Iterated Time Series Prediction with Ensemble Models

Jörg D. Wichard and Maciej Ogorzałek
Department of Electrical Engineering
AGH University of Science and Technology
al. Mickiewicza 30
30-059 Kraków, Poland
email: JoergWichard@web.de maciej@agh.edu.pl

ABSTRACT
We describe the use of ensemble methods to build proper models time series prediction. Our approach extends the classical ensemble methods for neural networks by using several different model architectures. We further suggest an iterated prediction procedure to select the final ensemble members. This is an extension of well know the cross-validation scheme for model validation.

KEY WORDS
Modelling, Time Series Prediction, Ensemble Models

1 Introduction

Ensemble building is a common way to improve the performance of the resulting model for classification and regression tasks, since it was noticed that an ensemble of individual predictors performs better than a single predictor in the average. This is based on the bias-variance decomposition of ensemble models which is described in detail in the next section.

Usually an ensemble consists of models taken from one single class, e.g. neural networks [1, 2, 3, 4], support vector machines [5] or regression trees [6].

We suggest a different strategy. We train several models from different model classes and combine them to build the final ensemble. This is done in order to introduce model diversity which is the central feature of the ensemble approach [3].

The novelty of our approach consists of building heterogeneous ensembles with several model classes combined with an iterated prediction scheme for final model selection. The models that we use to build the ensemble are presented in Section 3. In Section 4.1 we describe the model training and selection. We demonstrate the performance of our method on two benchmark problems in Section 5.

2 Ensembles

If we average the output of several different models $f_i(\mathbf{x})$, we call it an ensemble model

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{K} \omega_i f_i(\mathbf{x}). \qquad (1)$$

We further assume that the model weights ω_i sum to one $\sum_{i=1}^{K} \omega_i = 1$. The idea of averaging different models was developed in the neural network community [1, 2] and later it was pointed out by Krogh et al. [3], that the generalization error of the ensemble is lower than the mean of the generalization error of the single ensemble members. This holds in general, independent of the model class.

It is also known, that the generalization error of an ensemble model could be improved if the predictors on which averaging is done disagree and if their fluctuations are uncorrelated [7].

To see this we have to investigate the contribution of a single model in the ensemble to the generalization error. We consider the case where we have a given data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and we want to find a model $\hat{f}(\mathbf{x})$ that approximates y at future observations of $\mathbf{x}$. These future observations are assumed to come from the same source that generated the training set D, i.e. from the same (unknown) probability distribution P. It should be noted that f depends also on D. The expected generalization error $Err(\mathbf{x}, D)$ given a particular $\mathbf{x}$ and a training set D is

$$Err(\mathbf{x}, D) = E[(y - \hat{f}(\mathbf{x}))^2 | \mathbf{x}, D], \qquad (2)$$

where the expectation $E[\cdot]$ is taken with respect to the probability distribution P. We are now interested in

$$Err(\mathbf{x}) = E_D[Err(\mathbf{x}, D)],$$

where the expectation $E_D[\cdot]$ is taken with respect to all possible realizations of training sets D with fixed sample size N. According to [8] the bias/variance decomposition is

$$
\begin{aligned}
Err(\mathbf{x}) &= \sigma^2 + (E_D[\hat{f}(\mathbf{x})] - E[y|\mathbf{x}])^2 \\
&\quad + E_D[(\hat{f}(\mathbf{x}) - E_D[\hat{f}(\mathbf{x})])^2] \qquad (3) \\
&= \sigma^2 + (Bias(\hat{f}(x)))^2 + Var(\hat{f}(x)),
\end{aligned}
$$

where $E[y|\mathbf{x}]$ is the deterministic part of the data and σ^2 is the variance of y given $\mathbf{x}$.

Important for our consideration is the variance term, which could be decomposed in the following way:

$$Var(\hat{f}) = E\left[(\hat{f} - E[\hat{f}])^2\right] \qquad (4)$$

$$= E[(\sum_{i=1}^{K} \omega_i f_i)^2] - (E[\sum_{i=1}^{K} \omega_i f_i])^2$$

$$= \sum_{i=1}^{K} \omega_i^2 \left(E\left[f_i^2\right] - E^2\left[f_i\right] \right)$$

$$+ 2 \sum_{i<j} \omega_i \omega_j \left(E\left[f_i f_j\right] - E\left[f_i\right] E\left[f_j\right] \right),$$

where the expectation is taken with respect to D. The first sum in equ. (4) marks the lower bound of the ensemble variance and is the weighted mean of the variances of the ensemble members. The second sum contains the cross-terms of the ensemble members and vanishes if the models are completely uncorrelated [7]. That shows that the reduction in the variance of the ensemble is related to the degree of independence of the single models [4].

