Une nouvelle fonction de coût régularisante dans les réseaux de neurones artificiels : Application à la classification

Vincent Lemaire[1], Olivier Bernier, Daniel Collobert et Fabrice Clérot

[1] France-Télécom CNET DTL/DLI/TNT
Technopole Anticipa,
2 Avenue Pierre Marzin
22307 Lannion cedex FRANCE
Email : vins.lemaire@cnet.francetelecom.fr

Résumé

Une nouvelle méthode permettant un contrôle de la forme de la distribution des erreurs est présentée. Cette méthode est basée sur une nouvelle fonction de coût qui minimise la variance de l'erreur quadratique moyenne. Elle permet d'améliorer les performances en généralisation en maximisant la marge des MLP classifieurs. Cette méthode est testée et comparée avec succès à la méthode "classique" qui minimise l'erreur quadratique sur un problème de détection de visage à un MLP et sur un problème benchmark avec la méthode appelée Bagging.

1 Présentation

1.1 Motivation

Le choix du nombre de paramètres d'un réseau de neurones est primordial pour ses capacités d'apprentissage et de généralisation. Si le modèle est trop simple, il sera incapable d'apprendre la fonction souhaitée, s'il est trop complexe, il sera incapable de généraliser (c'est-à-dire de donner une réponse correcte pour un exemple non appris), bien que capable sans peine de passer par tous les points de l'ensemble d'apprentissage. En fait dans le second cas, l'espace des solutions admissibles est beaucoup trop grand. C'est pourquoi la fonction de côut est l'un des plus importants facteurs contrôlant les performances d'un multi-perceptron.

La fonction de coût la plus utilisée est la fonction quadratique [14] avec pour critère d'arrêt l'erreur quadratique moyenne obtenue sur l'ensemble de validation, nous appellerons cette méthode dans tous ce qui suit "MSE".

De nombreux algorithmes ont été proposés, pour accélérer l'apprentissage, pour trouver le "bon" pas d'apprentissage, pour trouver la bonne méthode d'arrêt de l'apprentissage, ..., mais ils sont très souvent destinés à la minimisation de cette fonction de coût MSE. On peut noter cependant que bien qu'elle vise à réduire la norme des erreurs commise par le réseau, elle n'a aucun pouvoir de contrôle sur la distribution de ces dernières au cours de l'apprentissage. Or, contrôler la "forme" de la distribution des erreurs lors de la phase d'apprentissage nous a semblé être un problème de grande importance notamment dans le cas des problèmes de classification ou la notion de marge s'impose d'elle-même. En d'autres termes est-il possible de contrôler la convergence de l'algorithme d'apprentissage de manière à contrôler la forme de la dite distribution et ainsi d'améliorer la robustesse de l'apprentissage ? Tout naturellement on pense alors à une nouvelle fonction de coût ou de régularisation. Celle que nous allons décrire peut être utilisée comme l'une ou comme l'autre.

Cette nouvelle fonction de coût est décrite dans la présente section ainsi que ses conditions d'utilisation. La deuxième section est une comparaison entre la nouvelle fonction de coût et la fonction de coût MSE. Ce, dans le cadre d'un problème de détection de visages à l'aide d'un unique multi-perceptron.

Après cette comparaison, dans la troisième section on s'attachera à réaliser une autre comparaison entre les deux fonctions de coût sur un autre problème de classification qualifié de benchmark. Le but sera alors de comparer les résultat obtenus par la technique de Bagging mise au point par Breiman [5] [3] et plus particulièrement les résultats affichés dans [12], ce avec la nouvelle fonction de coût et la fonction de coût MSE puis nous conclurons.

1.2 Description

Une manière de contrôler la forme de la distribution des erreurs au cours de l'apprentissage est la prise en compte d'un moment d'ordre 4 des erreurs : la variance des erreurs quadratiques, en plus de l'erreur quadratique classique.

La fonction de coût à minimiser est :

$$C^x = \sum_{i \in O} (d_i^x - s_i^x)^2 + \sum_{i \in O} \left(\frac{1}{P} \sum_{a=1}^{P} \left((d_i^a - s_i^a)^2 - \frac{1}{P} \sum_{b=1}^{P} (d_i^b - s_i^b)^2 \right)^2 \right) \quad (1)$$

qui peut s'écrire sous la forme de l'addition de deux coûts :

$$C^x = \sum_{i \in O} C_{quad}^x + \sum_{i \in O} C_{var}^x \quad (2)$$

où P est l'ensemble des exemples d'apprentissage, O est l'ensemble des neurones de sortie, s_i^x est la valeur du neurone de sortie i après la présentation de l'exemple x et d_i^x est la valeur désirée pour le neurone correspondant.

Il existe deux principales méthodes de modification des poids du réseau liées à la manière de calculer le gradient, soit en utilisant un gradient total qui est une méthode globale ou encore appelée batch, soit en utilisant un gradient partiel qui est appelée méthode stochastique. Bottou, L. présente dans [4] une comparaison des deux méthodes et il montre que la méthode stochastique est plus "rapide" que la méthode globale. Les propriétés de convergence de la backprop "standard" (telle que proposée dans [14, 10]), version stochastique et "batch" sont discutées dans [2].

Nous avons utilisé la méthode stochastique et dans ce cas le gradient attaché à neurone de sortie i pour une présentation d'un exemple x :

$$G_i^x = \frac{\partial C_i^x}{\partial a_i^x} = \frac{\partial}{\partial a_i^x} \left((d_i^x - s_i^x)^2 \right) + \frac{\partial}{\partial a_i^x} \left(\frac{1}{P} \sum_{a=1}^{P} \left((d_i^a - s_i^a)^2 - \frac{1}{P} \sum_{b=1}^{P} (d_i^b - s_i^b)^2 \right)^2 \right) \quad (3)$$

donc, si f est la fonction de transfert d'un neurone :

$$G_i^x = \frac{\partial C_i^x}{\partial a_i^x} = -2f'(a_i^x)(d_i^x - s_i^x) - \frac{4}{P} f'(a_i^x)(d_i^x - s_i^x) \left((d_i^x - s_i^x)^2 - \frac{1}{P} \sum_{e=1}^{P} (d_i^e - s_i^e)^2 \right) \quad (4)$$

gradient qui peut être écrit sous la forme :

$$G_i^x = -2f'(a_i^x)(d_i^x - s_i^x) - \frac{4}{P} f'(a_i^x)(d_i^x - s_i^x) \left((d_i^x - s_i^x)^2 - MSE \right) \quad (5)$$

ou encore :

$$G_i^x = G_{i_{quad}}^x (1 + \gamma) \quad (6)$$

avec

$$\gamma = \frac{2}{P} \left((d_i^x - s_i^x)^2 - MSE \right) \quad (7)$$

ou MSE représente l'erreur quadratique moyenne obtenue sur les exemples au cycle d'apprentissage du calcul de gradient.

