Améliorer la prévision multipas par réseaux de neurones récurrents

Mohammad Assaad, Romuald Boné, Hubert Cardot

Université François-Rabelais de Tours,
Laboratoire d'Informatique,
64, avenue Jean Portalis,
37200 Tours, France
{mohammad.assaad, romuald.bone, hubert.cardot}@univ-tours.fr

Résumé. Nous évaluons sur le problème de la prévision multipas un algorithme de boosting pour les réseaux de neurones récurrents (RNRs). Cet algorithme combine un grand nombre de RNRs, chacun d'entre eux étant généré en apprenant sur une version différente de l'ensemble d'apprentissage d'origine. Ces versions sont obtenues selon une variante de la méthode du boosting, qui permet de concentrer l'apprentissage sur les exemples difficiles mais, à la différence de l'algorithme d'origine, en prenant en compte tous les exemples disponibles. Nous l'appliquons au problème de la prévision pour différents horizons de trois séries temporelles de référence en testant à chaque fois trois fonctions de coût différentes et en variant les valeurs du paramètre principal. Nous comparons notre algorithme avec l'algorithme Back-Propagation Through Time (BPTT) et avec d'autres méthodes appliquées sur les mêmes ensembles de données, dont des approches locales.

1 Introduction

La dégradation des performances des systèmes de prévision de séries temporelles, dès lors que l'horizon de prévision augmente, est un problème bien connu. L'estimation de la valeur future d'une variable peut être raisonnablement fiable pour une prévision à court terme mais se dégrade lorsqu'on aborde le problème de la prévision à long terme. Toutefois la prévision multipas de séries temporelles est indispensable dans de nombreux domaines d'application, qui s'étendent de la modélisation des phénomènes naturels au contrôle de systèmes dynamiques en passant par la finance, le marketing, la météorologie, etc. La majeure partie de la littérature publiée traite du problème de la prévision de séries temporelles à un pas de temps. En effet, la prévision multipas demeure un problème difficile pour lequel les résultats obtenus par des extensions simples des méthodes développées pour la prévision à un pas sont souvent décevants. Par ailleurs, si beaucoup de méthodes obtiennent des résultats proches sur des problèmes de prévision à un pas, des différences significatives se font jour quand des extensions de ces méthodes sont employées sur des problèmes multipas.

Pour améliorer les performances obtenues, nous pouvons adapter des procédures générales qui ont démontré, sous certaines conditions, leur capacité à améliorer les performances de

divers modèles de base. Une de ces procédures est connue sous le nom de boosting (Schapire 1990).

Dans ce papier nous nous concentrons sur l'application d'un algorithme de boosting pour améliorer les performances de la prévision multipas de réseaux de neurones récurrents. Dans la prochaine section, nous présentons un état de l'art sur l'utilisation des réseaux de neurones pour la prévision multipas avant de décrire en section 3 notre algorithme, basé sur une adaptation due à Drucker de l'algorithme générique de boosting. Les résultats expérimentaux obtenus sur trois séries temporelles, une naturelle et deux chaotiques synthétiques, sont décrits dans la section 4 et montrent une amélioration sensible des performances.

2 La prévision multipas par réseaux de neurones

Les nombreuses méthodes développées pour la prévision de séries temporelles peuvent se regrouper en deux familles, les approches globales et les approches locales, selon qu'un modèle utilise l'ensemble des données pour son apprentissage ou que plusieurs modèles se répartissent ces données.

Dans une approche locale, une approximation par morceaux du comportement du système dynamique qui génère la série temporelle est établie. Un espace d'état synthétique est d'abord construit en employant une fenêtre temporelle glissante sur la séquence d'entrée et la séquence de sortie. Puis, une technique de quantification vectorielle est utilisée pour effectuer une segmentation de cet espace d'état. Finalement, un modèle local simple (habituellement linéaire) est développé pour chaque segment. Les approches locales se fondent sur les hypothèses que la dimension de l'espace d'état du système est inférieure à la dimension de la fenêtre temporelle et que le système a peu de comportements différents (quelques segments sont suffisants). Si ce n'est pas le cas, le nombre de paramètres explose et le nombre d'exemples pour chaque segment devient insuffisant pour un apprentissage fiable du modèle local correspondant.

L'approche locale s'est révélée efficace sur de nombreux problèmes de prévision multipas (Vesanto 1997) (Chudy et Farkas 1998) (McNames 2000) (Gers, Eck et al. 2001) (Cottrell, Simon et al. 2004) (Pavlidis, Tasoulis et al. 2005), probablement grâce à l'amélioration de la robustesse fournie par la quantification de l'espace d'état et à la simplicité des modèles locaux utilisés. La quantification vectorielle est parfois fournie par des réseaux de neurones à topologie contrainte, tels que les cartes auto-organisatrices (Vesanto 1997) (Cottrell, Simon et al. 2004), ou à topologie libre, comme les réseaux *neural-gas* ou les structures cellulaires dynamiques (Chudy et Farkas 1998). Quelques auteurs mentionnent pourtant des difficultés rencontrées par l'approche locale sur quelques problèmes (Vesanto 1997) (Gers, Eck et al. 2001).

