Analysis of Fast Input Selection: Application in Time Series Prediction

Jarkko Tikka, Amaury Lendasse, and Jaakko Hollmén

Helsinki University of Technology, Laboratory of Computer and Information Science, P.O. Box 5400, FI-02015 HUT, Finland
tikka@mail.cis.hut.fi
http://www.cis.hut.fi/tikka

Abstract. In time series prediction, accuracy of predictions is often the primary goal. At the same time, however, it would be very desirable if we could give interpretation to the system under study. For this goal, we have devised a fast input selection algorithm to choose a parsimonious, or sparse set of input variables. The method is an algorithm in the spirit of backward selection used in conjunction with the resampling procedure. In this paper, our strategy is to select a sparse set of inputs using linear models and after that the selected inputs are also used in the non-linear prediction based on multi-layer perceptron networks. We compare the prediction accuracy of our parsimonious non-linear models with the linear models and the regularized non-linear perceptron networks. Furthermore, we quantify the importance of the individual input variables in the non-linear models using the partial derivatives. The experiments in a problem of electricity load prediction demonstrate that the fast input selection method yields accurate and parsimonious prediction models giving insight to the original problem.

1 Introduction

Time series analysis is an important problem in natural and engineering sciences, both from the viewpoint of prediction and understanding of the behavior of the systems under study. There are numerous applications of time series analysis scattered in the published literature of econometrics, system identification, chemistry, statistics, pattern recognition, and neural networks [1]. It would be very appealing to be able to predict the behavior of the time series accurately, and at the same time to give insight to the system itself. Our target is to estimate time series prediction models that are both accurate and interpretable. By interpretability we mean that the models contain only a relatively small subset of input variables for the prediction. This gives emphasis to what is important in the prediction of system behavior. These kind of parsimonious, or sparse time series models are the focus of the paper. Inputs of the sparse models are selected from a large set of autoregressive input variables for a given past horizon. This approach tries to circumvent the problems of the high-dimensional input space, i.e. curse of dimensionality.

S. Kollias et al. (Eds.): ICANN 2006, Part II, LNCS 4132, pp. 161–170, 2006.

In the estimation of the sparse time series models, we rely on sparse regression techniques [2] and a backward selection strategy. In addition, resampling procedures [3] are used to take into account the inherent uncertainty of the finite data samples used in the estimation procedure. One of the main goals of the proposed method is to offer a fast and reliable method for input selection. In the first phase of the methodology, the linear model that is built is forced to be sparse. That is, we do not select the most accurate model, rather we select a compromise between sparsity and accuracy. In the second phase, the non-linear prediction model is constructed using the selected sparse set of inputs.

In this paper, we present an analysis of our previously published input selection method used for the problem of long-term time series prediction [4]. It is noteworthy, however, that the method is generally applicable to input selection problems. Our interest is to apply the method to input selection within time series prediction.

The rest of the article is organized as follows: Sect. 2 introduces relevant background in the time series prediction. Section 3 reviews our fast input selection procedure in the context of linear prediction models. Section 4 focuses on non-linear prediction models, i.e. multi-layer perceptron (MLP) networks, in which the selected variables are finally used. Also, sensitivity analysis of the MLP networks is presented. Section 5 presents the empirical experiments on which the findings are based on. Summary and Conclusions are presented in Sect. 6.

2 Time Series Prediction

In a time series prediction problem, future values of time series are predicted using the previous values. The previous and future values of time series are referred to inputs and outputs of the prediction model, respectively. One-step-ahead prediction is needed in general and it is called short-term prediction. If multi-step-ahead predictions are needed, it is known as long-term prediction.

Unlike short-term prediction, long-term prediction faces typically growing amount of uncertainties arising from various sources. For instance, an accumulation of errors and lack of information make the prediction more difficult. In the case of long-term prediction, there are several strategies to build prediction models. The direct and the recursive prediction are shortly described next.

2.1 Recursive Prediction Strategy

In the case of multi-step-ahead prediction, the recursive strategy uses the predicted values as known data to predict next ones. First, a one-step-ahead prediction is done $\hat{y}_t = f_1(y_{t-1}, y_{t-2}, ..., y_{t-l})$, where $y_{t-i}, i = 1, \ldots, l$ are the inputs. It is also possible to use external variables as inputs, but they are not considered here in order to simplify the notation. After that, the same model is used to predict two-step-ahead $\hat{y}_{t+1} = f_1(\hat{y}_t, y_{t-1}, y_{t-2}, \ldots, y_{t-l+1})$, where the predicted value of $\hat{y}_t$ is used instead of the true value, which is unknown. Then, the k-step-ahead predictions $y_{t+k-1}, k \geq 3$ are obtained iteratively. In the prediction of kth step, $l - k$ observed values and k predicted values are used as the inputs in the

case of $k < l$. When $k \geq l$, all the inputs are the predicted values. The use of the predicted values as inputs may deteriorate the accuracy of the prediction.

