ACSEG 2003 proceedings - Connectionist Approaches in Economics and Management Sciences
Nantes (France), 20-21 November 2003, pp. 515-534

# LE TEST DES METHODES NEURONALES : COMMENT UTILISER LES TECHNIQUES DE REECHANTILLONNAGE ?

M. Verleysen[(1)], A. Lendasse[(2)]

Université catholique de Louvain
[(1)]DICE, place du Levant, 3, [(2)]CESAME, av. G. Lemaître 4,
B-1348 Louvain-la-Neuve
verleysen@dice.ucl.ac.be, lendasse@auto.ucl.ac.be

*Résumé: Par rapport aux modèles linéaires ou à d'autres modèles statistiques classiques, les modèles connexionnistes offrent souvent des possibilités intéressantes, mais au prix d'un apprentissage plus difficile. Une des difficultés rencontrées consiste à fixer un certain nombre de "méta-paramètres", par exemple liés à la complexité du modèle (nombre d'unités dans une couche cachée, etc.). Souvent, il n'existe pas d'autre choix que d'en tester plusieurs valeurs possibles et de choisir la meilleure, au sens d'un certain critère. Il est dès lors indispensable de disposer de méthodes efficaces permettant d'évaluer les performances d'un modèle, même face aux contraintes liées à des applications réelles, en particulier le nombre limité de données disponibles. Cet article présente différentes méthodes de rééchantillonnage, destinées à être utilisées dans le contexte de modélisation non linéaire (par exemple connexionniste), et compare leurs avantages et inconvénients au point de vue de leurs performances et de leur complexité.*

*Mots clés : Validation, estimation de performances, rééchantillonnage, modèles non linéaires.*

# TESTING NEURAL MODELS : HOW TO USE RESAMPLING TECHNIQUES ?

*Abstract: With respect to linear and other classical statistical models, connexionnist ones offer interesting potentialities, despite their more difficult learning. One of the encountered difficulties consists in setting a number of "meta-parameters", for example linked to the complexity of the model (number of units in a hidden layer, etc.). Often, there is no other choice than testing several possible values and selecting the best one according to a predefined criterion. There is thus a need for efficient methods for the evaluation of the model performances, coping with the constraints from real-world applications (in particular the limited number of available data). This paper presents some resampling methods to be used in the context of nonlinear (including connectionist) modeling, and compares their advantages and drawbacks in terms of performances and complexity.*

*Keywords: validation, performance estimation, resampling, nonlinear models.*

ACSEG 2003 proceedings - Connectionist Approaches in Economics and Management Sciences
Nantes (France), 20-21 November 2003, pp. 515-534

## 1. INTRODUCTION

Les réseaux de neurones artificiels (Werbos 1974, Bishop 1995, Weigend A.S. *et al.* 1994) sont des outils de modélisation non linéaire, utilisés dans de nombreux contextes liés à l'identification, à la classification, à la prédiction, etc. Un modèle (non linéaire) est une relation mathématique entre des données dites d'entrées et une ou des sorties, faisant intervenir un certain nombre de paramètres. Par exemple, un polynôme du deuxième degré entre une entrée à valeurs réelles et une sortie à valeurs réelles également, est un modèle non linéaire faisant intervenir trois paramètres (les coefficients des trois termes du polynôme).

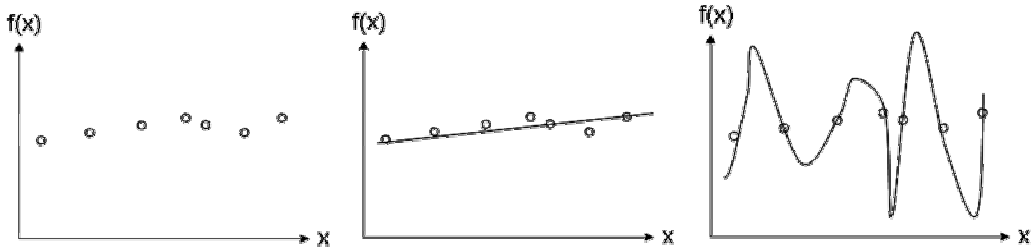
En général, la conception d'un modèle passe par deux étapes bien distinctes. La première est le choix d'un type de modèle, plus précisément d'une *structure de modèle*, ce qui revient à fixer la relation mathématique qui va être utilisée comme modèle, y compris le nombre de ses paramètres. Dans le cas de modèle polynomiaux par exemple, il s'agit de fixer l'ordre du polynôme. Ensuite, la relation mathématique étant fixée, il faut choisir au mieux ses paramètres, afin que le modèle se comporte "le mieux possible" sur un ensemble de données connues ("le mieux possible" étant une notion à définir précisément, et avec précautions).

Dans le cas des réseaux de neurones artificiels et d'autres modèles adaptatifs, la seconde étape s'appelle l'*apprentissage*, et fait l'objet de très nombreux travaux de recherches. Chaque modèle (perceptron multi-couches, réseaux à fonctions radiales de base, etc.) est connu avec son ou ses *algorithmes d'apprentissage*, plus ou moins complexes, destinés à estimer au mieux les paramètres du modèle. Même si de nombreux travaux de recherches perdureront encore longtemps en vue d'améliorer les performances des algorithmes d'apprentissage, des solutions satisfaisantes existent dans de nombreux cas.

Par contre, le choix de la structure de modèle (Lendasse *et al*. 2003) est un problème souvent plus difficile, et malgré tout moins étudié et résolu. Souvent, le choix d'un modèle est fait a priori en se basant sur une certaine expérience de l'utilisateur, avec des conséquences parfois catastrophiques… D'une part, le choix effectué peut se révéler tout simplement mauvais, entraînant des performances désastreuse pour l'ensemble du processus de modélisation. D'autre part, le manque de méthode objective de choix de structure de modèle entraîne une méfiance non justifiée à l'égard des méthodes neuronales, malgré leur potentiel.

Imaginons par exemple le (très simple) problème de modélisation illustré à la figure 1 a). Il s'agit de construire un modèle (de $\Re$ dans $\Re$) représentant au mieux un phénomène connu par l'intermédiaire de sept données seulement. En examinant la disposition des données, il vient immédiatement à l'esprit qu'un modèle très simple conviendra, tel que le modèle linéaire de la figure 1 b). Ce modèle commet de légères erreurs (il ne passe pas strictement par l'ensemble des sept données), mais il est quand même très probable qu'il représente correctement le phénomène inconnu à modéliser. Par contre, si l'on n'y prend garde, il est facile d'aboutir à un modèle trop complexe tel que celui de la figure 1 c): celui-ci commet moins d'erreur (il passe exactement pat les sept données considérées), mais n'est certainement pas ce qui était recherché.

ACSEG 2003 proceedings - Connectionist Approaches in Economics and Management Sciences
Nantes (France), 20-21 November 2003, pp. 515-534




a) b) c)

**Figure 1** : a) ensemble de données; b) modèle linéaire; c) modèle de complexité trop importante

Notons que le phénomène illustré à la figure 1 c), appelé *overfitting*[1], est nettement plus marqué lorsque la dimension de l'espace d'entrée est grande, et en tout cas plus grande que un. Lorsque cette dimension est grande, on arrive en effet facilement à des situations pour lesquelles le nombre de données semble (trop) faible par rapport à la complexité du phénomène à modéliser, favorisant ainsi l'apparition d'overfitting. Or c'est précisément quand les phénomènes à modéliser sont complexes et quand la dimension de l'espace d'entrée est importante, qu'il devient intéressant d'utiliser des réseaux de neurones artificiels, plutôt que d'autres modèles plus simples…

L'exemple de la figure 1 montre toute l'importance à apporter à deux concepts précis:

1. la définition d'un critère de performance approprié;
2. l'estimation de la valeur de ce critère, son calcul exact étant en général impossible.

Si l'on parvient à définir un critère de performance approprié (qui donc, dans l'exemple précédent, sera bon dans le cas du modèle 1 b) et mauvais dans le cas du modèle 1 c)), l'estimation de ce critère pour plusieurs structures de modèles nous donnera directement le choix de structure le plus adéquat.

La section 2 ci-dessous définit le concept d'erreur de généralisation comme critère de performance des modèles, et introduit les notions de régularisation et de validation, qui sont les deux principes à la base des méthodes d'estimation de cette erreur. La section 3 présente différentes méthodes de validation et les illustre sur un exemple simple, tandis que la section 4 compare ces méthodes sur un exemple de prédiction de séries temporelles.

