Salzburg Interest Group on Integrated Systems

siGis-018-1999 June 17, 1999

Evolutionary and Coevolutionary Approaches to Time Series Prediction Using Generalized Multi-Layer Perceptrons

1999 Congress on Evolutionary Computation
July 6-9, 1999, Washington DC

Helmut A. Mayer
helmut@cosy.sbg.ac.at

Roland Schwaiger
rschwaig@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

Evolutionary and Coevolutionary Approaches to Time Series Prediction Using Generalized Multi-Layer Perceptrons

Helmut A. Mayer
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
helmut@cosy.sbg.ac.at

Roland Schwaiger
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
rschwaig@cosy.sbg.ac.at

Abstract- The prediction of future values of a time series generated by a chaotic dynamical system is an extremely challenging task. Amongst several non-linear models employed for the prediction of chaotic time series artificial neural networks (ANNs) have gained major attention in the past decade. One widely recognized aspect of ANN design in order to achieve sufficient prediction performance is the structure of the network. We automatize this procedure by evolving ANN topologies of low complexity guiding the evolutionary process towards ANNs of increased generalization ability. Specifically, a genetic algorithm (GA) is utilized to construct the architecture of generalized multi-layer perceptrons (GMLPs) trained by error-back-propagation. Another less investigated but important factor of ANN prediction quality is the size and composition of the training data set (TDS). Henceforth, we subject the selection of training data to artificial evolution in the environment of an ANN with fixed structure. A natural way to exploit the mutual dependencies of ANN structures and TDSs is symbiotic (cooperative) coevolution, where the fitness of an ANN is equally credited to the TDS it has been trained with. We compare these methods (ANN evolution, TDS evolution, and coevolution) with a standard ANN architecture given in the literature by predicting the Mackey-Glass time series.

1 Introduction

Successful time series prediction is a major goal in many areas of research, e.g., biology, meteorology, physics, business and engineering. However, most time series of practical relevance are of nonlinear and chaotic nature which makes conventional, linear prediction methods inapplicable. Hence, a number of nonlinear prediction methods have been developed including artificial neural networks, though, not initially proposed for time series prediction, "exceed conventional methods by orders of magnitude in accuracy" [Lapedes and Farber, 1987].

1.1 ANN prediction

Ensuing the pioneering work of [Lapedes and Farber, 1987] substantial research has been carried out to utilize ANNs for the prediction of nonlinear and chaotic time series. Platt (1991) already observed the importance of the ANN architecture for accurate prediction and proposed a growing scheme using neurons with Gaussian activation function. In this procedure a neuron is added, whenever a new training pattern causes poor prediction of the network, hereby adapting the network topology to the specific problem [Platt, 1991]. A network growing scheme based on [Moody and Utans, 1994] is used by Cholewo and Zurada (1997) for the construction of instances of time delay and recurrent ANNs for time series prediction [Cholewo and Zurada, 1997]. Here, neurons are added sequentially up to a preset limit in order to improve training accuracy and to reduce training time. Yao and Liu (1997) report on time series prediction results using their EPNet system evolving ANN structure and weights simultaneously [Yao and Liu, 1997]. Another evolutionary prediction system has been presented by De Falco et al. (1998), where a fixed multi–layer perceptron architecture has been trained by evolving the weights of the connections [DeFalco et al., 1998].

1.1.1 Training data set selection

A less investigated problem concerns the size and composition of the training data set (TDS) for an ANN. A few researchers proposed sequential methods of TDS construction starting with a small initial TDS and selecting additional training patterns according to specific measures. These methods are subsumed under the term *Active Selection*. Representatives of this technique are Zhang's (1994) *Incremental Learning* [Zhang, 1994], Röbel's (1994) *Dynamic Pattern Selection* [Röbel, 1994], and Plutowski's (1994) analytical method based on the *Integrated Squared Bias* of training patterns [Plutowski, 1994]. Obviously, these methods are dependent on the composition of the initial TDS and the specific measure used to evaluate the "goodness" of training patterns.