2.2 Direct Prediction Strategy

In the direct strategy, the model $\hat{y}_{t+k-1} = f_k(y_{t-1}, y_{t-2}, \ldots, y_{t-l})$ is used for k-step-ahead prediction. The predicted values are not used as inputs at all in this approach, thus the errors in the predicted values are not accumulated into the next predictions. When all the values from y_t to y_{t+k-1} need to be predicted, k different models must be constructed. This increases the computational complexity, but more accurate results are achieved using the direct than the recursive strategy as shown in [4] and [5]. We only apply the direct strategy in this paper.

3 Fast Input Selection

Consider the situation that there are N measurements available from an output variable y and input variables $x_i, i = i, \ldots, l$. In the regression problems the usual task is to estimate the values of the output y using the inputs x_i. If the dependency is assumed to be linear it can be written mathematically

$$y_j = \sum_{i=1}^{l} \beta_i x_{j,i} + \varepsilon_j, \quad j = 1, \ldots, N \ . \tag{1}$$

The errors ε_j are assumed to be independently normally distributed with zero mean and common variance. All the variables are assumed to have zero mean and unit variance, thus the constant term is dropped out from the model. The ordinary least squares (OLS) estimates of the regression coefficients $\hat{\beta}_i$ are obtained by minimizing the mean squared error (MSE) [2].

The OLS estimates are not typically satisfactory [2]. Firstly, the generalization ability of the model may be improved by shrinking some coefficients toward zero or setting them exactly to zero. Secondly, if the number of inputs is large interpretation of the model might be difficult. Understanding or interpretability of the underlying process can be increased by selecting the subset of inputs which have the strongest effect in the prediction. Many approaches to input selection are presented in [2] and [6].

We propose an efficient input selection procedure in [4]. The algorithm is based on the bootstrap resampling procedure and it requires separate training and validation sets. However, it is straightforward to use other resampling procedures [3], e.g. k-fold cross-validation, instead of bootstrap.

The input selection procedure starts by estimating the linear model using all the available inputs. The sampling distributions of OLS estimates $\hat{\beta}_i$ and the standard deviation s_{tr} of the training MSEs are estimated using M times k-fold cross-validation. We have Mk different training sets and, thus, Mk estimates for the each coefficient β_i, which formulate the sampling distribution. In addition, we have Mk estimates for both training and validation MSE.

The median m_{β_i} is calculated from Mk estimates $\hat{\beta}_i$. The median is used as the location parameter for the distribution, since it is a reasonable estimate for skewed distributions and distributions with outliers. The width of the distribution of $\hat{\beta}_i$ is evaluated using the difference $\Delta_{\beta_i} = \hat{\beta}_i^{high} - \hat{\beta}_i^{low}$, where $\hat{\beta}_i^{high}$ is $Mk(1-q)$th and $\hat{\beta}_i^{low}$ is Mkqth value in the ordered list of the Mk estimates $\hat{\beta}_i$ [3] and q can be set, e.g., $q = 0.165$. With this choice of q, the difference Δ_{β_i} is twice as large as the standard deviation in the case of the normal distribution. The difference Δ_{β_i} describes well the width of both asymmetric and symmetric distributions.

The next step is to delete the least significant input variable. The ratio $|m_{\beta_i}|/\Delta_{\beta_i}$ is used as a measure of significance of the corresponding input variable. The input with the smallest ratio is pruned from the set of inputs. After that, the cross-validation procedure using the remaining inputs and pruning is repeated as long as there are variables left in the set of inputs.

The previous procedure removes inputs sequentially from the set of possible inputs. In the end, we have l different models. The purpose is to select a model which is as sparse as possible, but it still has comparable prediction accuracy. The initial model is selected based on the minimum validation error E_v^{min}. The final model is the least complex model whose validation error is under the threshold $E_v^{min} + s_{tr}^{min}$, where s_{tr}^{min} is the standard deviation of training MSE of the model having the minimum validation error. This means that we also include our uncertainty in the training phase into the selection of final model. The algorithmic details of the proposed method are presented in [4].

