fonction de la taille de l'ensemble d'apprentissage sont génériques. Nous nous attendons à ce que cela soit le cas à chaque fois que la croissance est bien contrôlée (soit par un algorithme d'élagage, soit par l'addition à la fonction de coût d'un terme de complexité [22]).

Remerciements

Les travaux présentés ici ont été effectués aux Laboratoires d'Electronique Philips S.A.S. (LEP). Nous tenons à remercier amicalement Didier Zwierski qui a été à l'origine d'une partie de ce travail. Nous remercions également Philippe Gentric (LEP) qui a fourni la base de données de caractères et avec qui nous avons eu de nombreuses discussions stimulantes.

Annexe : Mise en œuvre de l'apprentissage d'un trio

L'algorithme du trio pour construire des arbres hybrides de neurones a été mis en œuvre avec les choix de fonctions suivantes :

- fonction d'activation d'un neurone formel :

$$f(x;w) = \frac{1}{1 + exp(-2.\beta w.x)} \tag{4}$$

- fonction radiale (par exemple une gaussienne) :

$$g(x;\mu,\sigma) = exp(\frac{-[x-\mu]^2}{2.\sigma_0^2}) \tag{5}$$

Les fonctions d'activation étant différentiables dans chaque nœud du trio, on peut définir une fonction de coût différentiable à partir de la sortie du trio et appliquer un algorithme de descente de gradient pour minimiser cette fonction de coût. Les paramètres appris sont w, μ, et σ. Durant l'apprentissage d'arbres de neurones ou d'arbres hybrides, le paramètre β de pente de la fonction d'activation du noeud père est progressivement figé de manière à fixer plus rapidement l'hyperplan père que les fonctions

de classification des noeuds fils. Voici des critères de coût que nous avons utilisés en classification :

Soit x un vecteur d'entrée quelconque. Soit $h_d(x)$ la sortie désirée.

- le coût quadratique (local) :

$$E_{trio}(x) = \frac{1}{2} \cdot (h(x) - h_d(x))^2 \tag{6}$$

- la log-vraisemblance

$$E_{trio}(x) = -h_d(x) \cdot lnh(x) - (1 - h_d(x)) \cdot ln(1 - h(x)) \tag{7}$$

Minimiser ce critère local revient à maximiser la probabilité pour qu'un exemple x soit bien classé.

Le coût global est défini comme la moyenne des coûts locaux pour l'ensemble des exemples accessibles au trio. Cependant, pour des fonctions qui définissent l'appartenance à un fermé borné de l'espace de décision telles que les fonctions radiales, ces deux fonctions de coût présentent un inconvénient : à chaque étape, il faut choisir *a priori* quelle sera la classe représentée par l'intérieur de la région définie par la fonction à base radiale. Pour pallier cet inconvénient, le critère global suivant, basé sur l'entropie [18, 3, 21, 15] a été choisi pour permettre de déterminer automatiquement la meilleure configuration du système :

$$E = H_{trio} = \sum_{j=A,B} (\frac{N_j}{N}) \times H_j \tag{8}$$

H_j mesure l'incertitude liée à la réponse j du classifieur :

$$H_j = \sum_{i=A,B} p_{ij} \cdot log_2 p_{ij} \tag{9}$$

où p_{ij} est défini par :

$$p_{ij} = p(classe = i | y = j) \tag{10}$$

Ce critère se calcule en estimant ces probabilités par

$$p_{ij} = \frac{N_i^j}{N^j}$$

où N_i^j est le nombre d'exemples de la classe i, attribués à la classe j par le trio et N_j le nombre d'exemples attribués à la classe j par le trio. On utilise :

$$N_A^A = \sum_{x \epsilon X(trio)} h(x) \cdot h_d(x) \tag{11}$$

$$N_A^B = N_A - N_A^A \tag{12}$$

$$N_B^A = \sum_{x \epsilon X(trio)} h(x) \times (1 - h_d(x)) \tag{13}$$

$$N_B^B = N_B - N_B^A \tag{14}$$

$$N_A = \sum_{x \epsilon X(trio)} h(x) \tag{15}$$

$$N_B = N - N_A \tag{16}$$

Lorsqu'elle est minimisée, la fonction de coût H_{trio} permet d'isoler la configuration pour laquelle l'incertitude liée à la réponse du classifieur est la plus faible. Contrairement aux coûts précédemment cités qui peuvent être calculés localement, ce critère de coût est global ; comme l'algorithme de descente globale de gradient est très lent, durant nos simulations, nous ne calculons ce gradient que sur une partie des exemples. En revanche, l'évaluation du trio est globale. Nous avons également utilisé l'algorithme de la poche ("pocket algorithm") de manière à conserver à chaque étape la meilleure configuration de paramètres.