There are several ways to introduce model diversity to the ensemble in order to decorrelate the output of the individual ensemble members. A general approach is to train various models on selected subsets of the training data [3, 6] or to initiate the training algorithm with randomly chosen initial conditions [4]. A new approach was recently introduced by Bakker et al. [9], where the ensemble consists of representative models that are selected by clustering the model outputs.

We introduce model diversity in such a way, that we train several model classes on different subsets of the training data, using an extended cross validation scheme. An overview of the different model classes that we use for ensemble building is given in the next section.

3 Models

In this section we give a short overview about the models that we use for ensemble building and the related training procedures. The implementation of these models together with a more detailed description can be found in [10].

3.1 Linear and Polynomial Models

The d-dimensional linear model has the form $f(\vec{x}) = a_0 + \sum_{i=1}^{d} a_i x_i$, where a_0 is the offset. The model is trained on randomly selected subsets of the training data, where 20% of the training set is used for model validation and the best 66% of the validated models are used to build the average linear model. The coefficients $a_{i=0,\dots,d}$ are calculated with a standard least square method [11].

The polynomial model is given by $f(\vec{x}) = \sum_{i=1}^{P} a_i p_i(\vec{x})$, wherein the monoms have the form $p_i(\vec{x}) = \prod_{i=1}^{d} x_i^{n_i}$. We use an iterative term selection for the monoms wherein we add successively the terms to the polynomial model that decrease the out-of-sample error on a subset of the training data.

3.2 Nearest Neighbor Models

A k-Nearest-Neighbor model takes a weighted average over those observations z_i in the training set that are closest to the query point $\vec{x}$. This is,

$$f(\vec{x}) = \frac{1}{\sum w_i} \sum_{\vec{z}_i \in N_k(\vec{x})} w_i z_i, \qquad (5)$$

where $N_k(\vec{x})$ denotes the k-element neighborhood of $\vec{x}$, defined in a given metric. Common choices are the L_1, L_2 and the L_∞ metrics. To compensate for irrelevant input dimensions, distances are computed using a weighted metric:

$$d(\vec{x}, \vec{z}) = (\sum_{i=1}^{D} m_i(x_i - z_i)^M)^{\frac{1}{M}} \quad 0 \le m_i \le 1. \qquad (6)$$

The vector of metric coefficients $\vec{m}$ is adapted by a Genetic Algorithm. One vector of metric coefficients is an individual of the population. The fitness value is assigned to each individual according to it's error on the training data set. For our investigation we used the fast nearest neighbor algorithm ATRIA [12].

3.3 Nearest Trajectory Models

The nearest trajectory model is based on a strategy for time series prediction introduced by McNames [13]. It is based on the assumption that the time series stems from a dynamical system and the states can be reconstructed with a time delay embedding, which is possible for a large class of systems [14, 15, 16].

The nearest trajectory model looks for the nearest trajectory segments in the reconstructed state space instead of the nearest neighbors. The prediction is done with a local linear model of the closest trajectory points as described in [13]. The number of neighboring trajectories is chosen randomly at the start of the training algorithm.

3.4 Neural Networks

We use a multilayer feed-forward neural network (MLP: Multi Layer Perceptron) with the $\tanh(\vec{x})$ as nonlinear element. In order to increase the ensemble ambiguity, we initialize the weights with Gaussian distributed random numbers having zero mean and scaled variances, following a suggestion of LeCun et al. [17]. The number of hidden layers is chosen at random to be one or two and the numbers of neurons in also random (3-9 Neurons in the first layer, 4-32 in second layer). We use two different training procedures: A first order training algorithm based on the Rprop Algorithm [18] with the improvements given in [19]; The second order training algorithm is a Levenberg Marquart Gradient Descent [17]. As regularization method we use the common weight decay with the penalty term

$$P(\vec{w}) = \lambda \sum_{i=1}^{N} \frac{w_i^2}{1 + w_i^2}, \qquad (7)$$

where $\vec{w}$ denotes the N-dimensional weight vector of the MLP and the regularization parameter is small $\lambda = 0.001$.