## 2. GÉNÉRALISATION, RÉGULARISATION ET VALIDATION

L'exemple de la section 1 montre que la qualité d'un modèle ne doit pas être mesurée par l'erreur que celui-ci commet sur les données qui ont servi à son apprentissage. Au contraire, nous sommes intéressés par l'erreur que commettrait le modèle sur d'autres données, inconnues ou non utilisées lors de l'apprentissage. La difficulté vient bien évidemment du fait que si ces données sont inconnues, il est impossible de mesurer l'erreur commise par le

[1] Une traduction française parfois utilisée pour le terme *overfitting* est *surapprentissage*. Néanmoins, l'overfitting n'est pas toujours la conséquence d'un apprentissage mené trop loin, ce que pourrait laisser penser le terme surapprentissage. Pour cette raison, nous maintiendrons le terme en anglais dans la suite de ce texte, comme nous maintiendrons la dénomination anglaise pour le nom des méthodes de validation.

modèle sur ces mêmes données. De plus, le nombre limité de données disponibles dans des applications réelles interdit souvent d'en réserver une partie que l'on n'utilise pas lors de l'apprentissage. Cette section donne d'abord une définition générale de l'erreur de validation, puis montre deux voies à suivre pour son estimation, à savoir la régularisation et la validation.

### 2.1. Erreur de généralisation

Soit $\{x_t, y_t\} \in \Re^d \times \Re$, $1 \le t \le N$ les données disponibles pour l'apprentissage du modèle; l'ensemble de ces données est appelé *ensemble d'apprentissage*. Nous supposerons donc que le problème de modélisation à résoudre est une relation entre des données vectorielles de dimension $d$ en entrée, et des données scalaires en sortie (ceci pour une question de simplicité de notations; le raisonnement peut sans problème s'étendre à des données vectorielles en sortie). Dans le cas de l'exemple de la section 1, nous avions $d = 1$.

La relation $g$, inconnue, entre les données d'entrée et de sortie s'écrit

$$y_t = g(x_t) + \varepsilon_t , \tag{1}$$

où $\varepsilon_t$ représente l'erreur, par exemple une erreur de mesure.

Le problème de modélisation consiste à construire un modèle s'approchant le mieux possible de la relation inconnue $g(\cdot)$. Comme le principe de la validation détaillé ci-dessous consistera à construire un ensemble de modèles de structures différentes, nous noterons les modèles construits

$$\hat{y}_t = h^q(x_t, \theta(q)), \tag{2}$$

où $h^q(\cdot)$ est le modèle, $q$ permet d'identifier chaque structure de modèle ainsi construite, et $\theta$ représente l'ensemble des paramètres du modèle (ensemble dont la taille et la nature peuvent dépendre de la structure $q$).

Une fois la structure choisie (c'est-à-dire une fois la forme de relation $h^q(\cdot)$ déterminée), l'apprentissage d'un modèle consiste à trouver l'ensemble optimal $\theta^*(q)$ de paramètres du modèle, qui modélise le mieux possible les $n$ donnés d'apprentissage. Dans la plupart des cas, l'ensemble $\theta^*(q)$ est choisi de telle manière à ce qu'il minimise une erreur quadratique dite d'apprentissage:

$$E_{app}(q, \theta) = \frac{\sum_{t=1}^{N} \left( h^q(x_t, \theta(q)) - y_t \right)^2}{N} . \tag{3}$$

Notons que dans le membre de gauche de (3), la dépendance explicite de $\theta$ en fonction de $q$ a été omise, pour des questions de notations, mais que cette dépendance est évidemment à garder à l'esprit.

La définition de $E_{app}$ coïncide avec la notion plus intuitive d' "erreur commise sur les données d'apprentissage" mentionnée dans la section 1. Dans l'exemple de la figure 1, $E_{app}$ sera donc nul pour le modèle complexe de la figure 1 c), mais non nul pour le (meilleur) modèle de la figure 1 b).

En fait, il aurait été préférable de rendre minimum une autre erreur, celle commise par le modèle $h^q(\cdot)$ (muni de ses paramètres optimaux $\theta^*(q)$) sur d'autres données que celles utilisées lors de l'apprentissage. De plus, il aurait été intéressant que cette nouvelle erreur soit définie sur toutes les données possibles sur lesquelles le modèle pourra être utilisé, et donc sur un

nombre infini de données. Ceci conduit à la définition de l'*erreur de généralisation* (Lendasse *et al*. 2003):

$$E_{gen}(q,\theta*(q)) = \lim_{M\to\infty} \frac{\sum_{t=1}^{M}\left(h^q(x_t,\theta*(q)) - y_t\right)^2}{M}. \qquad (4)$$

Si maintenant on se replace dans le contexte d'un problème pour lequel on ne connaît pas la structure optimale $q^*$ à utiliser, le concept d'erreur de généralisation peut être utilisé: si plusieurs structures de modèles sont testées, et que $E_{gen}$ est estimée pour chacune d'entre elles, la meilleure structure correspondra à celle qui minimisera la valeur de $E_{gen}$.

Si l'on se restreint à une classe de modèles de complexité croissante (par exemple des polynômes, dont la complexité est mesurée par le nombres de paramètres), la différence entre erreurs d'apprentissage et de généralisation peut être illustrée par le diagramme typique de la figure 2. On y voit que de façon générale, une augmentation de la complexité ne fait que diminuer l'erreur d'apprentissage (comme dans l'exemple de la figure 1, où celle-ci devient nulle), alors qu'un optimum existe au niveau de l'erreur de généralisation.

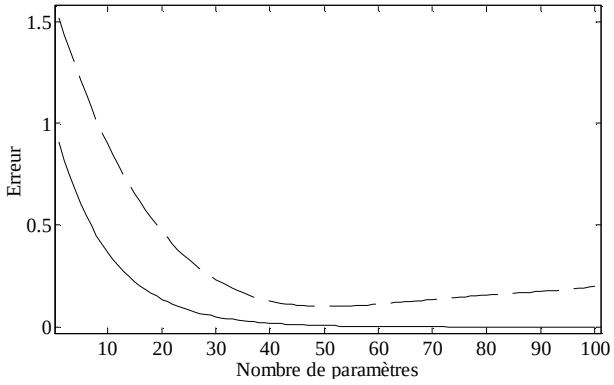



**Figure 2** : évolution typique des erreurs d'apprentissage (trait plein) et de généralisation (trait discontinu) en fonction de la complexité d'un modèle.

On peut donc en conclure que le problème de sélection de modèle se ramène à celui de trouver le meilleur estimateur possible de l'erreur de généralisation $E_{gen}$. En fait, nous verrons ci-dessous que cette condition est un peu trop restrictive: des estimateurs biaisés de $E_{gen}$ peuvent suffire, à condition que l'erreur d'estimation ne modifie pas l'ordre dans lequel différentes structures de modèle pourraient être classées. Mais considérons pour l'instant que le but est de trouver le meilleur estimateur possible de l'erreur de généralisation définie par (4).

Dans la suite de cette section, nous aborderons deux manières fort différentes utilisables pour estimer l'erreur de généralisation d'un modèle: la régularisation et la validation. Dans la section suivante, nous détaillerons différentes méthodes de validation, ainsi que leurs avantages et inconvénients.

### 2.2. Régularisation

La figure 2, ainsi que le raisonnement effectué plus haut sur l'exemple de la figure 1, nous suggèrent que l'erreur de généralisation (4) peut être approchée par la somme de l'erreur d'apprentissage (3), et d'un terme de pénalisation, qui augmente avec la complexité du modèle. En effet, d'une part il est naturel de penser que $E_{app}$ sera plus faible que $E_{gen}$: la première erreur est en effet celle qui est directement minimisée lors de l'apprentissage des paramètres $\theta$, sur les données appartenant à l'ensemble d'apprentissage; si cette minimisation a été efficacement réalisée, elle se doit d'être optimale par rapport aux données de l'ensemble d'apprentissage (utilisées pour le calcul de $E_{app}$), mais pas nécessairement par rapport à d'autres données (utilisées pour le calcul de $E_{gen}$). D'autre part, au fur et à mesure que la complexité d'un modèle croît, le risque d'overfitting, tel que défini dans la section précédente, croît, ce qui amènera la différence entre $E_{app}$ et $E_{app}$ à croître également. De façon générale, on peut donc exprimer l'erreur de généralisation sous la forme

$$E_{gen}(q, \theta^*(q)) = E_{app}(q, \theta^*(q)) + \lambda\Omega\,, \qquad (5)$$

où $\Omega$ est un terme reflétant la complexité du modèle, et $\lambda$ un facteur de pondération. Le second terme de (5) est appelé *terme de régularisation*.