Advantage of the described algorithm is the ranking of the inputs according to their explanatory power. The pruning starts from the least significant inputs and the resulting model includes only a few most significant ones. This might be useful information for interpretation of the underlying process. Also, the computational complexity of the proposed algorithm is linear $\mathcal{O}(l)$ with respect to the number of available inputs l. Therefore, it is applicable in the case of large number of inputs.

4 Non-linear Modeling Using MLP Networks

Although the linear models are easy to interpret and fast to calculate they are not accurate enough in some problems. The dependencies between the variables are described better using a non-liner model. However, many non-linear models are black-box models and interpretation is almost impossible. Our proposal is to use the selected inputs also in the non-linear model. Goals of this approach are to avoid the curse of dimensionality, over-parameterization, and overfitting in the non-linear modeling phase. In addition, the interpretability of the non-linear model increases, since only a subset of inputs is included to the model.

MLP networks are used in the non-linear modeling phase

$$\hat{y} = \mu + \sum_{j=1}^{p} \alpha_j \tanh(\sum_{i=1}^{l} w_{ij} x_i + b_j) \ , \tag{2}$$

where p is the number of neurons in the hidden layer, $\hat{y}$ is the estimate of the output y and μ, α_j, and w_{ij} are the weights of the network. It is known that only one hidden layer is required to approximate any continuous function if the number of connection weights is sufficient [7].

The number of connection weights is controlled by the number of neurons in the hidden layer. The selection of number of neurons is based on k-fold cross-validation. The optimal connection weights minimize MSE in the validation sets. Another option is to set the number of neurons to be large enough and to use weight decay (WD) to reduce the effective number of connection weights [8]. When WD is applied the cost function is

$$E = \frac{1}{N}\left(\sum_{j=1}^{N}(y_j - \hat{y}_j)^2 + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}\right) \quad , \tag{3}$$

where $\boldsymbol{\theta}$ is the vector containing all the parameters of the network and λ is the weight decay parameter. A proper value for λ can be selected by cross-validation. In WD, the values of weights are shrunk toward zero, but they are not set exactly to zero. So, it is very likely that WD does not perform input selection.

4.1 Sensitivity Analysis

It may not be enough that the most relevant inputs are found. In many applications, it is important to evaluate the way inputs contribute to explanation or prediction of the output.

The contribution of each input to the output can be evaluated using partial derivatives of the MLP network [9]. Partial derivatives (PAD) method gives two results. First, a profile of the output variations for a small changes of each input. Second, classification of the inputs in increasing order of relative importance. It is found that the PAD method gives stable results [9].

The partial derivative of MLP network (2) with respect to the input x_i is

$$d_i = \frac{\partial \hat{y}}{\partial x_i} = \sum_{j=1}^{p} \alpha_j (1 - I_j^2) w_{ij} \quad , \tag{4}$$

where $I_j = \tanh(\sum_{i=1}^{l} w_{ij} x_i + b_j)$. A set of graphs of the partial derivatives versus each corresponding input can be plotted. The graphs show the influence of the inputs on the output.

The sensitivity of the MLP output for the data set with respect to an input is calculated

$$\mathrm{SSD}_i = \sum_{j=1}^{N} d_{i,j}^2, \qquad \mathrm{SSD}_i \leftarrow \frac{\mathrm{SSD}_i}{\sum_{i=1}^{l} \mathrm{SSD}_i} \quad . \tag{5}$$

Sum of squared derivatives (SSD) value is achieved for each input. SSD_i is the sum over all the observations. In the end, the SSD_i values are scaled such that $\sum_{i=1}^{l} \mathrm{SSD}_i = 1$. The input having the highest SSD_i value influences most on the output. The ranking based on SSD values can be compared to the ranking obtained using the input selection method presented in Sect. 3.

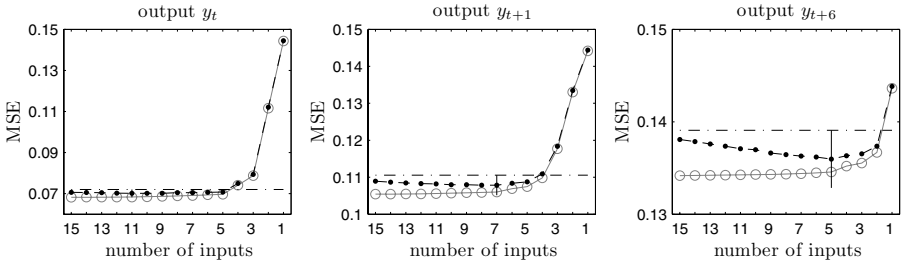

Fig. 1. Illustration of input selection, training error (*gray line*) and validation error (*black line*) as a function of the number of inputs in the linear model. The vertical line marks the minimum validation error and the horizontal dash-dotted line represents the threshold, which is used in the selection of the final model.