3.5 Perceptron Radial Basis Net

Perceptron Radial Basis Net (PRBFN) is and extension of MLP and Radial Basis Function (RBF) Networks. The PRBFN combine RBF and sigmoid units in the hidden layers. This hybrid network architecture together with a sophisticated training procedure was introduced by Cohen and Intrator [20]. The number and the centers of the hidden units are generated dynamically during the training and network parameters are refined by gradient descent, as described in [20].

4 Model Building

4.1 Iterated Prediction of Time Series

If we consider an equidistant sampled time series $\{x_\nu\}_{\nu=1,...,N}$, we can construct a d-dimensional state space vector $\vec{x}_n$ in the form

$$\vec{x}_n = (x_{(n-\lambda(d-1))}, x_{(n-\lambda(d-2))}, \ldots, x_n), \qquad (8)$$

where λ denotes the time lag. A "one-step ahead prediction" model $f(\vec{x})$ for iterated time series prediction has the form

$$\begin{aligned} f : \mathbf{R}^d &\rightarrow \mathbf{R} \\ f(\vec{x}_n) &= x_{n+1}. \end{aligned} \qquad (9)$$

We perform the iterated prediction in such a way, that we use the predicted value x_{n+1} to construct the next state space vector $\vec{x}_{n+1}$ which is used to predict the next time series sample x_{n+2} and so on. In this sense we define the 1-step ahead prediction as f^1, the 2-step ahead prediction as f^2 and the n-step ahead prediction as f^n.

In the field of nonlinear time series analysis, this method was suggested by Farmer and Sidorowich [21] in order to make short term predictions of chaotic systems.

A crucial question is the choice of the model class. If we have no prior knowledge about the nature of the process that generated the time series, we have to find a proper model architecture with an out of sample test. This is done during the model training. We train several models of different model classes and select the best ones regarding the iterated prediction error (see section 4.2).

4.2 Model Training and Cross Validation

In order to select models for the final ensemble we use a cross-validation scheme for model training (see [22] for a detailed discussion of the method). The cross validation is done in several training rounds on different subsets of the entire data set. In every training round the data is divided

in a training set and a test set, which is used for selecting the models for the final ensemble. Therefor we take the time series and build a data set M of input-out pairs $(\vec{x}_i, y_i)$, where the inputs are the state space vectors defined in equation 8 and the outputs are the one step ahead time series values $y_i = x_{i+1}$ from equation 9. We extract contiguous parts of length n from the input-out pairs as test sets M_{test} with

$$M_{test} = \{(\vec{x}_{i+\mu}, y_{i+\mu})\}_{\mu=1,...,n} \qquad (10)$$

to calculate the n-step iterated prediction error. The remaining part of the data M_{train} is used to train the models. In every training round of the cross-validation procedure, we use a set of different models that are initiated with randomly chosen parameters[1]. Each model is now trained to minimize the one step ahead prediction error

$$E_{train} = \sum_i (y_i - f(\vec{x}_i))^2 \qquad (11)$$

where the sum is taken over all members of the training set M_{train}. After the training each model fits (more or less) the data.

Now we want to prevent over-fitting and select the best model for the iterated prediction method. For that reason we kept the contiguous test set M_{test}. We calculate for every trained model the mean squared error (MSE) for the n-step ahead cross-validation

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - f^i)^2, \qquad (12)$$

where f^i is the i-step ahead prediction, starting with the first sample of the test set. We choose the model with the smallest error to become a member of the final ensemble.

If we repeat this procedure K times, we get K different models, trained and tested on different parts of the entire data set. These models are used to build the final ensemble

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{K} \frac{1}{K} f_i(\mathbf{x}). \qquad (13)$$

We use equal weights for each model, but other choices are also possible [2].

5 Numerical Simulations

Two data sets are used to illustrate the proposed method. The Laser Data and the Circuit data were part of to time series prediction competitions. The continuations of the time series were also made public, to compare the results of the different modeling strategies (see [23, 13] for further descriptions.

---

[1] These parameters are related to the certain model classes, for example the number of nearest neighbors, the maximal order of monoms in a polynomial model or the number of neurons in the hidden layers of the MLP.

