J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes,
Volume 2, Hors-Série 1, 5 (2003)
DOI : http://dx.doi.org/10.1051/bib-j3ea:2003505

Apprentissage statistique et techniques de ré-échantillonnage

Jean-Marc Martinez

Centre d'Études de Saclay
Département de Modélisation des Systèmes et Structures
F-91190 Gif-sur-Yvette, France

Jean-Marc.Martinez@cea.fr

Dans cet article, l'auteur, Jean-Marc Martinez, reprend un extrait d'un ouvrage qu'il a précédemment publié en collaboration :

G. Dreyfus, J.-M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria et L. Hérault, chapitre 3 *in* Réseaux de neurones – Méthodologie et applications (Éditions Eyrolles, 2002) p. 168-175.

http://www.editions-eyrolles.com/php.informatique/Ouvrages/9782212110197.php3

(reproduit avec l'autorisation des Éditions Eyrolles)

Apprentissage statistique et techniques de ré-échantillonnage

Jean-Marc Martinez
Centre d'Etudes de Saclay
Département de Modélisation des Systèmes et Structures
91190 Gif sur Yvette
Jean-Marc.Martinez@cea.fr

Ingénieur Chercheur au Département de Modélisation des Systèmes et Structures au Centre d'Etudes de Saclay, chargé d'affaires au CEA/DEN dans le domaine de la supervision, de l'analyse de sensibilité et des traitements des incertitudes en simulation numérique. Notre R&D porte sur les plans d'expériences numériques, les surfaces de réponse non linéaires, l'apprentissage statistique par réseaux de neurones et l'optimisation multicritère par algorithmes génétiques avec Michel.Dumas@cea.fr.

Mots clés : apprentissage statistique, réseaux de neurones, bootstrap.

Résumé

Les méthodes de ré-échantillonnage sont utilisées pour réaliser des estimations lorsqu'on ne connaît pas les lois de probabilité des variables à analyser. Dans les problèmes posés par la régression, notamment la régression par réseaux de neurones, elles permettent d'estimer l'erreur de généralisation, et d'évaluer, avec efficacité et robustesse, la variabilité du réseau par rapport aux données, élément clé du dilemme biais-variance qui conditionne l'élaboration de tout modèle statistique. Ces techniques très performantes sont gourmandes en temps de calcul, mais l'accroissement de la vitesse des calculateurs permet de plus en plus fréquemment leur mise en œuvre. Une nouvelle méthode est présentée, associant le *bootstrap* et le « early stopping » pour automatiser et contrôler l'apprentissage des réseaux de neurones. Elle a été développée dans le logiciel NeMo construit à partir du simulateur SNNS.

Ce texte est extrait du livre « **Réseaux de neurones – Méthodologie et Applications** » publié aux Editions **EYROLLES** en 2002 sous la Direction de Gérard Dreyfus.

Le bootstrap et les réseaux de neurones

Nous présentons une nouvelle approche permettant d'automatiser la construction et l'apprentissage des réseaux de neurones. Elle s'articule autour de la méthode statistique du *bootstrap* et de la technique de l'arrêt prématuré ou « early stopping » (cette dernière technique est présentée dans le chapitre 2). L'orientation prise est donc celle qui consiste à utiliser des réseaux suffisamment complexes puis à les régulariser par arrêt de l'apprentissage. Le *bootstrap* permet d'évaluer avec efficacité la variabilité du réseau et de son erreur par rapport aux données. Associé à l'arrêt prématuré, il permet le contrôle de l'apprentissage en optimisant automatiquement le nombre de cycles nécessaire, tout en fournissant les caractéristiques statistiques de l'erreur de généralisation.

