Model Selection with Cross-Validations and Bootstraps – Application to Time Series Prediction with RBFN Models

Amaury Lendasse[1], Vincent Wertz[1], and Michel Verleysen[2]

Université catholique de Louvain
[1] CESAME, av. G. Lemaître 3, B-1348 Louvain-la-Neuve, Belgium
{lendasse, wertz}@auto.ucl.ac.be,
[2] DICE, pl. du Levant 3, B-1348 Louvain-la-Neuve, Belgium
verleysen@dice.ucl.ac.be

Abstract. This paper compares several model selection methods, based on experimental estimates of their generalization errors. Experiments in the context of nonlinear time series prediction by Radial-Basis Function Networks show the superiority of the bootstrap methodology over classical cross-validations and leave-one-out.

1 Introduction

Nonlinear modeling has raised a considerable research effort since decades, including in the field of artificial neural networks. Nonlinear modeling includes the necessity to *compare* models (for example of different complexities) in order to select the "best" model among several ones. For this purpose, it is necessary to obtain a good approximation of the generalization error (or "expected loss") of each model (the generalization error being the average error that the model would make on an infinite-size and unknown test set). In this paper, the terms "model selection" will be used when several models must be compared based on estimations of their generalization errors, in order to select one of them.

Nowadays there exist some well-known and widely used methods able to fulfill this task. Among them we can cite:

- the hold-out (HO) which consist in removing data from the learning set and keeping them for validation; HO is also called "validation" [1], or "external validation" for example in chemometrics, etc.
- the Monte-Carlo cross-validation (or simply "cross-validation", CV), where several HO validation sets are randomly and sequentially drawn;
- the *k*-fold cross-validation, where the initial set is randomly split into *k* roughly equal parts, each one being used successively as a validation set;
- Leave-One-Out (LOO) is a *k*-fold cross-validation where the size of the validation set is 1;
- the bootstrap [2, 3], which consists in drawing sets *with replacement* from the original sample and using these sets to estimate the generalization errors (boostrap 632 and 632+ are improved versions of the original bootstrap):

All these methods of estimating generalization errors have been shown to be asymptotically roughly equivalent (see for example [4]), with some exceptions and limitations:

- LOO is less biased [5, 2] but its variance is unacceptable [6];
- cross-validation is consistent (i.e. converges to the generalization error when the size of the sample increases) if the size of the validation set grows infinitely faster than the size of the learning set (which is counter-intuitive!) [7];
- cross-validation is almost unbiased [2];
- bootstrap is downward biased but has a very low variance [2];
- most recent bootstrap methods (632 and 632+) are almost unbiased and also have a low variance [2].

Given this list of possible model selection methods and criteria, the purpose of this paper is to compare experimentally these methods in the context of (1) highly nonlinear regression (making complex model structures unavoidable) and (2) a small number of data points or input-output pairs (which is often the case in real-world applications in medium- or high-dimensional spaces).

An important argument in favor of simple, reliable model selection methods is the following. In the context of nonlinear regression or model identification, the learning time (or computational complexity) may be far from negligible. In some cases, for example when one tries a single model structure and has a large validation set at disposal, this might not be a concern. However, in most situations, first one has to try many different model structures (for example of different complexities, different number of neurons in the hidden layers of multi-layer perceptrons, etc.); and secondly the use of resampling methods (CV, LOO, bootstrap,…) is necessary because of the small size of the original sample at disposal. This might multiply the learning time by several hundreds or thousands, making computation time a real concern, even on up-to-date powerful machines.

Therefore, this paper will consider the various model selection methods in the context of nonlinear regression with relatively small sample, from the point of view of performances and computational complexity.

This will be illustrated on a standard time series prediction benchmark, the Santa Fe A series [8]. This particular example has been chosen for illustration because [8] (1) the ideal regressor is known, avoiding the supplementary problem of choosing its size, (2) if the rules of the Santa Fe competition are followed, we have a small set of data (1000), (3) the relation to learn is highly nonlinear, and (4) making the hypothesis of a stationary process, we have a very large test set at disposal (8000 data), used to obtain a very good approximation of the generalization error (independently from the model selection method) for comparison purposes. According to our previous experience [9], we chose to approximate this series by a Radial-Basis Function network with 100 Gaussian kernels, the learning being achieved by the method presented in [10].