5.1 Measure of Accuracy

We use the normalized mean squared error (NMSE) of the multi-step ahead prediction as a accuracy measure of the trained models (see [13] and the references therein). The error is calculated on contiguous parts of the data sets the are 'out-of-sample' (OOS) data in the way, that they were neither used to train the models nor to select them. The NMSE is defined as

$$NMSE(\tau) = \frac{\frac{1}{n_\mu} \sum_{i=1}^{n_\mu} (y_{\mu_i+\tau} - f^\tau(\vec{x}_{\mu_i}))^2}{\sigma^2(y)}, \quad (14)$$

where $y_{\mu_i+\tau}$ is the $\mu_i + \tau$ point in the OOS data set, μ_i is the first point in the i-th contiguous validation segment and n_μ is the overall number of validation sets. The τ-step ahead prediction starting with the point $\vec{x}_{\mu_i}$ is denoted as $f^\tau(\vec{x}_{\mu_i})$ and $\sigma^2(y)$ is simply the variance of the data

$$\sigma^2(y) = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 . \quad (15)$$

The NMSE as a function of the prediction length τ has a vivid interpretation. If the NMSE reaches lies near 1.0, the model has lost its predictive power and we can also use the mean of the time series as a simple predictor. The point were the NMSE reaches 1.0 marks the prediction horizon.

5.2 Laser Data

We applied our method to an experimental time series from a pumped far-infrared NH$_3$-Laser that was part of the Santa Fe Time Series Prediction and Analysis Competition in 1991 (see [23] and the references therein). The data set for the model training consists of 1000 points. We build an ensemble model with the parameters listed in table 1. The

Parameter	Value
Data points	1000
Cross-Validation rounds	60
Embedding Dimension d	30
Time Lag λ	1
Prediction steps for model selection	50
OOS-Validation sets n_μ	150

Table 1. Parameters of the ensemble model for the NH$_3$-Laser data.

final ensemble consist of 60 single models, according to the 60 cross-validation rounds. In detail we found 41 nearest trajectory models, 12 nearest neighbor models, 3 neural networks and 4 PRBFN-networks. The 150-step ahead prediction is shown in Figure 1. The most difficult part of this modeling task is to predict the breakdown of the amplitude that appears to cycles earlier than predicted with our model. As far as we know, there is no serious modeling approach that has solved this problem convincingly.
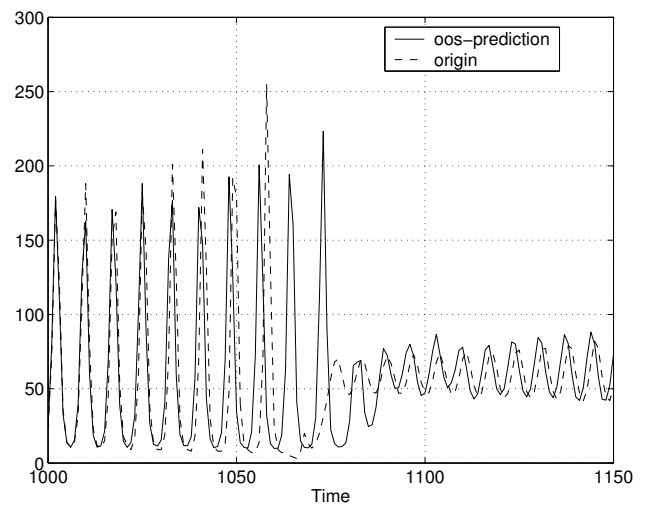

Figure 1. The 150-step prediction of the NH$_3$-Laser data and the original time series. The parameters of the ensemble are listed in Table 1. The model was able to accurately predict approximately 50 steps ahead. This is worse than the average prediction horizon which is around 100 steps.

Parameter	Value
Data points	2000
Cross-Validation rounds	30
Embedding Dimension d	50
Time Lag λ	1
Prediction steps for model selection	200
OOS-Validation sets n_μ	200

Table 2. Parameters of the ensemble model for Chua's Circuit.

Our model was able to predict the dynamics of the first 50 steps. Considering the NMSE we expect a prediction horizon of approximately 100 time steps. This indicates once more, that the breakdown of the amplitude is hard to predict, while the slowly growing cycles are relatively good natured.