Le *bootstrap*, proposé par [Efron 93] est une technique aujourd'hui très étudiée dans le cadre de l'inférence statistique, notamment pour les tests d'hypothèses et l'estimation des intervalles de confiance. Elle ne nécessite aucune hypothèse a priori sur les lois de distribution. Appliqué à la

régression, le *bootstrap* permet d'estimer les caractéristiques statiques de l'écart entre l'erreur d'apprentissage et l'erreur de généralisation. L'approche est particulièrement adaptée aux problèmes pour lesquels les échantillons d'exemples sont de petite taille. C'est le cas notamment du calcul scientifique et de la simulation de systèmes complexes. À partir d'une base de calculs, des fonctions analytiques sont construites par régression ou interpolation, pour être utilisées en lieu et place de modules plus coûteux en temps de calcul.

Dans le chapitre précédent, nous avons souligné l'importance de la validation des modèles (estimation de l'erreur de modélisation, d'intervalles de confiance, etc.) dans le cadre général de la modélisation notamment non linéaire. Dans le type d'applications mentionnées ci-dessus (remplacement d'un code de calcul complexe par une régression à partir de données engendrées par ce code), la problématique est exactement la même, à ceci près que les données issues de calculs ne sont généralement pas bruitées. Cette partie présente une alternative aux approches développées dans le chapitre précédent.

Principe du bootstrap

Nous allons illustrer le principe du *bootsrap* sur l'exemple du calcul de l'intervalle de confiance de l'espérance μ d'une variable aléatoire. L'exemple tiré de [Wonnacoot 90] a simplement pour but de montrer clairement le principe du *bootstrap*. En effet, pour cet exemple, l'intervalle de confiance de l'espérance d'une variable aléatoire est parfaitement déterminée à partir de la moyenne et de la variance calculée sur l'échantillon (vu au chapitre 2). Ce résultat découle du théorème de la limite centrale, selon lequel la distribution de la moyenne d'un échantillon converge assez rapidement vers une loi normale.

On considère un échantillon de la variable aléatoire composé de n=10 observations : $\mathbf{x} = (16, 12, 14, 6, 43, 7, 0, 54, 25, 13)$. La moyenne de l'échantillon est $\bar{X} = \Sigma_{i=1}^{10} x_i / 10 = 19.0$ et son écart type est $s = \sqrt{\Sigma_{i=1}^{10} (x_i - 19.0) / 9} = 17.09$. L'intervalle de confiance de l'espérance μ à 95% est :

$$\mu = \bar{X} \pm t_{.025} \frac{s}{\sqrt{n}} = 19.0 \pm 2.26 \frac{17.09}{\sqrt{10}} \approx 19 \pm 12 \quad \Rightarrow \quad 7 < \mu < 31$$

L'intervalle de confiance peut être également calculé par bootstrap. Il est alors obtenu par l'algorithme suivant.

À partir de l'échantillon initial, on simule de nouveaux échantillons, appelés « répliques », de taille n, par tirages aléatoires avec remise. Prenons par exemple l'échantillon initial défini précédemment $\mathbf{x} = (16,12,14,6,43,7,0,54,25,13)$. Par tirages aléatoires avec remise, on obtient par exemple la réplique suivante $\mathbf{x}^* = (54, 0, 16, 7, 43, 54, 0, 25, 25, 6)$, dans laquelle certaines valeurs de l'échantillon initial ne figurent pas, et d'autres apparaissent plusieurs fois. Plusieurs échantillons sont ainsi simulés. Pour chaque échantillon simulé une moyenne est calculée. L'intervalle de confiance à 95% est défini sur cet ensemble de moyennes. La simulation donne :

$$9 < \mu < 26$$

On note que l'intervalle obtenu par *bootstrap* est pratiquement identique à l'intervalle de confiance à 95% calculé précédemment et issu du théorème central limite.

Généralité du bootstrap

Le bootstrap ne fait appel à aucune hypothèse sur la distribution statistique sous jacente; d'où sa généralité et sa puissance.

Le *bootstrap* peut donc être appliqué à tout estimateur autre que la moyenne comme par exemple la médiane, le coefficient de corrélation entre deux variables aléatoires ou la valeur propre principale d'une matrice de variance covariance. Pour ces estimateurs, il n'existe pas de formule mathématique définissant l'erreur standard ou l'intervalle de confiance. Les seules méthodes applicables sont les méthodes dites de ré-échantillonnage qui procèdent par simulation d'échantillons comme le *bootstrap* ou le *jackknife* [Efron 93].