Une approche globale tente d'établir un modèle unique pour l'ensemble des comportements du système dynamique sous-jacent. Afin de pouvoir modéliser l'ensemble de ces comportements, des modèles complexes doivent être employés. De tels modèles sont souvent difficiles à apprendre et peuvent facilement diverger du comportement désiré.

De par leurs propriétés d'approximation universelle, les réseaux de neurones sont de bons modèles candidats pour les approches globales. Parmi les nombreuses architectures neuronales employées pour la prévision des séries temporelles, nous pouvons mentionner les perceptrons multicouches (PMCs) avec une fenêtre temporelle en entrée (Rumelhart, Hinton et al. 1986) (Cottrell, Rynkiewicz et al. 2001), les PMCs avec connexions Finite Impulse Response

(FIR) (Back et Tsoi 1992), les réseaux récurrents obtenus en utilisant des connexions Infinite Impulse Response (IIR) dans les PMCs (Wan 1994) ou avec des bouclages de la sortie vers les couches précédentes (Elman 1989; Czernichow, Piras et al. 1996), les réseaux récurrents simples (Suykens et Vandewalle 1995), les réseaux récurrents avec des connexions FIR (Boné et Crucianu 2002), et les réseaux récurrents avec des bouclages internes et en provenance de la sortie (Parlos, Rais et al. 2000). On peut noter que beaucoup d'auteurs incluent à divers endroits des connexions à délais qui fournissent une mémoire explicite du passé. Ces ajouts semblent être (implicitement et parfois explicitement) justifiés par le fait qu'ils facilitent l'apprentissage des dépendances à long terme présentes dans les données, qui sont généralement supposées utiles pour les problèmes de prévisions et en particulier pour des problèmes de prévision multipas. Le lien entre l'apprentissage des dépendances à long terme et les performances de la prévision multipas a été expérimentalement confirmé dans (Boné et Crucianu 2002).

Indépendamment de la nature des modèles utilisés, plusieurs méthodes permettent de traiter les problèmes de prévision multipas.

La première méthode, la plus répandue, consiste à mettre au point un prédicteur pour le problème à un pas de temps et à l'utiliser récursivement pour le problème multipas correspondant. Les estimations fournies par le modèle pour le pas de temps suivant sont représentées en entrée du modèle jusqu'à ce que l'horizon désiré pour la prévision soit atteint. Cette méthode simple est pénalisée par l'accumulation des erreurs durant les pas de temps successifs ; le modèle diverge rapidement du comportement désiré.

Une deuxième et meilleure méthode consiste à apprendre le prédicteur sur le problème à un pas de temps et, en même temps, à utiliser la rétropropagation des pénalités à travers les pas de temps afin de punir le prédicteur pour les erreurs cumulées. Quand les modèles sont des PMCs ou des RNRs, une telle procédure est directement inspirée de l'algorithme BPTT (Rumelhart, Hinton et al. 1986), qui effectue une descente de gradient sur l'erreur cumulée.

Dans la troisième méthode, appelée la méthode directe, le prédicteur traite directement le problème de prévision multipas. Des résultats expérimentaux et des considérations théoriques (Atiya, El-Shoura et al. 1999) tendent à prouver que la méthode directe se conduit toujours mieux que la première méthode et au moins aussi bien que la seconde. Une amélioration des performances est constatée si l'on utilise des RNRs que l'on entraîne en augmentant progressivement l'horizon de prévision (Suykens et Vandewalle 1995) ou en incluant les connexions à délais de la sortie du réseau à son entrée (Parlos, Rais et al. 2000).

Une méthode proche (Duhoux, Suykens et al. 2001) enchaîne plusieurs réseaux. Le premier apprend à prévoir à $t+1$, un second apprend à prévoir à $t+2$ en utilisant comme entrée supplémentaire la prévision du premier réseau et ainsi de suite jusqu'à l'horizon de prévision souhaité.

Récemment, deux méthodes de prévision sensiblement différentes ont été proposées. Dans (Jaeger 2004) un réseau de neurones récurrent de grande taille, appelé réservoir, est produit aléatoirement pour que ses neurones présentent un ensemble varié de comportements. En apprenant les poids sortants, les comportements sont combinés afin d'obtenir la prévision désirée. Dans (Schmidhuber, Wierstra et al. 2005), une approche évolutionniste est employée pour obtenir les poids des neurones cachés de RNRs et des méthodes comme la régression linéaire ou la programmation quadratique sont appliquées pour calculer les relations linéaires optimales entre la couche cachée et la sortie.