Theoretically, an optimal TDS w.r.t. a quality measure could be constructed in parallel, but is *"computationally impractical"*[Plutowski, 1994]. As a consequence, evolutionary methods generating complete TDSs in parallel seem to be a promising alternative. We have been reporting that evolutionary training data sets could enhance the classification performance of an ANN for a simple pattern recognition problem considerably [Mayer and Schwaiger, 1997]. Reeves and Taylor (1998) proposed an evolutionary training pattern selection scheme to improve the classification performance of radial basis function (RBF) networks [Reeves and Taylor, 1998]. As specific TDS compositions and ANN architectures influence each other, first investigations of coevolution of ANN topologies and TDSs have been presented in [Mayer, 1998b].

2 (Co)Evolution of ANNs and TDSs

The technical platform for the (co)evolution of ANNs and TDSs is the netGEN system [Huber et al., 1995] which has been designed for an arbitrary number of workstations in order to train individual ANNs in parallel.

The ANN's genetic blueprint is based on a direct encoding suggested by Miller et al. [Miller et al., 1989]. With this approach each connection is represented in a binary adjacency matrix called *Miller–matrix* (MM) describing the ANN architecture. Contrary to the original crossover operator exchanging rows and columns of the MM, we use standard 2–point crossover operating on a linearized MM where the triangle matrix of fixed 0s (feed–forward architecture) is not included.

The main enhancement in our *modified–Miller–matrix (MMM)* direct encoding scheme [Huber et al., 1995] are the *neuron markers* which are encoded in a separate section on the ANN chromosome (the two other sections contain learning parameters and all possible connections, respectively). From a biological viewpoint, the neuron markers are a simple analogue to activators/repressors regulating the expression of genes. Technically, each neuron marker (one bit) is transferred to the main diagonal (i, i) of the MMM adjacency matrix, indicating the absence or presence of a particular neuron i and its connections (Figure 1). As a consequence, most ANN chromosomes contain noncoding regions [1], as all connections associated with a specific neuron become obsolete, if the corresponding neuron marker is zero.

It has been shown [Mayer, 1998a] that the inclusion of noncoding segments decrease the disruptive effects of crossover which might have a positive effect on ANN evolution w.r.t. the often cited problem of "competing conventions" (or permutation problem). However, it should be noted that Hancock (1992) explicitely reported that "…simple crossover also fared well, suggesting that the permutation problem is not serious in practice" [Hancock, 1992]. Moreover, to our knowledge nobody has yet succeeded to give a quantitative analysis of the actual percentage of genotypes coding for identical phenotypes for a specific encoding.

The maximum number of hidden neurons (neuron markers) has to be set in advance with this encoding scheme, hence, it could be labeled as *evolutionary pruning*, since the system imposes an upper bound on the complexity of the network. One of the measures to avoid overfitting is the encoding of the number of training epochs with the additional benefit of another ANN parameter being set automatically. Two learning parameters for the selected training algorithm *resilient back–propagation* (RPROP) [Riedmiller and Braun, 1993] are also encoded in the genotype.

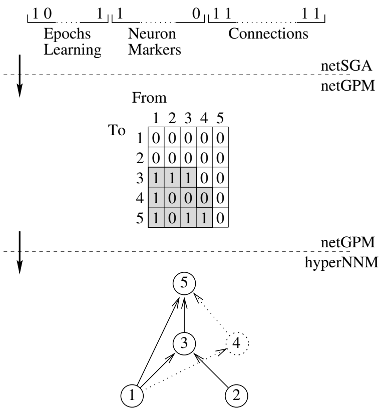

Figure 1: ANN encoding scheme.

The encoding of TDSs (Figure 2) is simple and straightforward. Each training pattern in the data store containing all available training patterns is represented by a single bit on the chromosome (bit mapping). Thus, the number of bits (chromosome length l) is equal to the number of training patterns in the data store. The actual TDS is built by selecting those patterns whose corresponding bits are of value 1. This enables the evolution of a TDS of variable size without multiple inclusion of patterns.

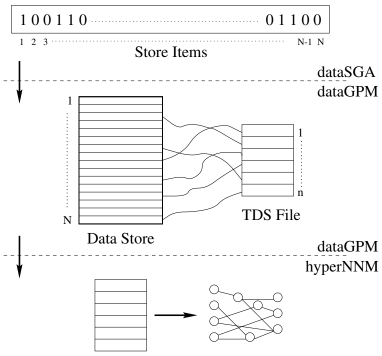

Figure 2: TDS encoding scheme.