5 Experiments

The two-phase modeling strategy described in the previous sections is applied to time series prediction. The data set used is the Poland electricity load time series[1] [5]. It contains daily measurements from the electricity load in Poland in the 1990's. The data set has 1400 observations in the training set and 201 observations in the test set. The training and test sets are not consecutive. The objective is to predict the electricity load one- (y_t), two- (y_{t+1}), and seven-day-ahead (y_{t+6}). We use direct prediction approach, i.e. we have to construct own model for each case. The data is normalized to zero mean and unit variance before the analysis.

5.1 Phase I: Input Selection

The maximum number of inputs is set to be $l = 15$, i.e. the available inputs are $y_{t-l}, l = 1, \ldots, 15$ in each of the three prediction cases. In the input selection, 10-fold cross-validation repeated $M = 100$ times is used. This choice produces 1000 estimates for the coefficients β_i, which is considered to be large enough for reliably estimating the distribution of the parameters in the linear model.

Figure 1 illustrates the input selection procedure. In all the three cases, it is notable that the validation error starts to increase only in the end. Almost all the inputs are pruned from the model then. If the final model had been selected according to the minimum validation error the number of inputs would have been 11, 7, and 5 in the case of one-, two, and seven-day-ahead prediction, respectively. However, even more parsimonious models are achieved when the thresholding is used. The final numbers of inputs are 5, 5, and 2 and the validation errors do not increase significantly.

In Fig. 2, the selected inputs for all the three models are visualized. The smaller the white number is in the selected inputs (in the black rectangles) the more important the corresponding input is in the prediction. In other words, the

[1] http://www.cis.hut.fi/projects/tsp/?page=Timeseries

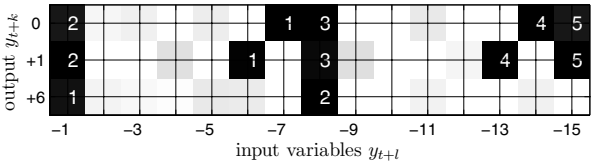

Fig. 2. The final models. The outputs $y_{t+k}, k = 0, +1, +6$ are in the vertical axis and the possible inputs $y_{t-l}, l = 1, \ldots, 15$ are in the horizontal axis. The selected inputs are denoted by black rectangles on each row and the white numbers indicate the ranking of the inputs according to the importance in the prediction.

number 1 indicates that the input was the last to prune from the model. For instance, the upper row shows that the one-day-ahead model has 5 inputs, which are $y_{t-7}, y_{t-1}, y_{t-8}, y_{t-14}$, and y_{t-15} in the decreasing order of importance. The model has nice interpretation, since the inputs correspond to values of 7, 1, 8, 14, and 15 days before the predicted value. It is plausible that the two most important inputs are the values of one week and one day before.

5.2 Phase II: Non-linear Modeling

Based on the results of the input selection, non-linear models are trained. Three MLP networks are constructed for each output: i) MLP using the selected inputs without weight decay, number of neurons (the maximum were 15) in the hidden layer selected by 10-fold cross-validation repeated five times, ii) MLP using the selected inputs with weight decay, number of neurons in the hidden layer was 20, and iii) MLP with all the inputs with weight decay, number of neurons in the hidden layer was 20. In the cases ii) and iii) MLPs are evaluated using 30 values of the regularization parameter λ, which are logarithmically equally spaced in the range $\lambda \in [10^{-4}, 10^{3}]$. The optimal value of λ is selected using 10-fold cross-validation repeated five times to increase the reliability of the results.

All the networks are trained using the Levenberg-Marquardt optimization method by back-propagating the error gradients [8]. Ten different initializations are used in the training of the networks in order to avoid local minima. The training errors were monotonically decreasing as a function of increasing complexity. In Fig. 3, the validation errors are shown as a function of λ for the cases ii) (*left*) and iii) (*right*). It is notable that the minimum errors are roughly on the same level, but the curve is flater in the left figure. Thus, a sparser grid for λ could be used, which would reduce the computational burden. In the case i), the minimum validation error is obtained using $p = 6$, $p = 6$, and $p = 7$ neurons in the hidden layer for the outputs y_t, y_{t+1}, and y_{t+6}, respectively.