Nous n'avons cité ici qu'une partie de la littérature récente sur la prévision neuronale mais les méthodes que nous avons mentionnées caractérisent bien les approches existantes.

Néanmoins, il est particulièrement difficile d'identifier une typologie des problèmes de prévision multipas et d'évaluer quelles méthodes sont les plus appropriées pour un type donné de problème.

L'approche que nous présentons pour la prévision multipas se base sur la troisième méthode et applique un algorithme de boosting adapté aux RNRs, qui possèdent une mémoire interne et ne nécessitent pas de fenêtre temporelle.

3 Boosting pour les réseaux de neurones récurrents

Pour améliorer les résultats des RNRs, nous pouvons employer une combinaison de modèles pour obtenir une estimation plus précise que celle obtenue par un modèle unique. Dans l'algorithme de boosting, le léger gain qu'un modèle de faible puissance peut apporter par rapport à une estimation aléatoire est amplifié par la construction séquentielle de plusieurs de ces modèles, qui se concentrent progressivement sur les exemples qui s'avèrent difficiles dans l'ensemble d'apprentissage d'origine. L'algorithme de boosting (Schapire 1990) (Freund et Schapire 1997) travaille par une application séquentielle d'un algorithme de classification aux versions re-pondérées des données d'apprentissage, et ensuite prend un vote majoritaire pondéré de la séquence des classificateurs ainsi produits. Freund et Schapire (1997) ont prolongé leurs idées pour appliquer l'algorithme adaboost aux problèmes de régression ; l'algorithme adaboost.R aborde le problème de régression en le ramenant à un problème de classification.

Récemment, une nouvelle approche de l'algorithme de boosting pour la régression a été proposée, basée sur l'ajustement des résidus (Mason, Baxter et al. 1999) (Duffy et Helmbold 2002). Au lieu d'apprendre sur un échantillon différent du même ensemble d'apprentissage, comme dans les algorithmes de boosting précédents, un régresseur est appris sur un nouvel ensemble d'apprentissage avec des sorties désirées différentes. Avant de présenter brièvement notre algorithme, étudié dans (Assaad, Boné et al. 2005) pour des problèmes de prévision à un pas de temps, mentionnons les quelques applications existantes de boosting pour la prévision des séries temporelles.

Dans (Cook et Robinson 1996) une méthode appartenant à la première famille des algorithmes de boosting présentée (Schapire 1990) est appliquée avec succès à la classification des phonèmes. Les modèles utilisés sont des RNRs, et les auteurs sont les premiers à noter l'influence de la mémoire interne des RNRs sur l'algorithme de boosting, sans en tirer partie cependant.

Un algorithme similaire de boosting est employé dans (Avnimelech et Intrator 1999), mais avec des PMCs comme régresseurs. L'algorithme de boosting utilisé doit être conforme aux restrictions imposées par le contexte général de l'application.

Dans notre cas, il doit pouvoir travailler quand une quantité limitée de données est disponible et accepter les RNRs comme régresseur. Nous avons suivi l'algorithme générique de (Freund 1990). Nos mises à jour sont basées sur la technique de (Drucker 1999), précédemment appliquée à des méthodes non-paramétriques dans (Borra et Di Ciaccio 2002) et aux arbres de régressions dans (Gey et Poggi 2006). Nous appliquons cependant une transformation affine aux poids avant de les utiliser (voir la définition de $D_{n+1}(q)$ dans le tableau 1) afin d'empêcher les RNRs d'ignorer purement et simplement les exemples plus faciles sur des problèmes comme celui des taches solaires. Puis, au lieu d'effectuer un échantillonnage avec remplacement selon la distribution mise à jour, nous préférons pondérer l'erreur calculée

pour chaque exemple d'apprentissage (en utilisant toutes les données) à la sortie du RNR avec la valeur de la distribution correspondant à l'exemple.

La décision d'utiliser toutes les données de l'ensemble d'apprentissage pour la mise au point de chaque régresseur permet de traiter des séries temporelles constituées d'une quantité limitée de données. Notre idée est que les modifications des poids $D_n(q)$ permettent de donner plus d'importance aux exemples difficiles pour l'apprentissage du régresseur suivant. Le paramètre k permet de régler cette prise en compte ; quand $k = 0$, les exemples difficiles ont la même pondération que les autres.

Pour combiner les régresseurs, nous employons la médiane pondérée (Drucker 1999), qui est moins sensible aux points aberrants que la moyenne pondérée. L'algorithme de boosting que nous proposons finalement est décrit dans le tableau 1.