Both GAs, netSGA (evolution of ANN architecture) and dataSGA (TDS evolution), use the same basic genetic operators which are given below. For the GA selection phase we chose *Binary Tournament Selection* without replacement.

With coevolution TDSs can be viewed as the (changing) environment of ANNs or vice versa. We employ symbiotic (cooperative) coevolution (Figure 3), where ANN fitness is equally credited to the TDS it has been trained with. To each individual of the ANN population a TDS individual is randomly assigned and used for ANN training. Thus, individuals being adapted to a small range of environments are less

[1] Non–coding DNA regions or *Junk DNA* can be found in abundance in biological systems, i.e. *Eukaryotes*.

likely to survive than "generalists". This constitutes one end of the coevolutionary spectrum, and we compare this setting (scenario B) with another extreme of possible coevulationary scenarios. Here, ANNs are evolved to a fixed TDS, and then the best ANN is kept "frozen" and TDSs are evolved for a number of generations (scenario A).

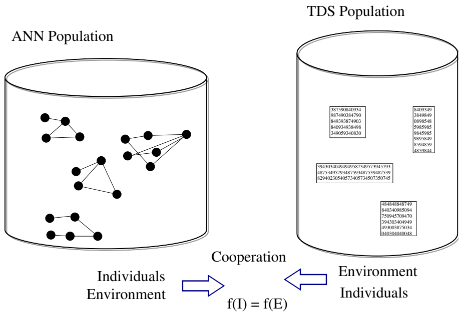

Figure 3: Symbiotic coevolution of ANN and TDS populations.

2.1 ANN and TDS fitness functions

The ANN fitness function comprises a complexity regularization term $\mathcal{E}_c = |C_{total}|$, with C_{total} being the set of all network connections. The *composite fitness function* $\mathcal{F}$ is given by a weighted sum of *model fitness* and *complexity fitness*

$$\mathcal{F} = \alpha_1 \frac{1}{1+\mathcal{E}_m} + \alpha_2 \frac{1}{1+\mathcal{E}_c} \tag{1}$$

with $\alpha_1 + \alpha_2 = 1.0$, and $\mathcal{E}_m$ being the model error. Earlier work [Huber et al., 1996] showed that α_2 in the range of $0.001-0.01$ is sufficient to guide the evolution towards ANNs of low complexity. Thus, we set $\alpha_2 = 0.01$. We define the model error for prediction tasks as the normalized root mean square (NRMS) on a separate validation set

$$\mathcal{E}_m = \sqrt{\frac{\sum_{i=1}^{N} (x_i - o_i)^2}{\sum_{i=1}^{N} (x_i - \overline{x})^2}}, \tag{2}$$

where x_i is the i-th value of the time series, o_i the prediction of the network, and $\overline{x}$ the mean of the time series subset defined by $i = 1 \ldots N$. TDS fitness is equal to the composite fitness of the ANN trained using the specific TDS.

2.2 GA and ANN Parameters

The following GA and ANN parameters have been used with all the experiments in this paper:
GA Parameters: Population Size = 50, Generations = 50, Crossover Probability p_c = 0.6, Mutation Probability p_m = $\frac{1}{l}$, Crossover = 2–Point, Selection Method = Binary Tournament.
ANN Parameters: Network Topology = Generalized Multi–Layer Perceptron, Activation Function (hidden neurons) = Sigmoid / (output neuron) = Identity, Output Function (all neurons) = Identity, Training = RPROP, Hidden Neurons (max 20), Learning Parameters Δ_0 (max 1.0), Δ_{max} (max 50.0), Number of Training Epochs (max 1000) = Evolutionary, Data Store = 1000 patterns, Validation Set = 2001 patterns, Test Set = 501 patterns.

3 Benchmark Problem

In the following experiments we evolve neural predictors for the *Mackey–Glass* [Mackey and Glass, 1977] time series generated by the differential delay equation

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t). \tag{3}$$

Setting $a = 0.2$, $b = 0.1$, and $c = 10$, the system exhibits chaotic behavior with $\tau > 16.8$. Choosing $\tau = 17$ yields a strange attractor with fractal dimension D approximately 2.1. Based on Takens' Theorem [Takens, 1981] Lapedes and Farber (1987) set the dimension of the embedding space to 4 and the time delay to 6 resulting in an ANN input vector $(x(t-18), x(t-12), x(t-6), x(t))$ for prediction of a future value $x(t+P)$. As the characteristic period of this time series is approximately 50, we set $P = 84$ as recommended at the CMU benchmark site (`www.boltz.cs.cmu.edu/benchmarks/`). The benchmark data is partitioned in a training set (3000 patterns, $t = 200-3200$) and a test set (500 patterns $t = 5000-5500$). We used the first 1000 patterns of the training set as data store and the next 2000 patterns for the validation set (test set unchanged).

