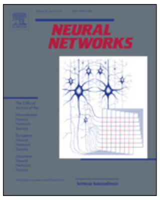
Neural Networks 42 (2013) 1–27

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Wavelet neural networks: A practical guide

Antonios K. Alexandridis[a,*], Achilleas D. Zapranis[b]

[a] University of Kent, School of Mathematics, Statistics and Actuarial Science, Cornwallis Building, P.O. CT2 7NF, Canterbury, UK
[b] University of Macedonia of Economic and Social Sciences, Department of Accounting and Finance, 156 Egnatia St., P.O. 54006, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 21 November 2010
Received in revised form 25 May 2012
Accepted 13 January 2013

Keywords:
Wavelet networks
Model identification
Variable selection
Model selection
Confidence intervals
Prediction intervals

ABSTRACT

Wavelet networks (WNs) are a new class of networks which have been used with great success in a wide range of applications. However a general accepted framework for applying WNs is missing from the literature. In this study, we present a complete statistical model identification framework in order to apply WNs in various applications. The following subjects were thoroughly examined: the structure of a WN, training methods, initialization algorithms, variable significance and variable selection algorithms, model selection methods and finally methods to construct confidence and prediction intervals. In addition the complexity of each algorithm is discussed. Our proposed framework was tested in two simulated cases, in one chaotic time series described by the Mackey–Glass equation and in three real datasets described by daily temperatures in Berlin, daily wind speeds in New York and breast cancer classification. Our results have shown that the proposed algorithms produce stable and robust results indicating that our proposed framework can be applied in various applications.

1. Introduction

Wavelet networks are a new class of networks that combine the classic sigmoid neural networks (NNs) and the wavelet analysis (WA). WNs have been used with great success in a wide range of applications. However a general accepted framework for applying WNs is missing from the literature. In this study, we present a complete statistical model identification framework in order to apply WNs in various applications. To our knowledge we are the first to do so. Although a vast literature about WNs exists, to our knowledge this is the first study that presents a step by step guide for model identification for WNs. Model identification can be separated in two parts, model selection and variable significance testing. In this study a framework similar to the one proposed by Zapranis and Refenes (1999) for the classical sigmoid NNs is adapted. More precisely, the following subjects were thoroughly examined: the structure of a WN, training methods, initialization algorithms, variable significance and variable selection algorithms, model selection methods and finally methods to construct confidence and prediction intervals. Only in Iyengar, Cho, and Phoha (2002) some of these issues are studied to some extent.

WA has proved to be a valuable tool for analyzing a wide range of time-series and has already been used with success in image processing, signal de-noising, density estimation, signal and image compression and time-scale decomposition. WA is often regarded as a “microscope” in mathematics, (Cao, Hong, Fang, & He, 1995), and it is a powerful tool for representing nonlinearities, (Fang & Chow, 2006). The major drawback of the WA is that it is limited to applications of small input dimension. The reason is that the construction of a wavelet basis is computationally expensive when the dimensionality of the input vector is relatively high, (Zhang, 1997).

On the other hand NNs have the ability to approximate any deterministic non-linear process, with little knowledge and no assumptions regarding the nature of the process. However the classical sigmoid NNs have a series of drawbacks. Typically, the initial values of the NN’s weights are randomly chosen. Random weight initialization is generally accompanied with extended training times. In addition, when the transfer function is of sigmoidal type, there is always significant change that the training algorithm will converge to local minima. Finally, there is no theoretical link between the specific parameterization of a sigmoidal activation function and the optimal network architecture, i.e. model complexity (the opposite holds true for WNs).

In Pati and Krishnaprasad (1993) it has been demonstrated that it is possible to construct a theoretical formulation of a feedforward NN in terms of wavelet decompositions. WNs were proposed by Zhang and Benveniste (1992) as an alternative to feedforward NNs which would alleviate the aforementioned weaknesses associated with each method. The WNs are a generalization of radial basis function networks (RBFN). WNs are one hidden

[*] Corresponding author.
E-mail addresses: A.Alexandridis@kent.ac.uk (A.K. Alexandridis), zapranis@uom.gr (A.D. Zapranis).

0893-6080/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2013.01.008

layer networks that use a wavelet as an activation function, instead of the classic sigmoidal family. It is important to mention here that the multidimensional wavelets preserve the "universal approximation" property that characterizes NNs. The nodes (or wavelons) of WNs are the wavelet coefficients of the function expansion that have a significant value. In Bernard, Mallat, and Slotine (1998) various reasons were presented in why wavelets should be used instead of other transfer functions. In particular, firstly, wavelets have high compression abilities, and secondly, computing the value at a single point or updating the function estimate from a new local measure, involves only a small subset of coefficients.

WNs have been used in a variety of applications so far, i.e. in short term load forecasting, (Bashir & El-Hawary, 2000; Benaouda, Murtagh, Starck, & Renaud, 2006; Gao & Tsoukalas, 2001; Ulugammai, Venkatesh, Kannan, & Padhy, 2007; Yao, Song, Zhang, & Cheng, 2000), in time series prediction, (Cao et al., 1995; Chen, Yang, & Dong, 2006; Cristea, Tuduce, & Cristea, 2000), signal classification and compression, (Kadambe & Srinivasan, 2006; Pittner, Kamarthi, & Gao, 1998; Subasi, Alkan, Koklukaya, & Kiymik, 2005), signal denoising, (Zhang, 2007), static, dynamic (Allingham, West, & Mees, 1998; Oussar & Dreyfus, 2000; Oussar, Rivals, Presonnaz, & Dreyfus, 1998; Pati & Krishnaprasad, 1993; Postalcioglu & Becerikli, 2007; Zhang & Benveniste, 1992), and nonlinear modeling, (Billings & Wei, 2005), nonlinear static function approximation, (Jiao, Pan, & Fang, 2001; Szu, Telfer, & Kadambe, 1992; Wong & Leung, 1998), to mention the most important. In Khayamian, Ensafi, Tabaraki, and Esteki (2005) WN were even proposed as a multivariate calibration method for simultaneous determination of test samples of copper, iron, and aluminum.

In contrast to classical "sigmoid NNs", WNs allow for constructive procedures that efficiently initialize the parameters of the network. Using wavelet decomposition a "wavelet library" can be constructed. In turn, each wavelon can be constructed using the best wavelet of the wavelet library. The main characteristics of these procedures are: (i) convergence to the global minimum of the cost function, (ii) initial weight vector into close proximity of the global minimum, and as a consequence drastically reduced training times, (Zhang, 1997; Zhang & Benveniste, 1992). In addition, WNs provide information for the relative participation of each wavelon to the function approximation and the estimated dynamics of the generating process. Finally, efficient initialization methods will approximate the same vector of weights that minimize the loss function each time.

In Zapranis and Alexandridis (2008) and Zapranis and Alexandridis (2009) we give a concise treatment of wavelet theory. For a complete theoretical background on wavelets and wavelet analysis refer to (Daubechies, 1992; Mallat, 1999). Here the emphasis is in presenting the theory and mathematics of wavelet neural networks.

The rest of the paper is organized as follows. In Section 2 we present the WN. More precisely in Section 2.1 the structure of a WN is described. In Section 2.2 various initialization methods were described. In Section 2.3 a training method of the WN is presented and in Section 2.4 the stopping conditions of the training are described. In Section 2.5 the various initialization methods are compared and evaluated. A model selection algorithm is described in Section 3 and is evaluated in two cases in Section 3.1. Next, various criteria for selecting significant variables are presented while a variable selection algorithm is analytically presented in Section 4.1. In Section 4.2 the proposed algorithm is evaluated in two cases. In Section 5 methods to estimate the model and variance uncertainty are described. In Section 5.1 a framework for constructing confidence intervals is presented while in Section 5.2 a framework for constructing prediction intervals is presented. In Section 5.3 the proposed framework for constructing confidence and prediction intervals is evaluated in two cases. In Section 6 the proposed framework is applied in real data described by temperature in Berlin. Similarly, our framework is applied in wind speed data in Section 7. In Section 8 a WN is constructed for breast cancer classification while in Section 9 the proposed framework is applied in modeling and predicting the chaotic Mackey–Glass equation. Finally, in Section 10 we conclude.

2. Wavelet neural networks for multivariate process modeling

2.1. Structure of a wavelet network

In this section the structure of a WN is presented and discussed. A WN usually has the form of a three layer network. The lower layer represents the input layer, the middle layer is the hidden layer and the upper layer is the output layer.

In the input layer the explanatory variables are introduced to the WN. The hidden layer consists of the hidden units (HUs). The HUs are often referred as wavelons, similar to neurons in the classical sigmoid NNs. In the hidden layer the input variables are transformed to dilated and translated version of the mother wavelet. Finally, in the output layer the approximation of the target values is estimated.

The idea of a WN is to adapt the wavelet basis to the training data. Hence, the wavelet estimator is expected to be more efficient than a sigmoid NN, (Zhang, 1993). In Billings and Wei (2005), Kadambe and Srinivasan (2006), Mellit, Benghamen, and Kalogirou (2006), Xu and Ho (1999) an adaptive WN was used. In Chen et al. (2006) a local linear WN was proposed. The difference is that the connections weights between the hidden layer and output layer are replaced by a local linear model. In Fang and Chow (2006) and Jiao et al. (2001) a multiwavelet NN is proposed. In this structure, the activation function is a linear combination of wavelet bases instead of the wavelet function. During the training phase, the weights of all wavelets are updated. The multiwavelet NN is also enhanced by the DWT. Their results indicate that the proposed model increases the approximation capability of the network. In Khayamian et al. (2005) a principal component-wavelet NN was introduced. In this context, first principal component analysis (PCA) has been applied to the training data in order to reduce the dimensionality. Then a WN was used for function approximation. In Zhao, Chen, and Shen (1998) a multidimensional wavelet-basis function NN was used. More precisely (Zhao et al., 1998) use a multidimensional wavelet function as the activation function in the hidden layer. Then the sigmoid function was used as an activation function in the output layer. (Becerikli, 2004) proposes a network with unconstrained connectivity and with dynamic elements (lag dynamics) in its wavelet processing units called dynamic WN.

In this study, we implement a multidimensional WN with a linear connection between the wavelons and the output. Moreover, in order for the model to perform well in the presence of linearity, we use direct connections from the input layer to the output layer. Hence, a network with zero HUs is reduced to the linear model.

The structure of a single hidden-layer feedforward WN is given in Fig. 1. The network output is given by the following expression:

$$g_\lambda(\mathbf{x};\mathbf{w}) = \hat{y}(\mathbf{x}) = w_{\lambda+1}^{[2]} + \sum_{j=1}^{\lambda} w_j^{[2]} \cdot \Psi_j(\mathbf{x}) + \sum_{i=1}^{m} w_i^{[0]} \cdot x_i. \qquad (1)$$

In that expression, $\Psi_j(\mathbf{x})$ is a multidimensional wavelet which is constructed by the product of m scalar wavelets, $\mathbf{x}$ is the input vector, m is the number of network inputs, λ is the number of HUs

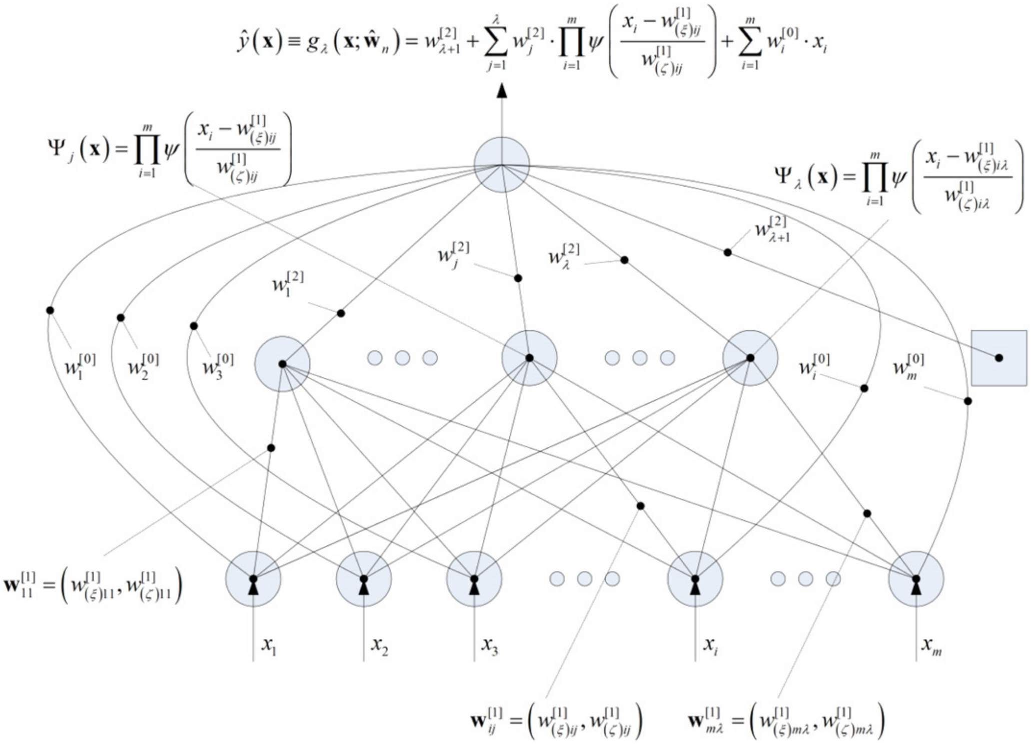

$\hat{y}(\mathbf{x}) \equiv g_\lambda(\mathbf{x}; \hat{\mathbf{w}}_n) = w_{\lambda+1}^{[2]} + \sum_{j=1}^{\lambda} w_j^{[2]} \cdot \prod_{i=1}^{m} \psi\left(\frac{x_i - w_{(\xi)ij}^{[1]}}{w_{(\zeta)ij}^{[1]}}\right) + \sum_{i=1}^{m} w_i^{[0]} \cdot x_i$
$\Psi_j(\mathbf{x}) = \prod_{i=1}^{m} \psi\left(\frac{x_i - w_{(\xi)ij}^{[1]}}{w_{(\zeta)ij}^{[1]}}\right)$
$\Psi_\lambda(\mathbf{x}) = \prod_{i=1}^{m} \psi\left(\frac{x_i - w_{(\xi)i\lambda}^{[1]}}{w_{(\zeta)i\lambda}^{[1]}}\right)$
$w_{\lambda+1}^{[2]}$
$w_j^{[2]}$
$w_\lambda^{[2]}$
$w_1^{[2]}$
$w_1^{[0]}$
$w_2^{[0]}$
$w_3^{[0]}$
$w_i^{[0]}$
$w_m^{[0]}$
$\mathbf{w}_{11}^{[1]} = \left(w_{(\xi)11}^{[1]}, w_{(\zeta)11}^{[1]}\right)$
x_1
x_2
x_3
x_i
x_m
$\mathbf{w}_{ij}^{[1]} = \left(w_{(\xi)ij}^{[1]}, w_{(\zeta)ij}^{[1]}\right)$
$\mathbf{w}_{m\lambda}^{[1]} = \left(w_{(\xi)m\lambda}^{[1]}, w_{(\zeta)m\lambda}^{[1]}\right)$

Fig. 1. A feedforward wavelet neural network.

and w stands for a network weight. The multidimensional wavelets are computed as follows:

$$\Psi_j(\mathbf{x}) = \prod_{i=1}^{m} \psi(z_{ij}) \tag{2}$$

where ψ is the mother wavelet and

$$z_{ij} = \frac{x_i - w_{(\xi)ij}^{[1]}}{w_{(\zeta)ij}^{[1]}}. \tag{3}$$

In the above expression, $i = 1, \ldots, m, j = 1, \ldots, \lambda + 1$ and the weights w correspond to the translation ($w_{(\xi)ij}^{[1]}$) and the dilation ($w_{(\zeta)ij}^{[1]}$) factors. The complete vector of the network parameters comprises: $w = \left(w_i^{[0]}, w_j^{[2]}, w_{\lambda+1}^{[2]}, w_{(\xi)ij}^{[1]}, w_{(\zeta)ij}^{[1]}\right)$. These parameters are adjusted during the training phase.

In bibliography three mother wavelets are usually suggested, the Gaussian derivative, the second derivative of the Gaussian, the so-called "Mexican Hat" and the Morlet wavelet.

The selection of the mother wavelet depends on the application and is not limited to the above choices. The activation function can be a wavenet (orthogonal wavelets) or a wave frame (continuous wavelets). Following (Becerikli, Oysal, & Konar, 2003; Billings & Wei, 2005; Zhang, 1994) we use as a mother wavelet the Mexican Hat function which proved to be useful and to work satisfactorily in various applications and is given by:

$$\psi(z_{ij}) = (1 - z_{ij}^2)e^{-\frac{1}{2}z_{ij}^2}. \tag{4}$$

2.2. Initialization of the parameters of the network

In WNs, in contrast to NNs that use sigmoid functions, selecting initial values of the dilation and translation parameters randomly may not be suitable, (Oussar et al., 1998). A wavelet is a waveform of effectively limited duration that has an average value of zero and localized properties hence a random initialization may lead to wavelons with a value of zero. Training algorithms like gradient descent with random initialization are inefficient, (Zhang, 1993), since random initialization affects the speed of training and may lead to a local minimum of the loss function, (Postalcioglu & Becerikli, 2007). Also, in sigmoid NNs, although a minimization of the loss function can be replicated with random initialization the values of the weights will be vary each time, (Anders & Korn, 1999).

Utilizing the information that can be extracted by the WA from the input dataset the initial values of the parameters w of the network can be selected in an efficient way. Efficient initialization will result to less iterations in the training phase of the network and training algorithms that will avoid local minimums of the loss function in the training phase. Finally, efficient initialization methods will approximate the same vector of weights that minimize the loss function each time.

Various methods have been proposed for an optimized initialization of the wavelet parameters. In Zhang and Benveniste (1992) the following initialization for the translation and dilation parameters is introduced:

$$w_{(\xi)ij}^{[1]} = 0.5\,(N_i + M_i) \tag{5}$$

$$w_{(\zeta)ij}^{[1]} = 0.2\,(M_i - N_i) \tag{6}$$

where M_i and N_i are defined as the maximum and minimum of input x_i.

$$M_i = \max_{p=1,\ldots,n}\left(x_{ip}\right) \tag{7}$$

$$N_i = \min_{p=1,\ldots,n}\left(x_{ip}\right). \tag{8}$$

In the above framework, the initialization of the parameters is based on the input domains defined by the examples of the training sample, (Oussar et al., 1998).

The initialization of the direct connections $w_i^{[0]}$ and the weights $w_j^{[2]}$ is less important and they are initialized in small random values between 0 and 1.

The previous heuristic method is simple and it its computational cost is almost negligible. However, it is not efficient as it is shown on the next section. The heuristic method does not guarantee that the training will find the global minimum. Moreover this method does not use any information that the wavelet decomposition can provide.

Recent studies proposed more complex methods that utilize the information extracted by the WA, (Kan & Wong, 1998; Oussar & Dreyfus, 2000; Oussar et al., 1998; Wong & Leung, 1998; Xu & Ho, 2002; Zhang, 1997). These methods are not optimal but a trade-off between optimality and efficiency, (He, Chu, & Zhong, 2002).

The implementation of these methods can be summed in the following three steps.

1. Construct a library W of wavelets.
2. Remove the wavelets that their support does not contain any sample points of the training data.
3. Rank the remaining wavelets and select the best wavelet regressors.

In the first step, the wavelet library can be constructed either by an orthogonal wavelet or a wavelet frame, (He et al., 2002; Postalcioglu & Becerikli, 2007). By determining an orthogonal wavelet basis the WN is simultaneously constructed. However, in order to generate an orthogonal wavelet basis, the wavelet function has to satisfy strong restrictions, (Daubechies, 1992; Mallat, 1999). In addition the fact that orthogonal wavelets cannot be expressed in closed form constitutes them inappropriate for applications of function approximation or process modeling, (Oussar & Dreyfus, 2000).

On the other hand constructing wavelet frames is very easy and can be done by translating and dilating the selected mother wavelet. The results from (Gao & Tsoukalas, 2001) indicate that a family of compactly supported non-orthogonal wavelets is more appropriate for function approximation. Due to the fact that a wavelet family can contain a large number of wavelets, it is more convenient to use a truncated wavelet family than an orthogonal wavelet basis, (Zhang, 1993).

However, constructing a WN using wavelet frames is not a straightforward process. The wavelet library may contain a large number of wavelets since only the input data were considered in the construction of the wavelet frame. In order to construct a WN the "best" wavelets must be selected. However, arbitrary truncations may lead to large errors, (Xu & Ho, 2005). In the second step, (Zhang, 1993) proposes to remove the wavelets that have very few training patterns in their support. Alternatively, in Cannon and Slotine (1995) magnitude based methods were used to eliminate wavelets with small coefficients.

In the third step, the remaining wavelets are ranked and the wavelets with the highest rank are used for the construction of the WN.

In Zhang (1994) and Zhang (1997) three alternative methods were proposed in order to reduce and rank the wavelets in the wavelet library: Residual Based Selection (RBS), Stepwise Selection by Orthogonalization (SSO) and Backward Elimination (BE).

