Bootstrap for neural model selection

Ryadh Kallel, Marie Cottrell, Vincent Vigneron

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

Bootstrap for neural model selection

Riadh Kallel[1], Marie Cottrell[1], Vincent Vigneron[1,2]

[1] MATISSE-SAMOS UMR 8595
90, rue de Tolbiac
75634 Paris cedex 13
kallel,cottrell@univ-paris1.fr

[2] CEMIF
40 rue du Pelvoux
91020 Evry Courcouronnes
vvigne@iup.univ-evry.fr

Abstract

Bootstrap techniques (also called *resampling computations techniques*) have introduced new advances in modeling and model evaluation [10]. Using resampling methods, the information contained in one observed data set is extended to many typical generated data sets. These procedures based on computer simulation and cross validation are the last resort when no classical inference is possible due to the intrinsic complexity of the problem: they can avoid to estimate the noise distribution from the residuals, like in Monte-Carlo approach which is based on a hypothesized noise distribution.

Resampling allows the modeler to construct a series of new samples which are based on the original data set, and then to estimate the stability of the parameters. Properties such as convergence and asymptotic normality can be checked for any particular observed data set. In most cases, the statistics computed on the generated data sets give a good idea of the confidence regions of the estimates. In this paper, we debate on the contribution of such methods for model selection, in the case of feedforward neural networks. The method is described and its effectiveness is checked through a number of examples.

1 Multilayer Perceptrons (MLP)

Suppose a set of n independent observations of a continuous variable y that we have to explain from a set of p explanatory variables $(x_1, x_2, \ldots, x_p)$. We want to use the non linear models called *Multilayer Perceptrons*. These models are nowadays commonly used for non linear regression, forecasting, pattern recognition, and are particular examples of artificial neural networks.

We consider in the following a multilayer perceptron (MLP) with p inputs, one hidden layer with H hidden units and one output layer.

In such a network, units are organized in successive layers with connexions connecting one layer to the following one. See Cheng et Titterington [2] or Hertz *et al.* [8] for details or references.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

The model can be analytically expressed in the following form : the output y is given by

$$y = \phi_0 \left(w_0 + \sum_{h=1}^{H} w_h \phi(b_h + \sum_{j=1}^{p} w_{jh} x_j) \right) + \epsilon \quad (1)$$

where ϵ is the residual term, with zero mean, variance σ^2 (with normal distribution or not),

y is a continuous variable,

ϕ_0 is the identity output function

ϕ is (in most cases) the sigmoid

$$\phi(x) = \frac{1}{1 + \exp(-x)}.$$

Let $\boldsymbol{\theta} = (w_0, w_1, \ldots, w_H, w_{11}, \ldots, w_{pH})$ be the parameter vector of the network and let $y(\boldsymbol{x}; \boldsymbol{\theta})$ the computed value for an input $\boldsymbol{x} = (x_1, \ldots, x_p)$ and a parameter vector $\boldsymbol{\theta}$. There are $H(p+1) + H + 1$ parameters to be estimated.

Classically, if there are numerous data, the first step consists in the division of the supplied data into two sets : a *test set* and a *training set*. The so-called training set $\{(\boldsymbol{x}_1; y_1), \ldots, (\boldsymbol{x}_m; y_m); (1 \leq i \leq m; m < n)\}$, is used to estimate the weights of the model by minimizing an error function $\frac{1}{m} \sum_{i=1}^{m} (y_i - y(\boldsymbol{x}_i; \boldsymbol{\theta}))^2$ using optimization techniques such as gradient descent, conjugate gradient or quasi-Newton methods...