4 Experimental Results

In order to compare the evolutionary neural predictors to an ANN of conventional architecture, we adopted the 4–10–10–1 network (standard) suggested in [Lapedes and Farber, 1987], and used the ANN parameters given in Section 2.2. Then, we utilized the complete data store (1000 patterns) as a fixed TDS for the evolution of ANN structure (annev). Next, we evolved a TDS in the fixed environment of the standard network (tdsev). Also, a TDS has been evolved in the fixed environment of the evolved ANN structure which constitutes coevolutionary scenario A (coevA), where each environment (first TDS, then ANN) is kept "frozen" for 50 generations. As with this scenario coevolution has been run for 100 generations, we allocated the same number of generations (100) to scenario B (coevB). Table 1 gives the NRMS and the absolute maximum prediction error $|\epsilon_{max}|$ for a single time series value over validation and test set for the evolved (and conventional) neural predictors.

The results are an indicator of the potential of evolutionary search for (sub)optimal ANN architectures and TDSs. Separate ANN or TDS evolution improves the prediction accu-

	Validation Set		Test Set	
	NRMS	$\|\epsilon_{max}\|$	NRMS	$\|\epsilon_{max}\|$
standard	0.3489	0.3010	0.3456	0.2712
annev	0.2124	0.1778	0.2109	0.1793
tdsev	0.2318	0.1610	0.2314	0.1478
coevA	0.1828	0.2100	0.1868	0.1885
coevB	0.1892	0.2375	0.1905	0.2096

Table 1: Normalized root mean square (NRMS) and absolute maximum prediction error $|\epsilon_{max}|$ of ANN predictors for the Mackey–Glass time series.

racy compared to a conventionally structured ANN. Coevolution even further improves the performance of the neural predictor. Though, at opposite ends of the coevolutionary spectrum, the two investigated coevolutionary scenarios exhibit similar performance which has also been observed in [Mayer, 1998b].

When comparing the results to previous work, we could not reproduce the accuracy of an NRMS around 0.05 as given in [Lapedes and Farber, 1987] and [Platt, 1991]. However, no detailed ANN training parameters were given in [Lapedes and Farber, 1987], while Platt (1991) used a different network type with 100+ hidden neurons. The best result for a recurrent network in [Cholewo and Zurada, 1997] is given by an NRMS = 0.46 (with $\tau = 30$ in Equation 3). For evolutionary prediction systems accuracies of NRMS = 0.02 in [Yao and Liu, 1997] and NRMS = 0.15 in [DeFalco et al., 1998] (both systems predicting $(t + 6)$) have been reported. Unfortunately, also different training and test sets have been used by these researchers.

Even more critical to the prediction results and their comparability seems to be the step size used with the numerical integration of the Mackey–Glass differential equation [2]. (E. g., CMU benchmark data have been generated using a second order Runge-Kutta method with step size 0.1, while in [Yao and Liu, 1997] a fourth order Runge-Kutta method with step size of 1.0 has been used.) As a larger step size results in a smoothing of the time series (low–pass filter), fine details are lost which might make predictions easier.

The actual predictions on the test set generated by different ANN architectures are shown in Figure 4.

The actual patterns selected from the data store in evolved TDSs are depicted in Figures 5 and 6.

Clearly, it can be observed that evolution favors patterns at and near local extrema of the time series segment in the data store. Moreover, patterns from aperiodic features of the time series are selected more frequently (e.g., the pattern cluster at $t = 400$). A closer look at Figure 6 reveals that the patterns of the coevolved TDS do not cover all local extrema and some time segments are only represented by few patterns (e.g., around $t = 1150$). This could be explained by the fact that the coevolving ANN architecture has adapted to

[2]Personal communication with Ivan De Falco.