Malgré le fait que de nombreux résultats alliant justification théorique et applicabilité existent en ce qui concerne le terme de régularisation, celui-ci souffre en général de plusieurs inconvénients.

- Suivant les cas, de nombreuses définitions de $\Omega$ existent. La complexité $\Omega$ peut se mesurer par rapport au nombre de paramètres d'un modèle (mais dans ce cas, le terme de régularisation risque de ne pas être continu, et donc pas dérivable); elle peut aussi se caractériser par des notions intuitivement liées à la complexité, comme l'intégrale de la dérivée seconde d'un modèle sur son domaine de définition (se référer à la figure 1 pour en être convaincu).
- Plusieurs définitions du terme de régularisation se basent sur des développements asymptotiques, faisant par exemple l'hypothèse d'un nombre de données tendant vers l'infini. Or c'est précisément dans des situations fort différentes, lorsque le nombre de données disponibles est faible, que l'intérêt de différencier $E_{app}$ et $E_{gen}$ existe. Sans en faire une généralité, on peut dire que des précautions extrêmes doivent être prises lorsque ces critères sont utilisés dans des cas réels difficiles.
- Même lorsque la définition de la complexité $\Omega$ est acquise, fixer une pondération $\lambda$ adéquate entre l'erreur d'apprentissage et la complexité peut relever de l'art de la divination, des méthodes objectives n'existant tout simplement pas dans de nombreux cas.

Signalons que les critères AIC d'Akaike (Akaike 1973), les critères BIC (Efron *et al*. 1998) et bien d'autres peuvent être exprimés sous la forme (5).

Même si, répétons-le, des résultats intéressants de régularisation sous la forme (5) existent, il reste donc difficile de généraliser ces résultats à toutes les techniques de modélisation non linéaire, et à les appliquer à de nombreux problèmes réels. Pour cette raison, les techniques de validation ci-dessous sont souvent préférées, malgré leur coût-calcul relativement élevé.

### 2.3. Validation

Au vu des difficultés qu'il y a à définir, a priori, un terme de régularisation de la forme (5), une autre possibilité consiste à vérifier a posteriori les performances d'un modèle. Le principe, simpliste mais efficace, consiste alors à construire un ensemble important de

modèles de structures $q$ différentes, à faire l'apprentissage des paramètres optimaux $\theta^*(q)$ pour chacun d'entre eux, et à sélectionner en final la structure (et ses paramètres optimaux) qui donnent la plus faible erreur de généralisation.

L'erreur de généralisation $E_{gen}$ ne peut néanmoins pas être calculée de façon exacte par (4): on ne dispose jamais d'un ensemble infini de données… On approximera alors l'erreur de généralisation par une moyenne sur un ensemble fini de données de validation:

$$\hat{E}_{gen}(q,\theta^*(q)) = \frac{\sum_{t=1}^{M}\left(h^q(x_t,\theta^*(q)) - y_t\right)^2}{M}. \qquad (6)$$

C'est ici qu'intervient la notion importante d'indépendance entre les ensembles d'apprentissage et de validation. Pour que l'estimation (6) détecte correctement le phénomène d'overfitting, il est indispensable que les données utilisées dans (6) ne soient pas les mêmes que celles utilisées dans (3). En présence d'un nombre fini de données, et sous l'hypothèse d'une distribution i.i.d. de celles-ci, il faut donc utiliser une partie des données disponibles pour l'apprentissage, et l'autre pour estimer l'erreur de généralisation. Néanmoins, ne pas utiliser cette seconde partie des données pour l'apprentissage revient à se priver d'une information importante, et donc à diminuer la qualité de l'apprentissage; le but des méthodes de validation détaillées dans la section 3 sera donc de permettre une estimation aussi précise que possible de l'erreur de généralisation, tout en évitant de retirer trop de données de l'ensemble d'apprentissage.

## 3. MÉTHODES DE VALIDATION

Le but de ce chapitre est de décrire et comparer différentes méthodes de validation (parfois appelées méthodes de rééchantillonnage), utilisées pour estimer au mieux l'erreur de généralisation commise par un modèle, tout en laissant le plus grand nombre possible de données disponibles pour l'apprentissage.

Les méthodes seront tout d'abord illustrées sur un exemple simple, artificiel, de modélisation d'une fonction de $\Re$ dans $\Re$. Les données connues $\{x_t, y_t\} \in \Re\times\Re$, $1 \le t \le 10$ sont représentées à la figure 3.

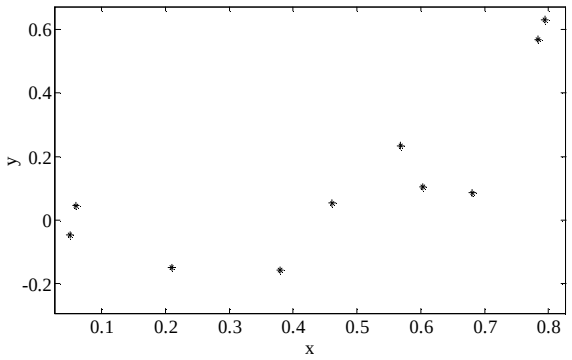



**Figure 3** : ensemble de 10 données utilisées pour illustrer les méthodes de validation.

Ces données artificielles ont été générées à l'aide de la relation :

$$y_t = -x_t + 2x_t^2 + \varepsilon_t \,, \tag{7}$$

avec ε une variable aléatoire uniformément distribué entre -0.167 et 0.167.

Le problème illustré par les méthodes de validation ci-dessous sera, étant entendu que l'origine (7) des données est considérée comme inconnue, de sélectionner la meilleure structure de modèle parmi les modèles suivants:

$$\begin{cases} \hat{y}_t = a + bx_t \\ \hat{y}_t = a + bx_t + cx_t^2 \\ \hat{y}_t = bx_t + cx_t^2 \\ \hat{y}_t = a + bx_t + cx_t^2 + dx_t^3 + ex_t^4 + fx_t^5, \end{cases} \tag{8}$$

correspondant respectivement à un modèle linéaire, un modèle quadratique, un modèle quadratique sans terme indépendant, et un modèle d'ordre 5. La figure 4 représente les approximations obtenues par ces quatre modèles, avec les paramètres optimaux (au sens de $E_{app}$ minimum) dans chaque cas.

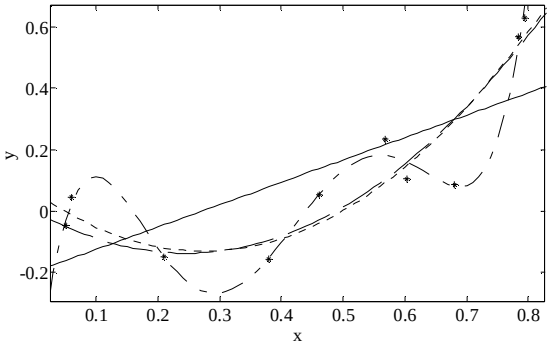



**Figure 4** : Approximation des données de la figure 3 par 4 modèles: modèle linéaire en trait continu, modèle quadratique en pointillé, modèle quadratique sans terme indépendant en tirets et modèle d'ordre 5 en tirets pointés.

Par (7), nous savons que le "vrai" modèle est celui quadratique sans terme indépendant. Le modèle linéaire n'est pas assez complexe et donnera donc une erreur d'apprentissage importante, tandis que le modèle d'ordre 5 mènera à un overfitting important. Ce sont ces propriétés que nous allons vérifier sur les différentes méthodes de validation décrites ci-dessous.

### 3.1. Validation simple

Une première technique de validation consiste tout simplement à diviser les données disponibles en deux ensembles (les ensembles d'apprentissage et de validation), sans qu'aune donnée ne soit commune (Ljung 1987). Un nombre non négligeable de données est bien entendu nécessaire dans l'ensemble de validation pour estimer correctement (6), ce qui réduit d'autant le nombre de données disponibles pour l'apprentissage. Souvent on garde 2/3 des données dans l'ensemble d'apprentissage pour en réserver 1/3 pour la validation (voir (Ljung, 1999) pour une justification de cette proportion). Pour éviter autant que possible tout

problème lié à une dérive des données (et donc à la remise en cause de leur caractère i.i.d.), les ensembles d'apprentissage et de validation sont tirés aléatoirement parmi les *N* données disponibles.