1. Initialiser les poids pour les exemples : $D_1(q) = 1/Q$, avec Q le nombre d'exemples d'apprentissage. Mettre le compteur d'itérations à 0 : $n = 0$
2. Répéter
   (a) Incrémenter n. Développer un RNR h_n en employant l'ensemble d'apprentissage au complet et en pondérant l'erreur quadratique calculée pour l'exemple q par $D_n(q)$, le poids de l'exemple q pour l'itération n ;
   (b) Mettre à jour les poids des exemples :
      (i) calculer $L_n(q)$ pour chaque $q = 1, \ldots, Q$ avec une des trois fonctions :

$$L_n^{linéaire}(q) = \left|y_q^{(n)}(x_q) - y_q\right| / S_n \;,\; L_n^{quadratique}(q) = \left|y_q^{(n)}(x_q) - y_q\right|^2 / S_n^2 \;,$$

$$L_n^{saturée}(q) = 1 - \exp\left(-\left|y_q^{(n)}(x_q) - y_q\right| / S_n\right), \text{ où } S_n = \sup_q \left|y_q^{(n)}(x_q) - y_q\right| \;;$$

      (ii) calculer $\varepsilon_n = \sum_{q=1}^{Q} D_n(q) L_n(q)$ et $\alpha_n = (1 - \varepsilon_n)/\varepsilon_n$;
      (iii) les poids des exemples deviennent

$$D_{n+1}(q) = \frac{1 + k \cdot p_{n+1}(q)}{Q + k}, \text{ avec } p_{n+1}(q) = \frac{D_n(q)\alpha_n^{(L_n(q)-1)}}{Z_n},$$

   jusqu'à $\varepsilon_n < 0{,}5$, où Z_n est une constante de normalisation ;
3. Combiner les RNRs en utilisant la médiane pondérée ou la moyenne pondérée.

TAB. 1 – *Notre algorithme de boosting pour la régression avec les réseaux de neurones récurrents.*

4 Expériences

Nous avons appliqué notre algorithme de boosting pour réseaux récurrents à la prévision multipas de trois séries temporelles de référence : une naturelle (les taches solaires) à dépendances temporelles trouvées expérimentalement à court terme et deux chaotiques synthétiques (Mackey-Glass 17 et Mackey-Glass 30) dont les dépendances à moyen et long termes sont contrôlées explicitement. Nous avons appliqué notre algorithme sur des RNRs compo-

sés d'un neurone d'entrée, un neurone de sortie, un neurone de biais et une couche cachée entièrement récurrente composée de neurones à fonctions de transfert tangente hyperbolique.

Nous avons exécuté 5 expériences pour chaque architecture, en initialisant aléatoirement les poids dans $[-0{,}3\,;\,0{,}3]$. La moyenne des résultats et l'écart type ont été déterminés après ces 5 essais. Nous avons fixé à 50 le nombre maximal n de RNRs pour chaque expérience.

Nous comparons nos résultats avec l'algorithme BPTT d'origine et avec deux algorithmes constructifs, *Constructive* BPPT (CBPTT) et *Exploratory* BPTT (EBPTT), qui ajoutent des connexions à délais aux RNRs (Boné et Crucianu 2002) selon des critères calculés pendant l'exécution de BPTT. Pour quantifier nos résultats, nous utilisons l'erreur quadratique moyenne normalisée (EQMN), qui est l'erreur quadratique moyenne divisée par la variance de la série temporelle, calculée sur l'ensemble de test. La racine de l'EQMN est parfois utilisée dans la littérature.

4.1 La série temporelle de taches solaires

Cet ensemble de données contient le nombre annuel de taches foncées sur le soleil de 1700 à 1979. Ces taches sont liées à l'activité du champ magnétique du soleil. Il est d'usage d'employer les données de 1700 à 1920 pour l'ensemble d'apprentissage et d'évaluer les performances du modèle sur l'ensemble de test correspondant à la période 1921-1979. Nous avons utilisé des RNRs ayant 12 neurones dans la couche cachée. Ce nombre de neurones correspond au meilleur résultat obtenu sur la série par BPTT sans boosting pour des prévisions à un pas de temps. Pour la même raison, les deux algorithmes CBPTT et EBPTT ont été appliqués sur un réseau avec 2 neurones cachés (Boné et Crucianu 2002).

Nos modèles ont deux paramètres, la fonction de coût qui peut être linéaire, quadratique ou saturée, et le paramètre k de notre algorithme de boosting. La valeur de k choisie ici correspond aux meilleurs résultats obtenus pour une prévision à un pas de temps sur l'ensemble de test. Par exemple, lin. 10 correspond à une fonction de coût linéaire avec $k = 10$. Pour déterminer ces paramètres, trois fonctions de coût avec des valeurs de k entre 5 à 150 ont été testées.

horizon h	Modèle							
	lin. 10		quad. 20		quad. 5		satu. 20	
	moy.	méd.	moy.	méd.	moy.	méd.	moy.	méd.
1	0,18	0,16	0,17	0,16	0,18	0,15	0,17	0,17
2	0,42	0,41	0,43	0,38	0,37	0,37	0,38	0,38
3	0,36	0,28	0,54	0,45	0,55	0,43	0,58	0,59
4	0,49	0,40	0,68	0,66	0,60	0,47	0,67	0,74
5	0,73	0,68	0,62	0,54	0,62	0,66	0,62	0,76
6	0,56	0,67	0,61	0,56	0,57	0,51	0,68	0,72
10	0,56	0,47	0,64	0,64	0,49	0,51	0,73	0,75
12	0,79	0,78	0,95	0,94	0,61	0,54	1,15	1,03