Algorithme du Bootstrap pour calculer un écart-type

Soit une variable aléatoire X obéissant à une loi de distribution F. On souhaite estimer un paramètre θ de F. Le paramètre θ est estimé à partir d'un n-échantillon $\mathbf{x} = (x_1, x_2, \cdots, x_n)$. On note $\hat{F}$ la distribution empirique, et $\hat{\theta} = s(\mathbf{x})$ l'estimation de θ réalisée à partir de l'échantillon $\mathbf{x}$. L'algorithme est :

1. Sélectionner B n-échantillons « bootstrapés » $\mathbf{x}^{*1}, \mathbf{x}^{*2}, \cdots, \mathbf{x}^{*B}$, chacun étant obtenu à partir de l'échantillon initial $\mathbf{x}$ par n tirages aléatoires avec remise
2. Calculer pour chaque n-échantillon *bootstrapé*, une réplique de l'estimation de θ par

$$\hat{\theta}(b) = s(\mathbf{x}^{*b}) \quad b = 1,2,\cdots,B$$

3. Estimer l'écart type à partir de l'erreur standard calculée sur l'ensemble des répliques

$$\hat{\theta}^*(.) = \sum_{b=1}^{B} \hat{\theta}^*(b) \Big/ B$$

$$\hat{\sigma}_B{}^2 = \sum_{b=1}^{B} \left(\hat{\theta}^*(b) - \hat{\theta}^*(.)\right)^2 \Big/ B - 1$$

Un des théorèmes démontrés par Efron, porte sur la consistance de l'estimateur *bootstrap*. L'estimation $\hat{\sigma}_B$ converge vers l'écart type $\sigma_{\hat{F}}(\hat{\theta}^*)$ du paramètre θ évalué sur la distribution de l'échantillon :

$$\lim_{B \to \infty} \hat{\sigma}_B = \sigma_{\hat{F}}$$

Cet algorithme peut s'appliquer à tout estimateur. Prenons l'exemple du calcul de la valeur propre principale lors d'une ACP. Elle correspond à la plus grande valeur propre de la matrice de variance covariance $X^T X$ des observations $X_{n \times p}$. Le *bootstrap* consiste à simuler des répliques $X^*_{n \times p}$ obtenues par n tirages aléatoires des lignes de la matrice $X_{n \times p}$. Puis la statistique (moyenne et écart type) pourra être établie sans difficulté. On voit ici la puissance de la méthode et sa facilité de mise en œuvre. On comprend aussi que cette méthode n'ait pas été très utilisée par le passé, en raison du nombre de calculs nécessaires : 50 à 200 répliques suffisent à estimer une moyenne, mais plusieurs milliers de répliques sont nécessaires si l'on souhaite déterminer des intervalles de confiance.

L'erreur de généralisation estimée par *bootstrap*

Nous avons insisté, dans le chapitre précédent, sur la nécessité d'estimer l'erreur de généralisation, et nous avons présenté l'estimation par « *leave-one-out* ». La technique du *bootstrap* permet également une estimation de cette erreur. Le principe est le même : il consiste à simuler B bases « bootstrapées ». Chaque base simulée peut contenir plusieurs fois le même exemple, en raison du tirage avec remise.

Loi binomiale des bases bootstrapées

A chaque tirage, tous les exemples ont la même probabilité p=1/n, en notant n le nombre d'exemples. Le nombre d'apparitions d'un exemple dans une base *bootstrapée* suit donc une loi binomiale $B(n, p = 1/n)$. La probabilité q'un exemple apparaisse k fois est donnée par $P(k) = C_n^k p^k (1-p)^{n-k}$ [Saporta 90]

La probabilité qu'un élément n'apparaisse pas dans la base *bootstrapée* est donc $P(0) = (1-1/n)^n$. Pour n suffisamment grand $P(0)_{n\to\infty} = e^{-1} \approx 0.368$. En moyenne, 37% des exemples ne seront donc pas utilisés en apprentissage et pourront être utilisés pour estimer l'erreur de généralisation.