On s'aperçoit que ce gradient peut être vu comme le gradient habituel, dû à l'erreur quadratique, mais corrigé, pondéré, par γ qui représente une mesure entre l'erreur quadratique commise sur l'exemple x et l'erreur quadratique moyenne commise sur l'ensemble des exemples. Dans tout ce qui suivra nous appellerons cette méthode d'apprentissage "VMSE" pour *"***V***ariance and* **M***ean* **S***quared* **E***rror"*. Le principe du calcul du gradient d'un neurone caché est à l'identique de la rétropropagation de l'erreur quadratique.

1.3 La loi de modification des poids

En reprenant l'équation (5) on pose :

$$G_i^x = G_{i_{quad}}^x + G_{i_{var}}^x \quad (8)$$

avec

$$G_{i_{quad}}^x = -2f'(a_i^x)(d_i^x - s_i^x) \quad (9)$$

$$G_{i_{var}}^x = -\frac{4}{P} f'(a_i^x)(d_i^x - s_i^x) \left((d_i^x - s_i^x)^2 - MSE \right) \quad (10)$$

La loi de modification des poids devient alors :

$$\Delta w_{ij}^{t+1}$$

$$= \alpha_{quad} G^x_{i_{quad}} s_j + \alpha_{var} G^x_{i_{var}} s_j + \beta \Delta w_{ij}^t \qquad (11)$$

avec α_{quad} le pas d'apprentissage sur l'erreur quadratique, α_{var} le pas d'apprentissage sur la variance de l'erreur quadratique et β l'inertie (terme qui permet d'accélérer la convergence [11]).

1.4 Actualisation de la MSE

Du fait de la nécessité de calculer l'erreur quadratique moyenne et la variance de l'erreur quadratique moyenne après chaque modification des poids le temps de calcul peut devenir très important. En effet pour chaque présentation d'un exemple il faut réaliser P propagation et 1 rétropropagation. Aussi afin de réduire les temps de calcul nous utiliserons l'algorithme suivant qui, comme nous le verrons plus loin permet d'introduire une contrainte plus forte sur le contrôle de la forme de la distribution des erreurs et d'ajouter de la stabilité au processus d'apprentissage :

- ⋆ pour toutes les itérations
  - pour tous les exemples x
    - calcul de $f_w(x)$
    - calcul de G^x_{quad} et G^x_{var} pour les neurones de sorties
    - calcul de G^x_{quad} et G^x_{var} pour les neurones cachés
    - modifications des poids
  - calcul de l'erreur quadratique moyenne et de la variance de l'erreur quadratique moyenne

1.5 Normalisation des pas d'apprentissage

Dans les différentes études comparatives que nous allons présenter ci-après et pour étudier l'influence du terme ajouté dans la fonction de coût on pose :

$$\nu = (\alpha_{var}{}'/\alpha_{quad}) = (\alpha_{var} N/\alpha_{quad}) \qquad (12)$$

ou N représente la taille de l'ensemble utilisé dans le calcul de la MSE.

2 Étude comparative : Classification

2.1 Introduction

Dans le cadre d'une classification par discrimination le but recherché est la détermination d'un classifieur qui, à chaque observation x (vecteur de $\mathbb{R}^p$) associe une classe parmi un ensemble fixé de classes. Nous nous intéresserons particulièrement ici au cas où le classifieur est un perceptron multi-couches et où on utilise q neurones de sortie (où q est le nombre de classes). Le codage habituellement utilisé consiste alors à attribuer le code $(d_1, ..., d_q)$ à la classe h avec $d_i = a$ si $i = h$, $d_i = b$ si $i \neq h$ et $a \geq b$. Les valeurs habituellement utilisées sont $(a, b) = (+1, -1)$ ou $(1, 0)$ ou encore $(0.9, 0.1)$ ... La sortie du réseau est un vecteur $\mathbf{y}$ de dimension q, cependant si $q = 2$ on peut se contenter d'utiliser un réseau à une seule sortie.

Pour les multi-perceptrons qui utilisent des fonctions d'activation et des poids à valeurs continues, quelques valeurs particulières des sorties du réseaux de neurones sont utilisées comme valeurs désirées pour les différentes classes du problème de classification à traiter. Par conséquent on observe deux types d'erreurs : l'erreur d'apprentissage liée à la minimisation de la fonction de coût que nous appellerons dans ce qui suit l'erreur d'estimation qui est la différence entre ce qu'on voulait obtenir en sortie du réseau, pour un exemple donné, et ce qu'on obtient réellement à la fin de l'apprentissage ; et l'erreur de classification quand l'erreur d'estimation est supérieure à un seuil pré-établi.

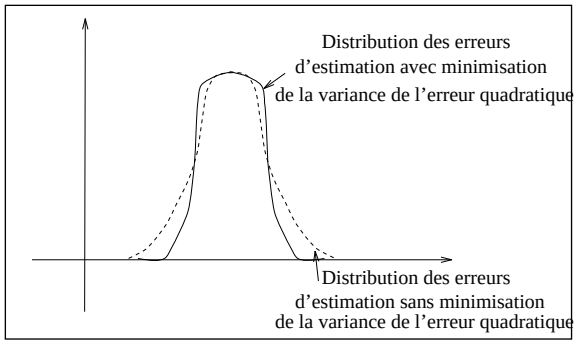

FIG. 1 - Influence de la minimisation de la variance de l'erreur quadratique sur la distribution des erreurs d'estimation.

Les bornes de l'erreur de généralisation d'un système composite de classifieurs ont été précédemment établies [15][8] et sont basées sur la notion de marge de classification. La taille de la marge dépend à la fois de l'erreur quadratique moyenne obtenue et de la distribution des erreurs d'estimation soit donc par conséquent de la variance de l'erreur quadratique obtenue sur chaque classe

(voir figure 1). Ainsi la performance en terme de bonne classification dépend de la forme particulière de la distribution des erreurs d'estimation. Par conséquent, le choix d'une fonction de coût appropriée, de manière à contrôler la forme de la distribution des erreurs d'estimation peut être crucial pour obtenir une solution raisonnable au problème de classification posé.

Nous proposons ici d'utiliser la nouvelle fonction de coût décrite précédemment. En effet cette méthode approche le problème en prenant en compte un moment d'ordre 4, la variance de l'erreur quadratique, introduit dans la fonction de coût à minimiser lors de la phase d'apprentissage pour améliorer les résultats en généralisation. Cette approche peut être utilisée lorsque des méthodes [15] [5] utilisant plusieurs réseaux de neurones afin d'améliorer la marge de classification nécessiteraient un temps de calcul trop long pour des applications industrielles. Nous étudierons néanmoins son intérêt dans le cadre de la méthode dite du Bagging un peu plus loin.

