ICEA 92, TIZI-OUZOU, ALGERIE, 4-6 Mai 1992

Réseaux de Neurones Formels pour le Filtrage Adaptatif en Traitement du Signal Architectures et Algorithmes.

Sylvie Marcos, O.Macchi, C.Vignat

Laboratoire des Signaux et Systèmes, Plateau de Moulon, 91192 Gif sur Yvette, France

G.Dreyfus, L.Personnaz, P.Roussel et O.Nerrand

ESPCI, 10 rue Vauquelin, 75005 Paris, France.

RESUME :

Le travail présenté ici s'insère dans un projet plus important visant à juger de l'apport potentiel des réseaux de neurones pour réaliser des opérations de filtrage non linéaire en traitement du signal. Cette communication propose des architectures de réseaux de neurones ainsi que des algorithmes d'apprentissage, réellement adaptatifs, prenant en compte le caractère temporel des signaux à traiter. Cette approche est originale parce que dans la plupart des applications, les réseaux de neurones ne traitent que des informations statiques. Lors de la communication, une application sur le codage prédictif de la parole par réseaux de neurones sera présentée.

I. INTRODUCTION

Le filtrage adaptatif est un sujet d'intérêt pratique considérable dans de nombreuses applications du traitement du signal telles que la modélisation de signaux et l'identification de systèmes. Ces applications apparaissent dans les domaines des communications, du radar, du sonar, de l'électronique biomédicale et de la géophysique.

Le point commun de ces opérations de traitement de l'information que peuvent réaliser des filtres adaptatifs est leur capacité à apprendre à identifier une fonction mathématique f, à partir d'exemples $\{x_i, f(x_i)\}$, ainsi qu'à poursuivre une éventuelle évolution de f au cours du temps. Classiquement, le filtrage adaptatif utilise une fonction linéaire pour approximer f dont les paramètres (les poids à retard) sont obtenus en optimisant un critère qui mesure la distance entre f et la fonction linéaire. Cette contrainte limite la capacité d'un filtre adaptatif à extraire des informations contenues dans les statistiques d'ordre supérieur à deux des données. Quand un processus est non gaussien, l'utilisation d'un filtre adaptatif linéaire donne une solution sous optimale. Or, il apparait de plus en plus que le caractère non gaussien est un phénomène couramment rencontré dans le domaine des communications et du traitement du signal. Plusieurs solutions ont été proposées pour tenter d'introduire des non linéarités dans le traitement du signal classique : généralisation des filtres linéaires à des filtres polynomiaux (filtres de Volterra), fonctions de base radiales.

Les réseaux de neurones formels (RNF) sont connus pour être des approximateurs universels: toute fonction multivariable non linéaire suffisamment régulière peut être approchée, avec une précision donnée, par un réseau de neurones, pourvu que celui-ci ait une architecture appropriée et qu'il soit soumis à un apprentissage efficace [1]. Il apparait donc important d'étudier leur apport potentiel pour réaliser des opérations de filtrage non linéaire en traitement du signal.

Les RNF ont encore assez peu été utilisés en traitement du signal, au sens où la plupart du temps ils traitent des informations statiques (images, segments de parole, formes) et non le caractère temporel de ces informations. On peut cependant remarquer dans la littérature un intérêt croissant pour ce sujet [2]. Un RNF est un ensemble de cellules non linéaires élémentaires qui sont interconnectées entre elles et réalisent ensemble des opérations complexes d'optimisation, de classification, de reconnaissance de formes. Ce système neuronal est caractérisé par un ensemble de paramètres qui sont les poids des interconnexions entre deux neurones; l'un des problèmes essentiels consiste à déterminer ces poids de façon à ce que le système réalise bien la fonction qu'on lui a demandée. De plus, il est d'un extrême intérêt que ce système puisse acquérir de lui-même, par l'intermédiaire de ces poids, l'information à

stocker. C'est ce qu'on appelle l'apprentissage du RNF. Il est réalisé afin de répondre à un critère que l'on se donne, et qui est, dans le cas de l'apprentissage supervisé, une somme d'erreurs (pondérée ou non) quadratiques sur tous les neurones de sortie du réseau et sur tous les exemples de l'ensemble d'apprentissage.