5.3 Chua's Circuit

The data set was part of the time series competition of the International Workshop on *Advanced Black-Box Techniques for Nonlinear Modeling* in 1998 in Leuven, Belgium [24]. It stems from a nonlinear transform of a 5-scroll generalized Chua's Circuit (see [25] for a detailed description). The data set consists of 2000 points. We build an ensemble model, consisting of the models listed in Section 3 with the parameters shown in Table 2. The final ensemble consist of 30 single models, according to the 30 cross-validation rounds. In detail we found 15 nearest trajectory models, 7 nearest neighbor models, 6 neural networks and 2 PRBFN-networks. The 200-step ahead prediction is shown in Fig-
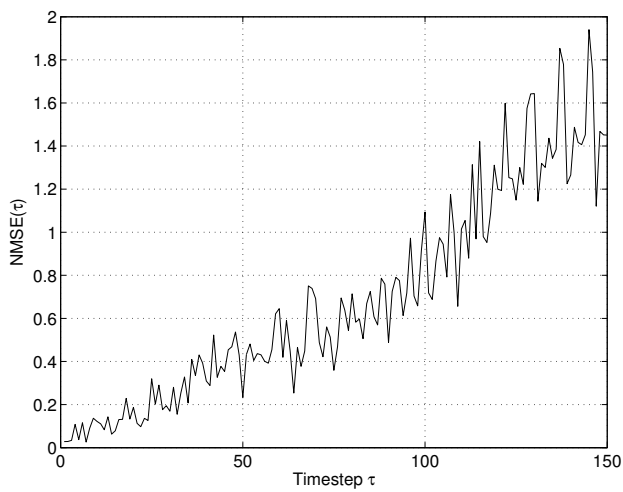
Figure 2. The NMSE versus τ, the number of iterated prediction steps for the NH$_3$-Laser data. The average prediction horizon of the ensemble is approximately 100 time steps.

ure 3.

6 Conclusion

We have demonstrated that heterogeneous ensembles together with an iterated prediction procedure for model selection provides a powerful tool for time series prediction. The performance on benchmark problems shows that this method can compete against the methods that are known to perform well on this data sets.

Acknowledgments

References

[1] L.K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[2] M. P. Perrone and L. N. Cooper, "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks," in *Neural Networks for Speech and Image Processing*, R. J. Mammone, Ed., pp. 126–142. Chapman-Hall, 1993.
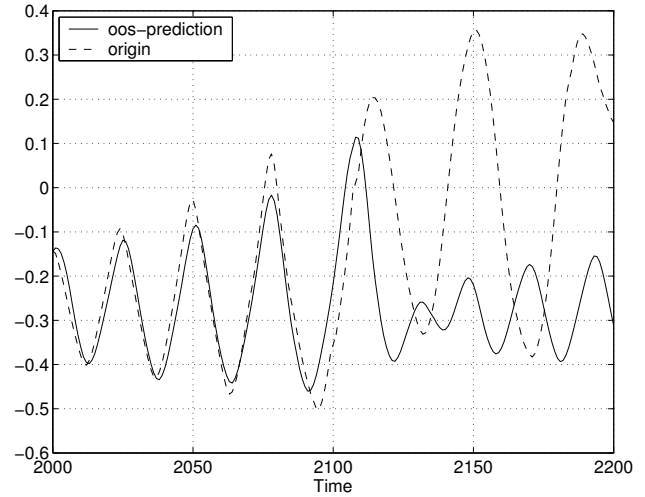
Figure 3. The 200-step prediction of the data from Chua's Circuit and the original time series. The parameters of the ensemble are listed in Table 2. In this case the ensemble model was able to predict the dynamics of the first 100 time steps. This is more than the average prediction horizon of the ensemble.
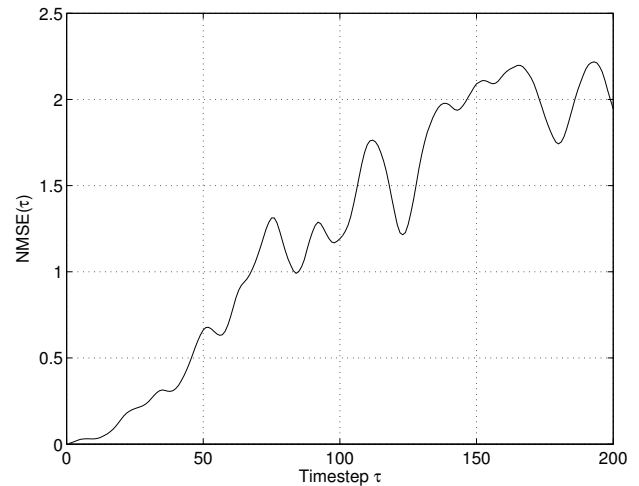