TAB. 2 – *Meilleurs résultats (EQMN) parmi 5 essais pour la moyenne et la médiane pondérées sur l'ensemble de test des taches solaires en fonction de l'horizon h.*

Le tableau 2 récapitule les meilleurs résultats obtenus par notre algorithme. Pour cela, nous cherchons pour chaque valeur de l'horizon h le meilleur résultat normalisé après les 5

essais de chaque configuration (fonction de coût ; paramètre k). Nous pouvons observer que l'utilisation de la médiane pondérée offre en général de meilleures performances que la moyenne pondérée, quelles que soient la configuration et la valeur de h .

Peu de comparaisons ont été effectuées pour cet ensemble de données, les résultats dans la littérature concernant la prévision à un pas de temps. Le tableau 3 donne les meilleurs résultats moyens de l'EQMN pour un horizon de prévision jusqu'à 12. Pour les déterminer, pour chaque valeur de h , nous calculons la moyenne des 5 essais pour chaque fonction de coût et pour chaque valeur du paramètre k , puis nous retenons les meilleurs résultats issus de ces calculs. Pour notre algorithme, l'écart type associé est au moins 5 fois inférieur à la moyenne, indépendamment de la fonction de coût utilisée.

Tous les algorithmes testés se comportent mieux que l'algorithme BPTT d'origine et montrent une dégradation rapide lorsque l'horizon de prévision augmente, cependant notre algorithme de boosting se montre le plus performant.

horizon	modèle						
h	BPTT	CBPTT	EBPTT	lin. 10	quad. 20	quad. 5	sat. 20
1	0,24	0,17	0,19	0,18	0,17	0,18	0,18
2	0,88	0,69	0,53	0,43	0,40	0,43	0,42
3	1,14	0,99	0,79	0,54	0,54	0,56	0,67
4	1,22	1,17	0,80	0,67	0,73	0,64	0,76
5	1,01	0,99	0,88	0,74	0,69	0,73	0,77
6	1,02	1,01	0,84	0,73	0,68	0,65	0,74
10	-	-	-	0,64	0,69	0,67	0,75
12	-	-	-	0,86	0,97	0,77	1,09

TAB. 3 – *Meilleurs résultats moyens de l'EQMN sur l'ensemble de test des taches solaires en fonction de h (agrégation par médiane pondérée, qui donne les meilleurs résultats).*

Avec cette série, notre algorithme développe environ 9 réseaux avec les deux fonctions linéaire et quadratique et 30 réseaux avec la fonction saturée, comme pour la prévision à un pas de temps. Le nombre moyen de réseaux reste constant quand l'horizon augmente.

4.2 Les séries temporelles de Mackey-Glass

Les données de Mackey-Glass sont produites par une équation non-linéaire (Casdagli 1989) :

$$\frac{dx(t)}{dt} = -0{,}1 \cdot x(t) + \frac{0{,}2 \cdot x \cdot (t-\tau)}{1 + x^{10} \cdot (t-\tau)}$$

Selon la valeur de τ , la série obtenue peut converger asymptotiquement vers un point fixe, avoir un comportement périodique ou encore chaotique si $\tau > 16{,}8$. Nous avons retenu comme habituellement dans la littérature les deux constantes de temps, $\tau = 17$ (la série ainsi obtenue est appelée MG17) et $\tau = 30$ (MG30). Les données sont échantillonnées avec une période de 6 pas de temps. Les 500 premières valeurs ont été dédiées à l'ensemble d'apprentissage et les 100 valeurs suivantes à l'ensemble de test.

La plupart des résultats dans la littérature concernent un pas de temps en avant (6 pas si nous considérons la période de l'échantillonnage).

Comme pour la série précédente, nous avons effectué nos tests pour la prévision multipas sur les deux séries MG17 et MG30 avec une architecture de RNRs à 7 neurones dans la couche cachée, configuration trouvée optimale pour la prévision à un pas de temps, que ce soit avec l'algorithme BPTT, CBPTT ou EBPTT. Nous avons considéré un horizon h entre 1 et 11 pas de temps, limite au delà de laquelle les performances se dégradent fortement.