Un filtre adaptatif est un système optimisé et adapté à l'environnement sans intervention extérieure, et dont l'optimisation en présence de fluctuations de l'environnement au cours du temps est maintenue. Cette adaptation répond à un critère et a la particularité d'être effectuée continuellement pendant toute la durée du fonctionnement du système.

Cette communication traite des réseaux de neurones utilisés comme des filtres adaptatifs non linéaires. Dans la première partie, nous concevons des architectures de réseaux de neurones susceptibles de traiter des signaux temporels. En particulier, nous généralisons la notion de filtre transverse ou récursif au cas de RNF non bouclés (feedforward) et bouclés (feedback). Dans la seconde partie, nous proposons des algorithmes adaptatifs originaux permettant un apprentissage permanent des RNF. Le cas plus délicat des réseaux bouclés est étudié plus particulièrement. Lors de la communication, nous présenterons une application du traitement du signal où l'utilisation des réseaux de neurones est susceptible d'améliorer les performances.

II. LES RESEAUX DE NEURONES UTILISES COMME FILTRES NON LINEAIRES

II.1 Modèle de neurone à temps discret

Un neurone formel est une cellule élémentaire (Fig.1) qui calcule la quantité

$$y(n) = f(v(n)) \ , \qquad v(n) = \sum_{i=1}^{M} c_i \, u_i(n) \tag{II.1}$$

où n indique simplement le numéro du vecteur d'entrée et non pas forcément le temps ; $y(n)$ est la sortie ou état du neurone, $v(n)$ son potentiel d'action, $\{c_i\}$ les poids synaptiques, les $u_i(n)$ sont les entrées externes et f une fonction d'activation. La fonction f pourrait être a priori quelconque, cependant les auteurs utilisent généralement des fonctions impaires, croissantes, à seuils, à saturation ou sigmoïde [1]. On supposera simplement dans la suite que f est dérivable : c'est l'hypothèse qui a permis de concevoir l'algorithme de rétropropagation du gradient [3] et ainsi de redonner un essor à l'étude des RNF. Un RNF est un ensemble de cellules élémentaires définies ci-dessus qui sont fortement connectées entre elles : l'état du neurone j est une entrée du neurone i ; on note c_{ij} le poids synaptique reliant le neurone j au neurone i.

Un filtre (Fig.2) fait correspondre à une suite de données ordonnées dans le temps, $u(0)$, $u(1)$, $u(2)$, … ,$u(n)$, …, une autre suite de données $y(0)$, $y(1)$, $y(2)$, … , $y(n)$, …. Dans le cas d'un filtre linéaire transverse, représenté sur la figure 2, la sortie $y(n)$ s'écrit à chaque instant n,

$$y(n) = C^{\mathrm{T}} U(n), \tag{II.2}$$

avec $U(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}}$ et $C = [c_0, c_1, \ldots, c_{M-1}]^{\mathrm{T}}$. Un filtre linéaire récursif (Fig.3) est déterminé par les relations

$$y(n) = C^{\mathrm{T}} X(n), \tag{II.4}$$

avec $X(n) = [u(n), u(n-1), \ldots, u(n-M+1), y(n-1), \ldots, y(n-N),]^{\mathrm{T}}$ et où cette fois le vecteur des paramètres du filtre est de dimension $M+N$ et est composé de deux blocs A et B de dimensions respectives M et N, désignant respectivement les parties transverses et récursives du filtre: $C = [A^{\mathrm{T}} : B^{\mathrm{T}}]^{\mathrm{T}}$.