2 Model Selection Methods

This section presents the different methods enumerated in the introduction. The initial data used for the learning phase are the N pairs (x_i, y_i) with x_i representing the d-

dimensional input vectors and y_i the scalar outputs. Theses pairs form the learning dataset *X*. Each of the methods below computes an approximation of the generalization error obtained with a model *g*. The generalization error is defined by:

$$E_{gen}(g) = \lim_{N\to\infty} \sum_{i=1}^{N} \frac{(g(x_i) - y_i)^2}{N}, \tag{1}$$

with (x_i, y_i) the elements of an infinite test dataset and $g(x_i)$ the approximation of y_i obtained with the model *g*. The selected model will be the model minimizing this estimate of the generalization error.

2.1 Monte-Carlo Cross-Validation

The consecutive steps of the Monte-Carlo cross-validation method are:

1. One randomly draws without replacement some elements of the dataset *X*; these elements form a new learning dataset X_{learn}. The remaining elements of *X* form the validation set X_{val}. Usually, two third of the elements of *X* are used in X_{learn} and one third in X_{val} [1]; this rule will be used in the following.
2. The training of the model *g* is done using X_{learn} and the error $E_k(g)$ is calculated according to:

$$E_k(g) = \frac{\sum_{i=1}^{N/3} \left(g(x_i^{val}) - y_i^{val} \right)^2}{N/3}, \tag{2}$$

with (x_i^{val}, y_i^{val}) the elements of X_{val} and $g(x_i^{val})$ the approximation of y_i^{val} by model *g*.
3. Steps 1 and 2 are repeated *K* times, with *K* as large as possible. The error $E_k(g)$ is computed for each repetition *k*. The average error is defined by:

$$\hat{E}_{gen}(g) = \frac{\sum_{k=1}^{K} E_k(g)}{K}. \tag{3}$$

A particular case of the Monte-Carlo cross-validation method is the Hold-Out method, where the number *K* of repetitions is equal to 1.

2.2 *k*-fold Cross-Validation

The *k*-fold cross-validation method is a variant of the Monte-Carlo cross-validation method. The consecutive steps of this method are:

1. One divides the elements of the dataset *X* into *K* sets of roughly equal size. The elements of *the* k^{th} set form the validation set X_{val}. The other sets form a new learning dataset X_{learn}.
2. The training of the model *g* is done using X_{learn} and the error $E_k(g)$ is calculated according to:

$$E_k(g) = \frac{\sum_{i=1}^{N/K} \left(g(x_i^{val}) - y_i^{val} \right)^2}{N/K}, \tag{4}$$

with (x_i^{val}, y_i^{val}) the elements of X_{val} and $g(x_i^{val})$ the approximation of y_i^{val} by model *g*.

3. Steps 1 and 2 are repeated for *k* varying from 1 to *K*. The average error is computed according to (3).

A particular case of the *k*-fold cross-validation method is the Leave-One-Out, where *K* is equal to *N*.

2.3 Bootstrap

The consecutive steps of the bootstrap method, developed by Efron [2], are:

1. In the dataset *X*, one draws randomly *N* samples with replacement. These new samples form a new learning set X_{learn} with the same size as the original one. The validation set X_{val} is the original learning set *X*. This process is called re-sampling.
2. The training of the model *g* is done using X_{learn} and the errors $E_k^{val}(g)$ and $E_k^{learn}(g)$ obtained with this model are calculated according to the following equations:

$$E_k^{learn}(g) = \frac{\sum_{i=1}^{N} \left(g(x_i^{learn}) - y_i^{learn} \right)^2}{N}, \tag{5}$$

with $(x_i^{learn}, y_i^{learn})$ the elements of X_{learn} and $g(x_i^{learn})$ the approximation of y_i^{learn} obtained by model *g*;

$$E_k^{val}(g) = \frac{\sum_{i=1}^{N} \left(g(x_i^{val}) - y_i^{val} \right)^2}{N}, \tag{6}$$

with (x_i^{val}, y_i^{val}) the elements of X_{val} and $g(x_i^{val})$ the approximation of y_i^{val} by model *g*.

3. The optimism $D_k(g)$, a measure of performance degradation (for the same model) between a learning and a validation set, is computed according to:

$$D_k(g) = E_k^{val}(g) - E_k^{learn}(g). \tag{7}$$

4. Steps 1,2 and 3 are repeated *K* times, with *K* as large as possible. The average optimism $\hat{D}(g)$ is computed by:

$$\hat{D}(g) = \frac{\sum_{k=1}^{K} D_k(g)}{K}. \tag{8}$$

5. Once this average optimism is computed, a new training of the model *g* is done using the initial dataset *X*; the learning error $E_m^I(g)$ is calculated according to:

$$E_m^I(g) = \frac{\sum_{i=1}^{N} \left(g(x_i) - y_i \right)^2}{N}, \tag{9}$$

with (x_i, y_i) the elements of *X* and $g(x_i)$ the approximation of y_i obtained using the model *g*.