Considérons un problème de classification à deux classes C_1, C_2 classifiées à l'aide d'un réseau discriminant à une sortie. Le but de la phase d'apprentissage est d'obtenir les sorties suivantes pour le réseau :

- si $x \in C_1$ alors $f_w(x) = d_1$
- si $x \in C_2$ alors $f_w(x) = d_2$

avec x le vecteur d'entrée, d_1, d_2 les sorties désirées respectivement pour un exemple de la classe C_1, C_2 et $f_w(x)$ la réponse donnée par le réseau de neurones. A la fin de la phase d'apprentissage, ce réseau de neurones a une erreur quadratique moyenne m_1 sur la classe C_1 avec une variance σ_1^2 et une erreur quadratique moyenne m_2 sur la classe C_2 avec une variance σ_2^2 tel que :

$$\sigma_1^2 = \frac{1}{n_1} \sum_{\substack{a=1 \\ \mathrm{a} \in \mathrm{C}_1}}^{n_1} \left[(d^a - s^a)^2 - \frac{1}{n_1} \sum_{\substack{b=1 \\ \mathrm{b} \in \mathrm{C}_1}}^{n_1} (d^b - s^b)^2 \right]^2 \quad (13)$$

$$\sigma_2^2 = \frac{1}{n_2} \sum_{\substack{c=1 \\ \mathrm{c} \in \mathrm{C}_2}}^{n_2} \left[(d^c - s^c)^2 - \frac{1}{n_2} \sum_{\substack{d=1 \\ \mathrm{d} \in \mathrm{C}_2}}^{n_2} (d^d - s^d)^2 \right]^2 \quad (14)$$

ou :

- n_p est le nombre d'exemples de la classe C_p ;
- s^x est la valeur de la sortie du réseau pour l'entrée x ;
- d^x est la sortie désirée pour l'entrée x.

Comme décrit en 1.2 on peut prendre en compte la minimisation de la variance de l'erreur quadratique et, par extension, des variances σ_1^2 et σ_2^2, en ajoutant à l'habituelle fonction de coût, basée que sur l"erreur quadratique, un terme associé à la variance de l'erreur quadratique observée sur chaque classe. Le but étant ici d'augmenter la marge de classification.

L'expression de la nouvelle fonction de coût sur un problème de classification à deux classes devient :

$$C^x = (d^x - s^x)^2 + \frac{1}{n_1} \sum_{\substack{a=1 \\ a \in C_1}}^{n_1} \left[(d^a - s^a)^2 - \frac{1}{n_1} \sum_{\substack{b=1 \\ b \in C_1}}^{n_1} (d^b - s^b)^2 \right]^2 + \frac{1}{\mathrm{n}_2} \sum_{\substack{\mathrm{c}=1 \\ \mathrm{c} \in \mathrm{C}_2}}^{\mathrm{n}_2} \left[(\mathrm{d}^\mathrm{c} - \mathrm{s}^\mathrm{c})^2 - \frac{1}{\mathrm{n}_2} \sum_{\substack{\mathrm{d}=1 \\ \mathrm{d} \in \mathrm{C}_2}}^{\mathrm{n}_2} (\mathrm{d}^\mathrm{d} - \mathrm{s}^\mathrm{d})^2 \right]^2 \quad (15)$$

L'expression du gradient de la fonction de coût C pour le neurone de sortie i et un exemple $x \in C_1$ est :

$$\left.\frac{\partial C^x}{\partial w_{ij}}\right|_{k \in C1} = \frac{\partial}{\partial w_{ij}} \left[(d^x - s^x)^2 \right] + \frac{\partial}{\partial w_{ij}} \left[\frac{1}{n_1} \sum_{\substack{a=1 \\ a \in C_1}}^{n_1} \left[(d^a - s^a)^2 - \frac{1}{n_1} \sum_{\substack{b=1 \\ b \in C_1}}^{n_1} (d^b - s^b)^2 \right]^2 \right] + \frac{\partial}{\partial w_{ij}} \left[\frac{1}{n_2} \sum_{\substack{c=1 \\ c \in C_2}}^{n_2} \left[(d^c - s^c)^2 - \frac{1}{n_2} \sum_{\substack{d=1 \\ d \in C_2}}^{n_2} (d^d - s^d)^2 \right]^2 \right] \quad (16)$$

Le gradient, qui ne dépend par de la variance observée sur la classe C_2, est par conséquent :

$$\left.\frac{\partial C^x}{\partial w_{ij}}\right|_{k \in C1} = s_j^x(-2f'(a^x)(d^x - s^x) - s_j^x \frac{4}{n_1} f'(a^x)(d^x - s^x)) \left[(d^x - s^x)^2 - \frac{1}{n_1} \sum_{\substack{e=1 \\ e \in C_1}}^{n_1} (d^e - s^e)^2 \right] \quad (17)$$

ou a^x est la valeur de la somme pondérée à l'entrée du neurones i pour l'exemple x.

Le calcul est à l'identique pour un exemple appartenant à la classe C_2. On retrouve donc la formulation proposée en 1.3 et le calcul du gradient attaché aux neurones cachés ne change pas dans sa forme sauf qu'il comprend deux termes. La loi de modification des poids est :

$$\Delta w_{ij}^{t+1} = \alpha_{quad} G_{quad}^x + \alpha_{var} G_{var}^x + \beta \Delta w_{ij}^t \quad (18)$$

avec α_{quad} le pas d'apprentissage sur l'erreur quadratique, α_{var} le pas d'apprentissage sur la variance de l'erreur quadratique et β l'inertie. On peut noter ici qu'il est possible de différencier le pas d'apprentissage sur la variance de l'erreur quadratique de la classe C_1 par rapport à la classe C_2 (et vice versa). Il est aussi possible de ne calculer qu'une seule variance, indépendamment à la classe, et chercher à ne minimiser que cette variance "globale".

2.2 Conditions expérimentales

Le problème de classification à deux classes que nous allons utiliser pour notre étude comparative est un problème de détection de visages. La nouvelle fonction de coût est testée sur le "pré-réseau" de l'application MULTRAK [1], qui est un système temps réel pour la détection et le suivi automatique de personnes lors d'une vision conférence. Ce système est capable de détecter et de suivre continuellement la position des visages présents dans son champ de vision. Le coeur du système est un réseau de neurones modulaire [6] qui détecte les visages avec précision. Le pré-réseau est utilisé comme filtre et doit être beaucoup plus rapide que le réseau modulaire sans dégrader le taux de détection des visages pour l'ensemble du système. Pour des questions de performances temps réel, la vitesse du pré-réseau est critique et impose de n'utiliser qu'un seul réseau de neurones discriminant.