Nous avons évalué notre méthode sur les deux ensembles avec les paramètres correspondant aux meilleurs résultats obtenus par notre algorithme de boosting pour la prévision à un pas de temps : (linéaire, 150), (quadratique, 100) et (saturée, 100) pour MG17 et (linéaire, 300), (quadratique, 200) et (saturée, 150) pour MG30. Pour déterminer ces paramètres, trois fonctions de coût avec des valeurs du paramètre k variant entre 5 et 200 pour MG17 et entre 5 et 300 pour MG30 ont été testées.

horizon h	modèle					
	lin. 150		quad. 100		sat. 100	
	moy.	méd.	moy.	méd.	moy.	méd.
1	0,18	0,13	0,18	0,15	0,20	0,13
2	0,26	0,20	0,24	0,20	2,36	0,20
3	0,45	0,44	0,59	0,50	0,52	0,49
4	0,70	0,47	0,60	0,45	0,65	0,35
5	6,26	0,76	9,43	0,86	11,9	1,13
6	10,8	1,47	13,7	1,27	28,2	2,07
10	16,6	7,33	18,6	7,69	24,6	11,4
11	17,4	8,65	23,0	8,53	27,9	14,9

TAB. 4 – *Meilleurs résultats EQMN* (*10^3) *parmi 5 essais pour la moyenne et la médiane pondérées sur l'ensemble de test de MG17 en fonction de l'horizon h.*

horizon h	modèle					
	lin. 300		quad. 200		Sat. 150	
	moy.	méd.	moy.	méd.	moy.	méd.
1	0,41	0,41	0,46	0,43	0,46	0,45
2	0,53	0,43	15,8	0,44	35,4	0,54
3	42,8	0,52	32,9	0,51	62,4	0,57
4	16,2	0,40	42,1	0,40	78,8	0,44
5	0,76	0,78	0,63	0,63	0,81	0,66
6	3,96	1,62	2,32	1,73	2,02	1,61
10	26,9	2,59	42,0	3,00	13,6	2,26
11	29,7	5,41	27,6	4,97	64,4	6,35

TAB. 5 – *Meilleurs résultats EQMN* (*10^3) *parmi 5 essais pour la moyenne et la médiane pondérées sur l'ensemble de test de MG30 en fonction de l'horizon h.*

Les tableaux 4 et 5 montrent respectivement les meilleurs résultats obtenus par notre algorithme de boosting sur les deux séries MG17 et MG30. Les performances obtenues sur ces

séries en retenant la médiane pondérée comme opérateur d'agrégation sont meilleures que celles issues de l'utilisation de la moyenne pondérée, ceci pour chaque configuration et sur toutes les valeurs de h. Le tableau 6, pour MG17, et la figure 1, pour MG30, montrent les fortes améliorations obtenues par rapport à l'algorithme BPTT d'origine et par rapport aux algorithmes constructifs. Sur les deux séries, nous remarquons que nos résultats, quelle que soit la fonction de coût utilisée, sont sensiblement meilleurs que les résultats obtenus par les autres algorithmes. La faiblesse de l'erreur des algorithmes CBPTT et EBPTT pour l'horizon de temps $h = 4$ peut être expliquée par une autocorrélation négative forte pour cette valeur.

horizon	modèle					
h	BPTT	CBPTT	EBPTT	lin. 150	quad. 100	sat. 100
1	22	13	1	0,17	0,16	0,17
2	179	124	101	0,24	0,28	0,25
3	145	124	16	0,57	0,57	0,52
4	8	7	4	0,57	0,54	0,52
5	266	253	181	0,98	1,26	1,27
6	321	321	232	2,11	15,2	4,66
10	336	331	219	14,1	12,2	15,0
11	289	218	252	9,80	12,0	16,8

TAB. 6 – *Meilleurs résultats moyens EQMN* (*10^3) *sur l'ensemble de test de MG17 en fonction de l'horizon h (agrégation par médiane pondérée).*

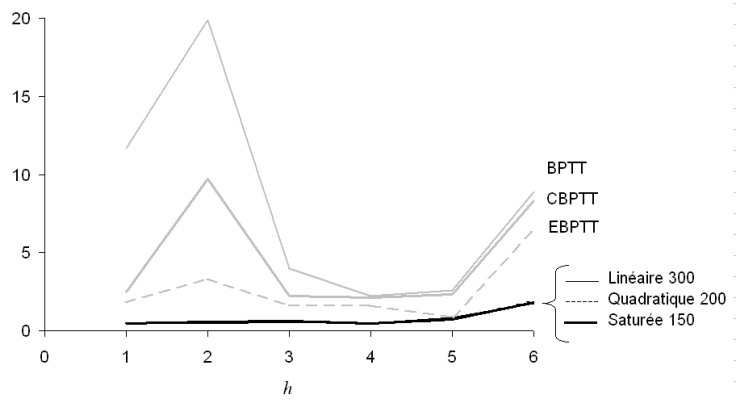

Fig. 1 – *Meilleurs résultats moyens EQMN* (*10^3) *sur l'ensemble de test de MG30 en fonction de l'horizon h (agrégation par médiane pondérée).*

Les comparaisons avec d'autres résultats publiés concernant le problème de prévision multipas ne peuvent être effectuées que pour un horizon de 14 ; les résultats présentés ici sont inférieurs à ceux des méthodes locales proposées dans (Vesanto 1997) (Chudy et Farkas 1998) ou de la méthode décrite dans (Chen, Yang et al. 2005), mais pour les RNRs appris par notre algorithme, sensiblement moins de données ont été utilisées pour l'apprentissage (500 comparés à 3000 ou à 10000), puisque nous avons retenu la série MG17 habituelle (Casdagli 1989) (Wan 1994).