Cette méthode n'est justifiable que lorsque le nombre *N* de données est très important (vis-à-vis de la dimension de l'espace d'entrée et de la complexité de la relation à approximer). Dans les autres cas, elle peut s'avérer catastrophique, comme dans le cas de l'exemple décrit plus haut. La table 1 donne la valeur de $E_{gen}$ estimée par la méthode de validation simple, sur les quatre modèles (8).

| Modèle linéaire | Modèle quadratique | Modèle quadratique sans terme indépendant | Modèle d'ordre 5 |
|---|---|---|---|
| 0.0370 | 0.0181 | 0.0118 | 0.0038 |

**Table 1** : estimation de l'erreur de généralisation par la méthode de validation simple, sur les quatre modèles (8).

On voit que le meilleur modèle résultant de la table 1 est celui d'ordre cinq, ce qui n'est pas le résultat que nous espérions. Malgré les mauvaises performances de la méthode de validation simple, celle-ci est encore toujours utilisée par de nombreux utilisateurs de réseaux de neurones artificiels, et pire par de nombreux chercheurs dans le domaine. Sauf dans des cas précis et exceptionnels (par exemple lorsque le nombre de données disponibles est très grand, au vu de la dimension de l'espace considéré, et donc que les données peuvent être considérées comme fort redondantes), cette technique est à proscrire.

### 3.2. Monte-Carlo cross validation

L'inconvénient majeur de la technique de validation simple est que ses résultats dépendent intimement du tirage aléatoire des données, entre les deux ensembles d'apprentissage et de validation (Lendasse *et al*. 2003). Un remède possible consiste alors à effectuer plusieurs tirages de ce type, à estimer $E_{gen}$ pour chacun de ces tirages, et à retenir la moyenne de ces estimations. Si l'indice *j* est utilisé pour caractériser les différents tirages ($1 \leq j \leq J$), l'erreur de généralisation pour chacun de ceux-ci peut s'écrire:

$$E_j\left(q, \theta_j^*(q)\right) = \frac{\sum_{x_t \in V_j} \left(h^q\left(x_t, \theta_j^*(q)\right) - y_t\right)^2}{size\left(V_j\right)}, \qquad (9)$$

avec $V_j$ l'ensemble des données de validation (non utilisées lors de chaque apprentissage), et size($V_j$) le nombre de données qui s'y trouvent (par exemple *N*/3 par analogie avec la validation simple). L'erreur de généralisation sera alors la moyenne des erreurs (9):

$$\hat{E}_{gen}(q) = \frac{\sum_{j=1}^{J} E_j\left(q, \theta_j^*(q)\right)}{J}. \qquad (10)$$

Précisons que la dépendance explicite de l'estimation de l'erreur de généralisation par rapport aux paramètres optimaux du modèle a été omise du membre de gauche de (10), car chaque modèle considéré dans (9) et donc dans le membre de droite de (10) possède ses propres paramètres optimaux.

La table 2 donne l'estimation de l'erreur de généralisation par la méthode Monte-Carlo cross-validation, sur l'exemple (8), pour différentes valeurs de *J*. On y voit qu'à partir d'une valeur de $J = 100$, les deux modèles quadratiques ressortent. Bien entendu, une valeur élevée de *J* signifie un grand nombre de tirages aléatoires suivis d'un nombre identique d'apprentissages de modèles. Dans le cas des réseaux de neurones artificiels, l'apprentissage d'un seul modèle pouvant être long (en terme de temps-calcul), ceci peut conduire à des contraintes incompatibles avec la réalité d'applications de modélisation.

| *J* | Modèle linéaire | Modèle quadratique | Modèle quadratique sans terme indépendant | Modèle d'ordre 5 |
|---|---|---|---|---|
| 10 | 0.0184 | 0.0229 | 0.0201 | 0.0097 |
| 100 | 0.0652 | 0.0275 | 0.0251 | 13.8062 |
| 1000 | 0.0743 | 0.0276 | 0.0257 | 154.8485 |
| 10000 | 0.0798 | 0.0267 | 0.0250 | 208.5566 |

**Table 2** : estimation de l'erreur de généralisation par la méthode de Monte-Carlo cross-validation, sur les quatre modèles (8), pour différentes valeurs du nombre *J* de tirages effectués.

### 3.3. *K*-fold cross-validation

Si l'augmentation du nombre de tirages (et donc du nombre d'évaluations de (9) dans la méthode de Monte-Carlo cross-validation) permet d'utiliser successivement les mêmes données, tantôt pour l'apprentissage, tantôt pour la validation, leur caractère aléatoire empêche toute certitude quant au fait que chaque donnée soit, au moins approximativement, utilisée le même nombre de fois pour l'apprentissage ou pour la validation. Seule une valeur prohibitive de *J* permettrait d'atteindre cette certitude.

Une variante consiste alors à découper l'ensemble des *N* données disponibles en *K* sous-ensembles disjoints de taille approximativement identiques (Kohavi 1995). On peut alors obtenir *K* évaluations de (9), en utilisant à chaque fois *K*-1 sous-ensembles pour l'apprentissage et un pour la validation. Chacun des *K* sous-ensemble joue alors une et une seule fois le rôle d'ensemble de validation, ce qui répond à l'argument ci-dessus. L'estimation de l'erreur de généralisation est alors calculée par (10), où $J = K$.

### 3.4. Leave-one-out

Le leave-one-out est une méthode de *K*-fold cross-validation poussée à l'extrême, lorsque $K = N$ (Efron *et al*. 1998, Kohavi 1995). En d'autres mots, disposant de *N* données d'apprentissage, *N* modèles sont construits sur base de *N*-1 données, et testés sur l'unique donnée restante. Pour chacun des modèles, on calcule donc l'erreur

$$E_j\left(q, \theta_j^*(q)\right) = \left(h^q\left(x_j, \theta_j^*(q)\right) - y_t\right)^2 \qquad (11)$$

sur la donnée $x_j$ écartée de l'apprentissage, et l'estimation de l'erreur de généralisation qui en résulte est

$$\hat{E}_{gen}(q) = \frac{\sum_{j=1}^{N} E_j\left(q, \theta_j^*(q)\right)}{N}. \qquad (12)$$

La combinaison des équations (11) et (12) ressemble fort à l'équation (3), donnant la valeur de l'erreur d'apprentissage d'un modèle. La différence, essentielle, réside cependant dans le fait que chaque terme $E_j$ dans (12) est calculé sur un modèle différent, de paramètres $\theta_j^*$, et construit sur base de $N$-1 données, alors qu'il s'agissait d'un unique modèle (et donc d'un ensemble unique $\theta^*$ de paramètres) dans l'équation (3).

La table 3 donne l'estimation de l'erreur de généralisation par la méthode du leave-one-out, sur l'exemple (8). On y voit que le résultat escompté n'est pas obtenu: l'estimation de l'erreur de généralisation est en effet la plus faible sur le modèle d'ordre 5. Il est connu que la méthode du leave-one-out ne donne de bons résultats que sur des ensembles relativement grands de données, ce qui n'est pas le cas de cet exemple ($N$ = 10). Précisons que si le nombre de calculs pour obtenir (12) semblent proportionnel au nombre de données $N$, il existe des moyens permettant de calculer chaque terme (11) à partir de l'erreur d'apprentissage (3), dans le cas de modèles linéaires. Cette amélioration est cependant impossible dans le cas de modèles non linéaires, sauf à recourir à des approximations difficilement justifiables.

| Modèle linéaire | Modèle quadratique | Modèle quadratique sans terme indépendant | Modèle d'ordre 5 |
|---|---|---|---|
| 0.0488 | 0.0153 | 0.0146 | 0.0045 |

**Table 3** : estimation de l'erreur de généralisation par la méthode du leave-one-out sur les quatre modèles (8).

### 3.5. Bootstrap

Le Bootstrap (Efron *et al.* 1998, Efron *et al.* 1997, Efron 1983) est une méthode relativement récente qui a été développée afin d'estimer certains paramètres statistiques comme la moyenne, la variance, etc. Dans le cas de la sélection de structure de modèle, le paramètre qui est estimé est l'erreur de généralisation. Mais ce qui est particulier au Bootstrap, c'est que cette erreur n'est pas calculée directement. Ce qui est en réalité estimé, c'est la différence entre l'erreur de généralisation et l'erreur d'apprentissage calculée sur l'ensemble initial de données. Cette différence est appelée *optimisme*. L'estimée de l'erreur de généralisation sera donc la somme de l'erreur d'apprentissage et de l'optimisme :

- L'erreur d'apprentissage est calculée en utilisant tous les points de l'ensemble des données. Il n'est donc pas nécessaire de diviser les données en plusieurs ensembles.
- L'optimisme est ensuite calculé en utilisant une technique de rééchantillonnage détaillée ci-dessous ; celle-ci n'est pas très différente des techniques de division en ensembles d'apprentissage et de validation qui ont été utilisées dans les sections précédentes.