Pour réaliser des fonctions de filtrage, il est nécessaire de modéliser le comportement d'un neurone en introduisant des synapses à retard. Dans le cas d'un réseau de plusieurs neurones à retards synaptiques, l'état du neurone i à l'instant n s'écrit

$$z_i(n) = f_i(v_i(n)) \ , \qquad v_i(n) = \sum_{j \in P_i} \sum_{\tau_{ij}=0}^{q_{ij}} c_{ij,\tau}\, z_j(n - \tau_{ij}) \qquad \text{(II.3)}$$

où $v_i(n)$ est le potentiel du neurone i, n est, cette fois, le temps, P_i l'ensemble des indices des neurones dont l'état est l'entrée du neurone i, les τ_{ij} sont les retards dans les connexions entre les neurones i et j et enfin où q_{ij} est le retard maximum entre les neurones i et j (Fig.4). Il apparait ainsi que l'information peut transiter d'un neurone i à un neurone j par l'intermédiaire de plusieurs synapses, chacune introduisant un retard et possédant son propre coefficient de connexion $c_{ij,\tau}$. Cette modélisation implique évidemment que $c_{ij,0} = 0, \ \forall \ i$. On dira que le neurone i est statique si $i \notin P_i$ et $q_{ij} = 0 \ \forall j \in P_i$; un réseau sera dit statique s'il est composé de tels neurones. Un réseau feedforward peut être un réseau dynamique ou statique suivant qu'il possède des synapses à retard ou pas, alors qu'un réseau bouclé fait intervenir nécessairement des synapses à retards.

On voit alors aisément qu'un filtre transverse est un seul neurone linéaire non bouclé et qu'un filtre récursif est un seul neurone linéaire bouclé comportant des synapses à retard. On peut alors imaginer le potentiel que peut avoir un réseau de neurones pour réaliser des fonctions plus complexes de filtrage non linéaire.

II.2 Réseaux de neurones feedforward utilisés comme filtres transverses

Dans un réseau feedforward ou direct, l'information circule toujours dans le même sens, de l'entrée du réseau vers la sortie; il n'y a pas de boucle de retour. L'architecture d'un réseau feedforward, c'est à dire la topologie des connexions et la distribution des retards peut être complètement ou partiellement imposée a priori par le problème à traiter qui est déterminé par la séquence d'entrées sorties à traiter. Dans le cas général d'un réseau statique on considére un réseau complètement connecté.

N'importe quel réseau feedforward possédant des retards synaptiques peut être représenté sous une forme canonique [4] formée d'un réseau feedforward statique dont les entrées sont retardées d'une unité de temps. Le réseau feedforward, aussi général qu'il soit, peut alors être considéré comme un filtre transverse général (général dans le sens où il est non linéaire et possède de nombreuses connexions) dont la sortie à l'instant n vérifie la relation :

$$y(n) = \Phi(u(n), u(n-1), \ldots, u(n - M + 1)) \qquad \text{(II.5)}$$

où Φ représente la fonction non linéaire globalement réalisée par le réseau. Dans (II.5), le nombre M représente la mémoire maximale du système. L'évolution du réseau dont le résultat global est exprimé par (II.5) est régie par

$$\begin{aligned} &z_i(n) = u(n - i + 1) \ ; \qquad i = 1, \ldots, M, \\ &z_i(n) = f_i(v_i(n)) \ , \ v_i(n) = \sum_{j<i} c_{ij} z_j(n) \ \ ; \quad i = M+1, \ldots, M+\nu+1 \end{aligned} \qquad \text{(II.6)}$$

où $z_i(n)$ est l'état du neurone i quand $u(n)$ est la dernière entrée externe, et où $y(n) = z_{M+\nu+1}(n)$. Les fonctions f_i sont du type de celle de (II.1) sauf $f_{M+\nu+1}$ qui est la fonction identité.

II.3 Réseaux de neurones bouclés comme filtres récursifs

Les filtres récursifs ont été introduits pour deux raisons essentielles : (i) la récursivité confère au filtre une mémoire infinie avec un nombre fini de paramètres; (ii) pour modéliser des systèmes récursifs, il est naturel d'introduire des filtres récursifs. Ces deux raisons sont encore valables pour justifier l'utilisation de réseaux de neurones bouclés (ou récurrents). Comme on l'a dit plus haut, les réseaux bouclés ont été utilisés d'abord comme mémoires associatives. Ici, les réseaux bouclés sont vus comme des filtres récursifs non linéaires.

Contrairement à un réseau feedforward où l'état des neurones à un instant n est indépendant de l'état des neurones à l'instant initial, un réseau bouclé se définit par le fait que l'état des neurones à l'instant n dépend de l'état à l'instant initial d'un certain nombres de neurones. Le nombre minimal de ces neurones N est appelé l'ordre du réseau.