The resulting least squares estimator of $\boldsymbol{\theta}$ is denoted by $\hat{\boldsymbol{\theta}}$, and the resulting lack of fit for training is the *learning error*

$$MSE_a = \frac{1}{m} \sum_{i=1}^{m} \left(y_i - y(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}) \right)^2. \quad (2)$$

The training set is used to derive the coefficients (weights) of the model and the resulting model is tested on the test set. A good regression method would generalize well on examples that have not been seen before, by learning the underlying function without the associated noise. The *test error* can be defined by

$$MSE_t = \frac{1}{n-m} \sum_{i=m+1}^{n} \left(y_i - y(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}) \right)^2. \quad (3)$$

Most optimization techniques (that are variants of gradient methods) provide local minima of the error function and not a global one. Practically, different learning conditions (initialization of weights, learning adaptation parameter, sequential order in the sample presentation, ...) give different solutions that it is difficult to compare. It is not easy to know if a minimum is reached, because the decrease of the error function is slow, an over-learning phenomenon can occur, etc...For these reasons, numerous stopping and validation techniques are proposed, see for example Borowiak [1], or Hertz *et al* [8].

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

For multilayer perceptrons, the choice of a model is equivalent to the choice of the *architecture* of the network. If one has to select a model among a lot of them, an exhaustive method would consist in exploring the whole set of possible models, and in testing all these models on the given problem. The estimation of the performances is then a very crucial point, all the more so since many factors intervene to complicate this evaluation. It is necessary to be certain that the convergence has occurred, to have at disposal a good quality criterion which allows to decide what is the *best model*.

2 Bootstrap for parameter estimation

Bootstrap techniques were introduced by Efron [5] and are simulation techniques based on the empirical distribution of the observed sample. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ a n-sample, with an unknown distribution function $\mathcal{F}$, depending on an unknown real parameter $\boldsymbol{\theta}$. The problem consists in estimating this parameter $\boldsymbol{\theta}$ by a statistic $\hat{\boldsymbol{\theta}} = s(\boldsymbol{x})$ from the sample $\boldsymbol{x}$ and in evaluating the estimate accuracy, although the distribution $\mathcal{F}$ is unknown.

In order to evaluate this accuracy, B samples are built from the initial sample $\boldsymbol{x}$, by re-sampling. These samples are called *bootstrapped samples* and denoted by x^{*b}.

A *bootstrapped sample* $\boldsymbol{x}^{*b} = (x_1^{*b}, \ldots, x_n^{*b})$ is built by a random drawing (with repetitions) in the initial sample $\boldsymbol{x}$: $P_U(x_i^{*b} = x_j) = \frac{1}{n};\ \ i, j = (1, \ldots, n)$, where P_U is the uniform distribution on the original data set $\boldsymbol{x} = (x_1, \ldots, x_n)$.

The distribution function of a bootstrapped sample $\boldsymbol{x}^{*b}$ is $\hat{\mathcal{F}}$, i.e. the empirical distribution of $\boldsymbol{x}$. A bootstrap replicate of the estimator $\hat{\boldsymbol{\theta}} = s(\boldsymbol{x})$

$\hat{\boldsymbol{\theta}}^{*b} = s(\boldsymbol{x}^{*b})$. For example, for the mean of the sample $\boldsymbol{x}$, the estimator is $s(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i$, and a bootstrap replicate will be $s(\boldsymbol{x}^{*b}) = \frac{1}{n}\sum_{i=1}^{n} x_i^{*b}$.

Then, the bootstrap estimate of the standard deviation of $\hat{\boldsymbol{\theta}}$ denoted by $\hat{\sigma}_{boot}(\hat{\boldsymbol{\theta}})$ is given by

$$\hat{\sigma}_{boot}(\hat{\boldsymbol{\theta}^*}) = \left[\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\boldsymbol{\theta}}^{*b} - \hat{\boldsymbol{\theta}}^{*}(.)\right)^2\right]^{\frac{1}{2}}$$

and

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

It is computed by replacing the unknown distribution function $\mathcal{F}$ with the empirical distribution $\hat{\mathcal{F}}$. In conjonction with these re-sampling procedures, hypothesis tests and confidence regions for statistics of interest can be constructed.

In the following, the method we propose as a tool to select a MLP model is similar to the bootstrap method, since it relies on re-sampling techniques, but it is non parametric.