In the framework of RBS, first the wavelet that best fits the output data is selected. Then the wavelet that best fits the residual of the fitting of the previous stage is repeatedly selected. RBS is considered as a very simple method but not an effective one, (Juditsky, Zhang, Delyon, Glorennec, & Benveniste, 1994). However if the wavelet candidates reach a very large number, computational efficiency is essential and the RBS method may be used, (Juditsky et al., 1994). In Kan and Wong (1998) and Wong and Leung (1998) the RBS algorithm was used for the synthesis of a WN. In Xu and Ho (2002) a modified version of the RBS algorithm was used. More precisely an Orthogonalized Residual Based Selection (ORBS) algorithm is proposed for the initialization of the WN. The ORBS method combines both the RBS and the Orthogonalized Least Squares (OLS) method. In this way high efficiency is obtained while relatively low computational burden is maintained.

The SSO method is an extension of the RBS first proposed by Chen, Billings, and Luo (1989), Chen, Cowan, and Grant (1991). In order to initialize the WN the following procedure is followed: First the wavelet which best fits the output data is selected. Then the wavelet that best fits the residual of the fitting of the previous stage together with the previous selected wavelet is repeatedly selected. In other words the SSO considers the interaction or the non-orthogonality of the wavelets. The selection of the wavelets is performed using the modified Gram–Schmidt algorithm that has better numerical properties and is computationally less expensive than the ordinary Gram–Schmidt algorithm, (Zhang, 1997). SSO is considered to have good efficiency while it is not computationally expensive. In Oussar and Dreyfus (2000) an algorithm similar to SSO was proposed.

In contrast to previous methods, the BE starts the regression by selecting all the available wavelets from the wavelet library. Then the wavelet that contributes the least in the fitting of the training data is repeatedly eliminated. The drawback of BE is that it is computationally expensive but it is considered to have good efficiency.

In Zhang (1997) the exact number of arithmetic operations of each algorithm are presented. More precisely, for the RSO at each step i the computational cost is $2nL - 2in + 6n - 1$ where n is length of the training samples, L is the number of the wavelets in the wavelet basis. Similarly, the computational cost of the SSO algorithm at each step is $8nL - 6in + 9n + 5L$. Roughly speaking, the SSO is 4 times more computational expensive than the RSO algorithm. Finally, the operations needed in the BE method is $2n(L-i) + 4\left(L^2 + i^2\right) - 8iL - (L-i)$. In addition at the beginning of the BE algorithm an $L \times L$ matrix must be inverted. If the number of HUs and as a result the number of the wavelets that must be selected is $HUs > L/2$ then less steps are performed by the BE algorithm while in the case of $HUs < L/2$ the contrary is true, (Zhang, 1997).

All methods described above are used just for the initialization of the dilation and translation parameters. Then the network is further trained in order to obtain the vector of the parameters $w = \hat{\mathbf{w}}_n$ which minimizes the cost function.

It is clear that additional computational burden is added in order to initialize efficiently the WN. However the efficient initialization significantly reduces the training phase hence the total amount of computations is significantly smaller than in a network with random initialization.

2.3. *Training a wavelet network with back-propagation*

After the initialization phase, the network is further trained in order to find the weights which minimize the cost function.

In Cristea et al. (2000) genetic algorithms were used to train a WN while in Li and Chen (2002) a learning algorithm by applying least trimmed squares was proposed. He et al. (2002) suggest an hierarchical evolutionary algorithm. In Xu and Ho (2005) the Levenberg–Marquardt algorithm was applied. Chen et al. (2006) combine an adaptive diversity learning particle swarm optimization and gradient descent algorithms in order to train a WN. However, most evolutionary algorithms including particle swarm optimization, are inefficient and cannot avoid certain degeneracy and local minimum completely, (Zhang, 2009). Also evolutionary algorithms suffer from fine-tuning inefficiency, (Chen et al., 2006; Yao, 1999). On the other hand the Levenberg–Marquardt is one of the fastest algorithms for training NNs. The main drawback of this algorithm is that it requires the storage and the inversion of some matrices that can be quite large.

The above algorithms originate from classical sigmoid NNs, as they do not take advantage of the properties of wavelets, (Zhang, 2007, 2009). Since a wavelet is a function whose energy is well localized in time–frequency, (Zhang, 2007, 2009) use sampling theory in order to train a WN in both uniform and non-uniform data. Their results indicate that their proposed algorithm has global convergence.

In our implementation the ordinary back-propagation (BP) was used. BP is probably the most popular algorithm used for training WNs, (Fang & Chow, 2006; Jiao et al., 2001; Oussar & Dreyfus, 2000; Oussar et al., 1998; Postalcioglu & Becerikli, 2007; Zhang, 1997; Zhang & Benveniste, 1992; Zhang, 2007). Ordinary BP is less fast but also less prone to sensitivity to initial conditions than higher order alternatives, (Zapranis & Refenes, 1999).

The basic idea of BP is to find the percentage of contribution of each weight to the error. The error e_p for pattern p is simply the difference between the target (y_p) and the network output $(\hat{y}_p)$. By squaring and multiplying by 1/2 we take the pairwise error E_p which is used in network training:

$$E_p = \frac{1}{2}\left(y_p - \hat{y}_p\right)^2 = \frac{1}{2}e_p^2. \tag{9}$$

The weights of the network were trained to minimize the mean quadratic cost function (or loss function):

$$L_n = \frac{1}{n}\sum_{p=1}^{n} E_p = \frac{1}{2n}\sum_{p=1}^{n} e_p^2 = \frac{1}{2n}\sum_{p=1}^{n}\left(y_p - \hat{y}_p\right)^2. \tag{10}$$

Other functions can be used instead of (10) however the mean quadratic cost function is the most commonly used. The network is trained until a vector of weights $w = \hat{\mathbf{w}}_n$ that minimizes the proposed cost function is found. The previous solution corresponds to a training sample of size n. Computing the parameter vector $\hat{\mathbf{w}}_n$ is always done by iterative methods. At each iteration t the derivative of the loss function with respect to the network weights is calculated. Then, the updating of the parameters is performed by the following (delta) learning rule:

$$w_{t+1} = w_t - \eta\frac{\partial L_n}{\partial w_t} + \kappa\left(w_t - w_{t-1}\right) \tag{11}$$

where η is the learning rate and it is constant. The complete vector of the network parameters comprises:

$$w = \left(w_i^{[0]}, w_{(\xi)ij}^{[1]}, w_{(\zeta)ij}^{[1]}, w_j^{[2]}, w_{\lambda+1}^{[2]}\right).$$

A constant momentum term, defined by κ, is induced which increases the training speed and helps the algorithm to avoid oscillations. The learning rate and momentum speed take values between 0 and 1. The choice of the learning rate and the momentum depends on the application and the training sample. Usually, values between 0.1 and 0.4 are used.

The partial derivative of the cost function with respect to a weight w is given by:

$$\begin{aligned}\frac{\partial L}{\partial w} &= \frac{1}{2n}\sum_{p=1}^{n}\frac{\partial E_p}{\partial w} = \frac{1}{2n}\sum_{p=1}^{n}\frac{\partial E_p}{\partial \hat{y}_p}\frac{\partial \hat{y}_p}{\partial w}\\ &= \frac{1}{n}\sum_{p=1}^{n} -\left(y_p - \hat{y}_p\right)\frac{\partial \hat{y}_p}{\partial w} = \frac{1}{n}\sum_{p=1}^{n} -e_p\frac{\partial \hat{y}_p}{\partial w}.\end{aligned} \tag{12}$$

The partial derivatives with respect to each parameter, $\frac{\partial \hat{y}_p}{\partial w}$, and with respect to each input variable, $\frac{\partial \hat{y}_p}{\partial x_i}$, are presented in Appendix.

The above training methodology falls to the category of off-line training. This means that the weights of the networks are updated after all training patterns are presented to the network. Alternatively, one can use on-line training methods. In on-line methods the weights are changed after each presentation of a training pattern. For some problems, this method may yield effective results, especially for problems where data arrives in real time, (Samarasinghe, 2006). Using on-line training it is possible to reduce training times significantly. However, for complex problems it is possible, that on-line training, to create a series of drawbacks. First, there is a possibility that the training will stop before the presentation of each training pattern to the network. Second, by changing the weights after each pattern, they could bounce back and forth with each iteration, possibly resulting in a substantial amount of wasted time, (Samarasinghe, 2006). Hence, in order to ensure the stability of the algorithms, offline training is used in this study.

2.4. Stopping conditions for training

After the initialization phase of the network parameters w, the weights $w_i^{[0]}$, $w_j^{[2]}$ and parameters $w_{(\xi)ij}^{[1]}$ and $w_{(\zeta)ij}^{[1]}$ are trained during the learning phase for approximating the target function. A key decision related to the training of a WN is when the weight adjustment should end. Under the assumption that the WN contains the number of wavelets that minimizes the prediction risk the training is stopped when one of the following criteria is met — the cost function reaches a fixed lower bound or the variations of the gradient or the variations of the parameters reaches a lower bound or the number of iterations reaches a fixed maximum, whichever is satisfied first. In our implementation the fixed lower bound of the cost function, of the variations of the gradient and of the variations of the parameters were set to 10^{-5}.

2.5. Evaluating the initialization methods

As it was mentioned in the previous section the initialization phase is a very important on the construction and training of a WN. In this section we compare four different initialization methods. The heuristic, the SSO, the RBS and the BE methods, that constitute the bases for alternative algorithms and can be used with the BP training algorithm, will be tested.

The four initialization methods will be compared in three stages. First the distance between the initialization and the underlying function as well as the training data will be measured. Second the number of iterations needed to train the WN will be compared. Finally, the difference of the final approximation of the trained WN and the underlying function and the training data will be examined. The four initialization methods will be tested in two cases. First on a simple underlying function and second on a more complex function that incorporates large outliers.

2.5.1. Example 1

In the first case the underlying function $f(x)$ is given by:

$$f(x) = 0.5 + 0.4\sin(2\pi x) + \varepsilon_1(x)x \in [0, 1] \tag{13}$$

where x is equally spaced in [0, 1] and the noise $\varepsilon_1(x)$ follows a normal distribution with mean zero and a decreasing variance:

$$\sigma_\varepsilon^2(x) = 0.05^2 + 0.1(1 - x^2). \tag{14}$$

The four initialization methods will be examined using a WN with 2 HUs with learning rate 0.1 and momentum 0. The choice of the proposed structure of network will be justified in the next section. The training sample consists of 1.000 patterns.

Fig. 2 shows the initialization of the four algorithms for the first training sample. It is clear that the heuristic algorithm produces the worst initialization. However, even the heuristic approximation is still better than a random initialization. On the other hand the initialization of the RBS algorithm gives a better approximation of

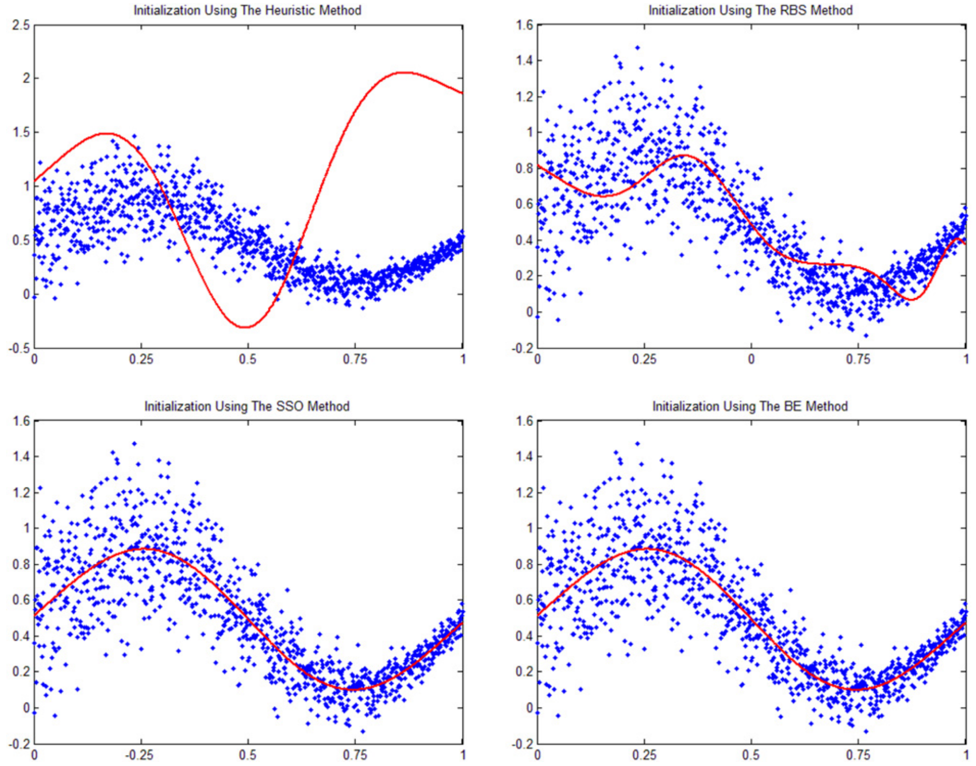


Fig. 2. Four different initialization methods of the first case.

the data however the approximation of the target function $f(x)$ is still not very good. Finally, both the SSO and the BE algorithms start very close to the target function $f(x)$.

The Mean Square Error (MSE) between the initialization of the network and the training data confirms the above results. More precisely the MSE between the initialization of the network and the training data is 0.630809, 0.040453, 0.031331 and 0.031331 for the heuristic, the RBS, the SSO and the BE respectively. Next we will test how close the initialization is to the underlying function. The MSE between the initialization of the network and the underlying function is 0.59868, 0.302782, 0.000121 and 0.000121 for the heuristic, the RBS, the SSO and the BE respectively. The results above indicate that both the SSO and the BE produce the best initialization for the parameters of the WN.

Another way to compare the initialization methods is to compare the number of iterations needed in the training phase until the solution $\hat{\mathbf{w}}_n$ is found. Also if the proposed initialization methods allow the training procedure to find the global minimum of the loss function will be examined.

First the heuristic method was used to train 100 networks with different initial conditions of the direct connections $w_i^{[0]}$ and weights $w_j^{[2]}$. Training 100 networks with perturbed initial conditions is expected to be sufficient to avoid any possible local minimums of the loss function (10). It was found that the smallest MSE between the target function $f(x)$ and the final approximation of the WN was 0.031331.

Using the RBS the training phase stopped after 617 iterations. The overall fit was very good and the MSE between the network output and the training data was 0.031401 indicating that the network was stopped before the minimum of the loss function was achieved. Finally, the MSE between the network output and the target function was 0.000676.

On the other hand, when initializing the WN with the SSO algorithm only 1 iteration was needed in the training phase and the MSE was 0.031331 while the MSE between the underlying function $f(x)$ and the network approximation was only 0.000121. The same results were achieved by the BE method. Finally, one implementation of the heuristic method needed 1501 iterations. All results are presented in Table 1.

Table 1
Initialization of the four methods.

Case 1	Heuristic	RBS	SSO	BE
MSE	0.031522	0.031401	0.031331	0.031331
MSE+	0.000791	0.000626	0.000121	0.000121
IMSE	0.630807	0.040453	0.031331	0.031331
IMSE+	0.598680	0.302782	0.000121	0.000121
Iterations	1501	617	1	1
Case 2				
MSE	0.106238	0.004730	0.004752	0.004364
MSE+	0.102569	0.000558	0.000490	0.000074
IMSE	7.877472	0.041256	0.012813	0.008304
IMSE+	7.872084	0.037844	0.008394	0.004015
Iterations	4433	3097	741	1107

Initialization criteria of the four methods for the two cases. Case 1 refers to function $f(x)$, case 2 to function $g(x)$ and case 3 to the Mackey–Glass equation.
RBS = Residual Based Selection SSO = Stepwise Selection by Orthogonalization BE = Backward Elimination MSE = MSE between the training data and the network approximation
MSE+ = MSE between the underlying function and the network approximation
IMSE = MSE between the training data and the network initialization
IMSE+ = MSE between the underlying function and the network initialization.

The results above indicate that the SSO and the BE algorithms give the same results and significantly outperform both the heuristic and the RBS algorithms. Moreover the above results indicate that having a very good initialization not only significantly reduces the needed training iterations and as a result the total needed training time but also a vector of weights $\hat{\mathbf{w}}_n$ that minimizes the loss function can be found.

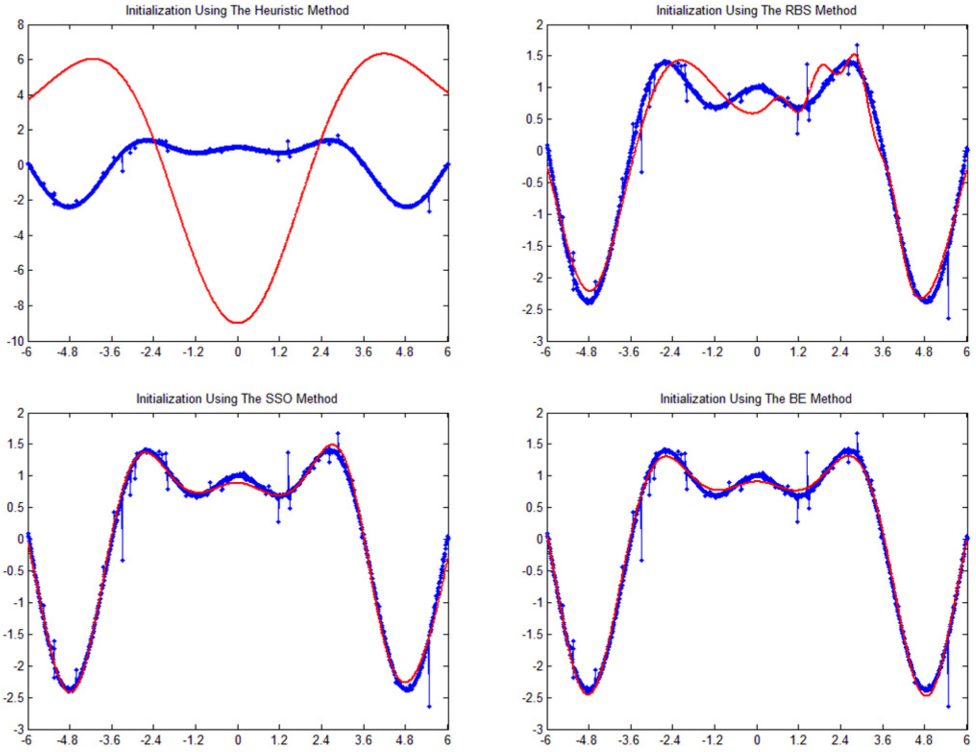


Fig. 3. Four initialization methods for the second case.

2.5.2. Example 2

Next a more complex case is introduced where the function $g(x)$ is given by:

$$g(x) = 0.5x\sin(x) + \cos^2(x) + \varepsilon_2(x) \quad x \in [-6, 6] \tag{15}$$

and $\varepsilon_2(x)$ follows a Cauchy distribution with location 0 and scale 0.05 and x is equally spaced in $[-6, 6]$. The training sample consists of 1.000 training patterns. While the first function is very simple the second one, proposed by Li and Chen (2002), incorporates large outliers in the output space. The sensitivity to the presence of outliers of the proposed WN will be tested. To approximate function $g(x)$ a WN with 8 HUs with learning rate 0.1 and momentum 0 is used. The choice of the proposed topology of the WN will be justified in the next section.

The results obtained in the second case are similar. A closer inspection of Fig. 3 reveals that the heuristic algorithm produces the worst initialization in approximating the underlying function $g(x)$. The RBS algorithm produces a significantly better initialization than the heuristic method however the initial approximation still differs from the training target values. Finally, both the BE and the SSO algorithms produce a very good initialization. It is clear that the first approximation of the WN is very close to the underlying function $g(x)$.

The MSE between the initialization of the network and the training data was 7.87472, 0.041256, 0.012813 and 0.008304 for the heuristic, the RBS, the SSO and the BE algorithms respectively. Also the MSE between the initialization of the network and the underlying function $g(x)$ was 7.872084, 0.037844, 0.008394 and 0.004015 for the heuristic, the RBS, the SSO and the BE respectively. The previous results indicate that the training phase using the BE algorithm starts very close to the target function $g(x)$.

Next the number of iterations needed in the training phase of each method was compared. Also, if the proposed initialization methods allow the training procedure to find the global minimum of the loss function was examined. The RBS algorithm stopped after 3097 iterations and the MSE of the final approximation of the WN and the training patterns was 0.004730. The MSE between the underlying function $f(x)$ and the network approximation was 0.000558. When initializing the WN with the SSO algorithm only 741 iterations were needed in the training phase and the MSE was 0.004752 while the MSE between the underlying function $g(x)$ and the network approximation was 0.000490. The BE needed 1107 iterations in the training phase and the MSE was 0.004364 while the MSE between the underlying function $g(x)$ and the network approximation was only 0.000074. Finally, one implementation of the heuristic method needed 4433 iterations and the MSE was 0.106238 while the MSE between the underlying function $g(x)$ and the network approximation was 0.102569. All results are presented in the second part of Table 1. In the second case the BE was slower than the SSO however the final approximation was significantly closer to the target function than any other method.