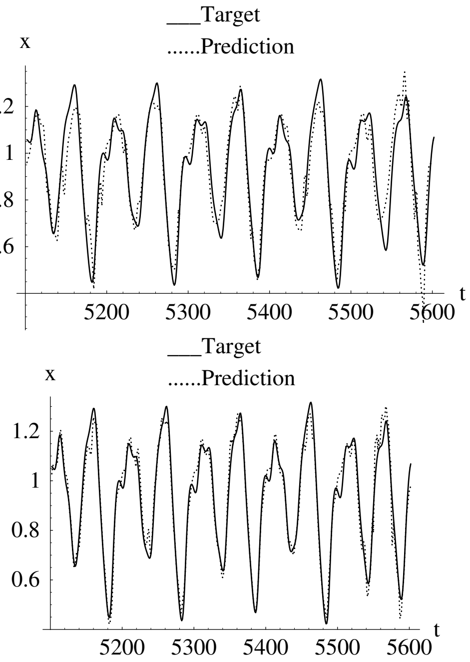

Figure 4: Test set predictions of ANNs *standard* (4–10–10–1) (top) and *coevA* (16 hidden / 171 connections) (bottom).

the specific time series and (partial) problem knowledge is represented by the network structure.

The data contained in the evolved TDSs can be also viewed in the light of sampling theory, where above TDSs are called irregular samples, i.e., sampling values at unequally spaced sample points. It would be worthwhile to compare the reconstructed time series using the original regular and the evolved irregular samples [Cenker et al., 1991]. It might further be argued that the error gradient (used for ANN training) generated by sample values at aperiodic features is more efficient in directing the training process to a “good” error minimum than the information given by values in nearly linear segments of the time series.

5 Outlook

Obviously, the presented evolutionary and coevolutionary approaches should be tested for a variety of prediction problems. Further research activities will focus on on–line (co)evolution, where ANN architectures and TDSs are constantly adapted to new data. This problem of changing data volume is of great practical relevance and occurs in a variety of real world problems (e.g., financial forecasting).

6 Acknowledgments

We wish to thank the *Austrian Center for Parallel Computation* (ACPC), Group Salzburg, for making available a cluster of DEC AXP workstations for this research. Also, we would like to thank two anonymous referees for helpful remarks.

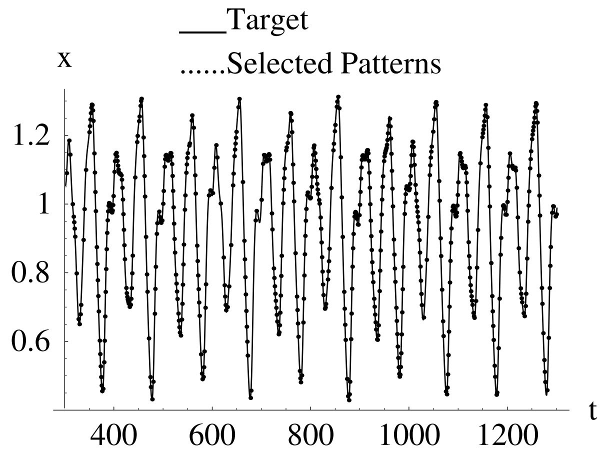

Figure 5: Selected Patterns of best evolved TDS *tdsev* (639 patterns).

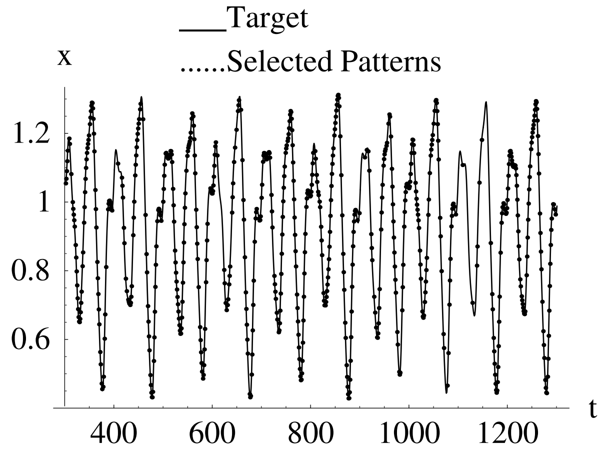

Figure 6: Selected Patterns of best coevolved TDS *coevB* (665 patterns).