Statistique de l'erreur de généralisation

L'écart entre l'erreur d'apprentissage calculée sur la base *bootstrapée* et l'erreur de test évaluée sur la base initiale, est considéré comme une variable aléatoire représentative de l'écart entre l'erreur d'apprentissage et l'erreur de généralisation.

Une statistique est faite sur l'ensemble de ces écarts (1 par base *bootstrapée*) afin d'estimer la loi de distribution de l'écart entre l'apprentissage et l'erreur de généralisation.

Soient B la base initiale des exemples et $B_b^*, b = 1, \cdots, N$ l'ensemble des répliques. Désignons par ε_b^* l'erreur d'apprentissage du réseau entraîné sur la réplique *k,* et par ε_b l'erreur du même réseau calculée sur la base initiale B. L'écart $\delta_b = \varepsilon_b - \varepsilon_b^*$ entre les 2 erreurs peut alors être considéré comme une variable aléatoire représentative du phénomène de sur apprentissage. Cet écart peut être considéré comme le biais apparaissant sur l'estimation de l'erreur de généralisation par l'erreur d'apprentissage. L'espérance $\bar{\delta}$ et la variance σ_δ du biais peuvent alors être estimée sur l'ensemble des valeurs δ_b:

$$\delta_b = \varepsilon_b - \varepsilon_{b_k}^* \qquad \bar{\delta} = \frac{1}{B}\sum_{b=1}^{B}\delta_b \qquad \sigma_\delta^2 = \frac{1}{B-1}\sum_{b=1}^{B}\left(\delta_b - \bar{\delta}\right)^2$$

La méthode NeMo

L'algorithme proposé précédemment a été programmé dans le logiciel NeMo. Le *bootstrap* y est associé à l'arrêt prématuré de l'apprentissage (« early stopping ») afin d'automatiser le contrôle de l'apprentissage du réseau.

Outil NeMo

NeMo est un outil développé au Centre d'Etudes de Saclay au Département de Modélisation de Systèmes et Structures à partir du simulateur SNNS (Stuttgart Neural Network Simulator) disponible sur *http://www-ra.informatik.uni-tuebingen.de/SNNS* visant à simplifier les tâches d'apprentissage et de test des réseaux de neurones.

L'utilisateur fixe a priori le nombre de cycles d'apprentissage N_c et le nombre B de répliques. NeMo effectue un nombre B d'apprentissages en sauvegardant à chaque cycle l'erreur quadratique moyenne d'apprentissage et de test. NeMo analyse les profils respectifs des erreurs d'apprentissage et de test pour choisir la valeur du nombre de cycles la plus appropriée.

Modélisation des erreurs

L'erreur quadratique moyenne EQMr est calculée sur les variables de sortie (estimées et désirées) centrées et réduites. L'analyse de l'erreur porte donc sur la part de la variance *non expliquée* par le modèle ou coefficient d'indétermination introduit au chapitre sur les pré-traitements des sorties.

Avant de donner le détail de la méthode, désignons par *j* le rang de la réplique et par *i* l'itération sur le nombre de cycles ; les erreurs quadratiques moyennes d'apprentissage et de test sont représentées par les 2 tableaux suivants :

$$\underbrace{\begin{bmatrix} \varepsilon_1^{*1} & \varepsilon_1^{*2} & \cdots & \varepsilon_1^{*B} \\ \varepsilon_2^{*1} & \varepsilon_2^{*2} & \cdots & \varepsilon_2^{*B} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_{\mathrm{Nc}}^{*1} & \varepsilon_{Nc}^{*2} & \cdots & \varepsilon_{Nc}^{*B} \end{bmatrix}}_{\text{erreur d'apprentissage}} \quad \underbrace{\begin{bmatrix} \varepsilon_1^{1} & \varepsilon_1^{2} & \cdots & \varepsilon_1^{B} \\ \varepsilon_2^{1} & \varepsilon_2^{2} & \cdots & \varepsilon_2^{B} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_{\mathrm{Nc}}^{1} & \varepsilon_{Nc}^{2} & \cdots & \varepsilon_{Nc}^{B} \end{bmatrix}}_{\text{erreur de test}}$$