La méthode du Bootstrap consiste en différentes étapes successives, détaillées ci-dessous pour une structure particulière $q$ de modèle, comme dans les sous-sections précédentes. Soit $I$ l'ensemble initial de $N$ données.

1. Dans l'ensemble initial, on tire $N$ points au hasard, avec répétition. Ce nouvel ensemble $A_j$ qui a la même taille que l'ensemble initial est utilisé comme ensemble

d'apprentissage. L'ensemble de validation (noté *V*) est formé des données de l'ensemble initial (*V* = *I*). Ce processus porte le nom de rééchantillonnage.

2. Pour chaque valeur de *j* (correspondant à une répétition du processus de bootstrap, $1 \le j \le J$), on répète les opérations suivantes.
   a. L'apprentissage d'un modèle de structure *q* est effectué sur l'ensemble $A_j$. L'erreur d'apprentissage est calculée:

$$E_j^{A_j,A_j}\left(q,\theta_j^*(q)\right)=\frac{\sum_{t=1}^{N}\left(h^q\left(x_t^{A_j},\theta_j^*(q)\right)-y_t^{A_j}\right)^2}{N}. \quad (13)$$

   Par rapport aux notations précédentes, le premier exposant $A_j$ dans l'expression de *E* dans le terme de gauche signifie que le modèle a été appris sur l'ensemble $A_j$, tandis que le second exposant $A_j$ signifie que le modèle a été testé sur $A_j$ également. Les exposants $A_j$ ajoutés aux données d'entrée $x_t$ et aux données de sortie $y_t$ insistent sur le fait qu'il s'agit des données provenant de l'ensemble $A_j$, et non de l'ensemble initial *I*.
   b. L'erreur de généralisation de ce même modèle est calculée également:

$$E_j^{A_j,V}\left(q,\theta_j^*(q)\right)=\frac{\sum_{t=1}^{N}\left(h^q\left(x_t^{V},\theta_j^*(q)\right)-y_t^{V}\right)^2}{N}. \quad (14)$$

   Cette fois, comme il s'agit d'une erreur de généralisation, la somme du membre de droite de (13) porte sur les données de l'ensemble de validation *V*, égal à l'ensemble initial *I*. Certaines données de *I* se retrouvant dans $A_j$, il y a bien un certain recouvrement dans les données utilisées dans les évaluations de (13) et (14).
   c. La différence entre ces deux erreurs reflète, sur un choix particulier d'ensembles, la différence entre les erreurs de généralisation et d'apprentissage, commises par un même modèle. Cette différence est appelée l'*optimisme*:

$$optimisme_j\left(q,\theta_j^*(q)\right)=E_j^{A_j,V}\left(q,\theta_j^*(q)\right)-E_j^{A_j,A_j}\left(q,\theta_j^*(q)\right). \quad (15)$$

3. Une évaluation globale de l'optimisme effectué par un modèle de structure *q* est alors donnée par la moyenne des optimismes calculés jusqu'ici:

$$\hat{optimisme}_j\left(q,\theta_j^*(q)\right)=\frac{\sum_{j=1}^{J} optimisme_j\left(q,\theta_j^*(q)\right)}{J}. \quad (16)$$

4. L'erreur d'apprentissage sur l'ensemble initial *I* est alors calculée:

$$E^{I,I}\left(q,\theta^*(q)\right)=\frac{\sum_{t=1}^{N}\left(h^q\left(x_t^{I},\theta^*(q)\right)-y_t^{I}\right)^2}{N}. \quad (17)$$

5. Une approximation de l'erreur de généralisation d'un modèle de structure *q* avec ses paramètres optimaux θ*(*q*) est alors obtenue en faisant la somme de son erreur d'apprentissage et de son optimisme:

$$\hat{E}_{gen}(q)=\hat{optimisme}(q)+E^{I,I}\left(q,\theta^*(q)\right). \quad (18)$$

Une manière élégante de comprendre le Bootstrap est illustrée ci-dessous en utilisant l'égalité suivante :

$$E_{gen}(q) = E_{gen}(q) - E^{I,I}(q,\theta*(q)) + E^{I,I}(q,\theta*(q)). \quad (19)$$

On définit *l'optimisme* comme l'écart entre l'erreur de généralisation et l'erreur d'apprentissage sur l'ensemble initial de points :

$$optimisme(q) = E_{gen}(q) - E^{I,I}(q,\theta*(q)), \quad (20)$$

ce qui donne

$$E_{gen}(q) = optimisme(q) + E^{I,I}(q,\theta*(q)). \quad (21)$$

Le terme *optimisme*(*q*) mesure l'écart entre l'erreur obtenue sur les données initiales et l'erreur obtenue sur un ensemble infini des données. La relation (15) mesure l'écart entre l'erreur obtenue sur un nouvel ensemble d'apprentissage et l'erreur obtenue sur l'ensemble initial, utilisé comme ensemble de validation (et comportant des données non utilisées lors de l'apprentissage). La méthode de Bootstrap suppose qu'en moyenne ces deux différences sont identiques. Cette hypothèse porte le nom de principe du *plug-in*, et peut s'exprimer sous la forme :

$$E_{gen}(q) - E^{I,I}(q,\theta*(q)) = \lim_{J\to\infty} \frac{\sum_{j=1}^{J} E_j^{A_j,V}\left(q,\theta_j*(q)\right) - E_j^{A_j,A_j}\left(q,\theta_j*(q)\right)}{J}. \quad (22)$$

La limite est approximée par la somme finie (15).

La méthode du bootstrap a été appliquée sur l'exemple (8), pour différentes valeurs de *J*. On y voit que la méthode parvient à identifier le modèle quadratique sans terme indépendant comme étant le modèle cherché, et ce même pour des petites valeurs de *J*.

| *J* | Modèle linéaire | Modèle quadratique | Modèle quadratique sans terme indépendant | Modèle d'ordre 5 |
|---|---|---|---|---|
| 10 | 0.0384 | 0.0209 | 0.0155 | 8.3898 |
| 100 | 0.0345 | 0.0301 | 0.0128 | 703.74 |
| 1000 | 0.0325 | 0.0807 | 0.0112 | 7846.5 |
| 10000 | 0.0333 | 0.1267 | 0.0118 | 6422.5 |

**Table 4** : estimation de l'erreur de généralisation par la méthode du bootstrap, sur les quatre modèles (8), pour différentes valeurs du nombre *J* de répétitions effectuées.

Il existe de nombreuses variantes ou améliorations de la méthode du bootstrap, basées sur le même principe. Sans entrer dans les détails, on peut citer:

- le boostrap 632 (Efron *et al*. 1997) qui consiste à corriger le biais (voir discussion ci-dessous) commis par la méthode du bootstrap, par une pondération différente des termes représentant les erreurs d'apprentissage et de généralisation.
- le fast boorstrap (Simon *et al*. 2003), qui consiste dans des conditions précises à constater la linéarité de l'optimisme en fonction du nombre de paramètres du modèle. Pour tester plusieurs modèles de structure *q* croissante, il suffit donc d'évaluer l'optimisme sur deux (ou un petit nombre) de modèles plutôt que sur l'ensemble d'entre eux.

### 3.6. Comparaison des méthodes de validation

Les sections 3.1 à 3.5 présentent plusieurs méthodes destinées à évaluer l'erreur de généralisation d'une structure donnée de modèles, afin d'ensuite sélectionner la structure la plus adéquate. Mais quelle méthode faut-il choisir pour un problème donné? De plus, les critères de régularisation abordés dans la section 2.2 ne sont-ils pas adéquats également, et plus simples à utiliser?