Nous avons donc entraîné deux pré-réseaux en tant que détecteur de visages : un en utilisant la nouvelle fonction de coût décrite précédemment et l'autre uniquement à l'aide de l'erreur quadratique. Chaque réseau de neurones est un multi-perceptron, utilisant des fonctions d'activation sigmoïdales, possédant 300 entrées (correspondant à une image de 15x20 pixels), une couche cachée de 8 neurones et une sortie. La base de donnée utilisée est constituée de 3 parties :

- Ensemble d'apprentissage : 7000 visages de face ou tournés et 7000 non visages ;
- Ensemble de validation : 7000 visages de face ou tournés et 7000 non visages ;
- Ensemble de test : 7000 visages de face ou tournés et 7000 non visages.

Pour comparer les deux fonctions de coût utilisées, différentes expérimentations ont été réalisées. Pour chaque expérimentation, 50 apprentissages ont été réalisés avec différentes initialisations des poids. Ceci permet d'obtenir, pour chaque condition expérimentale, la moyenne et l'intervalle de confiance de chaque résultat obtenu. Chaque apprentissage a été stoppé quand le coût sur l'ensemble de validation ne diminuait plus depuis 200 itérations. Dans ce cas l'erreur quadratique moyenne, la variance de l'erreur quadratique sur chaque classe, la marge et le taux de détection sont calculés sur la meilleure configuration des poids pour cet apprentissage sur chaque ensemble (apprentissage, validation, test) et sont détaillés ci-après.

Dans les parties suivantes nous étudions et comparons les deux fonctions de coût. Nous montrons que si le pas d'apprentissage sur le terme de variance ajouté est bien choisi, la variance sur l'ensemble d'apprentissage diminue ce qui accroît la marge sur cet ensemble et les performances en classification sur l'ensemble de test. Dans les figures qui vont suivre les résultats basés sur la nouvelle fonction de coût sont étiquetés "VMSE" et ceux basés uniquement sur l'erreur quadratique 'MSE'.

2.3 L'influence du terme sur la variance

Dans cette partie le paramètre α_{quad} relié à l'erreur quadratique a une valeur constante de 10^{-2}. L'influence de ν est examiné sur l'intervalle $[10^{-4} : 10^2]$ pour évaluer comment le gradient ajouté interagit avec le gradient de l'erreur quadratique. La valeur du terme sur l'inertie, β, a elle été fixée à 0.9. Les comparaisons sont réalisées pour l'erreur quadratique moyenne globale (pour tous les exemples quel que soient leurs classes d'appartenance) et pour la variance de l'erreur quadratique sur chaque classe. Les résultats pour le pré-réseau basé que sur l'erreur quadratique sont constants puisque α_{quad} est constant.

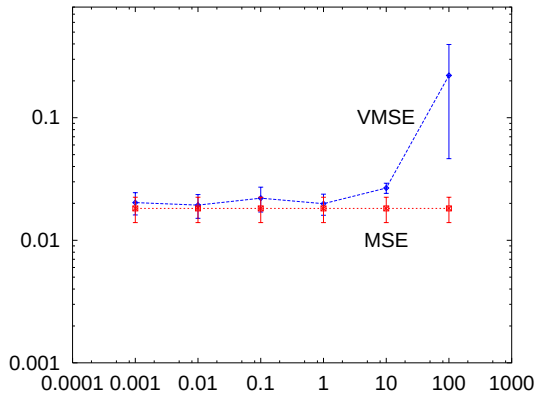

FIG. 2 - L'erreur quadratique moyenne globale sur l'ensemble d'apprentissage avec les deux fonctions de coût en fonction de ν.

La figure 2 montre les résultats obtenus sur l'erreur quadratique moyenne globale avec les deux fonctions de coût sur l'ensemble d'apprentissage. Pour $\nu \in [10^{-4} : 10]$, les deux fonctions de coût fournissent à peu près les mêmes résultats avec le même intervalle de confiance. Par contre, pour $\nu = 100$, l'erreur quadratique moyenne globale augmente beaucoup avec la nouvelle fonction de

coût. On voit ici que dans ce cas, $\alpha_{var}^{'}$ est trop grand, comparé à α_{quad}. La minimisation de la variance de l'erreur quadratique au cours de l'apprentissage empêche la minimisation de l'erreur quadratique et le réseau de neurones répond alors toujours le même résultat.

Les figures 3 et 4 montrent les résultats obtenus sur la variance de l'erreur quadratique associée à chaque classe de l'ensemble d'apprentissage. Pour $\nu \in [10^{-4} : 10^{-1}]$ les deux fonctions de coût exhibent des performances similaires. Pour $\nu \in [10^{-1} : 10]$ la nouvelle fonction de coût réduit les variances jusqu'à obtenir un gain de 37 % par rapport à la fonction de coût standard et ce avec un intervalle de confiance du même ordre. Par contre, pour $\nu = 100$, les variances sont aussi améliorées mais avec un intervalle de confiance inadéquat.

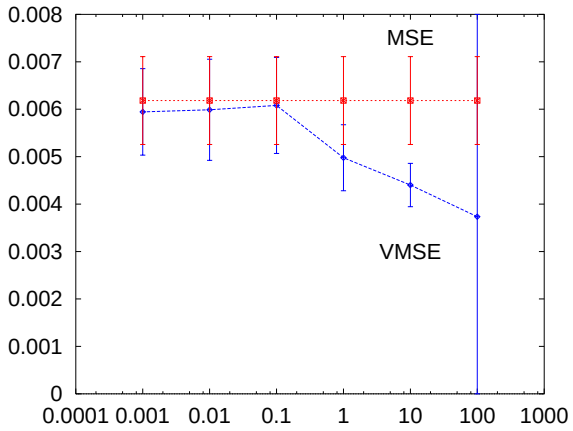

FIG. 3 - La variance de l'erreur quadratique pour la premièr classe (visages) sur l'ensemble d'apprentissage avec les deux fonctions de coût en fonction de ν.

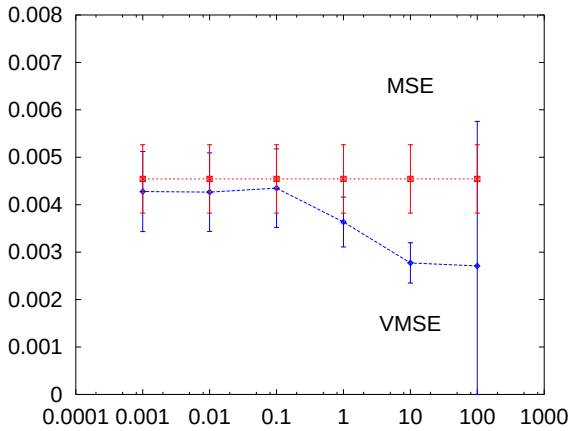

FIG. 4 - La variance de l'erreur quadratique pour la deuxième classe (non visages) sur l'ensemble d'apprentissage avec les deux fonctions de coût en fonction de ν.