6. Step 5 is repeated repeated *M* times, with *M* as large as possible. For each repetition *m* the learning error $E_m^I(g)$ is computed. The apparent error $\hat{E}^I(g)$ is defined as the average of errors $E_m^I(g)$ over the *M* repetitions. In the case of a linear model *g*, this repetition is not necessary; learning of a linear model gives a unique set of parameters, making all learning errors $E_m^I(g)$ equal. With nonlinear models, this repetition performs a (Monte-Carlo) estimate of the most probable apparent error obtained after training of *g*.
7. Now we have an estimate of the apparent error and of the optimism, their sum gives an estimate of the generalization error:

$$\hat{E}_{gen}(g) = \hat{E}^I(g) + \hat{D}(g)\,. \tag{10}$$

A particular case of bootstrap method is the bootstrap 632 [3]. In this method, $\hat{E}_{gen}(g)$ is calculated according to:

$$\hat{E}_{gen}(g) = .368\hat{E}^I(g) + .632\hat{D}^{632}(g)\,, \tag{11}$$

with $\hat{D}^{632}(g)$ an optimism estimated only on the data that are not selected during the re-sampling (see [3] for details).

3 Radial-Basis Function Networks and Time Series Prediction

The nonlinear models that have been chosen in this paper are the RBFN [11]. These approximators have the well-known property of universal approximation. Other approximators as Multi-Layers Perceptrons could have been chosen. However, the goal of this paper is not to compare different families of approximators: RBFN have been chosen for the simplicity of their training phase. Indeed, a possible learning algorithm for RBFN [10] consists in a three-folded strategy (separated computation of kernel centers, widths and weights). As a consequence, the weights are simply calculated as the solution of a linear system.

The problem of time series prediction is a common problem in many different fields as finance, electricity production, hydrology, etc. A set of successive points (the time series) is known; the goal is to predict the next unknown value. To perform this task, a model is needed, that associates some previous values of the time series to the next one. This model has the following form:

$$y(t) = g(y(t-1), y(t-2), ..., y(t-n)) , \tag{12}$$

where n is the lag order [1]. Model g can be linear or nonlinear. In this paper, the class of models we have chosen is a set of RBFN with different numbers C of Gaussian kernels but with a given size for the regressor. The problem of model selection is then to choose one of the models in the set, i.e.: how many Gaussian kernels must be used?

4 Experimental Results

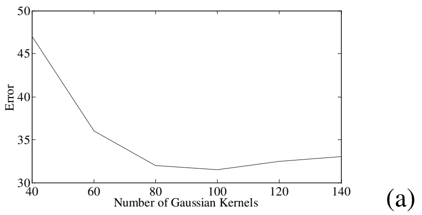
We illustrate the methods described in the previous section on a standard benchmark in time-series prediction. The Santa Fe A time series [8] has been chosen mainly for the large number of data available for the training stage (1000) as well as for the test stage (8000). These two numbers correspond to the rules of the Santa Fe competition, as detailed in [8]. According to [8], we also choose $n = 6$.

We trained 7 RBFNs on the Santa Fe learning dataset, for $C = 20$, 40, 60, 80, 100, 120 and 140 respectively. This number of models is kept small to avoid a too large computational time; indeed the goal is not to find the best among an infinite set of models but to compare the performances of the different model selection methods.

Having a huge (8000 samples) test set makes it possible to have a very good approximation of the true generalization error:

$$E_{gen}(g) = \lim_{N\to\infty} \sum_{i=1}^{N} \frac{(g(x_i) - y_i)^2}{N} \approx \sum_{i=1}^{8000} \frac{(g(x_i) - y_i)^2}{N} . \tag{13}$$

In practice, for similar reasons as those argued in favour of a Monte-Carlo evaluation (step 6) of the apparent error $\hat{E}^I(g)$ in the bootstrap (use of nonlinear models), a Monte-Carlo method (with $M = 100$ repetitions) is used to evaluate $E_{gen}(g)$ for each model. Then, according to Fig. 1 (a), the best RBFN model has a number C of Gaussian kernels equal to 100.



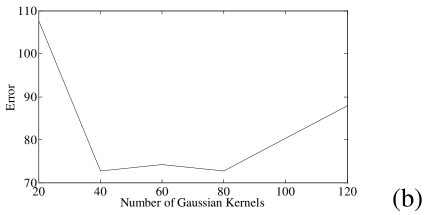


Fig. 1. (a) Estimation of the true generalization error; (b) Estimation of the generalization error by a Monte-Carlo cross-validation method.