Dans (Efron *et al*. 1998), on conseille d'utiliser les méthodes de régularisation pour les problèmes simples et celles de validation pour les problèmes complexes. Par problème complexe, on entend des problèmes où les modèles sont non linéaires et le nombre de paramètres n'est pas connu avec exactitude. De plus, de façon générale, les méthodes de validation sont plus robustes : elle donneront de bons résultats même si le nombre de données disponibles est fort réduit. En effet, les méthodes de régularisation du type AIC et BIC sont conçues en faisant l'hypothèse d'un nombre de données tendant vers l'infini. Elles ne donneront donc de bons résultats que lorsque ce nombre est grand (Stone 1997, Shao *et al*. 1995). Précisons quand même que le leave-one-out souffre du même inconvénient, comme mentionné dans la section 3.4.

En ce qui concerne les méthodes de validation, dans (Efron *et al*. 1998), il est affirmé que toutes les méthodes présentées sont asymptotiquement équivalentes. Ceci signifie que pour un nombre de données tendant vers l'infini, chacune des méthodes sélectionnera la même structure de modèle. Cela ne signifie en aucun cas que les estimations $\hat{E}_{gen}(q)$ de l'erreur de généralisation seront identiques pour chaque méthode; il se peut qu'une méthode surévalue systématiquement toutes les estimations par exemple (on parlera alors d'estimation *biaisée*). Néanmoins, on est assuré que dans un cas asymptotique, toutes les structures de modèles seront classées, par une méthode de validation particulière, dans un ordre tel que la structure donnant lieu à la valeur la plus faible de $\hat{E}_{gen}(q)$ soit bien la structure recherchée.

Par contre, ces méthodes se comportent de manières assez différentes sur des échantillons de petite taille. Dans (Efron 1993), on ajoute que les méthodes de *K*-Fold Cross-Validation et Leave-One-Out sont non biaisées mais de grande variance. Cela signifie que l'estimation de $\hat{E}_{gen}(q)$ sera bonne, mais uniquement pour un grand nombre de répétitions *J*. Cette affirmation est confirmée dans l'exemple utilisé tout au long de cette section. Les temps de calculs nécessaires pour ces méthodes sont donc le plus souvent prohibitifs. En particulier pour la méthode de Leave-One-Out, le nombre de répétitions est égal au nombre de données d'apprentissage; cette méthode nécessite donc en même temps beaucoup de données et beaucoup de temps calcul. Elle est donc déconseillée dans la plupart des cas. La méthode de *K*-fold cross-validation souffre des mêmes inconvénients, mais dans une moindre mesure; le choix du nombre *K* d'ensembles reste cependant crucial.

Il est montré dans (Efron *et al*. 1998) que la méthode de Monte-Carlo Cross-Validation est non biaisée avec une grande variance tandis que la méthode de Bootstrap possède une faible variance mais est biaisée. Il est également montré que ce biais dépend des données d'apprentissage et non des différentes structures de modèle qui sont testées. Néanmoins, même si une méthode est biaisée elle peut avoir les caractéristiques telles qu'elle conduit au bon choix de modèle comme expliqué ci-dessus. L'argument d'une faible variance est alors décisif: une faible variance signifie la possibilité de réaliser une nombre relativement faible d'itérations pour obtenir une estimation acceptable de l'erreur de généralisation. Comme toutes les méthodes de validation sont très gourmandes en temps-calcul, cet argument fait

pencher le choix en faveur du bootstrap. Le bootstrap 632, mentionné dans la section 3.5, possède les mêmes atouts que le bootstrap, en même temps qu'un biais plus faible pour l'estimation de l'erreur de généralisation.

La table 5 résumé les caractéristiques des méthodes de validation décrites ci-dessous, sous l'angle du biais et de la variance dans l'estimation de $\hat{E}_{gen}(q)$.

| $\hat{E}_{gen}(q)$ | **Biais** | **Variance** |
|---|---|---|
| Monte-Carlo Cross-Validation | Non | Grande |
| *K*-Fold Cross-Validation | Non | Grande |
| Leave-One-Out | Non | Grande |
| Bootstrap | Oui | Petite |
| Bootstrap 632 | Faible | Petite |

**Table 5** : synthèse des caractéristiques des méthodes de validation.

Efron ajoute que le Bootstrap 632 surpasse toutes les autres méthodes mais n'apporte pas de preuve à cette affirmation. Au contraire, certains auteurs comme Kohavi dans (Kohavi 1995), affirment que dans certains problèmes comme les problèmes de classification, la méthode de Monte-Carlo Cross-Validation donne de meilleurs résultats que le Bootstrap et le Bootstrap 632. Il n'existe donc malheureusement pas d'argument définitif pour choisir entre les différentes méthodes de validation; les caractéristiques ci-dessus doivent être examinées à la lumière de la complexité de la relation à approximer, du nombre de données disponibles, et du temps-calcul requis pour l'estimation de l'erreur de généralisation.

Dans la section qui suit, les méthodes de Monte-Carlo cross-validation, de Leave-One-Out, de Bootstrap et de Bootstrap 632 vont être utilisées sur un problème réel de prédiction de séries temporelles, afin de les comparer.

## 4. EXEMPLE: PRÉDICTION DE SÉRIES TEMPORELLES

Cette section présente l'application et la comparaison des méthodes décrites ci-dessus, à un problème réel de prédiction de séries temporelles. Soit une fonction $y(i)$, supposée connue pour $0 \leq i \leq t$. Le problème consiste à prédire les valeurs inconnues de la série pour $i > t$, à partir de l'information contenue dans les valeurs connues. En pratique, on n'utilisera pour la prédiction qu'un ensemble fini, appelé régresseur, de valeurs passées, par exemple:

$$\hat{y}_{t+1} = h^q\left(y_t, y_{t-1}, y_{t-2}, y_{t-3}, \theta(q)\right). \qquad (23)$$

Dans (23), le régresseur est supposé formé des quatre dernières valeurs connues de la série; $h^q(\cdot)$ représente le modèle de prédiction utilisé, et $\theta(q)$ ses paramètres.

La série SantaFeA[2] a été proposée comme benchmark en 1994 par Neil Gershenfeld et Andreas Weigend (Weigend *et al*. 1994). Cette série est devenue un des benchmarks les plus

[2] Les données ont été collectées par Udo Huebner de la Phys.-Techn. Bundesanstalt, Braunschweig, Allemagne, en collaboration avec N. B. Abraham et C. O. Weiss. Ces données ont été enregistrées sur un laser infrarouge dans un état chaotique. Les mesures ont été faites sur un laser 14NH3 (81.5-micron), alimenté optiquement par une ligne P(13) d'un laser N2O via une transmission NH3 aQ(8,7).L'intensité a été enregistrée par un oscilloscope LeCroy.

utilisés dans le domaine de la prédiction de séries temporelles. Les 1000 premiers points sont utilisés comme ensemble d'apprentissage, et sont représentés à la figure 5.

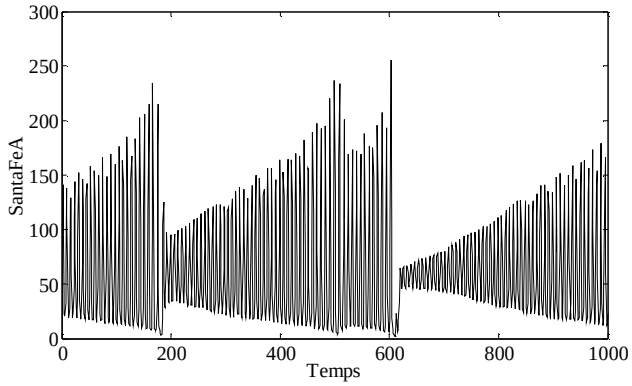



**Figure 5** : SantaFe A, ensemble d'apprentissage.

Cette série a été choisie comme benchmark car les auteurs de (Weigend *et al*. 1994) ont également fourni la suite de cette série, représentée à la figure 6, ce qui va permettre de valider le choix de la structure de modèle la plus adéquate. Le nombre de points dans cette suite est de 9200.