Ces résultats montrent que le terme de variance ajouté interagit avec le terme basé sur l'erreur quadratique. S'il est du même ordre il permet d'améliorer les résultats obtenus sur la variance de l'erreur quadratique de chaque classe. Une valeur bien choisie du pas d'apprentissage $\alpha_{var}^{'}$ améliore les résultats sur la variance de l'erreur quadratique associée à chaque classe sans dégrader l'erreur quadratique moyenne globale.

2.4 Augmentation de la marge

Une seconde expérimentation montre la relation entre la minimisation de la variance de l'erreur quadratique et la maximisation de la marge pour $\nu = 1$ (α_{quad}=0.01; $\alpha_{var}^{'}$=0.01).

Pour quantifier le pourcentage de la population à l'intérieur de la marge, proche de la frontière de décision, et par conséquent le taux de bien classés à l'extérieur de la marge nous avons déterminé un seuil θ (voir figure 5) pour différentes valeurs de la marge. Ce seuil est réglé de manière à obtenir le meilleur taux de détection en fonction de la marge sur l'ensemble d'apprentissage, sachant qu'un exemple est considéré bien classé si :

- $f_w(k) \leq \theta$ et k $\in C_1$
- $f_w(k) \geq \theta$ + Marge et k $\in C_2$

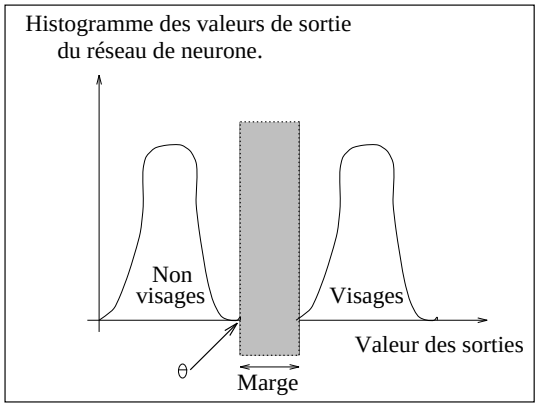

FIG. 5 - Explication du taux de bien classés en fonction de la marge

La figure 6 montre que pour une marge donnée le taux de détection avec la nouvelle fonction de coût est meilleur qu'avec la fonction de coût standard. Par conséquent pour un même taux de détection la marge est augmentée.

Par exemple, pour un taux de détection de 98.5 % (voir figure 6) la nouvelle fonction de coût fournit une marge de 0.17 ± 0.01 alors que la fonction de coût standard ne fournit que 0.12 ± 0.04. L'effet de la minimisation des variances est clairement visible ici. Il est de pousser la distribution des erreurs d'estimation sur les visages vers la droite et celle des non visages vers la gauche (voir figure 5).

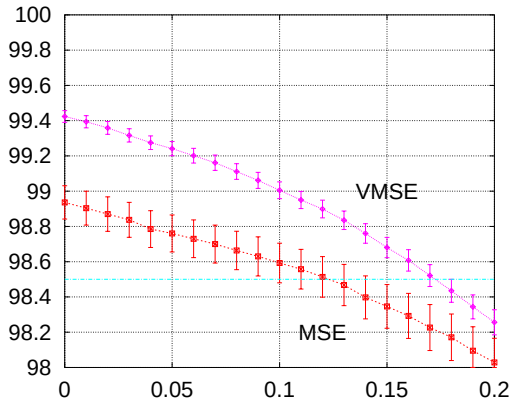

FIG. 6 - Le taux de bien classé en fonction de la marge pour l'ensemble d'apprentissage.

Le deuxième but est donc atteint : la minimisation de la variance de l'erreur quadratique sur chaque classe permet d'améliorer la marge de classification.

2.5 Une meilleure marge accroît les performances en généralisation

Cette partie montre que la maximisation de la marge sur l'ensemble d'apprentissage améliore les performances en généralisation (sur l'ensemble de test). La figure 7 montre l'effet d'une marge améliorée : la différence entre les deux courbes représente l'amélioration obtenue sur la marge. Avec la fonction de coût MSE et pour un taux de détection de 99.5 % le taux de fausse alarme est de 8 % alors qu'avec la nouvelle fonction de coût ce taux est de seulement 5 % ce qui représente une amélioration de 37 %.

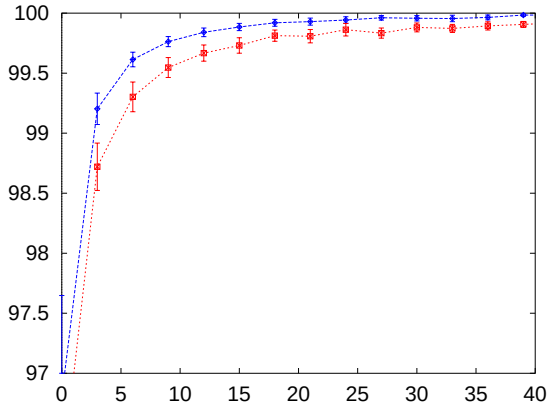

FIG. 7 - Le taux de détection pour les visages sur l'ensemble de test en fonction du taux de fausse alarme (non visage classés comme visages) pour les deux fonctions de coût.

2.6 Discussion

Une nouvelle méthode est ici proposée et testée pour améliorer les performances en généralisation sur un problème de classification à l'aide de multi-perceptron. Cette méthode est utilisée dans le réseau détecteur de visages appelé pré-réseau de l'application MULTRAK [1]. L'utilisation de notre nouvelle fonction de coût a permis d'améliorer les performances du pré-réseau comparé à la fonction de coût basée uniquement sur l'erreur quadratique : le taux de fausse alarme est réduit de 20 % (sur l'ensemble de test A de la base de donnée du CMU) pour le même taux de détection.

Cette méthode a été appliquée à un problème de classification à deux classes mais peut être étendue à des problèmes de classification avec plus de classes. Elle peut aussi être utilisé dans les méthodes qui utilisent plusieurs réseaux de neurones pour améliorer la généralisation comme nous allons le voir dans le problème suivant.