The previous examples indicate that SSO and BE perform similarly and outperform the other two methods whereas BE outperforms SSO in complex problems. Previous studies suggest that the BE is more efficient than the SSO algorithm however it is more computationally expensive. On the other hand in the BE algorithm the calculation of the inverse of the wavelet matrix is needed whose columns might be linear dependent, (Zhang, 1997). In that case the SSO must be used. However since the wavelets come from a wavelet frame this is very rare to happen, (Zhang, 1997).

3. Model selection

In this section we describe the model selection procedure. One of the most crucial steps is to identify the correct topology of the network. A desired WN architecture should contain as few HUs as necessary while at the same time it should explain as much variability of the training data as possible. A network with less HUs than needed would not be able to learn the underlying function while selecting more HUs than needed will result to an over-fitted model. Therefore, an algorithm to select the appropriate WN model for a given problem is necessary to be derived.

The usual approaches proposed in the literature are the early stopping, regularization and pruning. However all these methods have serious drawbacks. In early stopping method a more complex model than needed is used. Hence, a large number of weights must be trained. As a result large training times are expected. Moreover, the network incorporates a large number of connections most of them with small weights. In addition, a validation sample should be used, however, usually there is only a small amount of data available and splitting the data is not useful. Furthermore, growing validation errors indicate the reduction of network's complexity, (Anders & Korn, 1999). Finally, the solution $\hat{\mathbf{w}}_n$ of the network is highly dependent on the dividing of the data and the initial conditions, (Dimopoulos, Bourret, & Lek, 1995).

In regularization the penalty terms usually are chosen arbitrary without any theoretical justification, (Anders & Korn, 1999). Moreover a bad regularization parameter, δ, can severely restrict the growth of weights and as result the network will be under-fitted, (Samarasinghe, 2006). Finally in pruning methods the significance of each weight usually is not measured in a statistical way, (Anders & Korn, 1999). (Reed, 1993) presents an extensive survey on pruning methods. One of the disadvantages of pruning methods is that most of them do not take into account correlated weights. Two weights that cancel out each other do not have any effect at the output of the network however each weight may have a large effect, (Reed, 1993). Also the time when the pruning should stop is usually arbitrary, (Reed, 1993).

In contrast to previous constructive methods, on-line approaches do not require to determine the number of wavelets before the start of the training, (Wong & Leung, 1998). On-line training and synthesis methods allow the parameters to be updated after the presentation of each training pattern. New wavelets are added to the network when it is needed while wavelets that do not contribute to the performance of the network anymore are removed. In Cannon and Slotine (1995) and Wong and Leung (1998) online synthesis in the construction of the WN was used. Similarly, in Xu and Ho (1999) a WN proposed and introduced for adaptive nonlinear system identification In Xu and Ho (1999) the basis functions were selected on-line according to the local spatial frequency content of the approximated function. However the results from (Wong & Leung, 1998) indicate that this method is very prone to the initialization of the WN. Their results indicate that the suggested topology of a particular function approximation was varying from 4 to 10 HUs.

The previous methods do not use an optimal architecture of a WN. A very large WN is used and then various methods were developed to avoid over-fitting. Smaller networks usually are faster to train and need less computational power to build, (Reed, 1993).

Alternative the Minimum Prediction Risk (MPR) principle can be applied, (Efron & Tibshirani, 1993; Zapranis & Refenes, 1999). The idea behind MPR is to estimate the out-of-sample performance of incrementally growing networks. Assuming that the explanatory variables $\mathbf{x}$ were correctly selected and remain fixed, then the model selection procedure is the following: the procedure starts with a fully connected network with 0 HUs. The WN is trained and then the prediction risk is estimated. Then, iteratively a new HU is added to the network. The new WNs are trained and the new prediction risk is estimated at each step. The number of HUs that minimizes the prediction risk is the appropriate number of HUs that should be used for the construction of the WN.

The prediction risk measures the generalization ability of the network. More precisely, the prediction risk of a network $g_\lambda(\mathbf{x}; \hat{\mathbf{w}}_n)$ is the expected performance of the network on new data that were not introduced during the training phase and is given by:

$$P_\lambda = E\left[\frac{1}{n}\sum_{p=1}^{n}\left(y_p^* - \hat{y}_p^*\right)^2\right] \tag{16}$$

where $\left(\mathbf{x}_p^*, y_p^*\right)$ are the new observations that have not been used in the construction of the network $g_\lambda(\mathbf{x}; \hat{\mathbf{w}}_n)$ and $\hat{y}_p^*$ is the network output using the new observations, $g_\lambda(\mathbf{x}^*; \mathbf{w})$.

However finding a statistical measure that estimates the prediction risk is not a straightforward procedure. Since there is a linear relationship between the wavelons and the output of the WN, (Zhang, 1993, 1994, 1997; Zhang & Benveniste, 1992) propose the use of information criteria previously widely applied in linear models. More precisely, (Zhang, 1994) suggested that the Akaike's Final Prediction Error (FPE) can be used in various applications. More recently, (Zhang, 1997) suggested that the Generalized Cross-Validation (GCV) is an accurate tool for selecting the number of wavelets that constitutes the WN topology. In order to estimate the GCV the noise variance must be identified. In practice the noise variance σ^2 is not known. In that case it has to be estimated. An estimate is given by the MSE between the network output and the target data, (Zhang, 1997).

Because we do not have an *a priori* knowledge of the correct number of HUs or parameters of the WN we estimate the above criteria iteratively. The computational cost of these algorithms can be expressed as a function of WNs to be trained. For example, to estimate the prediction risk using the FPE or the GCV from 1 to 5 HU, 5 WNs must be trained.

The criteria described above for the estimation of the prediction risk are derived from linear models. Usually these models are based on assumptions that are not necessarily true in the framework of nonlinear nonparametric estimation. The hypothesis behind these information criteria is the asymptotic normality of the maximum likelihood estimators hence the information criteria are not theoretically justified for over-parameterized networks, (Anders & Korn, 1999).

Moreover, in fitting problems more complex than the least squares the number of parameters k is not known, (Efron & Tibshirani, 1993) and it is unclear how to compute the degrees of freedom, (Curry & Morgan, 2006), or the effective number of parameters described in Moody (1992).

In Zapranis and Refenes (1999) a different approach is presented. An analytical form of the prediction risk (16) was presented for the sigmoid NNs. However, the assumptions made by Zapranis and Refenes (1999) are not necessarily true in the framework of WNs and analytical forms are not available for estimating the prediction risk for WNs. Alternatively the use of sampling methods such as bootstrap and cross-validation can be employed since they do not depend on any assumptions regarding the model, (Efron & Tibshirani, 1993). The only assumption made by sampling methods is that the data are a sequence of independent and identically distributed variables. Another advantage of bootstrap and cross-validation is their robustness. In contrast to sampling methods both GCV and FPE require a roughly correct model to obtain the estimate of the noise variance.

The bootstrap and the ν-fold cross-validation approaches are analytically described in Efron and Tibshirani (1993). It is known that the simple estimation of the bootstrap approach is not very accurate, (Efron & Tibshirani, 1993). Hence, we estimate the improved estimation of the prediction risk following the suggestion of (Efron & Tibshirani, 1993). The number of new samples B is usually over 30, (Aczel, 1993; Efron & Tibshirani, 1993). It is clear that as the number of new samples B increases the bootstrap method becomes more accurate but also more computationally expensive. Cross-validation is an another standard tool for estimating the prediction error that makes an efficient use of the available information, (Efron & Tibshirani, 1993). The ν-*fold* cross-validation is applied as described in Efron and Tibshirani (1993). Again, it is expected that as the ν increases and the new samples B increase, the accuracy of the ν-*fold*cross-validation to increase but the method becomes computationally expensive. More precisely, the computational burden of the bootstrap and the ν-*fold*cross-validation methods are B times the computational cost of the FPE and GCV methods.

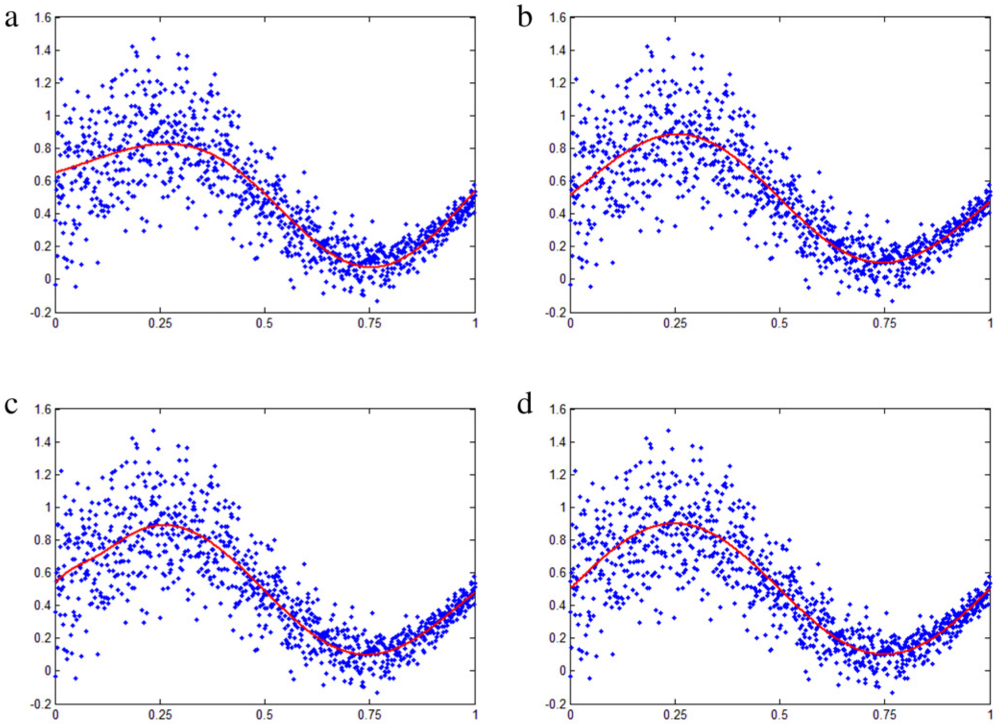


Fig. 4. Training a wavelet network with 1 (part a), 2 (part b) and 3 (part c) hidden units. In part (d) the target function is presented.

3.1. *Evaluating the model selection algorithm*

In order to find an algorithm that will work well with WNs and will lead to a good estimation of prediction risk we will compare, in this section, the various criteria as well as the sampling techniques discussed earlier.

More precisely, in this study we will compare the sampling techniques that are extensively used in various studies with sigmoid NNs and two information criteria previously proposed in the construction of a WN. More precisely, the FPE proposed by Zhang (1994), the GCV proposed by Zhang (1997), the bootstrap (BS) and the *v-fold* cross-validation (CV) methods proposed by Efron and Tibshirani (1993) and Zapranis and Refenes (1999) will be tested.

In order to evaluate each method the following procedure will be followed. First, the prediction risk according to each method will be estimated for a large number of HUs. Then the number of HUs that minimizes the prediction risk will be selected for the construction of the WN. The WN will be fully trained. Finally the MSE between the WN output and the target function will be estimated. The best network topology will be considered the one that produces the smallest MSE and shows no signs of over-fitting.

The four methods are evaluated using the functions $f(x)$ and $g(x)$ given by (13) and (15) respectively. Both training samples consist of 1.000 training patterns as in to the previous section. The WNs are trained with the BP algorithm with learning rate 0.1 and zero momentum. In order to estimate the prediction risk using the BS approach 50 new networks were created for each HU ($B = 50$). Similarly, the prediction risk using the CV method was estimated using 50 subsamples for each HU. In other words the 50-fold cross validation was used, ($\upsilon = 50$). All WNs were initialized using the BE algorithm since our results in the previous sections indicate that the BE outperforms the alternative algorithms.

3.1.1. *Example 1*

Table 2 presents the prediction risk and the suggested HUs for each information criterion for the two functions described previously. In the first case we estimate the prediction risk for a WN with zero HUs and iteratively one HU is added until a maximum number of 15 HUs. Three of the four criteria, the FPE the BS and the CV suggest that a WN with only 2 HUs is sufficient to model function $f(x)$. On the other hand, using the GCV, the prediction risk is minimized when a WN with 3 HUs is used. Fig. 4 shows the approximation of the WN to the training data using (a) 1 HU (b) 2 HUs and (c) 3 HUs. Part (d) of Fig. 4 shows the training data and the target function $f(x)$. It is clear that a WN with only 1 HU cannot learn the underlying function. On the other hand the WNs with 2 and 3 HUs approximate the underlying function very well. However when 3 HUs are used the network approximation is affected by the large variation of the noise in the interval [0, 0.25]. In order to confirm the above results the MSE between the output of the WN and the underlying target function $f(x)$ is estimated. The MSE is 0.001825 when a WN with only one HU is used. Adding one more HU, two in total, the MSE is reduced to only 0.000121. Finally, when 3 HUs are used the MSE increased to 0.000267. Hence, 2 wavelets should be used to construct a WN to approximate function $f(x)$. The results above indicate that the GCV suggested a more complex model than needed. Moreover a WN with 3 HUs shows signs of over-fitting.

Table 2
Prediction risk and hidden units for the four information criteria.

Case 1	FPE	GCV	BS	CV
Prediction Risk Hidden Units	0.02088 2	0.03966 3	0.04002 2	0.03991 2
Case 2				
Prediction Risk Hidden Units	0.00041 7	0.00077 14	0.00081 8	0.00078 8

Information criteria for the two cases. Case 1 refers to function $f(x)$, case 2 to function $g(x)$ and case 3 to the Mackey–Glass equation.
FPE = Final Prediction Error GCV = Generalized Cross-validation
BS = Bootstrap CV = 50-fold Cross-validation.

From Table 2 it is shown that the FPE criterion suggests 2 HUs however the prediction risk is only 0.02088 in contrast to GCV,

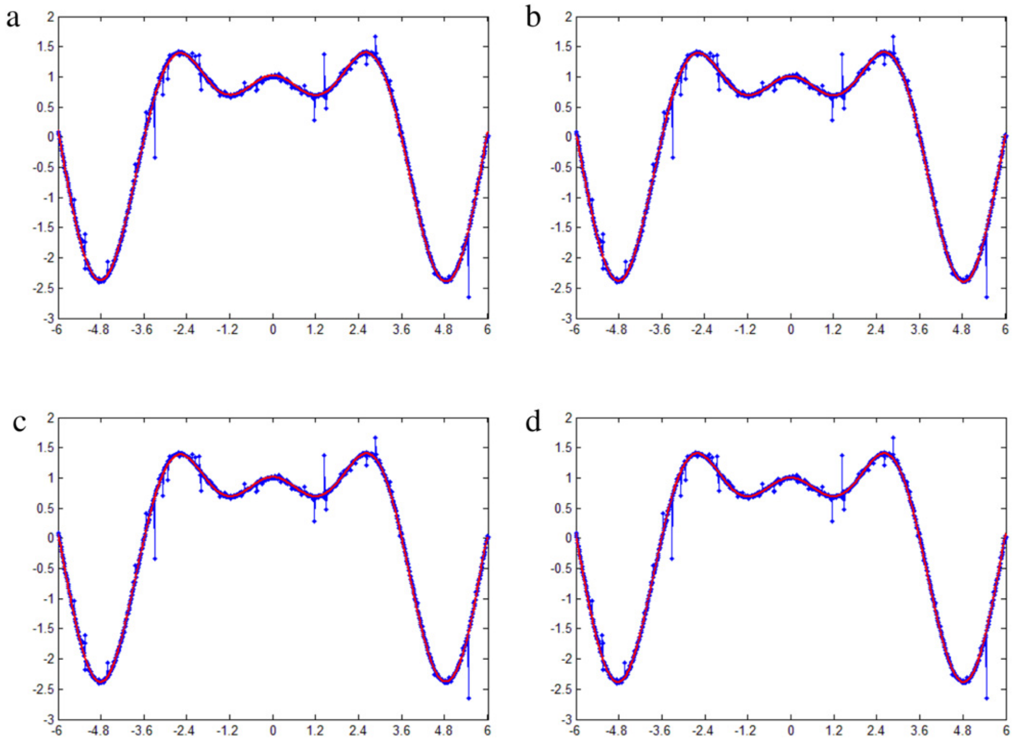


Fig. 5. Training a wavelet network with 7 (part a), 8 (part b) and 14 (part c) hidden units. In part (d) the target function is presented.

BS and CV which is 0.03966, 0.04002 and 0.03991 respectively. In order to find the correct magnitude of the prediction risk a validation sample is used to measure the performance of the WN with 2 HUs in out-of-sample data. The validation sample consists of 300 patterns randomly generated by (13). These patterns were not used for the training of the WN. The MSE between the network forecasts and the validation targets is 0.048751 indicating that the FPE criterion is too optimistic on the estimation of the prediction risk.

3.1.2. *Example 2*

In the second part of Table 2 the results for the second example are presented. As in the first case, the prediction risk for a WN with zero HUs is estimated and iteratively one HUs is added to the WN until a maximum number of 15 HUs is reached. The FPE criterion suggests that 7 HUs is appropriate for modeling the function $g(x)$. On the other hand, using the GCV, the prediction risk is minimized when a WN with 14 HUs is used. Finally using the BS and the CV criteria the prediction risk is minimized when a WN with 8 HUs is used. In Fig. 5 the approximation of the WN to the training data using (a) 7, (b) 8 and (c) 14 HUs is presented. Part (d) of Fig. 5 shows the target function $g(x)$ and the training data. It is clear that all networks produce similar results. In order to compare the above results, the MSE between the output of the WN and the underlying target function $g(x)$ was estimated. The MSE is 0.000239 when a WN with only 7 HUs is used. Adding one more HU, 8 in total, the MSE is reduced to only 0.000074. Finally, when 14 HUs are used the MSE increased to 0.000154. Hence, the optimum number of wavelet to approximate function $g(x)$ is 8. The results above indicate that the GCV suggests a more complex model while FPE suggest a simpler model than needed. Our results indicate that the sampling techniques outperform the information criteria again.

As reported in Table 2, the estimated prediction risk proposed by the FPE criterion is 0.00041 in contrast to GCV, BS and CV which is 0.00077, 0.00081 and 0.00078 respectively. In order to find the correct magnitude of the prediction risk a validation sample is used to measure the performance of the WN with 8 HUs in out-of-sample data. The validation sample consists of 300 patterns randomly generated by (15). These patterns were not used for the training of the WN. Our results indicate again that the FPE criterion is too optimistic on the estimation of the prediction risk.

A closer inspection of Fig. 5 reveals that the WN approximation was not affected by the presence of large outliers in contrast to the findings of Li and Chen (2002). In this study 8 HUs were used to construct the WN as it was proposed by ν-fold cross-validation and the BS while in Li and Chen (2002) the architecture of the WN had 10 HUs as it was proposed by the FPE criterion. Our results indicate that the FPE criterion does not perform as well as sampling techniques (bootstrap or ν-fold cross-validation).

3.1.3. *Model selection without training*

In Zhang (1997) the estimation of the preferred information criteria is performed after the initialization stage of the network. More precisely in the SSO and RBS the preferred information criteria is evaluated after the selection of each wavelet in the initialization stage. Similarly, when the BE algorithm is used, the preferred information criteria is evaluated after the elimination of each wavelet in the initialization stage. Since the initialization of the WN is very good, as presented in the previous section, the initial approximation is expected to be very close to the target function. Hence, a good approximation of the prediction risk is expected to be obtained. The same idea can also be applied when the BS or the CV are used. The above procedure is significantly less computational expensive.

However, the above procedure is similar to early stopping techniques. Usually early stopping techniques suggest a network with more HUs than necessary, though the network is not fully trained to avoid over-fitting, Samarasinghe (2006), while they do not work satisfactorily in complex problems, Samarasinghe (2006).

In the first case the results were similar to the case where the WNs were fully trained. More precisely, the FPE, the BS and the CV methods suggested that a WN with 2 HUs is sufficient to model $f(x)$ while GCV suggested a WN with 3 HUs. In the second case both the information criteria and the sampling techniques suggested that a WN with more than 14 HUs is needed to model function $g(x)$. The results above indicate that when more complex problems are introduced, as in the second case, this method does not work satisfactorily.

Since sampling techniques are computationally expensive methods, the FPE criterion can be used initially. Then the BS or the CV methods can be used in $+/-5$ HU around the HUs proposed by FPE in order to define the best network topology.

4. Variable selection

In real problems it is important to determine correctly the independent variables. In most problems there is a little information about the relationship of any explanatory variable with the dependent variable. As a result unnecessary independent variables are included in the model reducing its predictive power. In this section various methods for testing the significance of each explanatory variable will be presented and tested. The purpose of this section is to find an algorithm that constantly gives stable and correct results when it is used with WNs.

In linear models in order to determine if a coefficient, and as a result an input variable, is significant the t-stats or the p-values of each coefficient are examined. Applying the previous method in WNs is not a straightforward process since the coefficients (weights) are estimated iteratively and each variable contribute to the output of the WN linearly through the direct connections and nonlinearly through the HUs.