Fig. 1 (b) shows the estimation of the generalization error by a Monte-Carlo cross-validation method. The optimal number of Gaussian kernel is 80 (another number of kernels that could be chosen -according to the parsimony principle- is 40, corresponding to a local minimum).

Figure 2 (a) shows the estimation of the generalization error by a Leave-One-Out method. The optimal number of Gaussian kernels is 80.

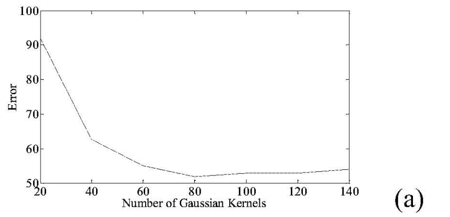


Fig. 2. (a) Estimation of the generalization error by Leave-One-Out; (b) Estimation of the generalization error by the classical bootstrap.

Fig. 2 (b) shows the estimation of the generalization error by the classical bootstrap. The optimal number of Gaussian kernels is between 80 and 120. In this case, again according to the parsimony principle, it is advised to choose $C = 80$.

Figure 3 shows the estimation of the generalization error by the bootstrap 632. The optimal number of Gaussian kernels is 100.

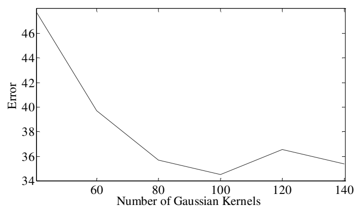


Fig. 3. Estimation of the generalization error by the bootstrap 632.

5 Conclusion

A number of similar experiments have been conducted, with the purpose of comparing model selection methods based on empirical estimations of the generalization error: Monte-Carlo cross-validation, Leave-One-Out, bootstrap, bootstrap 632. They all lead to similar conclusions: bootstrap methods, and in particular the bootstrap 632, give the best estimations of the generalization error.

Although theoretical results (briefly summarized in the introduction) show that all these methods are roughly asymptotically equivalent, this is no more the case in real applications, with limited number of data.

Concerning the computation time, the number of repetitions in the cross-validation and bootstraps may be tuned, which is not the case with the Leave-One-Out. This allows, in our 1000 learning data experiments, to strongly reduce the computational costs associated with cross-validation and Bootstraps.

However, in accordance with theoretical results, cross-validation has a much higher variance than bootstraps. Using the same number of repetitions in cross-validation and bootstraps highlights the advantages of the bootstraps.

Such results have also been published in different contexts, namely classification [5]. This paper extends these results to regression, and time series prediction in particular.

Acknowledgements. Michel Verleysen is Senior research associate at the Belgian National Fund for Scientific Research (FNRS). The work of A. Lendasse and V. Wertz is supported by the Interuniversity Attraction Poles (IAP), initiated by the Belgian Federal State, Ministry of Sciences, Technologies and Culture. The scientific responsibility rests with the authors.

References

1. Ljung, L.: System Identification - Theory for the user, 2nd ed, Prentice Hall, 1999.
2. Efron, B., Tibshirani, R. J. An introduction to the bootstrap, Chapman & Hall, 1993.
3. Efron, B, Tibshirani, R.: Improvements on cross-validation: The .632+ bootstrap method. J. Amer. Statist. Assoc. (1997) 92:548–560.
4. Stone, M.: An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion, J. Royal. Statist. Soc., B39, 44–7, 1977.
5. Kohavi, R.: A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, Proc. of the 14th Int. Joint Conf. on A.I., Vol. 2, Canada, 1995.
6. Efron, B.: Estimating the error rate of a prediction rule: improvements on cross-validation. Journal of American Statistical Association, (1983) **78**(382):316–331.
7. Shao, J., Tu, D.: The Jackknife and Bootstrap, Springer Series in Statistics, Springer-Verlag, New York (1995).
8. Weigend, A. S., Gershenfeld, N. A.: Times Series Prediction: Forecasting the future and Understanding the Past, Addison-Wesley Publishing Company (1994).
9. Simon, G., Lendasse, A., Wertz, V., Verleysen, M.: Fast approximation of the bootstrap for model selection, submitted to ESANN03, Bruges (Belgium), April 2003.
10. Benoudjit, N., Archambeau, C., Lendasse, A., Lee, J., Verleysen, M.: Width optimization of the Gaussian kernels in Radial Basis Function Networks, ESANN 2002, European Symposium on Artificial Neural Networks, Bruges (2002) 425–432.
11. Moody, J., Darken, C.: Learning with localised receptive fields, Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA (1989).