The prediction accuracy of the final models were evaluated using the test set, which is not used at all in the training and selection of the final models. Thousand bootstrap replications were drawn from the test set and MSE was calculated for each replication. The means and the standard deviations of MSE for each model are reported in Table 1. The sparse linear models are equally accurate as the full linear models, which indicates that the selected inputs are the most informative.

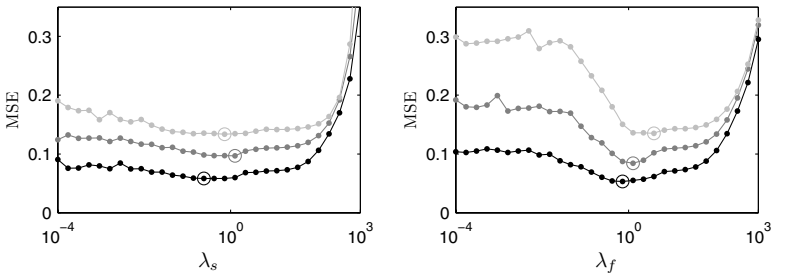

Fig. 3. Validation errors as a function of λ for one-day-ahead (*black line*), two-day-ahead (*dark gray line*), and seven-day-ahead (*light gray line*) prediction in the case of selected inputs (*left*) and all the inputs (*right*). Circles mark the minimums.

Table 1. MSEs and standard deviations of MSEs for the test set calculated using the bootstrap resampling procedure. n is the number of inputs, p is the number of neurons in the hidden layer, and λ is the regularization parameter.

	full linear linear $n = 15$	sparse linear model	MLP n-p-1 no WD	MLP n-20-1 with WD	MLP 15-20-1 with WD
1-day-ahead	0.054 (0.012)	0.055 (0.012) $n = 5$	0.038 (0.010) $n = 5$, $p = 6$	0.040 (0.010) $n = 5$, $\lambda = 0.24$	0.038 (0.010) $\lambda = 0.73$
2-day-ahead	0.085 (0.019)	0.086 (0.018) $n = 5$	0.074 (0.018) $n = 5$, $p = 6$	0.077 (0.017) $n = 5$, $\lambda = 1.27$	0.079 (0.016) $\lambda = 1.27$
7-day-ahead	0.118 (0.023)	0.116 (0.023) $n = 2$	0.116 (0.022) $n = 2$, $p = 7$	0.117 (0.023) $n = 2$, $\lambda = 0.73$	0.114 (0.023) $\lambda = 3.86$

In the cases of one- and two-day-ahead prediction, MLP with selected inputs without WD is the most accurate. It decreases MSE 30% and 13% compared to the full linear model in one- and two-day-ahead predictions, respectively. Also, it is slightly better than MLP with all the inputs. For seven-day-ahead prediction, MLP with all the inputs and WD has the lowest prediction error, although the errors of the other methods are nearly the same.

In Fig. 4, the relative importances of the inputs calculated by (5) are shown. The importances are averages over thousand bootstrap replications of the test set. In the case of one-day-ahead prediction and 5-6-1 MLP, the inputs are ranked in the order of decreasing importance as follows: y_{t-1}, y_{t-7}, y_{t-15}, y_{t-8}, and y_{t-14}. The ranking is nearly the same as with the linear models, see Fig 2. In 15-20-1 MLP, the five most important inputs in the order of decreasing importance are y_{t-1}, y_{t-7}, y_{t-2}, y_{t-8}, y_{t-15}. Four of them are same as obtained with the linear models. Also, in the cases of two- and seven-day-ahead prediction