Instead of removing the irrelevant variables one can reduce the dimensionality of the input space. An effective procedure for performing this operation is the PCA. PCA has many advantages and has been applied in many application with great success, Khayamian et al. (2005). In applications where WNs are used for prediction of future values of a target variable PCA can be proved very useful. On the other hand in applications where WNs are used for function approximation or sensitivity analysis PCA can be proved cumbersome. The main disadvantage of PCA is that the principal components are usually a combination of all the available variables. Hence, it is often very difficult to distinguish which variable is important and which is not statistical significant. In addition, extra care must be taken when linking the information resulted from principal components to the original variables.

PCA cannot always be used since a linear transformation among the explanatory variables is not always able to reduce the dimension of the dataset. Another disadvantage of the PCA is the fact that the directions maximizing variance do not always maximize information. Finally, PCA is an orthogonal linear transformation however the use of a WN implies a nonlinear relationship between the explanatory variables and the dependent variable.

In Wei, Billings, and Liu (2004) a novel approach for term and variable selection is presented. This method applies locally linear models together with orthogonal least squares in order determine which of the input variables are significant. This algorithm ranks the variables and determines the amount of the system output variance that can be explained by each term. The method assumes that non-linearities in the system are relatively smooth. Then local linear models are fitted in each interval. However the number of the locally linear models increases exponentially as the number of intervals for each independent variable is increased, Wei et al. (2004). Also, selecting the optimal operating regions for the locally piecewise linear models usually is computational expensive, Wei et al. (2004).

A similar approach is presented in Wei and Billings (2007) based on feature subset selection. In feature selection an optimal or suboptimal subset of the original features is selected, Mitra, Murthy, and Pal (2002). More precisely, in Wei and Billings (2007) a forward orthogonal search algorithm by maximizing the overall dependency is presented in order to detect the significant variables. This algorithm can produce efficient subsets with a direct link back to the underlying system. The proposed method assumes a linear relationship between sample features. However, this is assumption is not always true and will lead to a wider subset of explanatory variables.

Alternatively one can quantify the average effect of each input variable, x_j, on the output variable, y. Estimating the sensitivity of the WN output according to small input perturbations of variable x_j can be done either by applying the average derivative (AvgD) or the average elasticity (AvgL) where the effect is presented as a percentage and are given by the following equations:

$$\mathrm{AvgD}(x_j) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial \hat{y}}{\partial x_{ij}} \tag{17}$$

$$\mathrm{AvgL}(x_j) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial \hat{y}}{\partial x_{ij}}\right)\left(\frac{x_{ij}}{\hat{y}}\right). \tag{18}$$

Although AvgL conveys more information, in both criteria cancellations between negative and positive values are possible. A natural extension of the above criteria is the average derivative magnitude (AvgDM) and the average elasticity magnitude (AvgLM) given by

$$\mathrm{AvgDM}(x_j) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\partial \hat{y}}{\partial x_{ij}}\right| \tag{19}$$

and

$$\mathrm{AvgLM}(x_j) = \frac{1}{n}\sum_{i=1}^{n}\left(\left|\frac{\partial \hat{y}}{\partial x_{ij}}\right|\right)\left(\left|\frac{x_{ij}}{\hat{y}}\right|\right). \tag{20}$$

Eqs. (17)–(20) utilizes the average derivative of the output of the WN with respect to each explanatory variable. As in averaging procedure a lot of information is lost additional criteria are introduced.

The maximum and minimum derivative (MaxD, MinD) or the maximum and minimum derivative magnitude (MaxDM, MinDM) give additional insight of the sensitivity of the WN output to each explanatory variable. However, these criteria usually cannot be used on their own since they are appropriate only for some applications and are sensitive to inflection points, Zapranis and Refenes (1999). The mathematical expressions of the above criteria can be found in Zapranis and Refenes (1999).

Alternatively to sensitivity criteria, model fitness criteria such as the Sensitivity Based Pruning (SBP) proposed by Moody and Utans (1992) can be used. The SBP method quantifies a variable's relevance to the model by the effect on the empirical loss of the replacement of that variable by its mean. The SBP is given by:

$$SBP(x_j) = L_n(\mathbf{x};\hat{\mathbf{w}}_n) - L_n(\bar{\mathbf{x}}^{(j)};\hat{\mathbf{w}}_n) \tag{21}$$

where

$$\bar{\mathbf{x}}^{(j)} = \left(x_{1,t}, x_{2,t}, \ldots, \bar{x}_j, \ldots, x_{m,t}\right) \tag{22}$$

and

$$\bar{x}_j = \frac{1}{n}\sum_{t=1}^{n}x_{j,t}. \tag{23}$$

Additional criteria can be used like the ones presented in Dimopoulos et al. (1995). For additional information on the criteria presented above we refer to Zapranis and Refenes (1999).

4.1. An algorithm for selecting the significant variables

In order to statistically test whether a variable is insignificant and can be removed for the training dataset or not the distributions of the criteria presented in the previous section are needed. Without the distribution of the preferred measure of relevance it is not clear if the effects of the variable x_i on y are statistically significant, Zapranis and Refenes (1999). More precisely, the only information obtained by criteria described in the previous section is how sensitive is the dependent variable to small perturbations of the independent variable. It is clear that the smaller the value of the preferred criterion the less significant is the corresponding variable. However there is no information if this variable should be removed from the model or not.

In order to approximate asymptotically the distribution of the measures of relevance we use the bootstrap method. More precisely, a number of bootstrapped training samples can be created by the original training dataset. The idea is to estimate the preferred criterion on each bootstrapped sample. If the number of the bootstrapped samples is large then a good approximation of the empirical distribution of the criterion is expected to be achieved. Obtaining an approximation of the empirical distributions, confidence intervals and hypothesis tests can be constructed for the value of the criterion. The variable selection algorithm is analytically explained below and is illustrated in Fig. 6.

The procedure is the following: The algorithm starts with the training sample that consists of all available explanatory variables.

The first step is to create B bootstrapped training samples from the original dataset.

The second step is to identify the correct topology of the WN following the procedure described in the previous section and estimate the prediction risk.

The third step is to estimate the preferred measure of relevance for each explanatory variable for each one of the B bootstrapped training samples.

The fourth step is to calculate the p-values of the measure of relevance.

The fifth step is to test if any explanatory variables have a p-value greater than 0.1 If variables with a p-value greater than 0.1 exist then the variable with the largest p-value is removed from the training dataset else the algorithm stops.

The sixth step is to estimate the prediction risk and the new p-values of the reduced model. If the new estimated prediction risk is smaller than the prediction risk multiplied by a threshold (usually 1.05) then the decision of removing the variable was correct and we return to the fifth step.

If the new prediction risk is greater than the new prediction risk multiplied by a threshold (usually 1.05) then the decision of removing the variable was wrong and the variable must be reintroduced to the model. In this case the variable with the next largest p-value which is also greater than 0.1 is removed from the training sample and we return to step six. If the remaining variables have p-values smaller than 0.1 then the algorithm stops.

In order to have a good estimation of the prediction risk as well as an approximation of the distribution of the measure of relevance, a large number of bootstrapped samples B is needed. As B increases the accuracy of the algorithm also increases but also increases the computational burden. In Zapranis and Refenes (1999) two different bootstrap methods were presented, the local bootstrap and the parametric sampling, that are significantly less computationally expensive. However, the bootstrapped samples may significantly differ from the original sample. Hence, applying local bootstrap or parametric sampling may lead to wavelets outside their effective support, i.e. wavelets with value of zero, since wavelets are local functions with limited duration. In addition, in contrast to the case of NNs, the asymptotic distribution of the weights of a WN is not known. These observations constitute both local bootstrap and parametric sampling inappropriate for WNs.

Alternatively new samples from training patterns can be constructed. This can be done by applying bootstrap from pairs and train a WN for each sample. Since, the initialization of a WN is very good this procedure is not of a prohibited computational cost. The computational cost of this algorithm is the number of the WNs that must be trained which is B.

4.2. Evaluating the variable significance criteria

In this section the algorithm proposed in the previous section for selecting the significant explanatory variables will be evaluated. More precisely, the eight sensitivity criteria and the model fitness sensitivity criterion will be evaluated in the two functions, $f(x)$ and $g(x)$ given by (13) and (15) respectively.

4.2.1. Example 1

First a second variable is created which was randomly drawn from the uniform distribution within the range $(0, 1)$. Both variables are considered significant and constitute the training patterns $(\mathbf{x}_i, y_i)$ of the training dataset where $\mathbf{x}_i = \left\{x_{1,i}, x_{2,i}\right\}$ and y_i are the target values. A WN is trained in order to learn the target function $f(x)$ were both x_1 and x_2 are introduced to the WN as inputs patterns. The BE algorithm was used for the initialization of the WN. Using CV and BS the prediction risk is minimized when 3 HUs are used and it is 0.04194. The network converges after 3502 iterations. Comparing the results with the findings in previous section it is clear that including an irrelevant variable to our model increases the model complexity and the training time while the predictive power of the model is reduced.

Next, the algorithm described in the previous section will be applied in order to estimate the p-values of each criterion. More precisely, the BS method will be applied in order to estimate the asymptotic distributions of the various criteria. In order to approximate the empirical distributions of the various criteria 50 new bootstrapped samples were created and their corresponding p-values are presented in Table 3. A closer inspection of Table 3 reveals that the MaxD, MinD, MaxDM, MinDM, AvgDM and AvgLM suggest that both variables are significant and must remain on the model. On the other hand, the p-values obtained using the AvgL criterion wrongly suggests that the variable x_1 must be removed from the model. Finally, the SBP and AvgD correctly suggest that x_2 must be removed from the model. More precisely the p-values obtained using the AvgD are 0.0614 and 0.3158 for x_1 and x_2 respectively while the p-values obtained using the SBP are 0 and 0.9434 for x_1 and x_2 respectively. Finally, the p-value of x_1 using the SBP in the reduced model is 0 indicating that x_1 is still very significant. However, while the average value of SBP is almost the same in the full and the reduced model, the average value of AvgD is completely different in magnitude and sign.

The correctness of removing a variable from the model should always be further tested. As it was discussed in the previous section this can be done either by estimating the prediction risk or the $\bar{R}^2$ of the reduced model. The prediction risk in the reduced model was reduced to 0.0396 while it was 0.0419 in the full model. Moreover the $\bar{R}^2$ increased to 70.8% in the reduced model while it was 69.8% in the full model. The results indicate that the decision to remove x_2 was correct.

4.2.2. Example 2

The same procedure is repeated for the second case where a WN is used to learn the function $g(x)$ from noisy data. First a second variable is created which was randomly drawn from the uniform

Table 3
Variable significance testing for the first case using bootstrap.

	MaxD	MinD	MaxDM	MinDM	AvgD	AvgDM	AvgL	AvgLM	SBP
Full model (two variables)									
X_1	1.6242	−2.1524	2.2707	0.0031	−0.1079	0.6998	−0.0267	1.3498	0.0982
Std.	1.3929	2.3538	2.4426	0.0029	0.0758	0.0391	0.1651	0.4161	0.0045
p-value	0.0000	0.0000	0.0000	0.0000	0.0614	0.0000	0.6039	0.0000	0.0000
X_2	1.1038	−1.2013	1.4472	0.0003	0.0402	0.1369	0.1033	0.2488	0.0011
Std.	1.4173	2.6560	2.8320	0.0003	0.0477	0.0277	0.1010	0.1244	0.0013
p-value	0.0000	0.0000	0.0000	0.0179	0.3158	0.0000	0.4610	0.0000	0.9434
Reduced model (one variable)									
X_1	–	–	–	–	0.0800	–	–	–	0.0988
Std.	–	–	–	–	0.0433	–	–	–	0.0051
p-value	–	–	–	–	0.0000	–	–	–	0.0000

MaxD = Maximum Derivative
MinD = Minimum Derivative
MaxDM = Maximum Derivative Magnitude
MinDM = Minimum Derivative Magnitude
AvgD = Average Derivative
AvgDM = Average Derivative Magnitude
AvgL = Average Elasticity
AvgLM = Average Elasticity Magnitude
SBP = Sensitivity Based Pruning.

Table 4
Variable significance testing for the second case using bootstrap.

	MaxD	MinD	MaxDM	MinDM	AvgD	AvgDM	AvgL	AvgLM	SBP
Full model (two variables)									
X1	1.6485	−1.8391	1.9459	0.0006	0.0225	0.5412	0.2908	8.9262	0.4191
Std.	0.3555	0.7505	0.7475	0.0008	0.0736	0.0524	7.0110	5.9525	0.0589
p-value	0.0000	0.0000	0.0000	0.2867	0.9877	0.0000	0.8708	0.0000	0.0000
X2	10.0490	−7.7106	11.4443	0.0007	0.0269	0.4564	−0.1217	0.6045	0.0024
Std.	16.2599	9.5366	16.9065	0.0005	0.0923	0.2912	0.5508	0.7338	0.0085
p-value	0.07838	0.0762	0.1597	0.4158	0.6686	0.0000	0.7864	0.0000	0.8433
Reduced model (one variable)									
X1	–	–	1.7261	0.0009	–	–	–	–	0.4779
Std.	–	–	0.0916	0.0008	–	–	–	–	0.0255
p-value	–	–	0.0000	0.1795	–	–	–	–	0.0000

MaxD = Maximum Derivative
MinD = Minimum Derivative
MaxDM = Maximum Derivative Magnitude
MinDM = Minimum Derivative Magnitude
AvgD = Average Derivative
AvgDM = Average Derivative Magnitude
AvgL = Average Elasticity
AvgLM = Average Elasticity Magnitude
SBP = Sensitivity Based Pruning.

distribution within the range (0, 1). Both variables are considered significant and constitute the training patterns. A WN is trained in order to learn the target function $g(x)$ were both x_1 and x_2 are introduced to the WN as inputs patterns. The BE algorithm was used for the initialization of the WN. Using CV and BS the prediction risk is minimized when 10 HUs are used and it is 0.00336. The network approximation converges after 18 811 iterations. Again the inclusion of an irrelevant variable to our model increased the model complexity and the training time while the predictive power of the model was reduced.

Next, we estimate the *p*-values of the various criteria for the second case. The standard deviation and the *p*-values for all sensitivity and model fitness measures for the two variables of the second case are presented Table 4.

In Table 4 the analysis for the second case is presented. A closer inspection of Table 4 reveals that MaxD, MinD, AvgDM, and AvgLM suggest that both variables are significant and must remain in the model. On the other hand, the *p*-values obtained using the AvgL and AvgD criteria wrongly suggest that the variable x_1 must be removed from the model. Finally, the SBP, MaxD and MinDM correctly suggest that x_2 is not a significant variable and can be removed from the model. More precisely the *p*-values obtained using the MaxDM are 0 and 0.1597 for x_1 and x_2 respectively while the *p*-values obtained using the MinDM are 0.2867 and 0.4158 for x_1 and x_2 respectively. Finally, the *p*-values obtained using the SBP are 0 and 0.8433 for x_1 and x_2 respectively. Examining the reduced model, where only x_1 is used for the training of the WN, the *p*-values are 0 for x_1 when the MaxDM or the SBP criteria are used. On the other hand the *p*-value for x_1 is 0.1795, when the MinDM is used, indicating that x_1 is insignificant and should be also removed from the model.

Next, the correctness of removing a variable from the model is further tested. As it was discussed in the previous section this can be done either by estimating the prediction risk or the $\bar{R}^2$ of the reduced model. The prediction risk in the reduced model was reduced to 0.0008 while it was 0.0033 in the full model. Moreover the $\bar{R}^2$ increased to 99.7% in the reduced model while it was 99.2% in the full model.

The results from the previous simulated experiments indicate that the SBP gives constantly correct and robust results. In every case the SBP criterion correctly identified the irrelevant variable. Moreover the SBP criterion was stable and had the same magnitude and sign in both the full and reduced model.

The results of the previous cases indicate that when our algorithm is applied and the *p*-values are estimated, the performance of the remaining sensitivity criteria is unstable. In general the sensitivity criteria were not able to identify the insignificant variable.

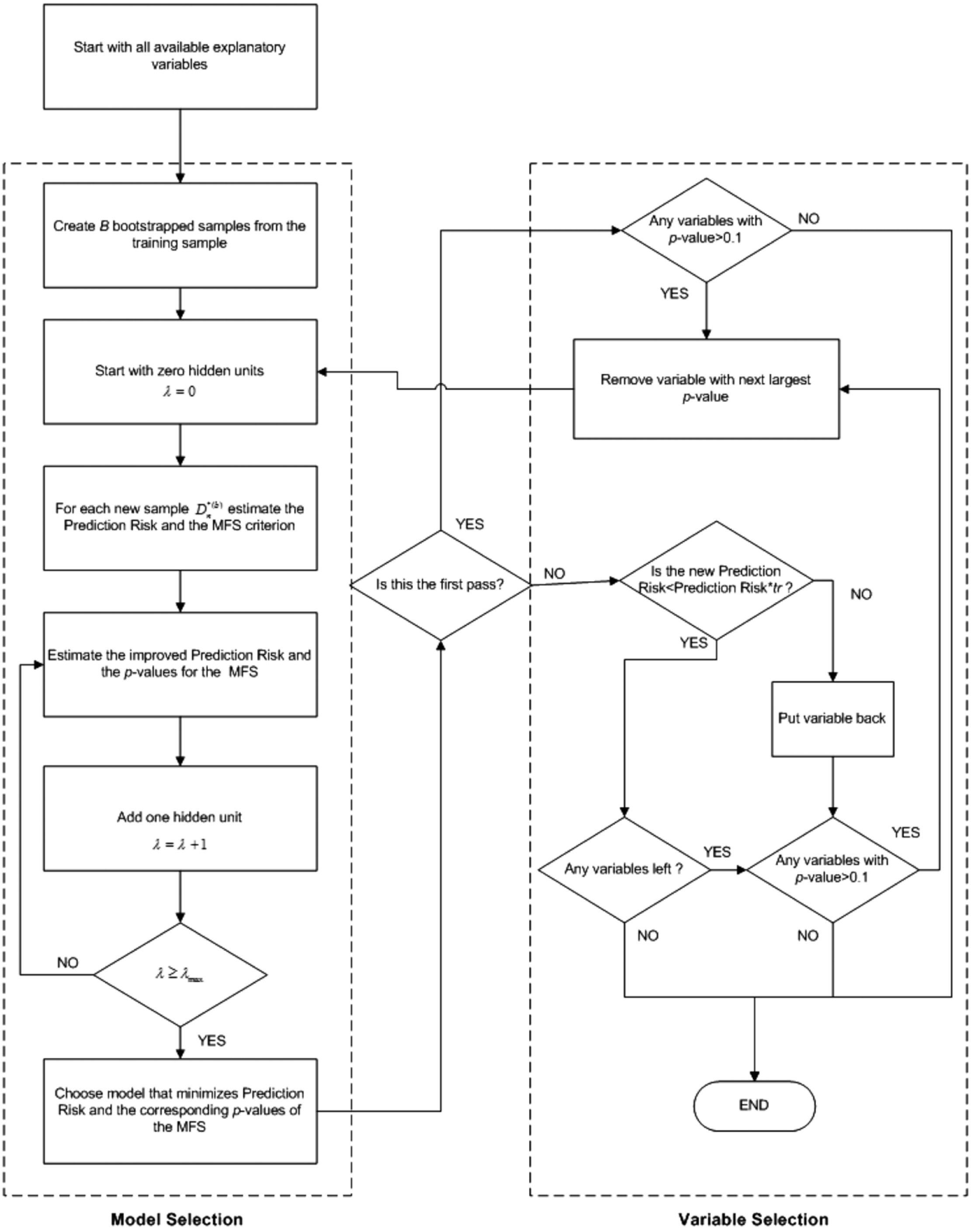

Start with all available explanatory variables
Create *B* bootstrapped samples from the training sample
Start with zero hidden units
$\lambda = 0$
For each new sample $D_n^{*(b)}$ estimate the Prediction Risk and the MFS criterion
Estimate the improved Prediction Risk and the *p*-values for the MFS
Add one hidden unit
$\lambda = \lambda + 1$
$\lambda \geq \lambda_{max}$
NO
YES
Choose model that minimizes Prediction Risk and the corresponding *p*-values of the MFS
Is this the first pass?
YES
NO
Any variables with *p*-value>0.1
NO
YES
Remove variable with next largest *p*-value
Is the new Prediction Risk<Prediction Risk*tr ?
NO
YES
Put variable back
Any variables left ?
YES
NO
Any variables with *p*-value>0.1
YES
NO
END
Model Selection
Variable Selection

Fig. 6. Model identification. Model selection and variable selection algorithms.

Moreover, they often suggested the removal of the significant variable x_1. The sensitivity criteria are application dependent and extra care must be taken when used, Zapranis and Refenes (1999). As their name suggest they are more appropriate for use in sensitivity analysis rather in variable significance testing.

5. Modeling the uncertainty

In the previous sections a framework were a WN can efficiently be constructed, initialized and trained was presented. In this section this framework is expanded by presenting two methods for estimating confidence and prediction intervals. The output of the WN is the approximation of the underlying function $f(\mathbf{x})$ obtained from the noisy data. In many applications and especially in finance, risk managers may be more interested in predicting intervals for future movements of the underlying function $f(\mathbf{x})$ than simply point estimates.