Après cette phase, NeMo détermine le nombre de cycles selon une heuristique rappelant la théorie des jeux. Un premier joueur « pessimiste » se place, pour chaque valeur du nombre de cycles, dans la pire des situations sur l'erreur de test :

$$\varepsilon_i^{\mathrm{Max}} = \mathrm{Max}_b \left\{ \varepsilon_i^{\,b} \right\}$$

Le second joueur détermine alors le nombre de cycles de façon à minimiser la pire des situations obtenues, c'est à dire celle qui correspond à l'erreur de test maximale :

$$N_{\mathrm{c}}^{\mathrm{optimal}} = \mathrm{Arg}_i \left\{ \mathrm{Min}\ \varepsilon_i^{\mathrm{Max}} \right\}$$

Cette stratégie sur le choix de $N_{\mathrm{c}}^{\mathrm{optimal}}$ peut être assouplie en ne retenant qu'une fraction de l'ensemble des B apprentissages. Pour la rendre plus robuste il suffit en effet d'exclure les cas extrêmes (outliers), c'est à dire les situations d'apprentissage très différentes de la moyenne. Par défaut NeMo détermine le nombre de cycles optimal sur le $90^{\text{ème}}$ percentile de l'erreur de test.

Percentile

Le $\alpha^{\text{ème}}$ percentile correspond à l'intervalle constitué des valeurs pour lesquelles la fonction de répartition est inférieure à α : une fraction $(1-\alpha)$ des valeurs maximales est exclue.

L'estimation du nombre optimal de cycles peut également être faite par la méthode du tri médian plus *stable* mais plus *risquée* car rejetant a priori 25% des cas : le dernier quartile correspondant aux erreurs de test les plus importantes.

Quartile

En notant F la fonction de répartition, les 1^{er} et $3^{\text{ème}}$ quartile Q_1 et Q_3 et la médiane Q_2 sont respectivement définis par $F(Q_1)=0.25$, $F(Q_2)=0.5$, $F(Q_3)=0.75$.

Tri-médian

Le tri médian correspond à 0.25 Q_1 (1^{er} quartile)+ 0.5 Q_2 ($2^{\text{ème}}$ quartile ou médiane) + 0.25 Q_3 (3ème quartile)

Après avoir déterminé le nombre de cycles optimal selon une des stratégies, NeMo lance un nouvel apprentissage fondé sur la totalité des exemples, avec, pour nombre de cycles, le nombre de cycles optimisé N_c^{optimal} défini à l'étape précédente. Pour ce dernier apprentissage, les mêmes paramètres d'apprentissage sont utilisés : la valeur initiale et la loi de décroissance du pas d'adaptation. En notant ε_a l'erreur moyenne calculée sur la base initiale, et $\bar{\delta}$ la valeur moyenne du biais, l'erreur de généralisation est estimée par :

$$\varepsilon_g = \varepsilon_a + \bar{\delta}$$

De façon plus générale, la fonction de répartition de l'erreur de généralisation est estimée par la fonction empirique de répartition du biais translatée de la valeur ε_a. On remarque l'apport du *bootstrap* associé au *early stopping* par rapport à la validation croisée :

- une certaine automatisation dans la construction du réseau en adaptant le nombre de cycle du *early stopping*,
- une plus grande estimation de la variabilté du modèle par rapport au jeu de données,
- estime des intervalles de confiance (marges, incertitudes),
- l'utilisation de l'ensemble des exemples pour construire le réseau.

Notons enfin que NeMo peut contrôler l'adéquation du modèle aux données : si le nombre de cycle optimisé est trop proche du nombre de cycle maximal fixé par l'utilisateur, l'erreur de test ne passe pas par un minimum ; l'utilisateur devra alors accroître la complexité du réseau (nobre de neurones cachés) ou augmenter le nombre de cycles d'apprentissage.

