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 for 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.**

I. 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 [1]. Usually an ensemble consists of models taken from one single class, e.g. neural networks [2], [3], [4], [5], support vector machines [6] or regression trees [7].
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 [4].
The novelty of our approach consists of building heterogeneous ensembles with several model classes combined with an iterated prediction scheme for final model selection. For the CATS Benchmark we propose a combined model strategy in order to cope with the different time scales of the data set. This paper is organized as follows. In the next Section we briefly introduce basic concepts of ensemble models. In Section III we present the models that we use to build the ensemble and in Section IV-A we describe the model training and model selection. In the end we demonstrate the results in section V.

II. Ensembles

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

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

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 [2], [3] and later it was pointed out by Krogh et al. [4], 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 [1]. 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

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

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

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

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:

$$\begin{aligned} Var(\hat{f}) &= E\left[(\hat{f} - E[\hat{f}])^2\right] \qquad (4) \\ &= E[(\sum_{i=1}^{K} \omega_i f_i)^2] - (E[\sum_{i=1}^{K} \omega_i f_i])^2 \\ &= \sum_{i=1}^{K} \omega_i^2 \left(E\left[f_i^2\right] - E^2\left[f_i\right]\right) \\ &\quad + 2\sum_{i<j} \omega_i \omega_j \left(E\left[f_i f_j\right] - E\left[f_i\right] E\left[f_j\right]\right), \end{aligned}$$

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 [1]. That shows that the reduction in the variance of the ensemble is related to the degree of independence of the single models [5].
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 [4], [7] or to initiate the training algorithm with randomly chosen initial conditions [5]. 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.

III. 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].

A. Linear and Polynomial Models

The d-dimensional linear model has the form

$$f(\vec{x}) = a_0 + \sum_{i=1}^{d} a_i x_i, \tag{5}$$

where a_0 is the offset. The coefficients are calculated with the standard method for ridge regression (see Hastie et al. [11] for a detailed description). The optimal ridge parameter is evaluated by performing a cross-validation on the training data.
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.

B. 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,

$$f(\vec{x}) = \frac{1}{\sum w_i} \sum_{\vec{z}_i \in N_k(\vec{x})} w_i z_i, \tag{6}$$

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:

$$d(\vec{x}, \vec{z}) = (\sum_{i=1}^{D} m_i (x_i - z_i)^M)^{\frac{1}{M}} \quad 0 \leq m_i \leq 1. \tag{7}$$

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].

C. 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.

D. 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

$$P(\vec{w}) = \lambda \sum_{i=1}^{N} \frac{w_i^2}{1 + w_i^2}\ , \tag{8}$$

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

E. Perceptron Radial Basis Net

Perceptron Radial Basis Net (PRBFN) is an 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].

IV. Model Building

A. Iterated Prediction of Time Series

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

$$\vec{x}_n = (x_{(n-\lambda(d-1))}, x_{(n-\lambda(d-2))}, \dots, x_n), \tag{9}$$

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

$$\begin{aligned} f : \mathbf{R}^d &\rightarrow \mathbf{R} \\ f(\vec{x}_n) &= x_{n+1}. \end{aligned} \tag{10}$$

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 IV-B).

B. Model Training and Cross Validation

In order to select models for the final ensemble we use a cross-validation scheme for model training (see [11] 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

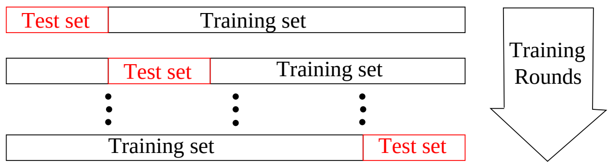

Fig. 1. In every training round the whole data set is divided in a training and a test set.

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 9 and the outputs are the one step ahead time series values $y_i = x_{i+1}$ from equation 10. We extract contiguous parts of length n from the input-out pairs as test sets M_{test} with

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

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

$$E_{train} = \sum_i (y_i - f^1(\vec{x}_i))^2 \tag{11}$$

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

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

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 (see FigureIV-B). These models are used to build the final ensemble

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

We use equal weights for each model to avoid over-fitting problems, but other choices are also possible (see [3] and [1] and the references therein).