3 Bootstrap applied to selection model for MLPs

Let $\mathcal{B}_0$ be a data set of size n,

$$\mathcal{B}_0 = \{(\boldsymbol{x}_1; y_1), \ldots, (\boldsymbol{x}_n; y_n); (1 \leq i \leq n)\}$$

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

where $\boldsymbol{x}_i$ is the i-th value of a p-vector of explanatory variables and y_i is the response to $\boldsymbol{x}_i$.

From the original data set $\mathcal{B}_0$ (called *initial base*), one generates B bootstrapped bases $\mathcal{B}_b^*, 1 \leq b \leq B$, (*i.e.* B uniform drawings of n data points in $\mathcal{B}_0$ with repetitions). For any generated data set $\mathcal{B}_b^*$, an estimator of the MLP parameter vector $\boldsymbol{\theta}$, denoted by $\hat{\boldsymbol{\theta}}^{*b}$, is found by application of the backpropagation algorithm [9] for example, but any minimization algorithm can be used. So the bootstrap procedure provides B replications $\hat{\boldsymbol{\theta}}^{*b}$ for model (1).

Then we use $\mathcal{B}_0$ as a test base, and evaluate for each $b = 1, \ldots, B$ and each $i = 1, \ldots, n$ the residual estimate

$$\epsilon_{test,i}^{*b} = y_i - y(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}^{*b}).$$

The study of the histogramms of these estimated residuals allows to evaluate the distribution of the error term ϵ, to control its *whiteness*, etc. For each bootstrapped sample $\mathcal{B}_b^*$, $b = 1, \ldots, B$, (that is for each $\hat{\boldsymbol{\theta}}^{*b}$), the sum of squares of the residuals on the test base $\mathcal{B}_0$ is computed:

$$TSSE(b) = \sum_{i=1}^{n} \left(\epsilon_{test,i}^{*b}\right)^2$$

as well as the mean of the squares of the residuals on the test base $\mathcal{B}_0$,

$$TMSE(b) = \frac{1}{n} \sum_{i=1}^{n} \left(\epsilon_{test,i}^{*b}\right)^2 .$$

So, we got a vector $TMSE$ whose mean value is $\mu_{boot} = \frac{1}{B} \sum_{b=1}^{B} TMSE(b)$ and standard deviation is $\sigma_{boot} = \frac{1}{B-1} \left[\sum_{b=1}^{B} \left(TMSE(b) - \mu_{boot}\right)^2\right]^{1/2}$.

These two values measure the residual variance of the model, estimated from the bootstrapped samples, and the stability of the parameter vector estimations. So this technique allows to evaluate a model from only one sample (without splitting it into a learning base and a test base, which decreases the number of data used for the estimation).

To choose between several architectures $M_1, M_2, \ldots$, these computations are repeated for each of them, and the best one will be this one that has the best compromise (the ideal is to simultaneously minimize μ_{boot} and σ_{boot}).

The approach is summarized in table 1.

Two main disadvantages must be outlined

- the *computer simulation time*: if n or p is high, computation time can be very long even with second-order optimization techniques as BFGS, but it still remains less than empirical exploration.
- the *repetition of extremal data*: the risk exists to select a re-sampling data set for which iterative methods will converge with difficulty. But ignoring these repetitions could introduce a bias.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

1. To generate B samples of size n by random drawings with repetitions in the initial base $\{\mathcal{B}_0\} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. Let us denote by $\{\mathcal{B}_b^*\} = \{(\boldsymbol{x}_1^{*b}, y_1^{*b}), \ldots, (\boldsymbol{x}_n^{*b}, y_n^{*b})\}$ the $b-$th bootstrapped sample, $b = 1, \ldots, B$.

2. For each bootstrapped sample, $b = 1, \ldots, B$, to estimate $\boldsymbol{\theta}$ by minimizing $\sum_{i=1}^{n} [y_i^{*b} - y(\boldsymbol{x}_i^{*b}; \boldsymbol{\theta})]^2$, we get $\hat{\boldsymbol{\theta}}^{*b}$.