In real datasets the training patterns usually are inaccurate since they contain noise or they are incomplete due to missing observations. Especially financial time series as well as temperature time series are dominated by these characteristics. As a result the validity of the predictions of our model (as well as of any other model) is questioned. The uncertainty that results from the data contributes to the total variance of the prediction and it is called the data noise variance, σ_ε^2, Breiman (1996), Carney, Cunningham, and Bhagwan (1999), Heskes (1997), Papadopoulos, Edwards, and Murray (2000).

On the other hand presenting to a trained network new data that were not introduced to the WN during the training phase, additional uncertainty is introduced to the predictions. Since the training set consist of a finite number of training pairs, the solution $\hat{\mathbf{w}}_n$ is likely not to be valid in regions not represented in the training sample, Papadopoulos et al. (2000). In addition, the iterative algorithm that is applied to train a network, often results to local minima of the loss function. This source of uncertainty that arises from misspecifications in model or parameter selection as well as from limitation of the training algorithm contributes also to the total variance of the prediction and it is called the model variance, σ_m^2, Papadopoulos et al. (2000).

The model variance and the data noise variance are assumed to be independent. The total variance of the prediction is given by the sum of two variances:

$$\sigma_p^2 = \sigma_m^2 + \sigma_\varepsilon^2. \tag{24}$$

If the total variance of a prediction can be estimated then it is possible to construct confidence and prediction intervals. The rest of the section is dedicated to this purpose.

In the framework of classical sigmoid NNs the proposed methods for constructing confidence and prediction intervals falls into 3 major categories: the analytical the Bayesian and the ensemble networks methods.

Analytical methods provide good prediction intervals, only if the training set is very large, De Veaux, Schumi, Schweinsberg, and Ungar (1998). They are based on the assumptions that the noise in the data is independent and identically distributed with mean zero and constant standard deviation. In real problems the above hypothesis usually does not hold. As a result there will be intervals where the analytical method either overestimates or underestimates the total variance. Finally, on analytical methods the effective number of parameters must be identified although pruning schemes like the Irrelevant Connection Elimination scheme can be used to solve this problem. On the other hand, Bayesian methods are computationally expensive methods that need to be tested further, Zapranis and Refenes (1999), Ζαπράνησ (2005). Results from Papadopoulos et al. (2000) indicate that the use of Bayesian methods and the increase in the computational burden is not justified by their performance. Finally, analytical and Bayesian methods are computationally complex since the inverse of the Hessian matrix must be estimated which under certain circumstances can be very unstable.

Finally, ensemble network methods create different versions of the initial network and then they combine the outputs to provide constancy to the predictor by stabilizing the high variance of a NN. In ensemble network methods the new versions of the network usually are created using bootstrap. The only assumption needed is that the NN provides an unbiased estimation of the true regression. Moreover, ensemble networks can handle non-constant variance. We suppose that the total variance of the prediction is not constant and is given by:

$$\sigma_p^2(\mathbf{x}) = \sigma_m^2(\mathbf{x}) + \sigma_\varepsilon^2(\mathbf{x}). \tag{25}$$

Two of the most often cited methods is the bagging, Breiman (1996), and balancing method, Carney et al. (1999), Heskes (1997). In this section we adapt these two methods in order to construct confidence and prediction intervals under the framework of WNs. A framework similar to the one presented in Carney et al. (1999) to estimate the total prediction variance, σ_p^2 and construct confidence and prediction intervals is adapted.

5.1. Confidence intervals

To generate confidence intervals the distribution of the accuracy of the network prediction to the true underlying function is needed. In other words the variance of the distribution of

$$f(\mathbf{x}) - \hat{y} \equiv f(\mathbf{x}) - g_\lambda\left(\mathbf{x}, \hat{\mathbf{w}}_n\right) \tag{26}$$

must be estimated.

The model variance σ_m^2 will be estimated using two different bootstrap methods, the bagging method proposed by Breiman (1996) and the balancing method proposed by Heskes (1997), Carney et al. (1999). Both methods are variation of the bootstrap method.

First $B = 200$ new random samples with replacement are created from the original training sample. Each new sample is used to train a new WN with the same topology as the original one, $g_\lambda(\mathbf{x}^{(*i)}; \hat{\mathbf{w}}^{(*i)})$, where $(*i)$ indicates the ith bootstrapped sample and $\hat{\mathbf{w}}^{(*i)}$ is the solution of the ith bootstrapped sample. Then each new network is evaluated using the original training sample $\mathbf{x}$. Next the average output of the B networks is estimated by:

$$g_{\lambda,avg}(\mathbf{x}) = \frac{1}{B}\sum_{i=1}^{B} g_\lambda\left(\mathbf{x}; \hat{\mathbf{w}}^{(*i)}\right). \tag{27}$$

It is assumed that the WN produces an unbiased estimate of the underlying function $f(\mathbf{x})$. This means that the distribution of $P\left(f(\mathbf{x})|g_{\lambda,avg}(\mathbf{x})\right)$ is centered on the estimate $g_{\lambda,avg}(\mathbf{x})$, Carney et al. (1999), Heskes (1997), Zapranis and Livanis (2005). Since, the WN is not an unbiased estimator (as any other model) it assumed that the bias component arising from the WN is negligible in comparison to the variance component, Carney et al. (1999), Zapranis and Livanis (2005). Finally, if we assume that the distribution of $P\left(f(\mathbf{x})|g_{\lambda,avg}(\mathbf{x})\right)$ is normal then the model variance can be estimated by:

$$\hat{\sigma}_m^2(\mathbf{x}) = \frac{1}{B-1}\sum_{i=1}^{B}\left(g_\lambda\left(\mathbf{x}; \hat{\mathbf{w}}^{(*i)}\right) - g_{\lambda,avg}(\mathbf{x})\right)^2. \tag{28}$$

In order to construct confidence intervals the distribution of $P\left(g_{\lambda,avg}(\mathbf{x})|f(\mathbf{x})\right)$ is needed. Since the distribution of $P\left(f(\mathbf{x})|g_{\lambda,avg}(\mathbf{x})\right)$ is assumed to be normal then the "inverse" distribution $P\left(g_{\lambda,avg}(\mathbf{x})|f(\mathbf{x})\right)$ is also normal. However this distribution is unknown. Alternatively it is empirically estimated by the distribution of $P\left(g_\lambda(\mathbf{x})|g_{\lambda,avg}(\mathbf{x})\right)$, (Carney et al., 1999; Zapranis & Livanis, 2005). Then the confidence intervals are given by:

$$g_{\lambda,avg}(\mathbf{x}) - t_{a/2}\hat{\sigma}_m \le f(\mathbf{x}) \le g_{\lambda,avg}(\mathbf{x}) + t_{a/2}\hat{\sigma}_m \tag{29}$$

where $t_{a/2}$ can be found in a Student's t table and $1-a$ is the desired confidence level.

However the estimator of the model variance, $\hat{\sigma}_m^2$, given by (28) is known to be biased, Carney et al. (1999), as a result wider confidence intervals will be produced. Carney et al. (1999) proposed a balancing method to improve the model variance estimator.

The B bootstrapped samples are divided in M groups. More precisely the 200 ensemble samples are divided in 8 groups of 25 samples each. Next the average output of each group is estimated:

$$\zeta = \left\{g_{\lambda,avg}^{(i)}(\mathbf{x})\right\}_{i=1}^{M}. \tag{30}$$

The model variance is not estimated just by the M ensemble output since this estimation will be highly volatile, Carney et al. (1999). In order to overcome this, a set of $P = 1000$ bootstraps of the values of ζ are created:

$$Y = \left\{\zeta_j^*\right\}_{j=1}^{P} \tag{31}$$

where

$$\zeta_j^* = \left\{ g_{\lambda,avg}^{(*j1)}(\mathbf{x}), g_{\lambda,avg}^{(*j2)}(\mathbf{x}), \ldots, g_{\lambda,avg}^{(*jM)}(\mathbf{x}) \right\} \tag{32}$$

is a bootstrapped sample of ζ. Then the model variance is estimated on each one of these sets by

$$\hat{\sigma}_j^{2*}(\mathbf{x}) = \frac{1}{M}\sum_{k=1}^{M}\left(g_{\lambda,avg}^{(*jk)}(\mathbf{x}) - g_{\lambda,avg}^{j}(\mathbf{x}) \right)^2 \tag{33}$$

where

$$g_{\lambda,avg}^{j}(\mathbf{x}) = \frac{1}{M}\sum_{k=1}^{M} g_{\lambda,avg}^{(*jk)}(\mathbf{x}). \tag{34}$$

Then the average model variance is estimated by taking the average of all $\hat{\sigma}_j^{2*}(x)$:

$$\hat{\sigma}_m^2(\mathbf{x}) = \frac{1}{P}\sum_{j=1}^{P}\hat{\sigma}_j^{2*}(\mathbf{x}). \tag{35}$$

This procedure is not computationally expensive since there is no need to train new networks. Hence, the complexity of both methods is similar and depends of the number B of the WNs that must be trained.

Following the same assumptions as in the bagging method, confidence intervals can be constructed. Since a good estimator of the model variance is obtained the improved confidence intervals using the balancing methods are given by:

$$g_{\lambda,avg}(\mathbf{x}) - z_{a/2}\hat{\sigma}_m \le f(\mathbf{x}) \le g_{\lambda,avg}(\mathbf{x}) + z_{a/2}\hat{\sigma}_m \tag{36}$$

where $z_{a/2}$ can be found in a standard Gaussian distribution table and $1 - a$ is the desired confidence level.

5.2. Prediction intervals

To generate prediction intervals the distribution of the accuracy of the network prediction to target values is needed. In other words the variance of the distribution of

$$y - \hat{y} \equiv y - g_\lambda\left(\mathbf{x}, \hat{\mathbf{w}}_n\right) \tag{37}$$

must be estimated.

In order to construct prediction intervals the total variance of the prediction, σ_p^2, must be estimated. As it was presented earlier the total variance of the prediction is the sum of the model variance and the data noise variance. In the previous section a method for estimating the model variance was presented. Here we emphasize on a method for estimating the data noise variance.

In order to estimate the noise variance σ_ε^2 maximum likelihood methods are used. First, the initial WN, $g_\lambda(\mathbf{x}; \hat{\mathbf{w}}_n)$, is estimated and the solution $\hat{\mathbf{w}}_n$ of the loss function is found. Since it is assumed that the estimated WN is a good approximation of the unknown underlying function, the vector $\hat{\mathbf{w}}_n$ is expected to be very close to the true vector $\mathbf{w}_0$ that minimizes the loss function. Hence, the noise variance can be approximated by a second WN, $f_\nu\left(\mathbf{x}; \hat{\mathbf{u}}_n\right)$, where the squared residuals of the initial WN are used as target values, Satchwell (1994). In the second WN, $f_\nu\left(\mathbf{x}; \hat{\mathbf{u}}_n\right)$, υ is the number of HUs and $\hat{\mathbf{u}}_n$ is the estimated vector of parameters that minimizes the loss function of the second WN. Since it is assumed that the estimated WN is a good approximation of the unknown underlying function, the vector $\hat{\mathbf{u}}_n$ is expected to be very close to the true vector $\mathbf{u}_0$ that minimizes the loss function. Hence the following cost function is minimized in the second network:

$$\sum_{i=1}^{n}\left\{ \left(g_\lambda(\mathbf{x}_i; \mathbf{w}_0) - y_i\right)^2 - f_\nu(\mathbf{x}_i; \mathbf{u}_0) \right\}^2 \tag{38}$$

and for a new set of observations, $\mathbf{x}^*$ that were not used in the training:

$$\hat{\sigma}_\varepsilon^2(\mathbf{x}^*) \approx f_\nu\left(\mathbf{x}^*; \mathbf{u}_0\right). \tag{39}$$

This technique assumes that the residuals errors are caused by variance alone, Carney et al. (1999). In order to estimate the noise variance, data that were not used in the training of the bootstrapped sample should be used. One way to do this is to divide the dataset in training and a validation set. However, leaving out these test patterns is a waste of data, Heskes (1997). Alternatively an unbiased estimation of the output of the WN, $\hat{y}_{ub}(x)$, can be approximated by:

$$\hat{y}_{ub}(\mathbf{x}) = \sum_{i=i}^{B} q_i^m \hat{y}_i(\mathbf{x}) \Big/ \sum_{i=i}^{B} q_i^m \tag{40}$$

where q_i^m is 0 if pattern m appears on the ith bootstrap sample and 1 otherwise. Constructing the new network $f_\nu(\mathbf{x}; \mathbf{u})$ we face the problem of model selection again. Using the methodology described in the previous section, the correct number of υ HUs is selected. Usually 1 or 2 HUs are enough to model the residuals. Finding the estimator of the noise variance the prediction intervals can be constructed:

$$g_{\lambda,avg}(\mathbf{x}^*) - t_{a/2}\hat{\sigma}_p(\mathbf{x}^*) \le f(\mathbf{x}^*) \le g_{\lambda,avg}(\mathbf{x}^*) + t_{a/2}\hat{\sigma}_p(\mathbf{x}^*) \tag{41}$$

where $t_{a/2}$ can be found in a Student's t distribution table and $1 - a$ is the desired confidence level. If the balancing method is used then the prediction intervals are given by:

$$g_{\lambda,avg}(\mathbf{x}^*) - z_{a/2}\hat{\sigma}_p(\mathbf{x}^*) \le f(\mathbf{x}^*) \le g_{\lambda,avg}(\mathbf{x}^*) + z_{a/2}\hat{\sigma}_p(\mathbf{x}^*) \tag{42}$$

where $z_{a/2}$ can be found in a standard Gaussian distribution table and $1 - a$ is the desired confidence level.

5.3. Evaluating the confidence and prediction intervals

In this section the bagging and balancing methods are evaluated in constructing confidence and prediction intervals. The two methods will be tested in the two function $f(x)$ and $g(x)$ given by (13) and (15) respectively.

In Fig. 7 the confidence intervals are presented for the first function. The first part of the Fig. 7 presents the confidence intervals using the bagging method while the second part presents the confidence intervals using the balancing method. Similarly, Fig. 8 presents the confidence intervals for the second function where the first part refers to the bagging method while the second part refers to the balancing method. It is clear that the confidence intervals using the balancing method are significantly narrower. This is due to the biased model variance estimator of the bagging method which results in overestimation of the confidence intervals, Carney et al. (1999).

The 95% prediction intervals of the first function, $f(x)$, are presented in Fig. 9. Again, the first part refers to the bagging method while the second part refers to the balancing. It is clear that both methods were able to capture the change in the variance of the noise. In both cases a WN with 2 HUs were used to approximate function $f(x)$ and a WN with 1 HUs to approximate the residuals in order to estimate the noise variance. In order to compare the two methods the Prediction Interval Correct Percentage (PICP) is used. PICP is the percentage of data points contained in the prediction intervals. Since the 95% prediction intervals were estimated, a value of PICP close to 95 is expected. The bagging prediction intervals contain 98% of the data points (PICP) while in the case of the balancing method the PICP = 95% and equal to the nominal value of 95%.

Next, the same analysis is repeated for the second function, $g(x)$. The 95% prediction intervals of $g(x)$ are presented in Fig. 10. The

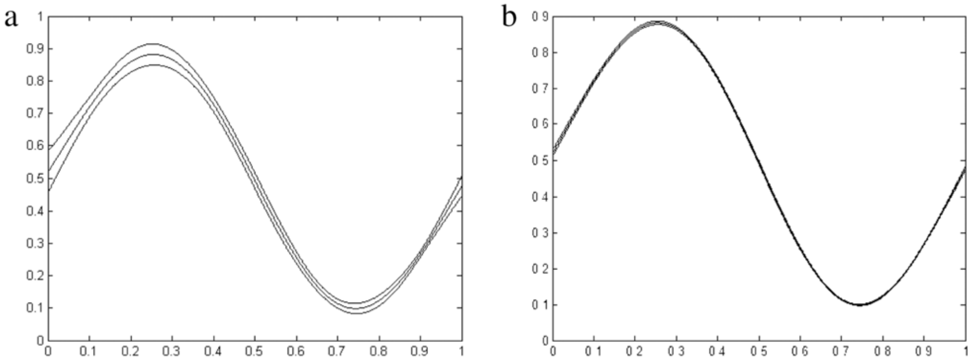


Fig. 7. Confidence intervals for the first case using the bagging (a) and balancing (b) method.

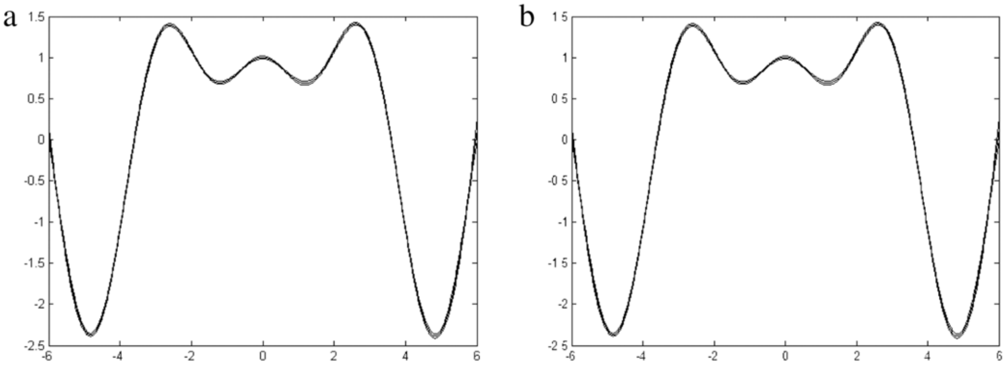


Fig. 8. Confidence intervals for the second case using the bagging (a) and balancing (b) method.

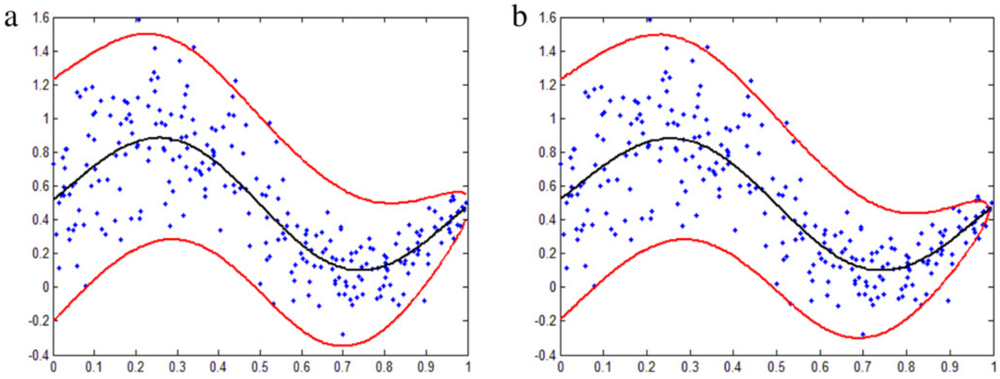


Fig. 9. Prediction intervals for the first case using the (a) bagging (PICP = 98%) and (b) balancing (PICP = 95%) method.

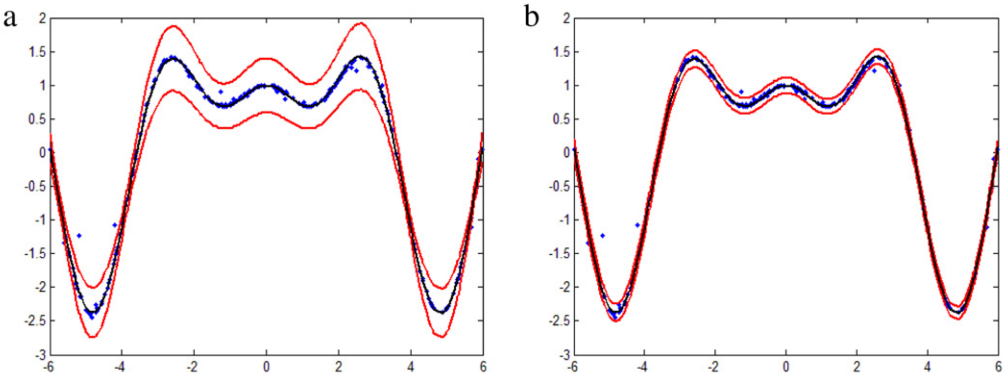


Fig. 10. Prediction intervals for the second case using the (a) bagging (PICP = 98.33%) and (b) balancing (PICP = 97.33%) method.

Table 5
Variable selection with backward elimination in Berlin.

Step	Variable to remove (lag)	Variable to enter (lag)	Variables in model	Hidden units (parameters)	n/p ratio	Empirical loss	Prediction risk
	–	–	7	5 (83)	43.9	1.5928	3.2004
1	X_6	–	6	2 (33)	110.4	1.5922	3.1812
2	X_7	–	5	1 (17)	214.3	1.5927	3.1902
3	X_5	–	4	1 (14)	260.2	1.6004	3.2056
4	X_4	–	3	1 (11)	331.2	1.5969	3.1914

The algorithm concluded in 4 steps. In each step the following are presented: which variable is removed, the number of hidden units for the particular set of input variables and the parameters used in the wavelet network, the empirical loss and the prediction risk.

first part refers to the bagging method while the second part refers to the balancing. In both cases a WN with 8 HUs were used to approximate function $g(x)$ and a WN with 2 HUs to approximate the residuals in order to estimate the noise variance. As in the previous case the two methods are compared using the PICP. For the bagging method the PICP $= 98.33\%$ while for the balancing method PICP $= 97.33\%$.