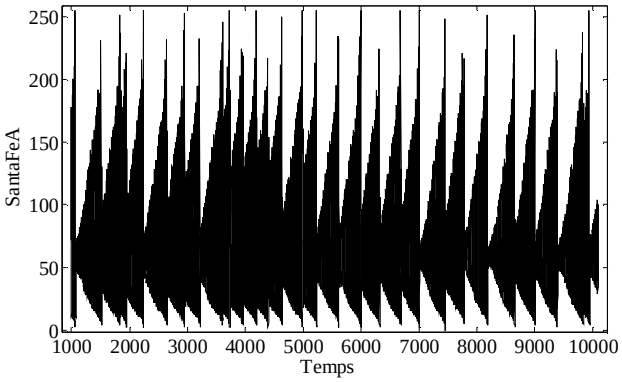



**Figure 6** : SantaFe A, ensemble de test.

Le modèle de prédiction utilisé pour cette série est

$$\hat{y}_{t+1} = h^q\left(y_t, y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}, \theta(q)\right), \tag{24}$$

où $h^q$ est un réseaux de neurones artificiels (modèle non linéaire) de type RBFN (Radial-Basis Function Networks), d'équation

$$y = h^q\left(x, \theta(q)\right) = \sum_{j=1}^{P} \lambda_j e^{-\frac{\left\|x - c_j\right\|^2}{2\sigma_j^2}}. \tag{25}$$

Dans (25), $y$ est la sortie du modèle (et correspondra donc à la valeur à prédire dans le cas de la prédiction de séries), $x$ est son entrée (vectorielle, correspondant au régresseur de l'équation (24)), et $c_j$, $\sigma_j$ et $\lambda_j$ sont ses paramètres. La question de l'apprentissage des paramètres $c_j$, $\sigma_j$ et $\lambda_j$ est un problème classique dans le domaine des réseaux de neurones artificiels, et est traité dans de nombreuses références. L'apprentissage utilisé pour les expériences décrites ci-dessous est détaillé dans (Verleysen 1996, Benoudjit 2002, Benoudjit 2003).

Les équations (24) et (25) possèdent deux "méta-paramètres" de structure qu'il convient de fixer: d'une part la taille du régresseur, et d'autre part le nombre *P* d'unités (noyaux gaussiens) dans le réseau RBFN. Dans les expériences qui suivent la taille du régresseur a été fixée a priori, selon (24). Le choix a été effectué en sa basant sur les résultats publiés dans (Weigend *et al*. 1994); notons cependant que la même méthodologie de choix de structure de modèle pourrait être appliquée au choix du régresseur. Cette méthodologie n'a donc été appliquée qu'à l'autre paramètre de structure, à savoir le nombre *P* d'unités dans le réseau RBFN.

Le fait de connaître un large ensemble de valeurs prédites (figure 6) de la série en question nous permet de calculer, a priori, le "meilleur" modèle. Par définition, le meilleur modèle est celui qui, après avoir ajusté ses paramètres sur un ensemble d'apprentissage, généralisera le mieux sur un ensemble infini de données inconnues (équation (26)). Dans le cas de la série SantaFe A, un ensemble infini de données n'est évidemment pas disponible; néanmoins, comme le nombre de données dans l'ensemble de test est très grand par rapport au nombre de données dans l'ensemble d'apprentissage, l'erreur de généralisation peut être adéquatement approchée par l'erreur commise sur les 9200 données non utilisées lors de l'apprentissage:

$$E_{gen}(q,\theta^*(q)) = \lim_{M\to\infty} \frac{\sum_{t=1}^{M}\left(h^q(x_t,\theta^*(q)) - y_t\right)^2}{M} \approx \frac{\sum_{t=1}^{9200}\left(h^q(x_t,\theta^*(q)) - y_t\right)^2}{9200}, \qquad (26)$$

où le dernier membre est évalué sur l'ensemble des points de validation.

Des réseaux RBFN avec de 20 à 140 (par pas de 20) unités ont été appris sur l'ensemble d'apprentissage, et testés sur l'ensemble de test. Le résultat est donné à la figure 7. On y voit que la meilleure structure comporte 100 noyaux gaussiens, et conduit à une erreur de généralisation d'environ 31.

L'objectif des tests effectués est donc de vérifier si les méthodes de validation décrites ci-dessus conduisent à un résultat similaire, surtout en ce qui concerne la structure de modèle à sélectionner. Bien entendu, ni les données de test ni le fait que nous savons quel est le résultat que nous aimerions obtenir (une structure à 100 unités) ne sont utilisés dans la suite des expériences: seul les 1000 premières données connues sont utilisées (figure 5), et séparées respectivement en ensemble d'apprentissage et de validation, selon la méthode utilisée.

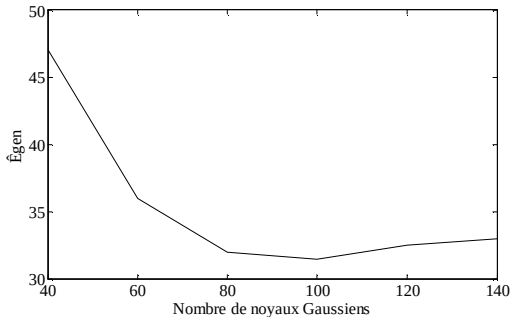



**Figure 7** : Erreur de généralisation sur l'ensemble de test en fonction du nombre de centroïdes.

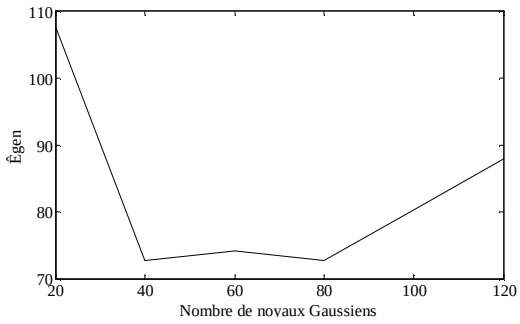



**Figure 8** : Estimation de l'erreur de généralisation obtenue par la méthode de Monte-Carlo cross-validation.

La figure 8 représente le résultat obtenu avec la méthode de Monte-Carlo Cross-Validation pour laquelle la répartition entre les ensembles d'apprentissage et de validation est de deux tiers / un tiers et en utilisant 100 répétitions ($J = 100$). On y voit que la structure idéale

contient 40 ou 80 noyaux gaussiens; selon le principe de parcimonie (destiné à combattre les risques d'overfitting), la structure à 40 unités devrait être choisie, ce qui constitue un écart important par rapport à la valeur attendue (100). De plus, l'estimation de l'erreur de généralisation (environ 72) est excessive.

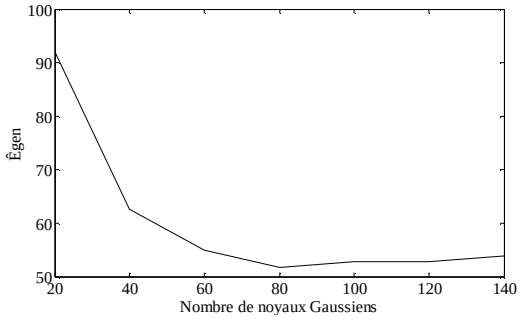



**Figure 9** : Estimation de l'erreur de généralisation obtenue par la méthode du leave-one-out.

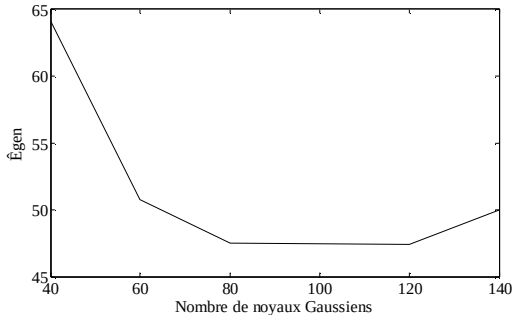



**Figure 10** : Estimation de l'erreur de généralisation obtenue par la méthode du bootstrap.

Les figures 9 et 10 montrent respectivement les estimations de l'erreur de généralisation obtenues par les méthodes de leave-one-out et de bootstrap (avec 100 tirages d'échantillons dans ce dernier cas). Dans le cas du leave-one-out, la structure sélectionnée (80 noyaux) se rapproche de la structure attendue, tandis que celle-ci est atteinte par la méthode du bootstrap. Les estimations des erreurs de généralisation se rapprochent également de la valeur attendue. Notons cependant que la méthode du leave-one-out nécessite un nombre d'itérations égal au nombre de données dans l'ensemble d'apprentissage, c'est-à-dire ici 1000, soit 10 fois plus que le nombre d'itérations utilisées pour la cross-validation et le bootstrap.