V. Main Results

We investigate two data sets with our modeling approach. The first time series was part of the Leuven Competition [22] and the second time series is the CATS benchmark from the IJCNN 2004.

A. 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 [22]. It stems from a nonlinear transform of a 5-scroll generalized Chua's Circuit (see [23] for a detailed description). The data set consists of 2000 points. We build an ensemble model based on 30 cross-validation rounds, consisting of the models listed in Section III. We build time-delay vectors with time lag $\lambda = 1$ and dimension $d = 50$ as defined in Equ. 9. The prediction of the following 200 data points is shown in Figure 2. As a measure of accuracy we used the Normalized Mean Squared Error ($NMSE$) for the τ-step iterated prediction, that was proposed by McNames [13]. It denotes as

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

[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.

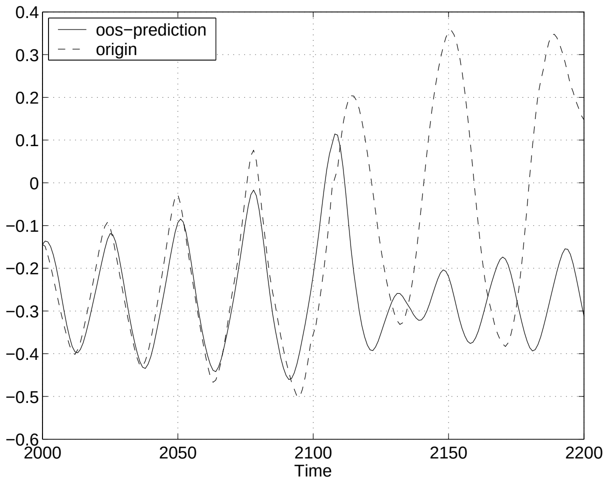

Fig. 2. The 200-step prediction of the data from Chua's Circuit and the original time series. In this case the ensemble model was able to predict the dynamics of the first 100 time steps.

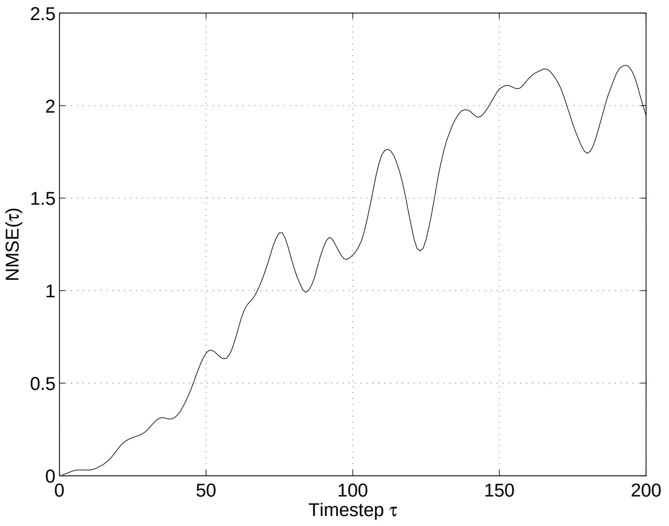

Fig. 3. The NMSE versus τ, the number of iterated prediction steps for the the data from Chua's Circuit. The average prediction horizon of the ensemble is approximately 60 time steps.

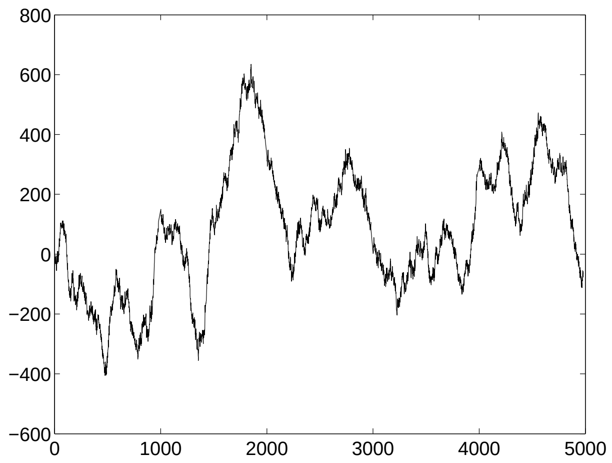

Fig. 4. The 5000 data points of the CATS Benchmark. The missing 100 points are left out in 5 blocks.

where $y_{\mu_i+\tau}$ is the $\mu_i + \tau$ point in the 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

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

The NMSE as a function of the prediction length τ has a vivid interpretation. If the NMSE is close to one, the model has lost its predictive power and we can also use the mean of the time series as a simple predictor. This marks the prediction horizon.