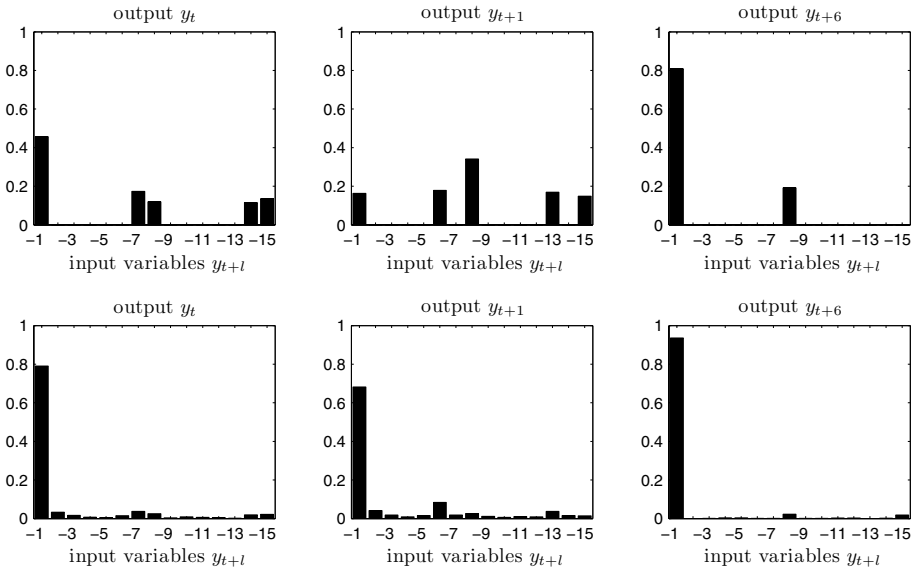

Fig. 4. Relative importances of the input variables $y_{t-l}, l = 1, \ldots, 15$ in 5-6-1 MLP network without WD (*above*), in 15-20-1 MLP with WD (*below*)

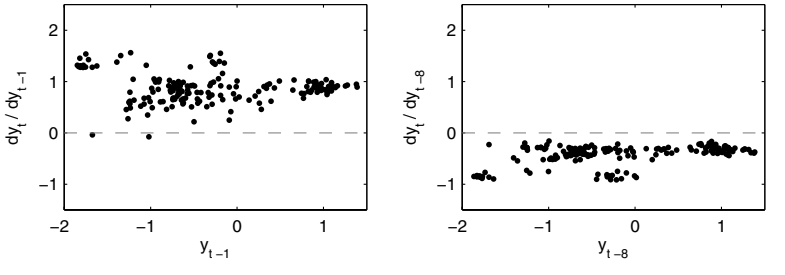

Fig. 5. The profiles of the inputs y_{t-1} (*left*) and y_{t-8} (*right*) in 5-6-1 MLP in one-day-ahead prediction

the relative importances of inputs in the MLP networks are nearly the same as with the linear model.

The contribution of the inputs y_{t-1} and y_{t-8} in the prediction of y_t with 5-6-1 MLP are shown in Fig. 5. The shown result is for the test set. The values of $\partial y_t / \partial y_{t-1}$ are positive, which means that y_t tends to increase while y_{t-1} increases. Although the relative importance of y_{t-8} is notably smaller than y_{t-1}, still the partial derivatives $\partial y_t / \partial y_{t-8}$ are clearly non-zero and negative. Thus, y_{t-8} has also contribution in the prediction. While y_{t-8} increases the output y_t tends to decrease.

6 Summary and Conclusions

A backward selection type algorithm with resampling for input selection in the context of time series prediction was presented. Experiments in an electricity load prediction demonstrated that the two phase strategy using input selection in a linear prediction model and subsequent non-linear modeling using MLP yields accurate prediction. In addition, this strategy was competitive to MLP network with all the inputs and a large number of neurons in the hidden layer trained with weight decay. The importance of inputs obtained using the linear models reflected also very well the importance of inputs in the non-linear models. The advantage of presented approach is sparsity in terms of the number of inputs and parameters in the final network. Sparsity of inputs makes the models more interpretable. A low number of parameters allows fast training of the networks and makes the models less prone to overfitting.

References

1. Weigend, A.S., Gershenfeld, N.A. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)
2. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning – Data Mining, Inference and Prediction. Springer Series in Statistics. (2001)
3. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall/CRC (1993)
4. Tikka, J., Hollmén, J., Lendasse, A.: Input Selection for Long-Term Prediction of Time-Series. In: Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN 2005). 1002–1009
5. Ji, Y., Hao, J., Reyhani, N., Lendasse, A.: Direct and Recursive Prediction of Time Series Using Mutual Information Selection. In: Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN 2005). 1010–1017
6. Ljung, L.: System Identification – Theory for the User. 2nd Edition. Prentice–Hall. (1999)
7. Hornik, K., Stinchcombe, M., White, H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks. **2** (1989) 359–366
8. Bishop, C.: Neural Networks in Pattern Recognition. Oxford Press (1996)
9. Gevrey, M., Dimopoulos, I., Lek, S.: Review and Comparison of Methods to Study the Contribution of Variables in Artificial Neural Network Models. Ecological Modelling **160** (2003) 249–264