Le nombre de réseaux dans nos expériences sur les deux séries MG17 et MG30 dépend fortement de la fonction de coût utilisée. Pour la fonction linéaire, avec la série MG17, notre algorithme développe entre 30 et 40 réseaux en moyenne, excepté pour les horizons $h = 4, 5, 7$ et avec la série MG30, notre algorithme développe entre 20 et 45 réseaux, excepté pour les horizons $h = 10, 11$. Pour ces cas, nous obtenons le nombre maximum de réseaux, fixé par l'utilisateur (50). Pour les deux séries, avec la fonction quadratique, le nombre moyen est légèrement supérieur alors que pour la fonction saturée, le nombre maximum de réseaux est produit pour tous les horizons.

5 Conclusion

Nous avons adapté l'algorithme de boosting au problème de l'apprentissage des données séquentielles pour la prévision, en utilisant comme régresseurs des RNRs. Ce nouvel algorithme avait déjà permis d'améliorer les résultats de l'approche globale pour la prévision à un pas de temps. Nous avons évalué ses performances pour le problème multipas.

Les résultats expérimentaux qui ont été obtenus sur les trois séries de référence montrent que notre algorithme de boosting pour les réseaux de neurones récurrents améliore fortement les prévisions à plusieurs pas de temps en avant. Le boosting semble renforcer la prise en compte des dépendances temporelles entre les données et les sorties désirées situées plusieurs pas de temps dans le futur. En effet, l'influence de l'algorithme de boosting se révèle moins notable pour les prévisions multipas avec la série des taches solaires alors que les dépendances à court terme sont connues pour être essentielles dans cette série. A contrario, le fait d'obtenir de meilleurs résultats pour les ensembles de données de Mackey-Glass peut être expliqué en notant que les dépendances à long terme jouent un rôle important pour MG17 et MG30. Remarquons que nos résultats pourraient être globalement améliorés par un réglage des paramètres directement sur le problème de prévision multipas, via un ensemble de validation par exemple. Le protocole et la taille des séries temporelles de référence retenues ne nous permettent cependant pas cette mise en œuvre.

Les approches locales gardent leur avantage quand on les compare à notre méthode si un nombre suffisant de données est disponible. Sinon, notre algorithme amplifie les capacités de prévision multipas des approches globales basées sur les RNRs. Des expérimentations supplémentaires devraient permettre de vérifier si les approches locales sont adaptées aux ensembles de données réelles bruitées et de les comparer avec notre méthode.

Références

Assaad, M., R. Boné et H. Cardot (2005). *Study of the Behavior of a New Boosting Algorithm for Recurrent Neural Network*. International Conference on Artificial Neural Networks, Warsaw, Poland.

Atiya, A. F., S. M. El-Shoura, S. I. Shaheen et M. S. El-Sherif (1999) A Comparison Between Neural Network Forecasting Techniques - Case Study: River Flow Forecasting. *IEEE Transactions on Neural Networks*, 10(2): 402-409.

Avnimelech, R. et N. Intrator (1999) Boosting Regression Estimators. *Neural Computation*, 11(2): 491-513.

Back, A. D. et A. C. Tsoi (1992). Stabilization Properties of Multilayer Feedforward Networks with Time-Delays Synapses. *Artificial Neural Networks*. J. T. I. Aleksander, Elsevier Science Publishers**.** 2: 1113-1116.

Boné, R. et M. Crucianu (2002). *An Evaluation of Constructive Algorithms for Recurrent Networks on Multi-Step-Ahead Prediction*. International Conference on Neural Information Processing, Singapore.

Borra, S. et A. Di Ciaccio (2002) Improving Nonparametric Regression Methods by Bagging and Boosting. *Computational Statistics and Data Analysis*, 38: 407-420.

Casdagli, M. (1989) Nonlinear Prediction of Chaotic Time Series. *Physica*, 35D: 335-356.

Chen, Y., B. Yang et J. Dong (2005) Time-series Prediction Using a Local Linear Wavelet Neural Network. *Neurocomputing*, 69: 449-465.

Chudy, L. et I. Farkas (1998) Prediction of Chaotic Time-Series Using Dynamic Cell Structures and Local Linear Models. *Neural Network World*, 8(5): 481-489.