Une forme canonique d'un réseau bouclé [4] est constituée d'un réseau feedforward ou non bouclé statique dont les sorties à l'instant n sont composées de la sortie du sytème global $Y(n)$ et des sorties d'un nombre N de neurones dits neurones d'état $S(n+1)$, et dont les entrées sont composées des entrées externes au réseau $U(n)$ et des sorties des neurones d'état retardées d'une unité de temps $S(n)$ (Fig.5). L'évolution du réseau est donnée par les équations d'état:

$$y(n) = \psi[S(n), U(n)], \tag{II.7}$$

$$S(n+1) = \varphi[S(n), U(n)]. \tag{II.8}$$

La sortie du sytème global peut aussi être une variable d'état. Le calcul de l'état des neurones se calcule comme dans un réseau feedforward; les valeurs de $S(n+1)$ sont les états de véritables neurones tandis que les entrées externes $U(n)$ et les variables d'état $S(n)$ ne sont pas l'état de neurones (elles ne sont pas calculées). Différentes architectures ont été considérées dans la littérature sur les réseaux de neurones, mais elles restreignent le type de fonction φ et ψ qui peuvent être réalisées.

III. ALGORITHMES ADAPTATIFS POUR DES RESEAUX DE NEURONES UTILISES COMME FILTRES

III.1 Algorithme adaptatif de type gradient

Un RNF utilisé comme filtre fait correspondre à une séquence de données d'entrée $\{u(n)\}$ une séquence de données de sortie $\{y(n)\}$. La tache que doit réaliser un RNF utilisé comme filtre adaptatif est assujettie à un critère d'optimisation. Dans beaucoup d'applications de la physique, un critère couramment utilisé est le critère des moindres carrés (MC). Ce critère est défini à partir d'une séquence d'entrée (éventuellement infinie) $\{u(n)\}$ et d'une séquence de réponses désirées correspondantes $\{d(n)\}$. Le critère MC sur une fenêtre glissante correspond à la minimisation de la fonctionnelle suivante:

$$I(n) \triangleq \frac{1}{2} \sum_{m=n-N_c+1}^{n} e(m)^2, \qquad e(m) = d(m) - y(m). \tag{III.1}$$

Le paramètre N_c doit être choisi en fonction de la durée de stationnarité du signal. Cette fonction de coût est à minimiser en fonction des paramètres C du réseau, $u(0), u(1), \ldots, u(n-1), u(n)$, et $d(0), d(1), \ldots, d(n-1), d(n)$ étant donnés. Cette fonction caractérise une surface d'erreur, qui dépend de n et qui n'admet pas forcément de minimum; elle peut également en admettre plusieurs. Dans l'hypothèse de l'existence d'un minimum $C_{opt}(n)$ même local, le gradient de $I(n)$ par rapport à C s'annule en les paramètres de ce minimum.

La recherche d'algorithmes adaptatifs pour optimiser un système est motivé par trois aspects i) la recherche directe d'une solution annulant le gradient de $I(n)$ est impossible dans le cas général d'un sytème non linéaire et/ou bouclé, ii) pour limiter la complexité des calculs, il est intéressant d'optimiser le système au fur et à mesure que l'information arrive, iii) remettre à jour l'optimisation du système quand l'environnement est évolutif. Dans la littérature sur les RNF, c'est surtout les points i) et ii) qui ont été pris en compte, tandis que le caractère adaptatif d'un algorithme regroupe à la fois les points i), ii) iii). Ainsi, dans un contexte de traitement du signal par un réseau de neurones, un algorithme sera dit adaptatif s'il calcule au fur et à mesure que le temps court, des modifications des paramètres du système à partir des informations passées et est capable de poursuivre des éventuels changements des caractéristiques du signal.

Une modification, à l'instant n, du vecteur contenant tous les coefficients du système, est calculée itérativement suivant l'algorithme du gradient stochastique:

$$C(n) = C(n-1) - \mu \left. \frac{\partial I(n)}{\partial C} \right|_{C=C(n-1)}. \qquad \text{(III.2)}$$

III.2 Calcul direct du gradient dans un réseau feedforward.