3 Étude comparative : Bagging

3.1 Introduction

Bagging, acronyme de Bootstrap Aggregating, est une procédure de stabilisation ([3] p55) qui émule la possession de répliques indépendantes d'un ensemble d'apprentissage. C'est une méthode assez récente [5] destinée à améliorer les performances des systèmes classifieur parmi d'autres [15], [8], [9], [13], [16]... Ces différentes méthodes ont montré l'intérêt qu'il y a à générer et à combiner plusieurs classifieurs afin d'augmenter la précision de l'ensemble grâce à la diminution de la variance de l'erreur commise. Le bagging a cependant l'avantage de ne pas avoir qu'une justification empirique mais aussi une justification théorique basée sur l'analyse du dilemme biais - variance [3], [7] et le comportement des systèmes comportant une combinaison de plusieurs classifieurs.

Supposons qu'on possède un ensemble de données d'apprentissage, de taille N, ou chacune de ces données appartient à une classe dont le nombre est K. Et une "machine à apprendre" à l'aide de laquelle on veut construire un classifieur étant donné les données d'apprentissage. Le classifieur obtenu aura alors une précision inhérente aux données apprises. Pour augmenter cette précision on veut pouvoir construire différents classifieurs à partir d'exemples appartenant à la même distribution de probabilité que les premiers. Malheureusement dans de nombreux problèmes la taille de l'ensemble d'apprentissage est limitée et le découper pour construire d'autres classifieurs peut diminuer la précision de chacun. Il s'agit alors de "créer" de nouveaux ensembles d'apprentissage à l'aide d'une approximation

bootstrap et de l'ensemble d'apprentissage que nous qualifierons d'originel. Pour ce faire et construire un deuxième ensemble d'apprentissage de la même taille que l'originel, on réalise N tirages indépendant avec remise dans l'originel. Ce deuxième ensemble sera donc différent de l'originel puisque chaque exemple qu'il contient peut apparaître plusieurs fois. Notons cependant que le deuxième ensemble provient de la même distribution de probabilité que le premier, la fréquence d'apparition des exemples étant juste différente. A présent à l'aide du nouvel ensemble on construit un deuxième classifieur. L'opération complète pouvant être réitérée autant de fois que l'on veut, notons ce nombre T. Finalement on définit le classifieur global comme étant la combinaison des T classifieurs ainsi construits. Sa réponse étant, pour un exemple présenté simultanément à tous les classifieurs construits, la réponse moyenne de ces derniers.

Notons que la nouvelle fonction de coût ne minimise pas la variance telle que considérée dans le dilemme biais - variance (voir [7]). Ici, dans cette étude comparative, le but est de rechercher si la méthode VMSE peut permettre, après avoir améliorer le pouvoir de classification d'un unique classifieur, d'améliorer celui d'une combinaison de classifieurs ou si au contraire elle perd de son intérêt. Autrement dit l'intérêt apporté par l'utilisation des ensembles de réseaux de neurones [5] peut il être augmenté en lui conjuguant la nouvelle fonction de coût proposée ?

3.2 Conditions expérimentales

Le problème qui nous intéresse ici est un problème de classification à deux classes sur un problème lié au crédit[1]. La base de données disponible possède 690 exemples comprenant chacun 15 attributs auxquels s'ajoute l'étiquette d'appartenance à la classe 1 ou 2. Les véritables noms des attributs ont été effacés pour devenir anonymes, on ne peut donc savoir à quoi ils correspondent. Cette base de données est intéressante parce qu'elle comprend un bon mélange d'attributs continus et discrets dont nous présentons la répartition et la normalisation que nous en avons effectué tableau 1. Il y a aussi des données manquantes : 37 exemples (5 %) ont un ou plusieurs attributs manquants (voir tableau 2).

Rappelons que le but ici est de comparer les résultats obtenus par la technique de Bagging mise au point par Breiman [5] [3] et plus particulièrement les résultats affichés dans [12] ce avec la nouvelle fonction de coût et la fonction de coût MSE. La répartition des exemples suivant les deux classes est : 307 exemples appartiennent à la classe 1, soit 44.5 %, et 383 à la classe 2, soit 55.5 %.

[1] On peut trouver ce fichier et d'autre informations sur les conditions de comparaison sur le site : ftp://ftp.ics.uci.edu/pub/machine-learning-databases/credit-screening/

Tab. 1 - La composition du fichier de données et la normalisation associée.

Attribut	Valeur de départ	Normalisation effectuée
A1	a, b	0.5, 1.0
A2	continue	[0.0,1.0]
A3	continue	[0.0,1.0]
A4	u, y, l, t	0.25, 0.50, 0.75, 1.0
A5	g, p, gg	0.3, 0.6, 0.9
A6	c, d, cc, i, j, k, m	0.07, ..., 0.56
	v, q, w, x, e, aa, ff	0.63, ..., 0.98
A7	v, h, bb, j, n, z, dd	0.11, ..., 0.77
	ff, o	0.88, 0.99
A8	continue	[0.0,1.0]
A9	t, f	0.3, 0.7
A10	t, f	0.3, 0.7
A11	continue	[0.0,1.0]
A12	t, f	0.3, 0.7
A13	g, p, s	0.2, 0.5, 0.8
A14	continue	[0.0,1.0]
A15	continue	[0.0,1.0]
A16	attribut de classe	0.1, 0.9
Ax	donnée manquante	-1.0

Tab. 2 - Répartition des attributs manquants.

Attribut	Nombre d'attributs manquants
A1	12
A2	12
A4	6
A5	6
A6	9
A7	9
A14	13

Nous avons utilisé la même procédure d'apprentissage que celle énoncée dans [12], qui est (voir figure 8) :

- création de duplications de la base de données de départ par tirage aléatoire uniforme avec remise ;
- pour chaque nouvelle base ainsi créée :
  - on subdivise la base en 10 blocs de taille égale ;
  - on réalise 10 apprentissages en utilisant 9 des blocs comme ensemble d'apprentissage et le dixième comme ensemble de "test" ;
  - le résultat final pour cette base est le résultat moyen obtenu sur les 10 apprentissages ;

– on donne les résultats obtenus en fonction du nombre de répliques de la base d'origine.

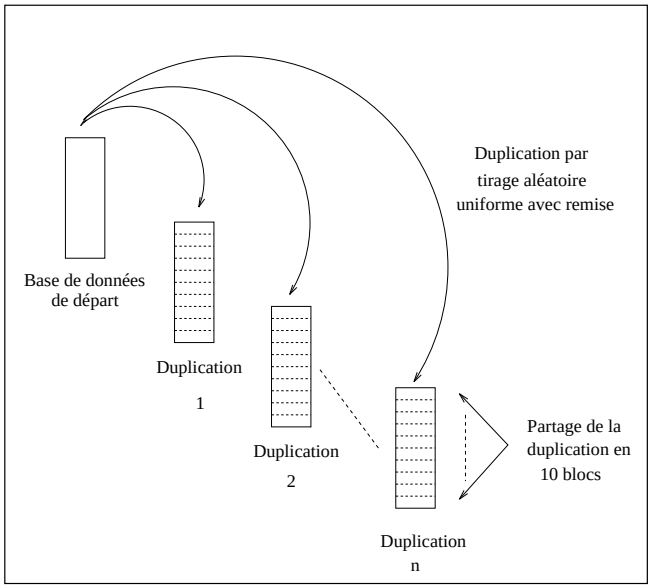

FIG. 8 - La "création" des différentes bases d'apprentissage.