D'autres choix concernent la stratégie de l'*arrêt de l'apprentissage d'un trio* ou *d'une branche*. Par exemple, comme critère d'arrêt de l'apprentissage de chaque trio, on teste si l'erreur en apprentissage est inférieure à une borne choisie à l'avance et si cette erreur varie peu. Le test d'arrêt de l'apprentissage d'une branche, c'est-à-dire la décision qui fixe si l'on doit remplacer un nœud fils par un trio de nœud, peut être effectué en mesurant les performances du nœud appris soit sur les exemples d'apprentissage accessibles au niveau du nœud (ce qui correspond aux résultats de simulation présentés dans la troisiéme partie de l'article), soit sur le sous-ensemble des exemples (accessibles à ce nœud) d'un ensemble dit de test, différent de la base d'apprentissage et différent de la base de validation. Les autres choix concernent les paramètres habituels d'un algorithme d'apprentissage, en particulier le pas ou gain d'apprentissage.

Références

[1] S-I. Amari. Mathematical foundations of neurocomputing. In *Proceedings of IEEE*, volume 78, September 1990

[2] S. I. Amari and N. Murata. Statistical theory of learning curves under entropic loss criterion. *Neural Computation*, 5:140–153, 1993.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[4] Monrocq C. Sélection de modèles linéaires et non linéaires. *Modulad*, pages 1–22, décembre 1994.

[5] C. Cortes, L. D. Jackel, S. Solla, V. Vapnik, and J. S. Denker. Learning curves: asymptotic values and rates of convergence. In *Neural Network for Computing Conference, Snowbird, Utah*, 1993.

[6] C. Cortès, L. D. Jackel, S. Solla, V. Vapnik, and Denker J. S. Learning curves: asymptotic values and rates of convergence. In *NIPS 6*, page 1. 1994.

[7] F. d'Alché Buc. *Modèles neuronaux et algorithmes constructifs pour l'apprentissage de règles de décision*. Université de Paris XI, 1993.

[8] F. d'Alché Buc and J.-P. Nadal. Asymptotic performances of a constructive algorithm. *Neural Processing Letters*, 2(2):1–4, 1995.

[9] F. d'Alché Buc, D. Zwierski, and J.-P. Nadal. Hybrid trees for supervised learning: towards extraction of decision rules. In *ECAI '94 Workshop on Combining Symbolic and Connectionist systems*, pages 133–142, august 1994.

[10] F. d'Alché Buc, D. Zwierski, and J.-P. Nadal. Trio-learning: a new strategy for building hybrid neural trees. *International Journal of Neural Systems*, 5(4):259–274, december 1994.

[11] Marchand M., Golea. A growth algorithm for neural network decision trees. *Europhys. Lett.*, (12):105–109, 1990.

[12] Friedman J. H. Multivariate adaptive regression splines. *The Annals of statistics*, (19):1–141, 1991.

[13] Guo H. and Gelfand S. B. Classification trees with neural network feature extraction. *IEEE Trans. on Neur. Net.*, 3(6), november 1992.

[14] D. Haussler, M. Kearns, H. S. Seung, and N. Tishby. Rigourous learning curves bounds from statistical mechanics. 1992.

[15] Sirat J. and Nadal J.-P. Neural trees: a new tool for classification. *Network*, (1):198–209, 1990.

[16] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[17] Dreyfus G., Kneri S., Personnaz L. Single layer learning revisited: a stepwise procedure for building and training a neural network. In F. Fogelman and J. Hérault, editors, *NATO Workshop Les Arcs*. New York Springer, 1989.

[18] Bichsel M. and Seitz P. Minimum class entropy: a maximum information approach to layered networks. *Neural Networks*, 2:133–141, 1989.

[19] Frean M. The upstart algorithm: a method for constructing and training feedforward neural networks. *Neurocomputing*, (2):423–438, 1990.

[20] Barron A. R. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, special issue on the Fourth ACM Workshop on Computational Learning Theory, 1991, 1993.

[21] Quinlan R. Induction of decision trees. *Machine Learning*, (1):81–106, 1986.

[22] J. Rissanen. *Stochastic complexity in statistical inquiry*. Word Scientific, Teaneck,New Jersey, 1989.

[23] H. Sompolinski, H. S. Seung, and N. Tishby. Learning from examples in large neural networks. *Phys. Rev. Lett.*, 65(13):1683–1686, 1990.

[24] Vapnik V., Levin E., and Le Cun Y. Measuring the vc-dimension of a learning machine, neural computation. *Neural Computation*, 6:851–876, 1994.

[25] V. Vapnik. *Estimation of dependances based on empirical data*. Springer-Verlag, 1984.

[26] M. Wynne-Jones. Constructive algorithm and pruning: improving the multi-layer perceptron, proc. of imacs, 1991.