The average prediction horizon of our ensemble is approximately 60 time steps which is comparable with the results of the best models of the KU Leuven Competition [23]. The NMSQ for the data set is shown in Figure 3. 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.

B. The Cats Benchmark

The data set consists of an artificial Time Series with 5000 data points wherein 100 values are missing. These missing values are divided in 5 blocks of 20 points that have to be predicted. The time series is shown in Figure 4. By visual inspection the time series shows an oscillatory regime on the large scale that seems to be corrupted with a kind of noisy random process on the small scale. A further investigation shows that this noise has a deterministic structure that changes over time, so it seems to be a stochastic process superposed by the dominating large scale dynamics. In order to cope with two dynamical systems we decided to split the prediction problem in two parts. We first build a model for the large scale, using a polynomial fit to cover the long term oscillations. The second step is focused on the noisy process where we build an ensemble model for the small scale dynamics that uses the iterated prediction scheme that we described above.

1) The Large Scale Model: As a first step we investigate the proper window length w and the order of the polynomial fit p for the first four blocks. This is done by a brute force search of the parameter space. We fit a polynomial model of order p to contiguous parts of the data set with the total length $2 \cdot w + 20$ where the 20 points in the middle are hold out from the fitting procedure and the MSE of the fit is calculated using these 20 points. This is done for several parts of the time series around the missing blocks. The parameter combination with the smallest MSE is used for the final fit of the missing points in that particular gap. The results are reported in Table I.

2) The Small Scale Model: If we consider the small scale dynamics as random noise then the polynomial fit described

Block	Range	Order p	Window Length w
1	980:1000	3	16
2	1980:2000	4	40
3	2980:3000	4	16
4	3980:4000	5	30

TABLE I

THE ORDER AND WINDOW LENGTH OF THE POLYNOMIAL FIT USED FOR THE MISSING POINTS IN THE FIRST FOUR BLOCKS.

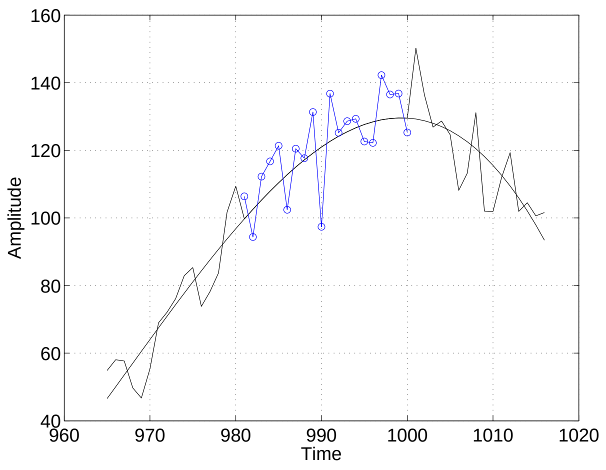

Fig. 5. The combined prediction for the first block of the CATS Benchmark. The solid line shows the polynomial fit and the circles show the added iterated prediction.

above would be a good solution of the prediction problem in the classical terms of statistics. But a further investigation of the time series leads to the conclusion, that we can find a deterministic part in the small scale dynamics that is covered with noise. Let $\{x_\nu\}_{\nu=1,\ldots,N}$ denote the original time series. We build the series of sequencing differences

$$x_n^* := x_n - x_{n-1}, \qquad (16)$$

and generate the d-dimensional state space vectors $\vec{x}_n^*$ as defined in Equ. 9. With this data set we construct a one-step ahead prediction model according to Equ. 10. We build simple ridge regression models for this purpose, because we only want to have an indication, if there is some deterministic part in the data. Further these models can be trained very fast and are known to be robust in the presence of noise.