Test de la méthode NeMo

Dans ce qui suit, nous montrons les résultats d'une expérience visant à valider la méthode. Le test consiste à comparer l'erreur moyenne estimée par NeMo à l'erreur réelle. L'erreur réelle est approchée selon le principe de la méthode de méthode de Monte Carlo, c'est à dire en effectuant un très grand nombre de calculs de l'erreur quadratique moyenne puis en effectuant sa moyenne. Nous avons mis en œuvre ***NeMo*** sur l'approximation de 2 fonctions analytiques non linéaires :

- $\phi_8(x)$ fonction de $R^8 \rightarrow R$
- $\phi_{12}(x)$ fonction de $R^{12} \rightarrow R$

Nous avons choisi ces superviseurs de façon à évaluer la méthode sur des problèmes d'approximations de fonctions suffisamment complexes (grande dimension de l'espace d'entrée). À l'aide de ces 2 superviseurs, nous avons créé plusieurs bases d'exemples en faisant varier le nombre d'exemples de 100 à 1500 par pas de 100. La loi de distribution retenue pour les entrées a été la loi uniforme sur l'intervalle [-1,1].

Les réseaux modèles retenus sont des réseaux non bouclés à 1 couche cachée. Les unités d'entrée et de sortie sont associées à la fonction d'activation identité et les unités cachées à la fonction d'activation logistique. Pour les bases créées par le premier superviseur ϕ_8, 5 réseaux modèles ont été proposés à NeMo comprenant respectivement 4, 6, 8, 10 et 12 unités cachés. Pour les bases engendrées par le second superviseur ϕ_{12} (espace d'entrée plus complexe), 6 réseaux ont été testés comprenant respectivement 10, 14, 18, 22, 26 et 30 unités en couche cachée.

Grande dimension

A noter la très faible densité des points dans R^{12} ; 1500,points dans R^{12} correspondent à un nombre moyen inférieure 2 par axe : $d^{12} = 1500 \mapsto d \approx 1{,}8$

L'erreur réelle est obtenue à partir de 10^6 tirages aléatoires en utilisant la même loi de génération des entrées (loi uniforme) et en calculant l'erreur moyenne quadratique réduite EQMr entre la sortie désirée et la sortie estimée.

Les figures ci-dessous présentent la comparaison (en échelle log-log) de l'erreur EQM_r « vraie » (en abscisse) à l'erreur estimée (en ordonnée) par NeMo. Les points visualisés correspondent aux

différents réseaux élèves construits sur l'ensemble des bases d'exemples. Chaque réseau a été entraîné 15 fois sur des bases d'exemples comprenant respectivement 100, 200, …, 1500 exemples.