It is clear that the balancing method produce an improved estimator of the model variance. Our results are consistent with those of Breiman (1996), Carney et al. (1999), Heskes (1997), Papadopoulos et al. (2000), Zapranis and Livanis (2005), Ζαπράνησ (2005). In all cases the intervals produced by the balancing method were significantly smaller while the PICP were considerable improved and closer to its nominal value.

6. Case study 1: modeling the daily average temperature in Berlin

In this section a real dataset is used to demonstrate the application of our proposed framework. More precisely using data from detrended and deseasonalized daily average temperatures (DATs) a WN will be constructed, initialized and trained. Also, at the same time the significant variables will be selected, in this case the correct number of lags. Finally, the trained WN will be used to construct confidence and prediction intervals.

The dataset consists of 3650 values, corresponding to the detrended and deseasonalized DATs of 10 years (1991–2000) in Berlin. In order for each year to have equal observations the 29th of February was removed from the data.

Using WNs the generalized version of detrended and deseasonalized is estimated nonlinearly and non-parametrically, that is:

$$\tilde{T}(t+1) = \varphi\left(\tilde{T}(t), \tilde{T}(t-1), \ldots\right) + e(t) \tag{43}$$

where $\tilde{T}$ is the detrended and deseasonalized DAT and $e(t)$ are the residuals of the WN.

For a concise treatment on modeling the temperature process refer to Zapranis and Alexandridis (2008), Zapranis and Alexandridis (2009). In the above expression, the length of the lag series must be selected.

6.1. Variable selection

The target values of the WN are the DATs. The explanatory variables are lagged versions of the target variable. Choosing the length of a lag distribution in linear models can be done by minimizing an information criterion like Akaike or Schwarz criteria. Alternatively the ACF and the PACF can be studied. The ACF suggests that the first 35 lags are significant. On the other hand the PACF suggests that the 6 first lags as well as the 8th and the 11th lag must be included on the model. However results from these methods are not necessarily true in nonlinear nonparametric models.

Alternatively, in order to select only the significant lags the variable selection algorithm presented in the previous section will be applied. Initially, the training set contains the dependent variable and 7 lags. Hence, the training set consists of 7 inputs, 1 output and 3643 training pairs.

In this study the relevance of a variable to the model is quantified by the SBP criterion which was introduced in the previous section.

Table 5 summarizes the results of the model identification algorithm for Berlin. Both the model selection and the variable selection algorithms are included in Table 5. The algorithm concluded in 4 steps and the final model contains only 3 variables. The prediction risk for the reduced model is 3.1914 while for the original model was 3.2004. On the other hand the empirical loss slightly increased from 1.5928 for the initial model to 1.5969 for the reduced model indicating that the explained variability (unadjusted) slightly decreased. However, the explained variability (adjusted for degrees of freedom) was increased for the reduced model to 64.61% while it was 63.98 initially. Finally, the number of parameters is significantly reduced in the final model. The initial model needed 5 HUs and 7 inputs. Hence, 83 parameters were adjusted during the training phase. Hence the ratio of the number of training pairs n to the number of parameters p was 43.9. In the final model only 1 HU and 3 inputs were used. Hence only 11 parameters were adjusted during the training phase and the ratio of the number of training pairs n to the number of parameters p was 331.2.

In Table 6 the statistics for the WN model at each step can be found. More precisely, the first part of Table 6 reports the value of the SBP and its p-value. In the second part of Table 6 various fitting criteria are reported. More precisely the Mean Absolute Error, the Maximum Absolute Error (Max AE), the Normalized Mean Square Error (NMSE), the Mean Absolute Percentage Error (MAPE), the $\bar{R}^2$, the empirical loss and the prediction risk.

In the full model, it is clear that the value of the SBP for the last three variables is very small in contrast to the first two variables. Observing the p-values, we conclude that the last four variables have p-value greater than 0.1 while the 6th lag has a p-value of 0.8826 strongly indicating a “not significant” variable. The WN was converged after 43 iterations. In general a very good fit was obtained. The empirical loss is 1.5928 and the prediction risk is 3.2004. The Max AE is 11.1823 while the MAE is 1.8080 and the NMSE is 0.3521. The MAPE is 3.7336. Finally the $\bar{R}^2 = 63.98\%$.

The statistics for the WN at step 1 are also presented in Table 6. The network had 6 inputs, 2 wavelets were used to construct the WN and 33 weights adjusted during the training phase. The WN converged after 17 iterations. By removing X_6 from the model, we observe from Table 6 that the p-value of X_5 became 0 while for X_7 and X_4 the p-values became 0.5700 and 0.1403 respectively. The empirical loss was slightly decreased to 1.5922. However the MAE and NMSE were slightly increased to 1.8085 and 0.3529 respectively. On the other hand the Max AE and the MAPE were decreased to 11.1446 and 3.7127 respectively. Next the decision of removing X_6 is tested. The new prediction risk was reduced to 3.1812 while the explained variability adjusted for degrees of freedom increased to 64.40%. Hence, the removal of X_6 reduced the complexity of the model while its predictive power was increased.

Table 6
Step by step variable selection in Berlin.

Variable	Full model		Step 1		Step 2		Step 3		Step 4	
	SBP	*p*-value	SBP	*p*-value	SBP	*p*-value	SBP	*p*-value	SBP	*p*-value
7	0.0026	0.7796	0.0031	**0.5700**	–	–	–	–	–	–
6	0.0032	**0.8826**	–	–	–	–	–	–	–	–
5	0.0053	0.6757	0.0131	0.0000	0.0206	**0.1907**	–	–	–	–
4	0.0161	0.3500	0.0149	0.1403	0.0216	0.1493	−0.0052	**0.4701**	–	–
3	0.2094	0.0000	0.2368	0.0000	0.2285	0.0000	0.1991	0.0000	0.2244	0.0000
2	1.1123	0.0000	1.0318	0.0000	1.0619	0.0000	0.9961	0.0000	0.9363	0.0000
1	9.8862	0.0000	10.0160	0.0000	9.9858	0.0000	10.0537	0.0000	10.1933	0.0000
MAE	1.8080		1.8085		1.8083		1.8093		1.8095	
Max AE	11.1823		11.1446		11.1949		11.0800		11.0925	
NMSE	0.3521		0.3529		0.3525		0.3526		0.3530	
MAPE	3.7336		3.7127		3.7755		3.7348		3.7171	
$\bar{R}^2$	63.98%		64.40%		64.59%		64.61%		64.61%	
Empirical loss	1.5928		1.5922		1.5927		1.6004		1.5969	
Prediction risk	3.2004		3.1812		3.1902		3.2056		3.1914	
Iterations	43		17		19		4		19	

The average SBP for each variable of 50 bootstrapped samples, the standard deviation and the *p*-value.
SBP = Sensitivity Based Pruning
MAE = Mean Absolute Error
Max AE = Maximum Absolute Error
NMSE = Normalized Mean Square Error
MSE = Mean Square Error
MAPE = Mean Absolute Percentage Error.

Table 7
Prediction risk at each step of the variable selection algorithm for the 5 first hidden units for Berlin.

Step \ HU	1	2	3	4	5
0	3.2009	3.2026	3.2023	3.2019	**3.2004**
1	3.1817	**3.1812**	3.1828	3.1861	3.1860
2	**3.1902**	3.1915	3.1927	3.1972	3.1974
3	**3.2056**	3.2077	3.2082	3.2168	3.2190
4	**3.1914**	3.2020	3.2182	3.2158	3.2169

At step 2, X_7, which had the largest $p - \text{value} = 0.5700$ at the previous step, was removed from the model. Table 6 shows the statistics for the WN at step 2. The new WN had 5 inputs, 1 HU was used and 17 weights adjusted during the training phase. The WN converged after 19 iterations. A closer inspection of Table 7 reveals that the removal of X_7 resulted to an increase in the error measures and a worse fit were obtained. The new $\bar{R}^2$ is 64.59%. The new prediction risk increased to 3.1902 which is smaller than the threshold. In other words, by removing X_7 the total predictive power of our model was slightly decreased; however, adding the variable X_7 on the model only 0.28% additional variability of our model was explained while the computational burden was significantly increased.

Examining the values of the SBP on Table 6 it is observed that the first two variables still have significantly larger values than the remaining variables. The *p*-values reveal that at in the third step the X_5 must be removed from the model since its *p*-value is 0.1907.

At step 3 the network had 4 inputs, 1 HU was used and 14 weights adjusted during the training phase. The WN converged after 4 iterations. When removing X_5 from the model we observe from Table 6 that only X_4 has a *p*-value greater than 0.1. Again the empirical loss and the prediction risk were increased. More precisely the empirical loss is 1.6004 and the prediction risk increased 0.48% to 3.2056. The new prediction risk is greater than the estimated prediction risk of the initial model about 0.16%. Again the increase in the prediction risk was significantly smaller than the threshold. On the other hand, the $\bar{R}^2$ was increased to 64.61% indicating an improved fit. Hence, the decision of removing X_5 was accepted.

In the final step the variable X_4 had $p\text{-value} = 0.4701$ and it was removed from the model. The network had 3 inputs, 1 wavelet was used for the construction of the WN and only 11 weights adjusted during the training phase. The WN converged after 19 iterations. After the removal of X_4 a new WN was trained with only one wavelet. The new empirical loss was decreased to 1.5969. The MAE and NMSE are 1.8095 and 0.3530 respectively while the Max AE and the MAPE are 11.0925 and 3.7171 respectively. Next the decision of removing X_4 was tested. The new prediction risk was reduced to 3.1914 while the explained variability adjusted for degrees of freedom was 64.61%. Hence, the removal of X_4 reduced the complexity of the model while its performance was increased. The *p*-values of the remaining variables are zero indicating that the remaining variables are characterized as very significant variables. Hence, the algorithm stops. Our proposed algorithm indicates that only the 3 most recent lags should be used while PACF suggested the first 6 lags as well as the 8th and the 11th lag.

Concluding, in the final model only three of the seven variables were used. The complexity of the model was significantly reduced since from 83 parameters in the initial model only 11 parameters have to be trained in the final model. In addition, in the reduced model the prediction risk minimized when only one HU was used while 5 HUs were needed initially. Our results indicate that the in-sample fit was slightly decreased in the reduced model. However when an adjustment for the degrees of freedom is made we observe that the $\bar{R}^2$ was increased to 64.61% from 63.98% in the initial model. Finally, the prediction power of the final and less complex proposed model was improved since the prediction risk was reduced to 3.1914 from 3.2004.

6.2. Model selection

In each step the appropriate number of HUs is determined by applying the model selection algorithm presented in Section 3. Table 7 shows the prediction risk for the first 5 HUs at each step of the variable selection algorithm for Berlin. Ideally, the prediction risk will decrease (almost) monotonically until a minimum is reached and then it will start to increase (almost) monotonically. The number of HUs that minimizes the prediction risk is selected for the construction of the model.

In the initial model, where all seven inputs were used, the prediction risk with one HU is only 3.2009. When one additional HU is added to the model the prediction risk increases. Then, as more HUs are added to the model the prediction risk monotonically decreases. The minimum is reached when 5 HUs are used and is

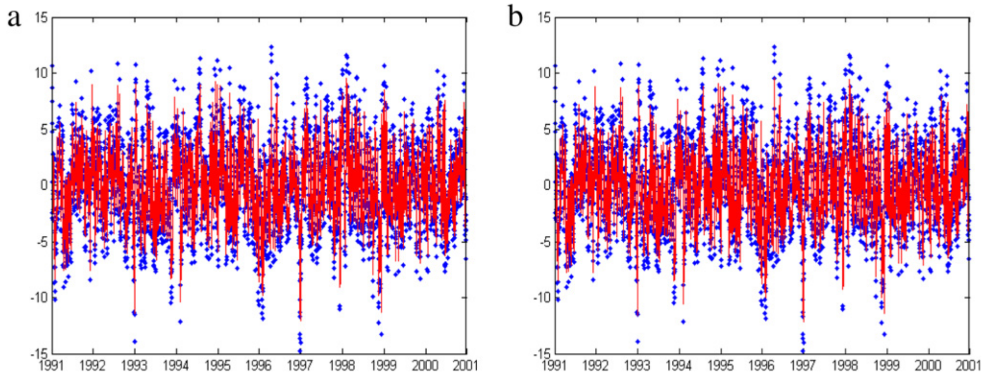


Fig. 11. Initialization of the final model for the temperature data in Berlin using the BE method (a) and the fit of the trained network with 1 HU (b). The WN converged after 19 iterations.

3.2004. When additional HUs are added in the topology of the model the prediction risk increases. Hence, the architecture of the WN contains 5 HUs. In other words, the 5 higher ranking wavelets should be selected form the wavelet basis in order to construct the WN. Observing Table 7 it is clear that the prediction risk at the initial model with only one HU is almost the same as in the model with 5 HUs. This due to the small number of parameters that were adjusted during the training phase when only 1 HU is used and not due to a better fit.

At the second step, when variable X_6 was removed, the prediction risk is minimized when 2 HUs are used. Similarly, at steps two, three and four the prediction risk is minimized when only one HU is used. Additional HUs does not improve the fitting or the predictive power of the model.

6.3. Initialization and training

After the training set and the correct topology of the WN are selected, the WN can be constructed and trained. The BE method is used to initialize the WN. A wavelet basis is constructed by scanning the 4 first levels of the wavelet decomposition of the DAT in Berlin.

The wavelet basis consists of 168 wavelets. However, not all wavelets in the wavelet basis contribute to the approximation of the original time-series. Following (Zhang, 1997) the wavelets that contain less than 5 sample points of the training data in their support are removed. 76 wavelets that do not significantly contributed to the approximation of the original time-series were identified. The truncated basis contains 92 wavelet candidates. Applying the BE method the wavelet are ranked in order of significance. The wavelets in the wavelet library are ranked as follows: the BE starts the regression by selecting all the available wavelets from the wavelet library. Then the wavelet that contributes the least in the fitting of the training data is repeatedly eliminated. Since only one HU is used on the architecture of the model, only the wavelet with the highest ranking is used to initialize the WN. Part (a) of Fig. 11 presents the initialization of the final model using only 1 HU. The initialization is very good and the WN converged after only 19 iterations. The training stopped when the minimum velocity, 10^{-5}, of the training algorithm was reached. The fitting of the trained WN can be found in part (b) of Fig. 11.

Next, various fitness criteria of the WN corresponding to the DAT in Berlin are estimated. Our results reveal that the WNs fit the DATs reasonable well. The overall fit for Berlin is $\bar{R}^2 = 64.61\%$ while the MSE is 5.4196 and the MAE is only 1.8090.

Next the Prediction of Sign (POS) as well the Prediction of Change in Direction (POCID) and the Independent Prediction of Change in Direction (IPOCID) are also estimated. These three criteria examine the ability of the network to predict changes, independently of the size of the change and they are referred as percentages. The POS measures the ability of the network to predict the sign of the target values, positive or negative. For analytical expressions of these criteria we refer to (Zapranis & Refenes, 1999). The POS for the detrended and deseasonalized DATs is very high and it is 81.49%. The POCID is 60.15% while the IPOCID is 52.30%.

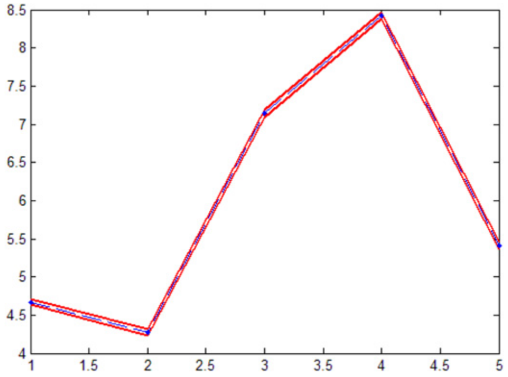


Fig. 12. Confidence intervals and the average WN output using the balancing method and 200 bootstrapped samples. The figure presents only the 5 first values for simplicity.

6.4. Confidence and prediction intervals

After the WN is constructed and trained it can be used for prediction. Hence, confidence and prediction intervals can be constructed. In this section both confidence and prediction intervals will be constructing using the balancing method. Using the BS method 200 training sample will be created and then they will be divided in 8 groups. In each group the average output of the WNs will be estimated. Next new 1000 bootstrapped samples will be created for the 8 average outputs in order to estimate the model variance given by (35). Then the confidence intervals are estimated with level of significance $a = 5\%$.

Fig. 12 presents the confidence intervals for the detrended and deseasonalized DAT in Berlin as well as the average WN output obtained from 200 bootstrapped samples. Because the intervals are very narrow in order to obtain a clear figure only the 5 first values are presented. Next, the prediction intervals are constructed for the out-of-sample dataset. The out-of-sample data consists of 365 values of detrended and deseasonalized DATs in Berlin for the period 2000–2001. In Table 8 the out-of-sample performance criteria are presented. The overall fit adjusted for degrees of freedom is $\bar{R}^2 = 59.27\%$. The NMSE is 0.3961 while the MAPE is only 2.4108. In Fig. 13 the prediction intervals together with

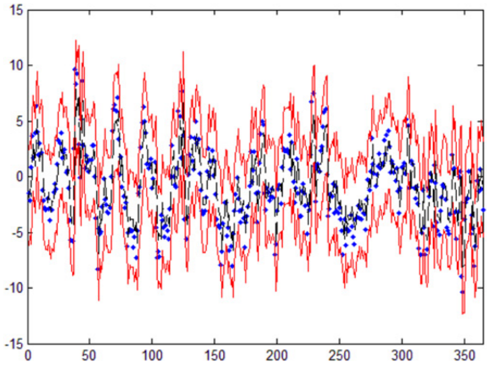


Fig. 13. Prediction intervals, the real data (dotted) and the average WN output (solid line) using the balancing method and 200 bootstrapped samples of the detrended and deseasonalized DATs in Berlin for the period 2000–2001. The PICP = 93.42%.

Table 8
Out-of-sample performance criteria for Berlin.

MAE	1.7340
Max AE	9.3330
NMSE	0.3961
MAPE	2.4108
$\bar{R}^2$	59.27%

the real data and the average forecast of the WN for the 200 bootstrapped samples. The PICP = 93.46%.

7. Case study 2: modeling the daily average wind speed in New York

In this section the proposed framework is applied on a second real dataset. More precisely daily average wind speeds (DAWS) are modeled and forecasted using a WN. The data were collected from NOAA and correspond to DAWS collected from New York, USA. The wind speeds are measured in 0.1 knots while the measurement period is between 1st of January 1988 and 28th of February 2008. The first 20 years are used for the estimation of the parameters while the remaining two months are used for the evaluation of the performance of the proposed model. In order for each year to have the same number of observations the 29th of February is removed from the data resulting to 7359 data points. The dataset is complete without any missing values.

Our aim is to select the correct number of lags that describe the dynamics of the detrended and deseasonalized DAWSs and then build a WN that can accurately forecasts the DAWSs. For a concise treatment on modeling the speed process refer to (Alexandridis & Zapranis, 2011). The predictive power of the proposed model is evaluated in the periods of January and February. These periods exhibits the highest variability and it is much harder to accurately predict the wind speed, (Alexandridis & Zapranis, 2011)

7.1. Variable selection

The first step is to identify the length of the lag series. The targets of the WN are the detrended and deseasonalized DAWSs while the explanatory variables are lagged versions of the target variable. Initially the training set contains the depended variable and 7 lags. Hence, the training set consists of 7 inputs and 7,293 training pairs. Again the relevance of each variable to the model is quantified by the SBP criterion.

Table 9 summarizes the results of the model identification algorithm in New York. Both the model selection and the variable selection algorithms are included in Table 9. The algorithm concluded after 4 steps and the final model contains only 3 variables. The prediction risk for the reduced model is 0.0937 while for the original model was 0.0938 indicating that the predictive power of the WN was slightly increased. On the other hand the empirical loss slightly increased from 0.0467 for the initial model to 0.0468 for the reduced model indicating that the explained variability (unadjusted) slightly decreased. Finally, the complexity of the network structure and the number of parameters were reduced in the final model. The initial model needed 1 HU and 7 inputs. Hence, 23 parameters were adjusted during the training phase and the ratio of the number of training pairs to the number of parameters was 317.4. In the final model only 2 HU and 3 inputs were used. Hence, only 18 parameters were adjusted increasing the ratio of the number of training pairs to the number of parameters to 405.6.

In Table 10 the statistics for the WN at each step can be found. More precisely, the first part of Table 10 reports the values of the SBP and its p-value. In the second part various fitting criteria are reported. A closer inspection of Table 10 revels that the various error measures are reduced in the final model. However, the values of $\bar{R}^2$ are relative small in all cases. This is due to the presence of large noise values compared to the small values of the underlying function.

Concluding, in the final model only three of the seven variables were used. The complexity of the model was reduced while at the same the prediction power of the reduced model was increased. However, a slightly worse fit in-sample was obtained.