The models are trained on 200 consecutive data points and the following 100 data points are used to perform an out-of-sample test. We compute the NMSE from Equ. 14 with $\tau = 1$ (one-step ahead prediction) for several segments of the time series. The results are shown in Fig. 6. The NMSE is lying below 0.9 for almost all tested segments.

We compare these results with the NMSE for a surrogate data set that we achieve by randomly shuffling the time series of sequencing differences from Equ. 16. This procedure destroys the time order of the series and the average NMSE for the linear models that are trained and performed on the randomized data set is closed to one. This is exactly what

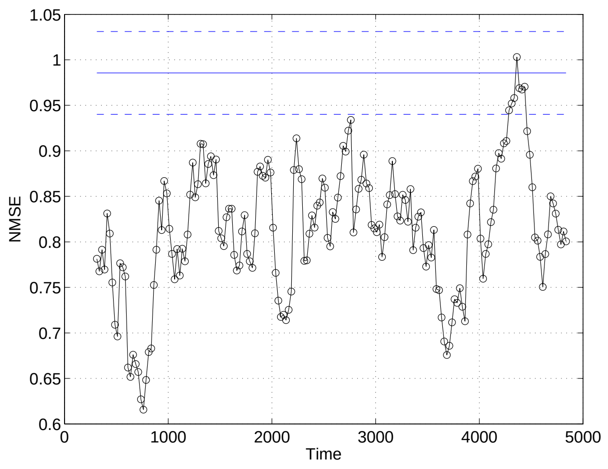

Fig. 6. The NMSE for several segments of the time series. Each dot marks the end of a segment with 300 data points. The average NMSE for the randomized data set is 0.98 (solid line) with a standard deviation of 0.046 shown as dotted lines.

we expect for a purely random process and it is a strong evidence, that the lower value of the NMSE for the series of sequencing differences is not an overfitting effect. This leads to the conjecture, that the small scale dynamics is not purely random noise but has a weak deterministic part, that could be described by a proper model.

In the above case the method of surrogate data acts as an indicator for determinism, while it is usually used to distinguish nonlinear determinism from a linear stochastic process [24].

For the small scale model we build state space vectors with $d = 14$ and perform an iterated 20-step ahead prediction. The results of the iterated prediction is combined with the polynomial fit described above to close the fore gaps of the CATS time series. The last 20 missing points are based on a pure iterated prediction without a polynomial fit. As an example we show the combined model for the first gap in Figure 5.

VI. Conclusion

We have demonstrated that ensembles models together with an iterated prediction procedure for model selection provides a powerful tool for time series prediction. The performance on KU Leuven Competition shows that this method can compete against the methods that are known to perform well on this data set. For the CATS Benchmark we used an advanced model building procedure that combines the robust polynomial fit of the large scale dynamics with an iterated ensemble prediction of the small scale dynamics.

Acknowledgments

The authors like to thank C. Merkwirth for fruitful discussions. The work was done as part of the Research Training Network *COSYC of SENS* No. HPRN-CT-2000-00158 within the 5th EU Framework Program of the European Community. This research has been also supported by AGH-University of Science and Technology grant 11.11.120.182.

REFERENCES

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

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

[3] 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. Chapman-Hall, 1993, pp. 126–142.

[4] 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., vol. 7. The MIT Press, 1995, pp. 231–238.

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

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

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

[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. Wichard, “ENTOOL - A Matlab Toolbox for Ensemble Modeling,” 2002. [Online]. Available: `http://chopin.zet.agh.edu.pl/˜wichtel/`

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

[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*, ser. Lect. Notes Math. Berlin: Sringer-Verlag, 1981.

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

[16] J. Stark, D. Broomhead, M. 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. Müller, “Effi cient BackProp,” in *Neural Networks: Tricks of the trade*, ser. Lecture Notes in Computer Science, G. Orr and K. Mller, Eds. Springer Verlag, 1998, vol. 1524, pp. 9–50.

[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. ICSC Academic Press, 2000, pp. 115–121.

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

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

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

[23] 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.

[24] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. Farmer, “Testing for nonlinearity in time series: The method of surrogate data,” *Physica D*, vol. 58, pp. 77–94, 1992.