3. The bootstrap standard deviation is given by:

$$\sigma_{boot} = \frac{1}{B-1} \left[\sum_{b=1}^{B} \left(TMSE(b) - \mu_{boot} \right)^2 \right]^{1/2},$$

where

$$\mu_{boot} = \frac{1}{B} \sum_{b=1}^{B} TMSE(b).$$

Table 1: Re-sampling algorithm (bootstrap procedure) used to compute μ_{boot} and σ_{boot} (typically $20 \leq B \leq 200$).

Many other re-sampling procedures have been proposed in the statistical literature: cross-validation, Jackkniffe,etc ... See Hamamoto [7] and Borowiak [1] for details.

4 Examples

4.1 Example 1: Linear model

We wish to illustrate the method on a simple linear case. Consider the problem of fitting a linear model

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_p x_p + \epsilon.$$

We simulate a data set $\mathcal{B}_0 = (x_1^{(i)}, x_2^{(i)}, y_i), i = 1, \ldots, 500$ by putting

$$x_1^{(i)} = i,\ x_2^{(i)} = i^{\frac{1}{2}},\ y_i = 2 + 0.7 x_1^{(i)} + 0.5 x_2^{(i)} + \epsilon_i$$

where ϵ_i is a random variable which possesses the distribution $\mathcal{N}(0, 4)$.

$B = 50$ bootstrapped samples are built, and three models with different architectures are compared.

Model M_1: $p = 2, y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \epsilon$
Model M_2: $p = 1, y = \theta_0 + \theta_1 x_1 + \epsilon$

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

Model M_3: $p = 3$, $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \epsilon$, with $x_3^{(i)} = i^{\frac{3}{2}}$ and $\theta_3 = 1$

For each model, we compute $\mu_{boot}(M_i)$ and $\sigma_{boot}(M_i)$.

In this case (see Tab.2 and Fig.1 on the left), it is evident that the best model is the model M_1, that is the true model.

4.2 Example 2: Non-linear modeling with simulated data

We simulate a data set $\mathcal{B}_0 = (x_1^{(i)}, x_2^{(i)}, y_i), i = 1, \ldots, 500$, by computing y_i as a noisy output of a multilayer perceptron, defined by

$p = 2$ input variables,

one hidden layer and 4 neurones on the hidden layer,

$\boldsymbol{\theta} = (0.5, -0.1, 0.2, 0.5, -0.4, 0.2, 0.1, 3, 0.3, 2, 0.5, 0.1, 0.2, 2, 0.2, 3, 0.1)$,

ϵ possesses a distribution $\mathcal{N}(0, 0.04)$.

$B = 50$ bootstrapped samples are built, and three models with different architectures are compared.

Model M_2: two inputs, one hidden layer with 2 hidden neurons
Model M_4: two inputs, one hidden layer with 4 hidden neurons
Model M_3: two inputs, one hidden layer with 6 hidden neurons

For each model, we compute $\mu_{boot}(M_i)$ and $\sigma_{boot}(M_i)$.

In this case (see Tab.2 and Fig.1 at the right), it is evident that the best model is the model M_2. It is not the true model, but it is the best. It is not so surprising since the Multilayer Perceptrons are always over-parametrized, and that there is no unicity of the multilayer perceptron function which can model a given function.

4.3 Example 3: Non linear model with real data

In this section, we study a real data set to set the efficiency of the model selection method that we propose.

The power peak control in the core of nuclear reactors is explored. The problem has already been studied in the past, namely by Gaudier [6], who constructed a neuronal model with 22 input variables, 2 hidden layers, (the first one with 26 neurons, the other with 40 neurons). The model accounts for physical localization of uranium bars and diffusion processes, and was set to reproduce the classical calculus code, while winning in terms of computing time.

$B = 50$ bootstrapped samples are built, and three models with different architectures are compared.