Figure 4. The NMSE versus τ, the number of iterated prediction steps for the the data from Chua's Circuit. The average prediction horizon of the ensemble is approximately 60 time steps.

[3] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds. 1995, vol. 7, pp. 231–238, The MIT Press.

[4] U. Naftaly, N. Intrator, and D. Horn, "Optimal ensemble averaging of neural networks," *Network, Comp. Neural Sys.*, vol. 8, pp. 283–296, 1997.

[5] G. Valentini and T.G. Dietterich, "Bias-variance analysis and ensembles of svm," in *Third International Workshop on Multiple Classifier Systems*, J. Kittler and F. Roli, Eds., vol. 2364 of *Lecture Notes in Computer Science*, pp. 222–231. Springer Verlag, New York, 2002.

[6] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[7] A. Krogh and P. Sollich, "Statistical mechanics of ensemble learning," *Physical Review E*, vol. 55, no. 1, pp. 811–825, 1997.

[8] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[9] B. Bakker and T. Heskes, "Clustering ensembles of neural network models," *Neural Networks*, vol. 16, no. 2, pp. 261–269, 2003.

[10] C. Merkwirth and J.D. Wichard, "ENTOOL - A Matlab Toolbox for Ensemble Modeling," 2002.

[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C : The Art of Scientific Computing*, Cambridge University Press, Cambridge, 2 edition, 1992.

[12] C. Merkwirth, U. Parlitz, and W. Lauterborn, "Fast exact and approximate nearest neighbor searching for nonlinear signal processing," *Phys. Rev. E*, vol. 62, no. 2, pp. 2089–2097, 2000.

[13] J. McNames, "A nearest trajectory strategy for time series prediction," in *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling*. K. U. Leuven Belgium, 1998.

[14] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence*, Lect. Notes Math. Sringer-Verlag, Berlin, 1981.

[15] T. Sauer, J.A. Yorke, and M. Casdagli, "Embedology," *J.Stat.Phys.*, vol. 65, pp. 579–618, 1991.

[16] J. Stark, D.S. Broomhead, M.E. Davies, and J. Huke, "Takens embedding theorems for forced and stochastic systems," *Nonlinear Analysis*, vol. 30, pp. 5303–5314, 1997.

[17] Y. LeCun, L. Bottou, G. Orr, and K. Mller, "Efficient BackProp," in *Neural Networks: Tricks of the trade*, G. Orr and K. Mller, Eds., vol. 1524 of *Lecture Notes in Computer Science*, pp. 9–50. Springer Verlag, 1998.

[18] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. of the IEEE Intl. Conf. on Neural Networks*, San Francisco, CA, 1993, pp. 586–591.

[19] C. Igel and M. Hüsken, "Improving the Rprop Learning Algorithm," in *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, H. Bothe and R. Rojas, Eds. 2000, pp. 115–121, ICSC Academic Press.

[20] S. Cohen and N. Intrator, "Automatic model selection in a hybrid perceptron/radial network," in *Multiple Classifier Systems*, F. Roli J. Kittler, Ed., vol. 2096 of *Lecture Notes in Computer Science*, pp. 440–454. Springer Verlag, 2001.

[21] J.D. Farmer and J.J. Sidorowich, "Predicting chaotic time series," *Phys. Rev. Lett.*, vol. 59(8), pp. 845 – 848, 1987.

[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics. Springer-Verlag, 2001.

[23] A.S. Weigend and N.A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley, Reading, MA, 1994.

[24] K. U. Leuven Belgium, *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling*, 1998.

[25] J. McNames, J. Suykens, and J. Vandewalle, "Winning Entry of the K.U. Leuven Time Series Prediction Competition," *International Journal of Bifurcation and Chaos*, vol. 9, no. 8, pp. 1485–1500, 1999.