Comme décrit dans la section 3.5, l'estimation de l'erreur de généralisation, dans le cas du bootstrap, provient de la somme de deux termes: l'erreur d'apprentissage, et une estimation de l'optimisme. Les figures 11 et 12 montrent ces deux termes; l'allure des courbes des figures 10 et 11 est consistante avec celle de la figure 2.

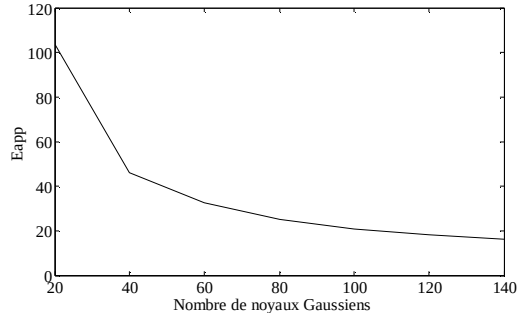



**Figure 11** : Méthode du bootstrap: erreur d'apprentissage

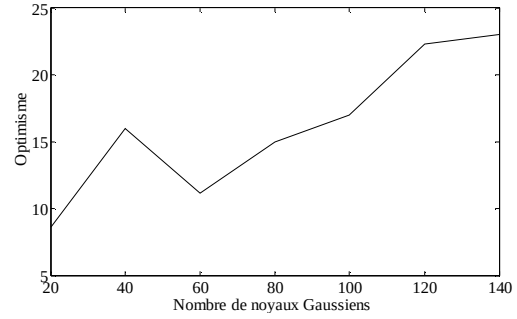



**Figure 12** : Méthode du bootstrap: estimation de l'optimisme.

Sur cet exemple, c'est en fait la méthode du bootstrap 632 qui donne les meilleurs résultats. Ceux-ci sont illustrés à la figure 13, où l'on voit que la structure sélectionnée comporte 100 noyaux gaussiens, et que l'estimation de l'erreur de généralisation est proche du résultat attendu.

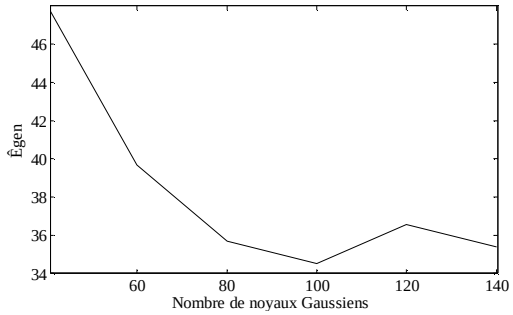



**Figure 13** : Estimation de l'erreur de généralisation obtenue par la méthode du bootstrap 632.

## 5. DISCUSSION ET CONCLUSION

Le problème de la sélection adéquate de la structure d'un modèle non linéaire, par exemple un réseau de neurones artificiels, est abordé dans ce papier sous l'angle des méthodes de validation. Différentes méthodes sont proposées: la validation simple, la Monte-Carlo cross-validation, la *K*-fold cross-validation, le leave-one-out, et le bootstrap. Ces méthodes sont comparées sous l'angle du biais et de la variance obtenues sur l'estimation de l'erreur de généralisation des modèles.

Toutes ces méthodes utilisent les données disponibles en les séparant (mais de manières différentes) en deux ensemble, que l'on appellera ensemble d'apprentissage et ensemble de validation. Dans la plupart des cas, l'ensemble d'apprentissage sera utilisé pour calculer les paramètres optimaux d'un modèle, une fois sa structure fixée, tandis que l'ensemble de validation sera utilisé pour sélectionner une structure parmi une famille. Cette façon de procéder entraîne le fait que toutes les données (apprentissage *et* validation) sont utilisées pour construire le modèle (paramètres *et* structure). L'estimation de l'erreur de généralisation obtenue est donc optimiste, par rapport à une estimation de cette erreur que l'on obtiendrait sur d'autres données tout à fait indépendantes, qui ne sont ni utilisées en apprentissage ni en validation. De façon tout à fait stricte, il faudrait donc se réserver un troisième ensemble de données, appelé ensemble de test, pour estimer correctement l'erreur de généralisation, une fois le modèle construit. C'est ce qui a été fait dans l'exemple de prédiction de séries temporelles illustré dans ce papier. Malheureusement, dans la littérature, les ensembles de validation et de test sont souvent confondus, sans qu'aucune précaution ne soit prise pour faire apparaître une éventuelle correction à apporter à l'estimation de l'erreur.

Au point de vue des méthodes de validation, les résultats illustrés ici, et d'autres effectués sur d'autres bases de données, semblent confirmer la supériorité des méthodes de bootstrap (bootstrap et bootstrap 632) par rapport aux autres. Le léger désavantage que ces dernières ont au point de vue du biais dans l'estimation de l'erreur de généralisation (désavantage qui peut n'avoir aucune conséquence sur la sélection de modèle) semble compensé largement par une variance plus faible; ceci entraîne une qualité d'estimation accrue pour un nombre fixé d'itérations, ou un nombre d'itérations nécessaires plus faible pour une même qualité.

## 6. REMERCIEMENTS

Les auteurs remercient Vincent Wertz pour ses remarques constructives et pour les précisions apportées quant aux notations utilisées. Michel Verleysen est Maître de Recherches du Fonds National de la Recherche Scientifique belge. Le travail d'Amaury Lendasse est financé par le programme belge des Pôles d'Attraction Interuniversitaires, mis en place par les Services fédéraux des affaires Scientifiques, Techniques et Culturelles de l'Etat belge. La responsabilité scientifique appartient à ses auteurs.

## 7. RÉFÉRENCES

Akaike H. (1973) Information theory and an extension of the maximum likelihood principle. In *2nd Int. Symp. on information Theory,* pp. 267-281, Budapest.

Benoudjit N. et Verleysen M. (2003), On the kernel widths in Radial-Basis Function Networks. Accepté pour publication dans Neural Processing Letters, Kluwer aca. pub.

Benoudjit N., Archambeau C., Lendasse A., Lee J. et Verleysen M. (2002) Width optimization of the Gaussian kernels in Radial Basis Function Networks. in *European Symposium on Artificial Neural Networks*. Bruges (Belgium), avril 2002, p. 425-432.

Bishop C. (1995) *Neural Networks for Pattern Recognition*. Cambridge University Press.

Efron B. (1983) Estimating the error rate of a prediction rule; improvement on cross-validation. Journal of the American Statistical Association, 78: p. 316-331.

Efron B. et Tibshirani R. (1997) Improvements on cross-validation: The .632+ bootstrap method. Journal of the American Statistical Association. 92(438): p. 548-560.

Efron B. et Tibshirani J. (1998) *An Introduction to the Bootstrap*. Monographs on Statistics and Appied Probability. CRC Press LLC.

Kohavi R. (1995) A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. in *Proc. of International Joint Conference on Artificial Intelligence*, San Mateo (USA), Morgan Kaufmann, p. 1137-1143.

Lendasse A., Wertz V., et Verleysen M. (2003) Model Selection with Cross-Validations and Bootstraps - Application to Time Series Prediction with RBFN Models, in *Artificial Neural Networks and Neural Information Processing,* LNCS 2714. Springer-Verlag. p. 573-580.

Ljung L. (1987) *System Identification - Theory for User*. Prentice-Hall ed. NJ.

Shao J. et Tu D. (1995) *The Jackknife and Bootstrap, Springer Series in Statistics*. New-York: Springer-Verlag.

Simon G., Lendasse A. et Verleysen M. (2003) Bootstrap for Model Selection: Linear Approximation of the Optimism, in *Computational Methods in Neural Modeling,* J. Mira, J.R. Alvarez eds, LNCS 2686, Springer-Verlag. p. 1182-1189.

Simon G., Lendasse A. et Verleysen M. (2003) Fast Approximation of the Bootstrap for Model Selection. *European Symposium on Artificial Neural Networks*, Bruges (Belgium), 23-25 avril 2003, p. 99-106.

Stone M. (1977) Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaikes Criterion. Journal of the Royal Statistical Society Series B- Methodological, 39(1): p. 44-47.

Verleysen M. et Hlavachova K. (1996) Learning in RBF Networks, in *International Conference on Neural Networks (ICNN)*. Washington, DC, juin 1996, special sessions volume p. 199-204.

Weigend A.S. et Gershenfeld N.A. (1994) *Times Series Prediction: Forcasting the future and Understanding the Past.* Addison-Wesley Publishing Company.

Werbos P. (1974) *Beyond regression: new tools for prediction and analysis in the behavioural sciences*. Harvard University.