Chaque réseau de neurones possède 15 neurones en entrée, 6 neurones cachés et un neurone en sortie. Nous précisons ici que le critère d'arrêt des différents apprentissages est basé uniquement sur l'erreur quadratique (tout comme les auteurs de [12]). Le terme ajouté dans la nouvelle fonction de coût est donc utilisé comme un facteur de régularisation ce qui ne change en rien la loi de modification des poids énoncé en 1.3. Son influence à été mesurée pour $\nu = 1$ et $\nu = 10$ (α_{quad}=0.01, β=0.9).

Chaque apprentissage a été stoppé quand le coût sur l'ensemble de test ne diminuait plus depuis 200 itérations. Dans ce cas l'erreur quadratique moyenne, la variance de l'erreur quadratique globale et le pourcentage d'erreur sont calculés sur la meilleure configuration des poids pour cet apprentissage sur l'ensemble de test et sont détaillés ci-après.

3.3 Résultats

Les résultats présentés dans les figures 9, 10, 11, 12 sont labélisé 'MSE' pour la fonction de coût n'utilisant que l'erreur quadratique, 'VMSE 1' et 'VMSE 10' respectivement pour $\nu = 1$ et $\nu = 10$.

On précise que pour $\nu = 0.1$ (ou inférieur) les résultats avec la méthode VMSE sont à l'identique de ceux obtenus avec la méthode MSE. Les valeurs de ν ont été choisies en fonction de l'expérience acquise sur le problème précédent de détection de visages.

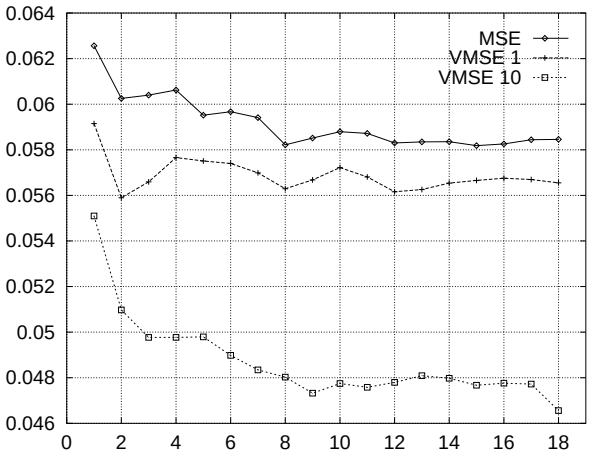

FIG. 9 - Résultats obtenus sur l'erreur quadratique moyenne globale en fonction du nombre de répliques.

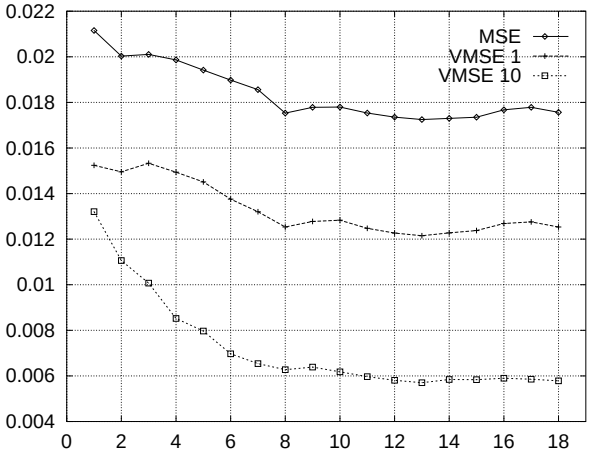

FIG. 10 - Résultats obtenus sur la variance de l'erreur quadratique en fonction du nombre de répliques pour la classe 1.

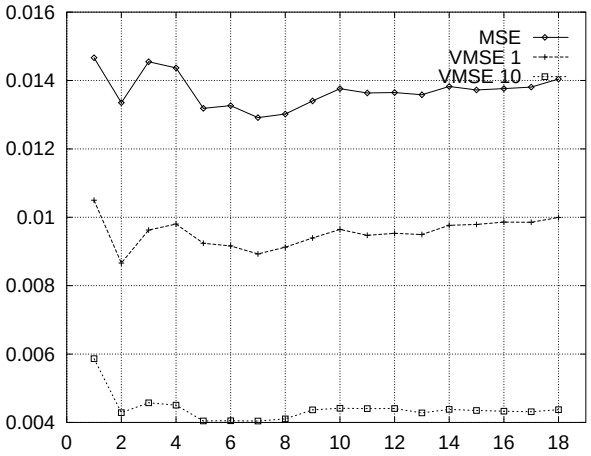

FIG. 11 - Résultats obtenus sur la variance de l'erreur quadratique en fonction du nombre de répliques pour la classe 2.

Plus précisément on trouve figure 9 le résultat obtenu sur l'erreur quadratique moyenne globale, figures 10, 11 le résultat respectivement sur la variance de l'erreur quadratique pour la classe 1, 2 et figure 12 le pourcentage de mal classés en fonction à chaque fois du nombre de répliques utilisées.

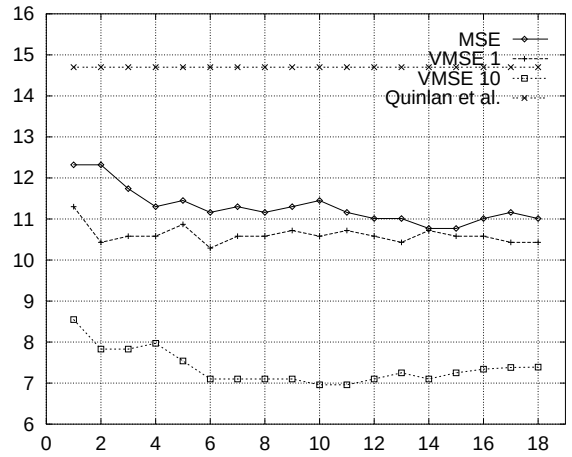

FIG. 12 - Résultats obtenus sur le pourcentage de mal classés en fonction du nombre de répliques.