7.2. Model selection

In this section the appropriate number of HUs is determined by applying the model selection algorithm. Table 11 shows the prediction risk for the first 5 HUs at each step of the variable selection algorithm for the DAWSs in New York. It is clear that only 1 HU is sufficient to model the detrended and deseasonalized DAWS in New York at the first three steps. Similarly, 2 HU are need for the last two steps.

7.3. Initialization and training

After the training set and the correct topology of the WN are selected, the WN can be constructed and trained. In this case study the BE method is used to initialize the WN. A wavelet basis is constructed by scanning the 4 first level of the wavelet decomposition of the detrended and deseasonalized DAWSs in New York.

The wavelet basis consists of 205 wavelets. In order to reduce the number of the wavelets in the wavelet basis, the wavelets that contain less than 6 sample points of the training data in their support are removed. The truncated basis contains 119 wavelet candidates. Applying the BE method the wavelets are ranked in order of significance. Since, only 2 HUs are used in the architecture of the model, the best two wavelets are selected. Part (a) of Fig. 14 presented the initialization of the final model using 2 HUs. The initialization is very good and the WN converged after 225 iterations. The training stopped when the minimum velocity, 10^{-5}, of the training algorithm was reached. The fitting of the trained WN can be found in part (b) of Fig. 14.

7.4. Prediction

In this section the proposed model will be evaluated out-of-sample. More precisely, the trained WN will be used in order to predict the future evolution of the dynamics of the wind speed process.

The out-of-sample data corresponds to the period of 1st of January 2008 to 28th of February 2008 and were not used for the

Table 9
Variable selection with backward elimination in New York.

Step	Variable to remove (lag)	Variable to enter (lag)	Variables in model	Hidden units (parameters)	n/p ratio	Empirical loss	Prediction risk
	–	–	7	1 (23)	317.4	0.0467	0.0938
1	X_7	–	6	1 (20)	365.0	0.0467	0.0940
2	X_5	–	5	1 (17)	429.4	0.0467	0.0932
3	X_6	–	4	2 (23)	317.4	0.0467	0.0938
4	X_4	–	3	2 (18)	405.6	0.0468	0.0937

The algorithm concluded in 4 steps. In each step the following are presented: which variable is removed, the number of hidden units for the particular set of input variables and the parameters used in the wavelet network, the empirical loss and the prediction risk.

Table 10
Step by step Variable selection in New York.

	Full model		Step 1		Step 2		Step 3		Step 4	
Variable	SBP	p-value	SBP	p-value	SBP	p-value	SBP	p-value	SBP	p-value
7	0.0000	**0.8392**	–	–	–	–	–	–	–	–
6	0.0000	0.7467	0.0000	0.4855	0.0000	**0.9167**	–	–	–	–
5	0.0000	0.6799	0.0000	**0.9467**	–	–	–	–	–	–
4	0.0000	0.5203	0.0000	0.7180	0.0000	0.2643	0.0000	**0.7480**	–	–
3	0.0001	0.1470	0.0001	0.0000	0.0001	0.4706	0.0001	0.4719	0.0003	0.0000
2	0.0010	0.0469	0.0010	0.0000	0.0010	0.0000	0.0009	0.0000	0.0010	0.0168
1	0.0141	0.0000	0.0141	0.0000	0.0137	0.0000	0.0140	0.0000	0.0135	0.0000
MAE	0.2430		0.2430		0.2428		0.2430		0.2429	
Max AE	1.7451		1.7453		1.7156		1.7541		1.6986	
NMSE	0.8832		0.8832		0.8833		0.8834		0.8834	
$\bar{R}^2$	11.68%		11.67%		11.67%		11.68%		11.65%	
Empirical Loss	0.0467		0.0467		0.0467		0.0467		0.0468	
Prediction Risk	0.0938		0.0940		0.0932		0.0938		0.0937	
Iterations	22		37		26		19		225	

The average SBP for each variable of 50 bootstrapped samples, the standard deviation and the p-value.
SBP = Sensitivity Based Pruning
MAE = Mean Absolute Error
Max AE = Maximum Absolute Error
NMSE = Normalized Mean Square Error
MSE = Mean Square Error
MAPE = Mean Absolute Percentage Error.

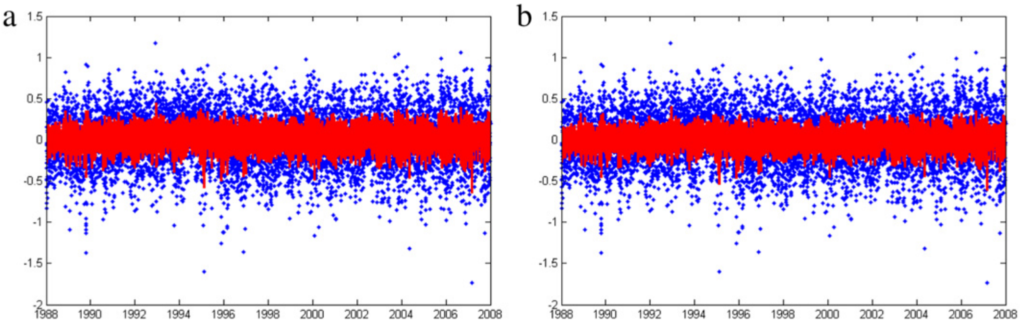


Fig. 14. Initialization of the final model for the wind data in New York using the BE method (a) and the fit of the trained network with 2 HU (b). The WN converged after 225 iterations.

Table 11
Prediction risk at each step of the variable selection algorithm for the 5 first hidden units for New York.

Step \ HU	1	2	3	4	5
0	**0.09378**	0.09380	0.09379	0.09379	0.09380
1	**0.09403**	0.09404	0.09403	0.09406	0.09406
2	**0.09321**	0.09324	0.09325	0.09326	0.09327
3	0.09384	**0.09380**	0.09384	0.09387	0.09386
4	0.09370	**0.09367**	0.09368	0.09373	0.09379

training of the WN. Note, that previous analysis indicates that the variance is higher in the winter period indicating that it is more difficult to forecast accurately DAWS for these two moths. The objective here is to estimate the cumulative wind speed index for the two periods. As a comparison the Historical Burn Analysis (HBA) method will be used. The HBA is a statistical method that estimates the performance of the index over the specific period the previous years. In other words, is the average of 20 years of the index of the period of January and February and it will be used as a benchmark. In Alexandridis and Zapranis (2011) the proposed framework is compared against various methods. The aim here is to show the accuracy of the proposed framework rather than an extensive comparison against various models.

In Table 12 the performance of the WN and the HBA are presented. In addition the actual values of the cumulative wind speed index for the two periods are presented in the final row. A closer inspection of Table 12 reveals that the WN can accurately predict the future evolution of the wind speed process.

First, the cumulative wind speed index is estimated for the period of January. The HBA is 345.5 while the actual index is 311.2. The prediction of the WN is 312.7 and it is very close to the real value of the index. Second, the index is estimated for the whole period between January and February. The HBA suggests that the index is 658.3 while the actual index is 600.6. The forecasted index using the WN is 591.1 which is again very close to real value.

Table 12
Prediction risk at each step of the variable selection algorithm for the 5 first hidden units for New York.

Months	HBA	WN	Actual
1	345.5	312.7	311.2
2	658.3	591.1	600.6

Our results indicate that WNs constitute an accurate and efficient tool for modeling the dynamics of the wind process in New York.

8. Case study 3: breast cancer classification in Wisconsin

In this section a different problem is considered. A WN will be constructed in order to classify breast cancer based on various attributes. The dataset were obtained by the UCI Machine Learning Repository and provided by (Mangasarian & Wolberg, 1990).

The Wisconsin breast cancer (WBC) data contains 699 samples. However, there are 16 missing values that are omitted, reducing the sample to 683 values. Each instance has one of two possible classes: benign or malignant. There are 239 (35%) malignant cases and 444 (65%) benign cases. The aim is to construct a WN that accurately classify each clinical case. The classification is based on 9 attributes: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses.

8.1. Variable selection

The target values of the WN are the two possible classes. The explanatory variables are the 9 attributes described in the previous section. In order to construct an accurate WN classifier the contribution of each attribute to the predictive power of the classifier must be tested. First, the significance of each attribute is examined. Hence, the initial training set consists of 9 inputs, 1 output and 683 training samples. Again, the relevance of each attribute is quantified by the SBP criterion. Applying the proposed variable selection the final model has only 6 variables while the predictive power of the model remains almost unchanged.

Table 13 summarizes the results of the model identification algorithm for the WBC data. Both the model selection and variable selection algorithm are included in Table 13. The algorithm concluded in 3 steps and the final model consists of 6 variables only. A closer inspection of Table 13 reveals that the empirical loss decreased from 0.0713 in the full model to 0.0426 to the reduced and simpler model. In addition, the prediction risk is also decreased from 0.1488 to 0.1135 indicating that the reduced model provides a better fitting to the data but also has a better forecasting ability. The results of the variable significance algorithm indicate that the uniformity of cell shape, marginal adhesion and mitoses should be removed from the input of the training sample in breast tumor classification.

8.2. Model selection

In Table 13 the results of the model selection algorithm are presented. In the full model a WN with 1 HU was constructed. Applying the model selection algorithm using 50 bootstrapped sample of the initial training set, the prediction risk was minimized when only 1 HU was used. The prediction risk for the full model was 0.1488 while the empirical loss was 0.0713. Similarly, in the final step, the reduced model needed 3 HU. The prediction risk was 0.1135 while the empirical loss was 0.0426 indicating that the reduced model provides a better fitting to the data but also has a better forecasting ability

8.3. Initialization and training

After the training set and the correct topology of the WN are selected, the WN can be constructed and trained. The BE method is used to initialize the WN. A wavelet basis is constructed by scanning the 4 first levels of the wavelet decomposition of the dataset.

The initial wavelet basis consists of 675 wavelets. However, not all wavelets in the wavelet basis contribute to the approximation of the original time-series. The wavelets that contain less than 8 sample points of the training data in their support are removed. The truncated basis contains 28 wavelet candidates. The MSE after the initialization was 0.173170 and the initialization needed 0.23 s to finish. The initialization is very good and the WN converged after only 264 iterations. The training stopped when the minimum velocity, 10^{-5}, of the training algorithm was reached. The MSE error after the training is 0.145352 and the total amount of time needed to train the network (initialization and training) was 1.53 s

8.4. Classification power of the full and the reduced model

In this section the predictive and classification power of the WN will be evaluated. More precisely, first the full model, including all 9 attributes will be tested using the leave-one-out cross validation. Then, a comparison will be made against the reduced model that uses only 6 attributes.

The full model is first trained using all training examples. The predictive power of the in-sample classifying power is presented in Table 14. The WN accuracy in sample is 97.65%. Also, the WN classified wrong the malignant tumors only 3 times. Next, the predictive power of the WN is evaluated out-of-sample using the leave-one-out cross-validation method. Each time a validation sample is created that consists of only one observation while the remaining pairs $(\mathbf{x}, y)$ are used for the training of a WN. In the next step another validation sample is created and a new WN is trained. The procedure is repeated until the WN classifies all pairs $(\mathbf{x}, y)$. The accuracy of the full model out-of-sample is 97.51% while the misclassification of the malignant cases is 4.

Next, the predictive power of the reduced model is evaluated in-sample and out-of-sample. In-sample the accuracy of the WN is 97.51% and 4 malignant cases were misclassified. Similarly, the accuracy of the WN out-of-sample is 97.36% with 5 misclassified malignant cases. A closer inspection of Table 14 reveals that the full model outperforms the reduced model only by one correct classification.

It is clear that the accuracy of the network remain practically the same although 3 classifiers were removed from the data. Hence, we can conclude that the information that comes from the uniformity of cell shape, marginal adhesion and mitoses does not contribute significantly in classifying breast tumors since the additionally accuracy is only 0.15%.

Our results indicate that a WN can successfully be used in breast cancer classification providing high classification accuracy. Moreover, the accuracy of the WN is higher than the ones presented in relevant studies, (Duch & Adamczak, 1998; Hassanien & Ali, 2006; Senapati, Mohanty, Dash, & Dash, 2011; Setiono & Liu, 1997; Wei & Billings, 2007).

9. Case study 4: the Mackey–Glass equation

In this section the proposed framework will be evaluated in a chaotic time series. Data from Mackey–Glass series were generated, (Mackey & Glass, 1977). The Mackey–Glass equation is a time-delay differential equation given by:

$$\frac{\partial x}{\partial t} = \frac{ax\,(t-\tau)}{1+x^{c}\,(t-\tau)} - bx(t). \tag{44}$$

Table 13
Variable selection with backward elimination in Wisconsin Breast Cancer Dataset.

Step	Variable to remove (lag)	Variable to enter (lag)	Variables in model	Hidden units (parameters)	n/p ratio	Empirical loss	Prediction risk
	–	–	9	1 (29)	23.6	0.0713	0.1488
1	X_9	–	8	1 (26)	26.7	0.0713	0.1485
2	X_4	–	7	3 (53)	12.9	0.0404	0.1136
3	X_3	–	6	3 (46)	14.8	0.0426	0.1135

The algorithm concluded in 4 steps. In each step the following are presented: which variable is removed, the number of hidden units for the particular set of input variables and the parameters used in the wavelet network, the empirical loss and the prediction risk.

Table 14
Classification power of the full and the reduced model.

Model	HU	Accuracy (%)	Epochs	Correct	Wrong	B/B	M/B	M/M	B/M
Full (out)	1	97.51	146	666	17	431	4	235	13
Full (in)	1	97.65	146	667	16	431	3	236	13
Reduced (out)	3	97.36	265	665	18	431	5	234	13
Reduced (in)	3	97.51	264	666	17	431	4	235	13

The algorithm concluded in 4 steps. In each step the following are presented: which variable is removed, the number of hidden units for the particular set of input variables and the parameters used in the wavelet network, the empirical loss and the prediction risk
(in) = in-sample
(out) = out-of-sample using leave-one-out cross-validation.
B/B = case is B/WN predicts B
B/M = case is B/WN predicts M
M/M = case is M/WN predicts M
M/B = case is M/WN predicts B.

The behavior of the Mackey–Glass equation depends on the choice of the time-delay τ. The common value that it is used is $\tau = 17$ and this will also be the case in this case study, (Hsu & Tenorio, 1992; Yingwei, Sundararajan, & Saratchandran, 1997). The Mackey–Glass equation with $\tau = 17$ has a chaotic behavior and an attractor with fractal dimension $\approx$2.1. The usual function approximation approaches are disadvantageous when the fractal dimension is greater than 2, (Cao et al., 1995; Iyengar et al., 2002). In addition the usual values of the parameters are $a = 0.2$, $b = 0.1$ and $c = 10$. The series initialized at $x_0 = 0.1$. In order for the initialization transients to decay the first 4000 data points were discarded, (Platt, 1991; Yingwei et al., 1997).

The series is predicted with $\upsilon = 50$ sample steps ahead using four past samples: $x_{n-\upsilon}, x_{n-\upsilon-6}, x_{n-\upsilon-12}, x_{n-\upsilon-18}$.

9.1. Model selection

In this section the appropriate number of HUs is determined by applying the model selection algorithm presented in Section 3. In Fig. 15 the prediction risk and the empirical loss for the Mackey–Glass equation is presented. The prediction risk was estimated for the first 35 HUs. It is clear that the empirical loss decreases (almost) monotonically as the complexity of the network increases. On the other hand the prediction risk decreases (almost) monotonically until a minimum is reached and then it will start to increase (almost) monotonically. The minimum value of the prediction risk is 0.002185 and is obtained when a WN with 21 HUs is used. Hence, 21 HUs were selected for the construction of the model.

9.2. Initialization and training

After the training set and the correct topology of the WN are selected, the WN can be constructed and trained. The BE method is used to initialize the WN. A wavelet basis is constructed by scanning the 4 first levels of the wavelet decomposition of the dataset.

The initial wavelet basis consists of 254 wavelets. However, not all wavelets in the wavelet basis contribute to the approximation of the original time-series. The wavelets that contain less than 6 sample points of the training data in their support are removed. The truncated basis contains 116 wavelet candidates. The MSE after the initialization was 0.220581 and the initialization needed 0.45 s to finish. The initialization is very good and the WN converged after only 1901 iterations. The training stopped when the minimum velocity, 10^{-4}, of the training algorithm was reached. In this case, the minimum velocity was slightly increased, from 10^{-5} to 10^{-4}, in order to avoid very large training times. The MSE error after the training is 0.000299 and the total amount of time needed to train the network (initialization and training) was 39.2 s. The $\bar{R}^2 = 99.30\%$ while the *POCID* $= 89.60\%$ and *IPOCID* $= 91.23\%$. The initialization of the WN and the final approximation after the training phase are presented in Fig. 16.

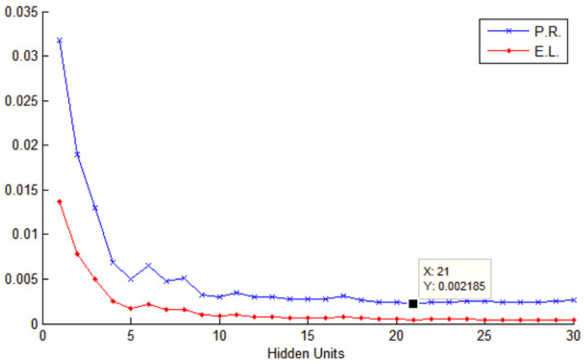


Fig. 15. The prediction risk (P.R.) and empirical loss (E.L.) for the first 30 hidden units for the Mackey–Glass equation.

9.3. Predicting the evolution of the chaotic Mackey–Glass time-series

In this section the performance of the WN out-of-sample is evaluated. The training dataset was consisting of 983 pairs while the out-of-sample dataset consists of 950 pairs. These additional data were not used for the training of the WN. The ability of the WN to forecast the evolution of the chaotic Mackey–Glass equation is presented in Fig. 17.

The WN has a very good generalization and forecasting ability. The MSE in the out-of-sample dataset is only 0.000347. Similarly, the $\bar{R}^2 = 99.14\%$ while the *POCID* $= 87.36\%$ and the *IPOCID* $= 90.41\%$ indicating that the WN can predict with great accuracy the changes in the direction of the chaotic system.

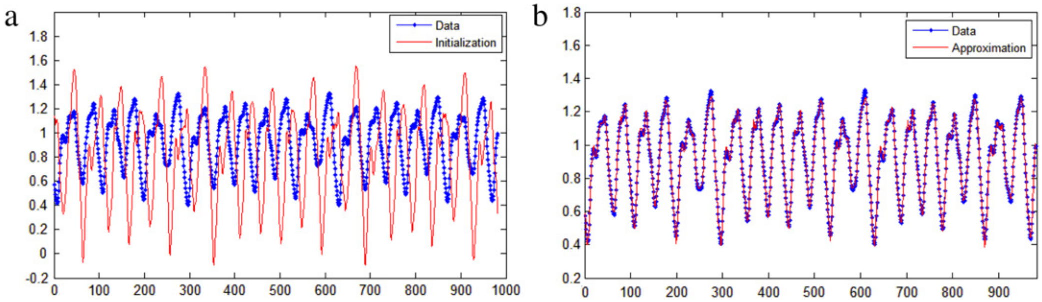


Fig. 16. The (a) initialization of the WN using the BE method and 21 HUs and (b) the approximation of the WN after the training phase for the Mackey–Glass equation.

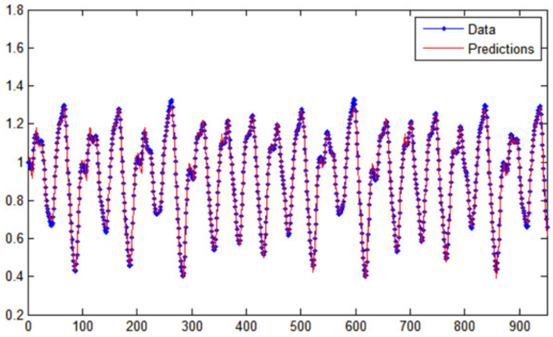


Fig. 17. Out-of-sample prediction of the WN using 21 hidden units.

9.4. Confidence and prediction intervals

After the WN is constructed and trained it can be used for prediction. Hence, confidence and prediction intervals can be constructed. In this section both confidence and prediction intervals will be constructing using the balancing method. Using the BS method 200 training sample will be created and then they will be divided in 8 groups. In each group the average output of the WNs will be estimated. Next new 1000 bootstrapped samples will be created for the 8 average outputs in order to estimate the model variance given by (35). Then the confidence intervals are estimated with level of significance $a = 5\%$. Unlike, the previous case studies and examples, in this case there is no additional noise to the underlying function. Moreover, the network fitting and prediction are very good, as it was mentioned in the previous section, with $\bar{R}^2 = 99.30\%$ in sample and $\bar{R}^2 = 99.14\%$ out-of-sample. As a result, it is expected that the variance $\sigma_p^2 = \sigma_m^2 + \sigma_\varepsilon^2$ to be very small.

Part (a) of Fig. 18 presents the confidence intervals and the true underlying function which is the Mackey–Glass equation. Since, the confidence intervals are very narrow, for clarity reasons, only a selected part is shown in Fig. 18. Its clear that the underlying function is always between the confidence intervals.

In addition, in part (b) Fig. 18 the prediction intervals for the out-of-sample dataset together with the read data and the average forecast of the WN for the 200 bootstrapped samples are presented. The PICP = 98.8%.