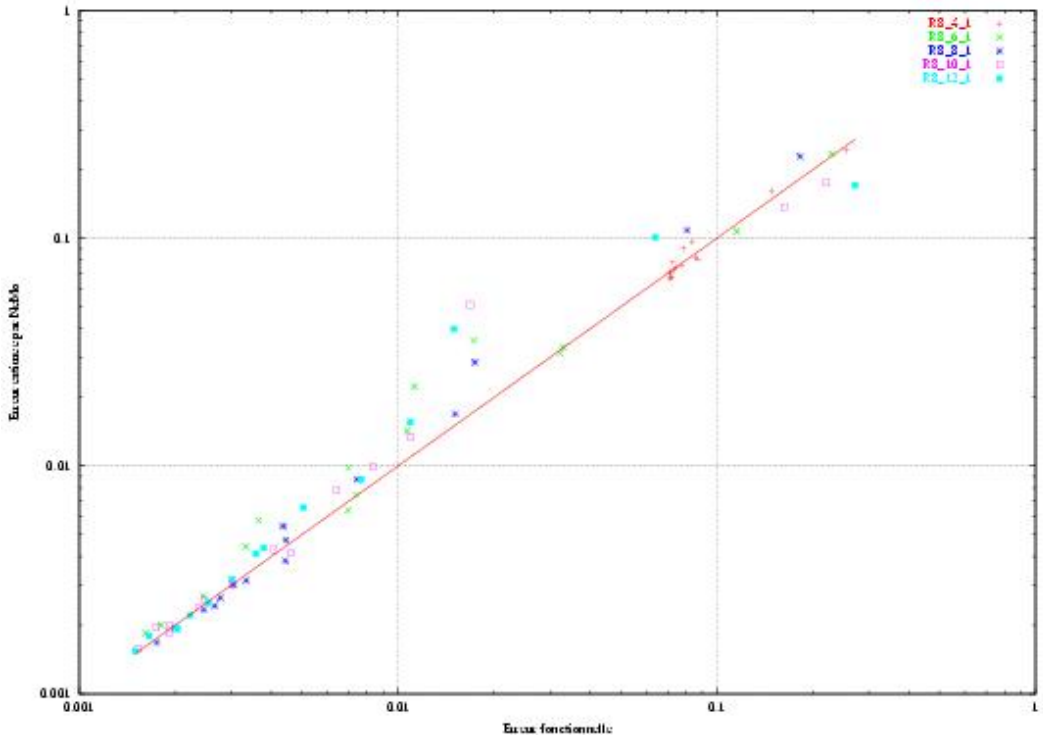

Figure 3.11. Générateur ϕ_8

L'analyse de l'ensemble des résultas illustrés par les figures 3.11 et 3.12 fait apparaître les propriétés essentielles de la méthode NeMo :

- l'erreur de généralisation est estimée avec précision, même dans les cas complexes (grand nombre d'entrées + faible nombre d'exemples)
- le bootstrap permet d'automatiser la régularisation du réseau aux données par contrôle de l'arrêt de l'apprentissage.

Les figure 3.11 et 3.12 font en effet apparaître des estimations de l'erreur de généralisation très proches des valeurs exactes. Les faibles valeurs de l'erreur correspondent aux apprentissages réalisés avec les bases d'exemples comportant suffisamment d'exemples. Pour ces cas, l'erreur estimée en ordonnée est quasiment égale à l'erreur vraie en abscisse.