Sur cette dernière figure on indique le meilleur résultat obtenu par Quilan. [12] ce dont on ne dispose pas pour les trois premiers résultats affichés. Dans le cas de l'utilisation de la fonction de coût MSE ($\nu = 0$) nous n'avons pas exactement retrouvé les résultats de Quilan. ce qui peut provenir d'une erreur de notre part quand à la compréhension des conditions expérimentales. Les résultats trouvés ici avec la méthode MSE sont néanmoins meilleurs et constituent donc de bonnes valeurs de comparaison.

On peut aussi noter qu'un des résultats de Breiman [5] montrant de façon expérimentale que le nombre de répliques nécessaire mais suffisant est de l'ordre de 10 après quoi les performances ne sont que très peu améliorées est retrouvé. Ceci étant plus particulièrement visible sur les figures 9, 10, 13.

3.4 Discussion

On s'aperçoit que les résultats obtenus avec la méthode 'VMSE' sont meilleurs que ceux obtenus avec la fonction 'MSE' tant sur l'erreur quadratique moyenne globale que sur sa variance et sur le pourcentage de mal classés. Pour ce dernier on présente figure 13 le gain obtenu en fonction du nombre de bases dupliquées.

On voit sur cette courbe que l'amélioration apportée en fonction du nombre de répliques utilisées est à peu près identique pour les trois courbes. L'amélioration des performances est conservé lorsque le nombre de répliques utilisées augmente.

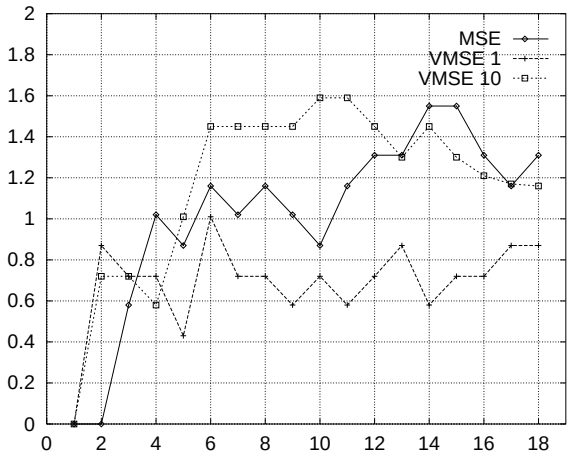

FIG. 13 - Amélioration des résultats obtenus sur le pourcentage de mal classés en fonction du nombre de répliques.

4 Conclusion

On a présenté dans cet article une nouvelle méthode destinée à améliorer les performances en généralisation des multi-perceptrons utilisés en tant que réseau discriminant. On a montré clairement la modification du critère d'apprentissage qui permet de minimiser à la fois les erreurs de classification et les erreurs d'estimation par une minimisation de la variance de l'erreur quadratique.

Cette méthode utilisée en tant que fonction de régularisation est très facilement implémentable même dans les cas où il y a plus que deux classes. En effet il suffit de calculer la MSE sur chaque neurone de sortie du réseau. Par exemple pour un problème à cinq classes et dans le cas où on utilise alors cinq neurones de sortie il faut calculer cinq MSE. Ces MSE ayant été calculées le calcul des cinq gradients G_{quad} est alors immédiat conformément à l'équation (10). Les modifications à apporter dans un programme d'apprentissage, où la méthode MSE était utilisée, sont alors réellement minimes.

Reste à savoir comment choisir de façon simple et efficace les paramètres α_{quad}, α_{var} et β. Pour β nous renvoyons le lecteur à [11] du fait que nous avons remarqué que nombre d'utilisateurs de réseaux de neurones introduisent ce terme dans la loi de modification des poids alors que d'autres n'y trouvent pas un grand intérêt ; expérimentalement parlant. Pour ce qui est de α_{quad} et de α_{var}, le critère de choix, comme nous l'avons montré, est imposé par leur ratio illustré dans cet article par la variable ν.

Une autre question non traitée dans cette article est de savoir si cette méthode peut être appliquée à des problèmes autres que ceux liés à la classification. Le

contrôle de la forme de la distribution des erreurs fonctionne t-il pour des problèmes de régression ou d'approximation de fonctions ? Nous répondrons que d'autres travaux dans ces domaines sont en cours [**?**].

Références

[1] O. Bernier, M. Collobert, R. Féraud, V. Lemaire, J.E. Viallet, and D. Collobert. MULTRAK : a system for Automatic Multiperson Localization and tracking in real-time. In *International Conference on Image Processing*, 1998.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. MA: Athena Scientific, Belmont, 1996. ISBN 1-886529-10-8.

[3] C. M. Bishop, editor. *Neural Networks and Machine Learning*, volume 168 of *F*, chapter I-11. NATO ASI Series, 1997.

[4] L. Bottou. *Une approche théorique de l'apprentissage connectioniste; applications à la reconnaissance de la parole.* PhD thesis, Université de Paris 11, Orsay, France, 1991.

[5] L. Breiman. Bagging predictors. Technical Report TR-421, University of California, Berkley, 1994.

[6] Féraud, R. and Bernier, O. Ensemble and modular approaches for face detection: a comparison. In *Neural Information Processing System*, volume 10, pages 472–478, december 1997.

[7] Geman, S., Bienenstock, E., and Doursat, R. Neural networks and the bias-variance dilemma. *Neural Computation*, 4:1–58, 1992.

[8] S. Holger and Y Bengio. Training method for adaptative boosting of neural networks. In *Neural Information Processing System*, 1998.

[9] R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In Morgan Kauffman, editor, *International Conference on Machine Learning*, volume 12, pages 304–311, San Francisco, 1995.

[10] Y. Le Cun. *Modèles connexionnistes de l'Apprentissage.* PhD thesis, Université Paris 6, 1987.

[11] D. Plaut, S. Nowlan, and G. Hinton. Experiments on learning by backpropagation. Technical Report CMU-CS-86-126, Department of computer science, Carnegie Mellon University, 1986.

[12] J. R. Quilan. Bagging, Boosting and C4.5. Technical report, University of Sydney, 1998.

[13] H. Ragavan and L. Rendell. Lokkahead feature construction for learning hard concepts. In Morgan Kauffman, editor, *International Conference on Machine Learning*, volume 10, pages 252–259, San Francisco, 1993.

[14] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In *Parallel Distribued Processing: Explorations in the Microstructures of Cognition*, volume 1, pages 318–362, 1986.

[15] R. E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin : A new explanation for the effectiveness of voting methods. In *Machines That Learn*, 1997. http://www.research.att.com/ schapire/.

[16] P. E. Utgogg and C. Brodley. An incremental method for finding multivariate splits for decision trees. In Morgan Kauffman, editor, *International Conference on Machine Learning*, volume 7, pages 58–65, San Francisco, 1990.