Cook, G. D. et A. J. Robinson (1996). *Boosting the Performance of Connectionist Large Vocabulary Speech Recognition*. International Conference in Spoken Language Processing, Philadelphia, PA.

Cottrell, M., G. Simon, A. Lendasse, J.-C. Fort et M. Verleysen (2004) Double Quantization of the Regressor Space for Long-Term Time Series Prediction: Method and Proof of Stability. *Neural Networks*, 17: 1169-1181.

Cottrell, M., J. Rynkiewicz, M. Mangeas et J. F. Yao (2001) Modèles de réseaux de neurones pour l'analyse des séries temporelles ou la régression : Estimation, Identification, Méthode d'élagage SSM. *Revue d'Intelligence Artificielle*, 15(3-4): 317-332.

Czernichow, T., A. Piras, K. Imhof, P. Caire, Y. Jaccard, B. Dorizzi et A. Germont (1996) Short Term Electrical Load Forecasting with Artificial Neural Networks. *Engineering Intelligent Systems*, 4(2): 85-99.

Drucker, H. (1999). Boosting Using Neural Nets. *Combining Artificial Neural Nets: Ensemble and Modular Learning*. A. Sharkey, Springer, 51-77.

Duffy, N. et D. Helmbold (2002) Boosting Methods for Regression. *Machine Learning*, 47: 153-200.

Duhoux, M., J. Suykens, B. De Moor et J. Vandewalle (2001) Improved Long-Term Temperature Prediction by Chaining of Neural Networks. *International Journal of Neural Systems*, 11(1): 1-10.

Elman, J. L. (1989) *Representation and Structure in Connectionist Models*. CRL Technical Report 89-03, San Diego, University of California.

Freund, Y. (1990). *Boosting a Weak Learning Algorithm by Majority*. Workshop on Computational Learning Theory.

Freund, Y. et R. E. Schapire (1997) A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1): 119-139.

Gers, F., D. Eck et J. Schmidhuber (2001). *Applying LSTM to Time Series Predictable Through Time-Window Approaches*. International Conference on Artificial Neural Networks, Vienna, Austria.

Gey, S. et J.-M. Poggi (2006) Boosting and Instability for Regression Trees. *Computational Statistics and Data Analysis*, 50: 533-550.

Jaeger, H. (2004) Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304: 78-80.

Mason, L., J. Baxter, P. L. Bartlett et M. Frean (1999). Functional Gradient Techniques for Combining Hypotheses. *Advances in Large Margin Classifiers*. A. J. Smola, P. L. Bartlett, B. Schölkopf and D. Schuurmans. Cambridge, MA, MIT Press**:** 221-247.

McNames, J. (2000). *Local Modeling Optimization for Time Series Prediction*. European Symposium on Artificial Neural Networks, Bruges, Belgium.

Parlos, A. G., O. T. Rais et A. F. Atiya (2000) Multi-Step-Ahead Prediction Using Dynamic Recurrent Neural Networks. *Neural Networks*, 13(7): 765-786.

Pavlidis, N. G., D. K. Tasoulis et M. N. Vrahatis (2005). *Time Series Forecasting Methodology for Multiple-Step-Ahead Prediction*. Fourth IASTED International Conference on Computational Intelligence, Calgary, Canada.

Rumelhart, D. E., G. E. Hinton et R. J. Williams (1986). Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. D. E. Rumelhart, J. McClelland. Cambridge, MA, MIT Press**:** 318-362.

Schapire, R. E. (1990) The Strength of Weak Learnability. *Machine Learning*, 5: 197-227.

Schmidhuber, J., D. Wierstra et F. J. Gomez (2005). *Evolino: Hybrid Neuroevolution / Optimal Linear Search for Sequence Learning.* 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland.

Suykens, J. A. K. et J. Vandewalle (1995) Learning a Simple Recurrent Neural State Space Model to Behave Like Chua's Double Scroll. *IEEE Transactions on Circuits and Systems-I*, 42: 499-502.

Vesanto, J. (1997). *Using the SOM and Local Models in Time-Series Prediction*. Workshop on Self-Organizing Maps, Espoo, Finland.

Wan, E. A. (1994). Time Series Prediction by Using a Connection Network with Internal Delay Lines. *Time Series Prediction: Forecasting the Future and Understanding the Past*. A. S. Weigend and N. A. Gershenfeld, Addison-Wesley. XV**:** 195-217.

Summary

We evaluate on multi-step-ahead prediction problems, a boosting algorithm for recurrent neural networks (RNNs). The algorithm combine a large number of RNNs, each of them is generated by training on a different set of examples. This algorithm is based on the boosting algorithm and allows concentrating the training on difficult examples but, unlike the original algorithm, by taking into account all the available examples. We study the behavior of our method for multi-step-ahead prediction problems (three time series of reference) with three loss functions and with different values of a parameter. We compare our algorithm to the standard back-propagation through time (BPTT) algorithm and to other methods applied to the same datasets, including local approaches.