10. Conclusions

In this study a complete statistical framework for constructing and using WNs in various applications was presented. Although a vast literature about WNs exists, to our knowledge this is the first study that presents a step by step guide for model identification for WNs. More precisely, the following subjects were examined: the structure of a WN, training methods, initialization algorithms, model selection methods, variable significance and variable selection methods and finally methods to construct confidence and prediction intervals. Finally the partial derivatives with respect to the weights of the network, to the dilation and translation parameters as well as the derivative with respect to each input variable are presented.

Our proposed framework was tested in two simulated cases, in three real dataset consisting of daily temperatures in Berlin, daily wind speeds in New York, Wisconsin breast cancer classifications and in predicting one chaotic time-series, the Mackey–Glass equation. Our results have shown that the proposed algorithms produce stable and robust results indicating that our proposed framework can be applied in various applications.

A multidimensional WN with a linear connection of the wavelons to the output and direct connections from the input layer to the output layer is proposed. The training is performed by the classic back-propagation algorithm.

One of the advantages of WNs is the allowance of constructive algorithms for the initialization of the WN. Four initialization methods were tested. The heuristic, the RSO, the SSO and the BE method. Our results indicate that SSO and BE perform similarly and outperform the other two methods whereas BE outperforms SSO in complex problems. Using the BE and SSO the training times were reduced significantly while the network converged to the global minimum of the loss function. The BE is more efficient than the SSO algorithm however it is more computationally expensive. On the other hand in the BE algorithm the calculation of the inverse of the wavelet matrix is needed which columns might be linear dependent. In that case the SSO must be used. However since the wavelets come from a wavelet frame this is very rare to happen. It is clear that additional computational burden is added in order to initialize efficiently the WN. However the efficient initialization significantly reduces the training phase hence the total amount of computations is significant smaller than a network with random initialization.

Model selection is a very important step. A network with less HUs than needed is not able to learn the underlying function while selecting more HUs than needed the network will be over-fitted, i.e. the network will start to learn the noise. Four techniques were applied to estimate the prediction risk, the FPE, the GCV, and two sampling techniques the BS and the CV. Our results indicate that the sampling techniques give more stable results than other alternatives. BS and CV found the correct network topology in both cases. Although FPE and GCV are extensively used in finding the topology of a WN, due to the linear relation of the wavelets and the original signal, our results indicate that both criteria should not be used in complex problems. Moreover our results indicate that early stopping techniques in complex problems tend to propose more complex problems than needed.

In order to identify the significance of each explanatory variable 9 criteria were presented. These are the weights of the direct connections between the input and the output variable, 8 sensitivity criteria and one model fitness criterion. In order to statistically test whether a variable is insignificant and can be removed for the training dataset or not the distributions of these

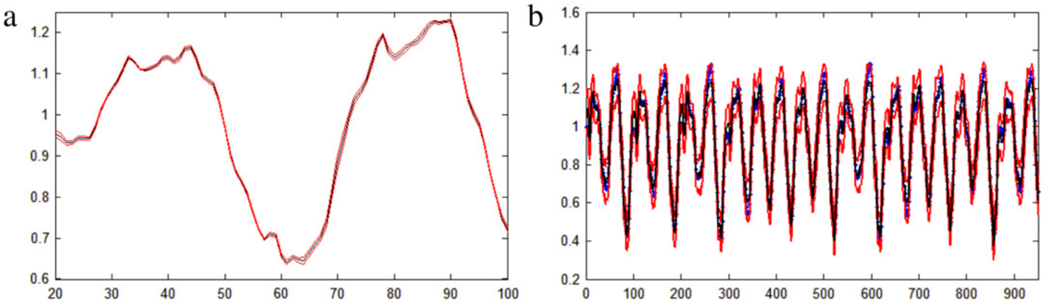


Fig. 18. In-sample confidence intervals (a) and out-of-sample prediction intervals (b) for the Mackey–Glass equation using the balancing method (PICP = 98.8%).

criteria were estimated. Our results indicate that only SBP correctly identifies the insignificant variable and produce correct and robust results in all cases. On the other hand using the AvgDM or the AvgLM the resulting p-values are inconclusive and very volatile on the bootstrapped samples. After each variable is removed it is very important to test the correctness of this decision. This can be done by checking the prediction risk or the $\bar{R}^2$ of the reduced model. In all cases, when the irrelevant variable was removed the prediction risk decreased while the $\bar{R}^2$ increased.

Next, a framework for constructing confidence and prediction intervals was presented. Two methods originating from the sigmoid NNs were adapted, the bagging and the balancing method. Our results indicate that the bagging method overestimates the model variance and as a result wider intervals are constructed. On the other hand the balancing method produces an unbiased estimator of the model variance. Our results are consistent with previous studies.

Although a framework for selecting an appropriate model was presented the adequacy of the final model must be further tested. This is usually done by examining the residuals by various criteria. However, the selection of these criteria depends on the nature of the underlying function and the assumptions made while building the model.

In order to ensure the stability and robustness of the proposed algorithms online training or synthesis of the WN was avoided. The adaption of the proposed methods to switching models and adaptive networks is left as a future research.

Appendix

A. Partial derivatives w.r.t. the bias term

$$\frac{\partial \hat{y}_p}{\partial w^{[2]}_{\lambda+1}} = 1.$$

B. Partial derivatives w.r.t. the direct connections

$$\frac{\partial \hat{y}_p}{\partial w^{[0]}_i} = x_i \quad i = 1, \ldots, m.$$

C. Partial derivatives w.r.t. the linear connections between the wavelets and the output

$$\frac{\partial \hat{y}_p}{\partial w^{[2]}_j} = \Psi_j(\mathbf{x}) \quad j = 1, \ldots, \lambda$$

D. Partial derivatives w.r.t. the translation parameters

$$\begin{aligned}\frac{\partial \hat{y}_p}{\partial w^{[1]}_{(\xi)ij}} &= \frac{\partial \hat{y}_p}{\partial \Psi_j(\mathbf{x})} \cdot \frac{\partial \Psi_j(\mathbf{x})}{\partial \psi(z_{ij})} \cdot \frac{\partial \psi(z_{ij})}{\partial z_{ij}} \cdot \frac{\partial z_{ij}}{\partial w^{[1]}_{(\xi)ij}} \\ &= w^{[2]}_j \cdot \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}) \cdot \frac{-1}{w^{[1]}_{(\zeta)ij}} \\ &= -\frac{w^{[2]}_j}{w^{[1]}_{(\zeta)ij}} \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}).\end{aligned}$$

E. Partial derivatives w.r.t. the dilation parameters

$$\begin{aligned}\frac{\partial \hat{y}_p}{\partial w^{[1]}_{(\zeta)ij}} &= \frac{\hat{y}_p}{\partial \Psi_j(\mathbf{x})} \cdot \frac{\partial \Psi_j(\mathbf{x})}{\partial \psi(z_{ij})} \cdot \frac{\partial \psi(z_{ij})}{\partial z_{ij}} \cdot \frac{\partial z_{ij}}{\partial w^{[1]}_{(\zeta)ij}} \\ &= w^{[2]}_j \cdot \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}) \cdot \frac{x_i - w^{[1]}_{(\xi)ij}}{w^{[1]\,2}_{(\zeta)ij}} \\ &= -\frac{w^{[2]}_j}{w^{[1]}_{(\zeta)ij}} \frac{x_i - w^{[1]}_{(\xi)ij}}{w^{[1]\,2}_{(\zeta)ij}} \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}) \\ &= -\frac{w^{[2]}_j}{w^{[1]}_{(\zeta)ij}} z_{ij} \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}) \\ &= z_{ij} \frac{\partial \hat{y}_p}{\partial w^{[1]}_{(\xi)ij}}.\end{aligned}$$

F. Partial derivatives w.r.t. the input variables

$$\begin{aligned}\frac{\partial \hat{y}_p}{\partial x_i} &= w^{[0]}_i + \frac{\sum_{j=1}^{\lambda} w^{[2]}_j \partial \Psi_j(\mathbf{x})}{\partial \psi(z_{ij})} \cdot \frac{\partial \psi(z_{ij})}{\partial z_{ij}} \cdot \frac{\partial z_{ij}}{\partial x_i} \\ &= w^{[0]}_i + \sum_{j=1}^{\lambda} w^{[2]}_j \cdot \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}) \cdot \frac{1}{w^{[1]}_{(\zeta)ij}} \\ &= w^{[0]}_i + \sum_{j=1}^{\lambda} \frac{w^{[2]}_j}{w^{[1]}_{(\zeta)ij}} \cdot \psi(z_{1j}) \cdots \psi'(z_{ij}) \cdots \psi(z_{mj}) \\ &= w^{[0]}_i - \sum_{j=1}^{\lambda} \frac{\partial \hat{y}_p}{\partial w^{[1]}_{(\xi)ij}}.\end{aligned}$$

References

Aczel, A. D. (1993). *Complete business statistics* (2nd ed.). IL: Irwin: Homewood.

Alexandridis, A., & Zapranis, A. (2011). Wind derivatives: modeling and pricing. In *Paper presented at the financial engineering and banking society* (F.E.B.S.).

Allingham, D., West, M., & Mees, A. I. (1998). Wavelet reconstruction of nonlinear dynamics. *International Journal of Bifurcation and Chaos*, *8*(11), 2191–2201.

Anders, U., & Korn, O. (1999). Model selection in neural networks. *Neural Networks*, *12*(2), 309–323.

Bashir, Z., & El-Hawary, M. E. (2000). Short term load forecasting by using wavelet neural networks. In *Paper presented at the the proc. of Canadian conference on electrical and computer engineering*.

Becerikli, Y. (2004). On three intelligent systems: dynamic neural, fuzzy and wavelet networks for training trajectory. *Neural Computation and Applications*, *13*, 339–351.

Becerikli, Y., Oysal, Y., & Konar, A. F. (2003). On a dynamic wavelet network and its modeling application. In *Lecture Notes in Computer Science*: *2714* (pp. 710–718).

Benaouda, D., Murtagh, G., Starck, J.-L., & Renaud, O. (2006). Wavelet-based nonlinear multiscale decomposition model for electricity load forecasting. *Neurocomputing*, *70*, 139–154.

Bernard, C., Mallat, S., & Slotine, J.-J. (1998). Wavelet interpolation networks. In *Paper presented at the the proc. of ESANN'98*.

Billings, S. A., & Wei, H.-L. (2005). A new class of wavelet networks for nonlinear system identification. *IEEE Transactions on Neural Networks*, *16*(4), 862–874.

Breiman, I. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Cannon, M., & Slotine, J.-J. E. (1995). Space-frequency localized basis function networks for nonlinear system estimation and control. *Neurocomputing*, *9*, 293–342.

Cao, L., Hong, Y., Fang, H., & He, G. (1995). Predicting chaotic time series with wavelet networks. *Physica D*, *85*, 225–238.

Carney, J. G., Cunningham, P., & Bhagwan, U. (1999). Confidence and prediction intervals for neural network ensembles. In *Paper presented at the IJCNN'99*.

Chen, Y., Billings, S. A., & Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identifcation. *International Journal of Control*, *50*, 1873–1896.

Chen, Y., Cowan, C., & Grant, P. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions in Neural Networks*, *2*(1991), 302–309.

Chen, Y., Yang, B., & Dong, J. (2006). Time-series prediction using a local linear wavelet neural wavelet. *Neurocomputing*, *69*, 449–465.

Cristea, P., Tuduce, R., & Cristea, A. (2000). Time series prediction with wavelet neural networks. In *Paper presented at the the proc. of 5th seminar on neural network applications in electrical engineering*.

Curry, B., & Morgan, P. H. (2006). Model selection in neural networks: some difficulties. *European Journal of Operational Research*, *170*(2), 567–577.

Daubechies, I. (1992). *Ten lectures on wavelets*. Philadelphia, USA: SIAM.

De Veaux, D. R., Schumi, J., Schweinsberg, J., & Ungar, H. L. (1998). Prediction intervals for neural networks via nonlinear regression. *Technometrics*, *40*(4), 273–282.

Dimopoulos, Y., Bourret, P., & Lek, S. (1995). Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, *2*(6), 1–4.

Duch, W., & Adamczak, R. (1998). Statistical methods for construction of neural networks. In *Paper presented at the 5th international conference on neural inf proceedings*.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. USA: Chapman & Hall.

Fang, Y., & Chow, T. W. S. (2006). *Lecture notes in computer science*: *vol. 3971*. *Wavelets based neural network for function approximation* (pp. 80–85).

Gao, R., & Tsoukalas, H. I. (2001). Neural-wavelet methodology for load forecasting. *Journal of Intelligent & Robotic Systems*, *31*, 149–157.

Hassanien, A., & Ali, J. (2006). Rough set approach for classification of breast cancer mammogram images. In V. Di Ges, F. Masulli, & A. Petrosino (Eds.), *Fuzzy logic and applications, Vol. 2955* (pp. 224–231). Heidelberg: Springer Berlin.

He, Y., Chu, F., & Zhong, B. (2002). A hierarchical evolutionary algorithm for constructing and training wavelet networks. *Neural Computing & Applications*, *10*, 357–366.

Heskes, T. (1997). Practical confidence and prediction intervals. In *Paper presented at the advances in neural information processing systems 9*.

Hsu, W., & Tenorio, M. F. (1992). Plastic network for predicting the Mackey–Glass time series. In *Paper presented at the international joint conference on neural networks, IJCNN*. Baltimore, MD, USA.

Iyengar, S. S., Cho, E. C., & Phoha, V. V. (2002). *Foundations of wavelet networks and applications*. USA: CRC Press.

Jiao, L., Pan, J., & Fang, Y. (2001). Multiwavelet neural network and its approximation properties. *IEEE Transactions on Neural Networks*, *12*(5), 1060–1066.

.Juditsky, A., Zhang, Q., Delyon, B., Glorennec, P.-Y., & Benveniste, A. (1994). Wavelets in identification — wavelets, splines, neurons, fuzzies: how good for identification?.

Kadambe, S., & Srinivasan, P. (2006). Adaptive wavelets for signal classification and compression. *International Journal of Electronics and Communications*, *60*, 45–55.

Kan, K.-C., & Wong, K. W. (1998). Self-construction algorithm for synthesis of wavelet networks. *Electronic Letters*, *34*, 1953–1955.

Khayamian, T., Ensafi, A. A., Tabaraki, R., & Esteki, M. (2005). Principal component-wavelet networks as a new multivariate calibration model. *Analytical Letters*, *38*(9), 1447–1489.

Li, S. T., & Chen, S.-C. (2002). Function approximation using robust wavelet neural networks. In *Paper presented at the the proc. of ICTAI'02*.

Mackey, M. C., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, *197*(4300), 287–289.

Mallat, S. G. (1999). *A wavelet tour of signal processing*. San Diego: Academic Press.

Mangasarian, O. L., & Wolberg, W. H. (1990). Cancer diagnosis via linear programming. *SIAM News*, *23*(5), 1–18.

Mellit, A., Benghamen, M., & Kalogirou, S. A. (2006). An adaptive wavelet-network model for forecasting daily total solar-radiation. *Applied Energy*, *83*, 705–722.

Mitra, P., Murthy, C. A., & Pal, S. K. (2002). Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), 301–312.

Moody, J. E. (1992). The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems. In J. E. Moody, S. J. Hanson, & R. P. Lippman (Eds.), *Advances in neural information processing systems Vol. 4*. San Mateo CA: Morgan Kaufmann.

Moody, J. E., & Utans, J. (1992). Principled architecture selection for neural networks: application to corporate bond rating prediction. In *Paper presented at the neural networks in the capital markets*.

Oussar, Y., & Dreyfus, G. (2000). Initialization by selection for wavelet network training. *Neurocomputing*, *34*, 131–143.

Oussar, Y., Rivals, I., Presonnaz, L., & Dreyfus, G. (1998). Trainning wavelet networks for nonlinear dynamic input output modelling. *Neurocomputing*, *20*, 173–188.

Papadopoulos, G., Edwards, J. P., & Murray, F. A. (2000). Confidence estimation methods for neural networks: a comparison. In *Paper presented at the ESANN*.

Pati, Y. C., & Krishnaprasad, P. S. (1993). Analysis and synthesis of feedforward neural networks using discrete affine wavelet transforms. *IEEE Transactions on Neural Networks*, *4*(1), 73–85.

Pittner, S., Kamarthi, S. V., & Gao, Q. (1998). Wavelet networks for sensor signal classification in flank wear assessment. *Journal of Intelligent Manufacturing*, *9*, 315–322.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, *3*, 213–225.

Postalcioglu, S., & Becerikli, Y. (2007). Wavelet networks for nonlinear system modelling. *Neural Computing & Applications*, *16*, 434–441.

Reed, R. (1993). Prunning algorithms — a survey. *IEEE Trans Neural Networks*, *4*, 740–747.

Samarasinghe, S. (2006). *Neural networks for applied sciences and engineering*. New York, USA: Taylor & Francis Group.

Satchwell, C. (1994). Neural networks for stochastic problems: more than one outcome for the input space. In *Paper presented at the NCAF conference*.

Senapati, M., Mohanty, A., Dash, S., & Dash, P. (2011). Local linear wavelet neural network for breast cancer recognition. *Neural Computing & Applications*, 1–7.

Setiono, R., & Liu, H. (1997). Neurolinear: from neural networks to oblique decision rules. *Neurocomputing*, *17*(1), 1–24.

Subasi, A., Alkan, A., Koklukaya, E., & Kiymik, M. K. (2005). Wavelet neural network classification of eeg signals by using ar model with mle pre-processing. *Neural Networks*, *18*, 985–997.

Szu, H., Telfer, B., & Kadambe, S. (1992). Neural network adaptive wavelets for signal representation and classification. *Optical Engineering*, *31*, 1907–1916.

Ulugammai, M., Venkatesh, P., Kannan, P. S., & Padhy, N. P. (2007). Application of bacterial foraging technique trained artificial and wavelet neural networks in load forecasting. *Neurocomputing*, *70*, 2659–2667.

Wei, H.-L., & Billings, S. A. (2007). Feature subset selection and ranking for data dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(1), 162–166.

Wei, H.-L., Billings, S. A., & Liu, J. (2004). Term and variable selection for non-linear system identification. *International Journal of Control*, *77*(1), 86–110.

Wong, K.-W., & Leung, A. C.-S. (1998). On-line successive synthesis of wavelet networks. *Neural Processing Letters*, *7*, 91–100.

Xu, J., & Ho, D. W. C. (1999). Adaptive wavelet networks for nonlinear system identification. In *Paper presented at the the proc. of american control conference.* San Diego, USA.

Xu, J., & Ho, D. W. C. (2002). A basis selection algorithm for wavelet neural networks. *Neurocomputing*, *48*, 681–689.

Xu, J., & Ho, D. W. C. (2005). A constructive algorithm for wavelet neural networks. In *Lecture notes in computer science (3610)*, (pp. 730–739).

Yao, S. J., Song, Y. H., Zhang, L. Z., & Cheng, X. Y. (2000). Wavelet transform and neural networks for short-term electrical load forecasting. *Energy Conversion & Management*, *41*, 1975–1988.

Yao, X. (1999). Evolving artificial neural networks. *Proc IEEE*, *87*(9), 1423–1447.

Yingwei, L., Sundararajan, N., & Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, *9*(2), 461–478.

Zapranis, A., & Alexandridis, A. (2008). Modelling temperature time dependent speed of mean reversion in the context of weather derivetive pricing. *Applied Mathematical Finance*, *15*(4), 355–386.

Zapranis, A., & Alexandridis, A. (2009). Weather derivatives pricing: modelling the seasonal residuals variance of an ornstein–uhlenbeck temperature process with neural networks. *Neurocomputing*, *73*, 37–48.

Zapranis, A., & Refenes, A. P. (1999). *Principles of neural model indentification, selection and adequacy: with applications to financial econometrics*. Springer-Verlag.

Zapranis, A. D., & Livanis, E. (2005). Prediction intervals for neural network models. In *Paper presented at the proceedings of the 9th WSEAS international conference on computers*.

Zhang, Q. (1993). Regressor selection and wavelet network construction.

Zhang, Q. (1994). Using wavelet network in nonparametric estimation (No. 2321): Techincal Report, INRIA.

Zhang, Q. (1997). Using wavelet network in nonparametric estimation. *IEEE Transactions on Neural Networks*, *8*(2), 227–236.

Zhang, Q., & Benveniste, A. (1992). Wavelet networks. *IEEE Transactions on Neural Networks*, *3*(6), 889–898.

Zhang, Z. (2007). Learning algorithm of wavelet network based on sampling theory. *Neurocomputing*, *71*(1), 224–269.

Zhang, Z. (2009). Iterative algorithm of wavelet network learning from nonuniform data. *Neurocomputing*, *72*, 2979–2999.

Zhao, J., Chen, B., & Shen, J. (1998). Multidimensional non-orthogonal wavelet-sigmoid basis function neural network for dynamic process fault diagnosis. *Computers and Chemical Engineering*, *23*, 83–92.

Ζαπράνης, Α. (2005). *Χρηματοιοκονομική και Νευρωνικά Συστήματα*. Αθήνα: Κλειδάριθμος.