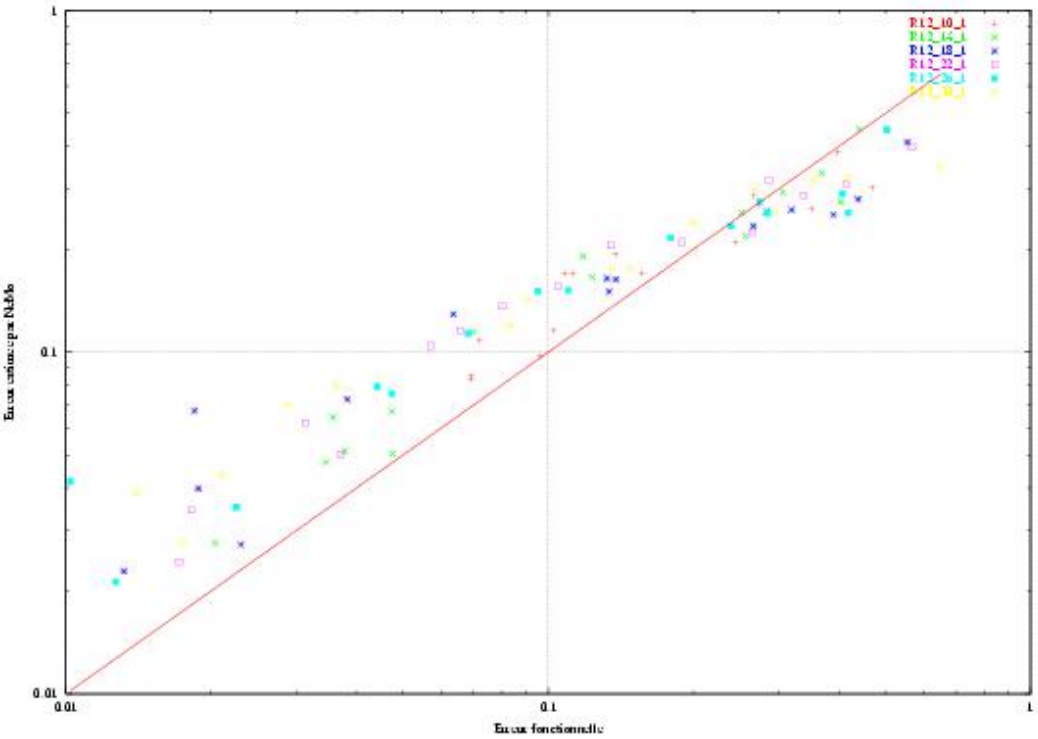

Figure 3.12 Générateur ϕ_{12}

Il faut noter une légère surestimation sur 4 cas parmi 75 entre les valeurs 0.01 et 0.02 pour le cas ϕ_8 (figure 3.11) et une moindre précision sur le cas plus complexe ϕ_{12} (figure 3.12). Pour ce dernier cas, la régression porte sur une relation de R^{12} à R avec un maximum de 1500 points pour représenter la relation. Il apparaît une surestimation de l'erreur pour les faibles valeurs et une sous-estimation pour les valeurs supérieures à 0.2. Néanmoins, malgré la grande dimension de l'espace d'entrées, la relation de R^{12} dans R est correctement modélisée à partir de quelques centaines d'exemples.

Conclusions

Plusieurs points peuvent être tirés de cette étude.

- Les réseaux construits automatiquement sont suffisamment bien régularisés même dans les cas les plus difficiles lorsque le nombre d'exemples est faible. La statistique apportée par le *bootstrap* permet le contrôle automatique de l'arrêt prématuré de l'apprentissage et fournit une statistique robuste de l'erreur de généralisation.
- Le deuxième point est lié au problème de la dimension de l'espace d'entrée. Même dans l'exemple de la relation de R^{12} dans R, quelques centaines de points suffisent à la représentation de la relation. Dans de nombreux problèmes, des relations non linéaires peuvent ainsi être facilement approchées à partir d'une densité d'exemples faible.. A noter qu'à partir d'un certain niveau de complexité, les réseaux construits et régularisés sur un même échantillon semblent équivalents. Des réseaux différents peuvent être adaptés pour représenter la même relation.

Dans le cadre de la théorie de l'apprentissage statistique, la régularisation des modèles peut être contrôlée et donc optimisée par bootstrap. Cette voie est à approcher des méthodes plus formelles

fondées sur la théorie proposée par [Vapnik 95]. L'enjeu étant l'adaptation des capacités de représentation (dimension VC) du modèle aux données. Dans ce cadre, les méthodes statistiques de ré-échantillonnage apportent de réelles solutions par leur facilité de mise en œuvre et surtout, reconnaissons le, par les puissances de calculs aujourd'hui disponibles sur nos bureaux.

Les méthodes de ré-échantillonnage comme le bootstrap permettent également d'estimer les effets de leviers caractérisant l'importance des exemples dans la construction du modèle. Cette technique est particulièrement efficace dans le cadre des plans d'expériences numériques. Elle est actuellement développée dans le logiciel NeMo.

Bilbliographie

[Dreyfus 02], Gérard Dreyfus, Jean-Marc Martinez, Manuel Samuelides, Mirta B. Gordon, Fouad Badran, Sylvie Thiria, Laurent Hérault, Réseaux de neurones – Méthodologie et Applications, Eyrolles 2002

[Efron 93], Bradley Efron, Robert J. Tibshirani, An introduction to the Bootstrap, Chapman & Hall, 1993

[Saporta 90], Gilbert Saporta, Probabilités Analyse des données et Statistique, Editions Technip, 1990

[Thiria 97] Sylvie Thiria, Yves Lechevallier, Olivier Gascuel, Stéphane Canu, Statistique et méthodes neuronales, Dunod 1997

[Vapnik 95], Vladimr N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995

[Wonnacoot 90] Thomas H. Wonnacoot, Ronald J. Wonnacott, Statistique Economie-Gestion-Sciences-Médecine, Economica, 4ème édition, 1990