Model M_{40}: 22 inputs, two hidden layers with respectively 26 and 40 hidden neurons

Model M_{35}: 22 inputs, two hidden layers with respectively 26 and 35 hidden neurons

Model	μ_{boot}	σ_{boot}
M_1	3.9525	0.0155
M_2	3.9020	0.5985
M_3	3.9475	0.4259

Model	μ_{boot}	σ_{boot}
M_2	0,04277	0.00019
M_4	0.04271	0.00029
M_6	0.04277	0.00028

Model	μ_{boot}	σ_{boot}
M_{30}	0,0473	0.0052
M_{35}	0.0599	0.0069
M_{40}	0.0492	0.0049

Table 2: μ_{boot} and σ_{boot} for the three models in each example.

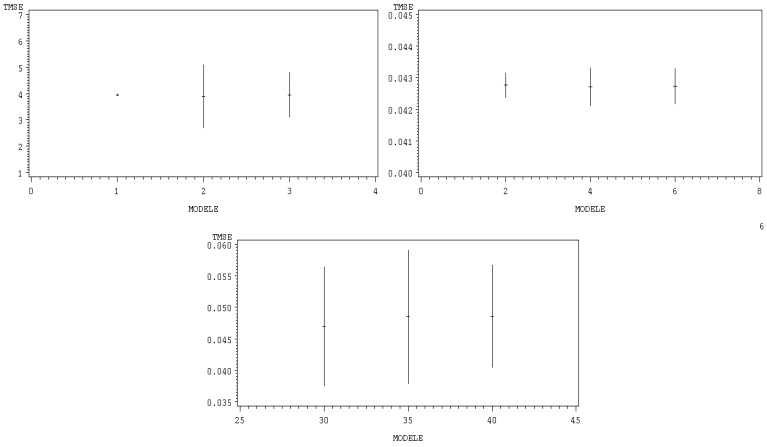

Figure 1: Boxplots for μ_{boot} for the three models in each example.

Model M_{30}: 22 inputs, two hidden layers with respectively 26 and 30 hidden neurons

For each model, we compute $\mu_{boot}(M_i)$ and $\sigma_{boot}(M_i)$.

In this case (see lower table Tab.2 and lower figure Fig.1), the conclusion is not evident, perhaps the model M_{30} seems to be the best, (its residual variance is the smallest), but the more stable is the model M_{40}. In that case, it is necessary to study other architectures, more different from the 3 that we have considered.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 61-68

5 Provisional conclusion

These examples indicate that our techniques can be used for a great variety of situations. But, even if the first results are promising, we have now to apply them to many other cases, and to try to prove theoretical results in order to assess our method.

References

[1] D.S. Borowiak (1990) *Model discrimination for nonlinear regression models*, Marcel Dekker, New York.

[2] B. Cheng, D.M. Titterington (1994) - Neural networks : a review fom a statistical perspective. *statistical science*, **9**,n° 1, p 2–54.

[3] McCullagh, P. et Nelder, J. A. (1988) Generalized Linear Models, Chapman & Hall, seconde édition, Monographs on Statistics and Applied Probability 37.

[4] Efron, B and Tibshirani R. (193) An introduction to the bootstrap, Chapman and Hall.

[5] B. Efron (1979) The convex hull of a random set of points. *Biometrika*, **52**, p 331–342.

[6] F. Gaudier (1998) Optimisation et réseaux de neurones pour le repositionnement des barres de combustible nucléaire, Thèse de doctorat de l'université Paris VI, ENS Cachan.

[7] Y. Hamamoto, S. Uchimura et S. Tomita (1997) - A bootstrap technique for Nearest neighbor classifier design. *IEEE Transactions on PAMI*, **19**,n° 1, p 73–79.

[8] J. Hertz, A. Krogh et R. Palmer (1991) *Introduction to the theory of neural computation*, Addison-Wesley, Redwood City, CA.

[9] D.E. Rumelhart, G.E. Hinton, R.J. Wiliams (1986) Learning internal representations by error propagation, *Parallel distributed processing*, 1**8**, Cambridge, MIT Press.

[10] A. Zapranis, A.-P. Refenes (1999) *Principles of Neural Model Identification, Selection and Adequacy*, Springer, London.