Le calcul direct du gradient repose sur le développement le plus immédiat de (III.1), soit

$$\left. \frac{\partial I(n)}{\partial c_{pq}} \right|_{C=C(n)} = - \sum_{m=n-N_c+1}^{n} e(m) \left. \frac{\partial y(m)}{\partial c_{pq}} \right|_{C=C(n)}. \qquad \text{(III.3)}$$

Les dérivées partielles de la sortie du système $y(m)$ par rapport aux coefficients c_{pq} $(p > q)$ se calculent à partir des équations d'évolution (II.6) du réseau feedforward. Il vient

$$\begin{aligned}
&\frac{\partial z_i(m)}{\partial c_{pq}} = 0 \quad ; \; p > i \text{ ou } i = 1, \ldots, M, \\
&\frac{\partial z_i(m)}{\partial c_{pq}} = f'_i(v_i(m)) z_q(m) \; ; \; p = i \text{ and } i > M, \\
&\frac{\partial z_i(m)}{\partial c_{pq}} = f'_i(v_i(m)) \sum_{p<j<i} c_{ij} \frac{\partial z_j(m)}{\partial c_{pq}} \; ; \; p < i \text{ and } i > M.
\end{aligned} \qquad \text{(III.4)}$$

Ainsi le calcul des $\frac{\partial y(m)}{\partial c_{pq}}$ se fait de proche en proche par propagation, dans un réseau feedforward linéarisé dont les poids sont $f'_i(v_i(m))c_{ij}$, des dérivées partielles des sorties des neurones situés plus en amont du réseau [5]. Dans le cas où $N_c \geq 1$, il faut calculer les N_c dérivées partielles intervenant dans (III.3) avec les poids fixés à leur valeur $C = C(n)$.

III.3 Calcul du gradient par rétropropagation dans un réseau feedforward.

Le calcul du gradient par rétropropagation fait intervenir des dérivées partielles différentes de celles utilisées dans le calcul direct. A partir de (III.1), on écrit

$$\left. \frac{\partial I(n)}{\partial c_{pq}} \right|_{C=C(n)} = \frac{1}{2} \sum_{m=n-N_c+1}^{n} \frac{\partial e^2(m)}{\partial v_p(m)} \left. \frac{\partial v_p(m)}{\partial c_{pq}} \right|_{C=C(n)}. \qquad \text{(III.5)}$$

D'après (II.6), on calcule aisément $\frac{\partial v_p(m)}{\partial c_{pq}} = z_q(m)$ ainsi que

$$\frac{\partial e^2(m)}{\partial v_p(m)} = \sum_{l \geq p} \frac{\partial e^2(m)}{\partial v_l(m)} \frac{\partial v_l(m)}{\partial v_p(m)} = f'_p(v_p(m)) \sum_{l \geq p} c_{lp} \frac{\partial e^2(m)}{\partial v_l(m)}. \qquad \text{(III.6)}$$

La dernière relation exprime que les dérivées partielles $\frac{\partial e^2(m)}{\partial v_p(m)}$ peuvent être calculées à partir de celles déjà calculées en aval du réseau. Le gradient est calculé par rétropropagation des dérivées partielles dans un réseau feedforward linéarisé dont les paramètres sont $c_{lp} f'_p(v_p(m))$.

On remarquera que la sommation se fait sur le premier indice l dans (III.6) alors qu'elle se faisait sur le deuxième indice dans le calcul direct (III.4).

Remarquons enfin que le calcul par rétropropagation est moins complexe en calculs que le calcul direct pour un réseau feedforward de neurones non linéaires.

III.4 Evaluation du gradient dans un réseau bouclé.