Bibliography

[Cenker et al., 1991] Cenker, C., Feichtinger, H. G., and Herrmann, M. (1991). Iterative Algorithms in Irregular Sampling: A First Comparison of Methods. In *Proceedings of ICCCP'91*, pages 483–489.

[Cholewo and Zurada, 1997] Cholewo, T. J. and Zurada, J. M. (1997). Sequential Network Construction for Time Series Prediction. In *Proceedings of the IEEE International Joint Conference on Neural Networks*.

[DeFalco et al., 1998] DeFalco, I., Iazzetta, A., Natale, P., and Tarantino, E. (1998). Evolutionary Neural Networks for Nonlinear Dynamics Modeling. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *5th International Conference Parallel Problem Solving from Nature*, pages 593–602, Berlin. Springer.

[Hancock, 1992] Hancock, P. J. B. (1992). Genetic Algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In Whitley, D., editor, *Proceedings of COGANN Workshop at the International Joint Conference on Neural Networks*. IEEE Press.

[Huber et al., 1995] Huber, R., Mayer, H. A., and Schwaiger, R. (1995). netGEN - A Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by means of Genetic Algorithms. In *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3, Forschungsbericht Nr. 580, Dortmund*.

[Huber et al., 1996] Huber, R., Schwaiger, R., and Mayer, H. A. (1996). On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks. In *World Congress on Neural Networks*, pages 1063–1066. International Neural Network Society, Lawrence Erlbaum Associates, Inc.

[Lapedes and Farber, 1987] Lapedes, A. and Farber, R. (1987). Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling. LA–UR LA–UR–87–2662, Los Alamos National Laboratory, Los Alamos, NM.

[Mackey and Glass, 1977] Mackey, M. and Glass, L. (1977). Oscillation and Chaos in Physiological Control Systems. *Science*, 197(287).

[Mayer, 1998a] Mayer, H. A. (1998a). ptGAs–Genetic Algorithms Evolving Noncoding Segments by Means of Promoter/Terminator Sequences. *Evolutionary Computation*, 6(4):361–386.

[Mayer, 1998b] Mayer, H. A. (1998b). Symbiotic Coevolution of Artificial Neural Networks and Training Data Sets. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *5th International Conference Parallel Problem Solving from Nature*, pages 511–520, Berlin, Germany. Springer.

[Mayer and Schwaiger, 1997] Mayer, H. A. and Schwaiger, R. (1997). Towards the Evolution of Training Data Sets for Artificial Neural Networks. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 663–666. IEEE Press.

[Miller et al., 1989] Miller, G. F., Todd, P. M., and Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo, California. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[Moody and Utans, 1994] Moody, J. and Utans, J. (1994). *Architecture Selection Strategies for Neural Networks: Application to Corporate Bond Rating Prediction*. Neural Networks in the Capital Markets. J. Wiley & Sons.

[Platt, 1991] Platt, J. (1991). A Resource–Allocating Network for Function Interpolation. *Neural Computation*, 3(2):213–225.

[Plutowski, 1994] Plutowski, M. (1994). *Selecting Training Examplars for Neural Network Learning*. PhD thesis, University of California, San Diego.

[Reeves and Taylor, 1998] Reeves, C. R. and Taylor, S. J. (1998). Selection of Training Data for a Neural Network by a Genetic Algorithm. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *5th International Conference Parallel Problem Solving from Nature*, pages 633–642, Berlin. Springer.

[Riedmiller and Braun, 1993] Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA.

[Röbel, 1994] Röbel, A. (1994). Dynamic Pattern Selection: Effectively training Backpropagation Neural Networks. In *Proceedings of the International Conference on Artificial Neural Networks, ICANN'94*.

[Takens, 1981] Takens, F. (1981). Detecting Strange Attractors in Turbulence. In Rand, D. and Young, L.-S., editors, *Dynamical Systems and Turbulence, Lecture Notes in Mathematics No. 898*, Berlin. Springer.

[Yao and Liu, 1997] Yao, X. and Liu, Y. (1997). A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 8(3):694–713.

[Zhang, 1994] Zhang, B.-T. (1994). Accelerated Learning by Active Example Selection. *International Journal of Neural Systems*, 5(1):67–75.