L'évaluation du gradient nécessaire à l'adaptation des paramètres d'un réseau bouclé est beaucoup plus complexe que dans le cas d'un réseau feedforward. En effet, on voit sur les équations d'état (II.7, II.8) de la représentation canonique d'un réseau bouclé, que la sortie globale du système ainsi que les états des neurones cachés du réseau feedforward associé sont fonction non seulement des entrées externes au réseau, mais aussi des variables d'état qui sont elles mêmes fonction de la sortie du réseau et des variables d'état calculées aux instants précédents. Autrement dit, chacun des termes d'erreur $e(m)$ intervenant dans le critère $I(n)$ de (III.1) ne dépend plus seulement des M dernières entrées externes $u(m)$, ..., $u(m - M + 1)$, et des M dernières réponses désirées correspondantes, mais de toutes les valeurs de u et les réponses désirées depuis l'instant initial. Par conséquent, pour calculer le gradient de $I(n)$ en un certain point C, que ce soit par le calcul direct ou par la rétropropagation, il faudrait théoriquement calculer chaque $e(m)$ et chaque dérivées partielles intervenant dans (III.4) ou (III.6), avec toutes les données depuis l'instant initial et à C fixé. Dans le cas d'un algorithme itératif voire adaptatif, il faudrait refaire ce calcul pour chaque C en cours.

La représentation d'état d'un dispositif bouclé temporellement (Fig.5), qu'il soit linéaire ou non linéaire permet une interprétation "spatiale" particulièrement simple du système. En effet, à un instant m quelconque, le dispositif peut être représenté de façon équivalente (Fig.6) en "déroulant" le dispositif de départ en la concaténation des m cellules élémentaires identiques ou copies qui le composent. Dans la pratique, la taille m de ce réseau feedforward croit avec m et il est irréalisable de recalculer avec des coefficients fixés à leur dernière valeur les quantités nécessaires à l'évaluation du gradient.

Une simplification consiste à tronquer et à initialiser correctement le déroulement du système de la Fig.6 sur une longueur fixe N_t réalisant un compromis entre précision du calcul du gradient, complexité des calculs, places mémoires. Ce système (Fig.7) est alors lui même un réseau feedforward dont les paramètres et les variables d'état, d'entrée et de sortie sont à l'instant m, $c_{pq}^{l}(m)$ et $z_i^l(m)$, c'est à dire indicés par le numéro de la copie l et supposés indépendants. La modification des paramètres à un instant n, est alors la somme sur N_c instants et sur N_t copies des modifications de chacun des paramètres.

A partir de ce réseau feedforward, différents type d'algorithmes ont été envisagés [5] suivant la technique de calcul choisie (calcul direct du gradient ou rétropropagation) mais aussi le type d'initialisation du système de la Fig.7. En effet, la rétropropagation suppose implicitement que les dérivées partielles à l'entrée du réseau de la Fig.7 sont nulles et que l'on connaisse des réponses désirées pour toutes les variables d'état de la dernière copie alors que le calcul direct s'accomode de n'importe quel type d'initialisation. Autrement dit le critère de la complexité qui faisait préférer la rétropropagation au calcul direct dans un réseau feedforward ne peut plus être pris seul en compte dans le cas des réseaux bouclés; un calcul direct avec des initialisations convenables peut donner de meilleures performances que la rétropropagation.

IV. CONCLUSION

Traditionnellement, les RNF sont utilisés pour la classification ou la reconnaissance de formes statiques. Nous avons présenté ici quelques points essentiels concernant l'architecture et les algorithmes d'apprentissage des RNF, qui apparaissent quand on veut utiliser les RNF pour faire du filtrage adaptatif non linéaire. Lors de la communication, l'application des RNF au codage prédictif de la parole, qui utilise classiquement un prédicteur linéaire et un quantificateur, sera présentée.

REFERENCES

[1] K. HORNIK, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol.2, pp. 359-366, 1989.
[2] R.J.WILLIAMS, D.ZIPSER, "A learning algorithm for continually running fully recurrent neural networks", Neural Computation 1, p. 4-27, 1990.

[3] D.E. RUMELHART et al. "Learning internal Representation by Error Propagation" in "Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Vol. 1. Foundations, MIT Press, 1986, D. Rumelhart, J. McCleveland Editors.
[4] O.NERRAND et al., "Neural network training schemes for non-linear adaptive filtering and modelling", Proceedings of IJCNN, Seattle, 1991
[5] S.MARCOS et al., "A unified framework for gradient algorithms used for filter adaptation and neural network training", Int. Journal of Circuit Theory and Applications, to be published.