

## A TEMPORAL MINIMUM DESCRIPTION LENGTH POLICY FOR EVOLVING NEURAL NETWORKS

**REZA DERAKHSHANI**

Department of Computer Science and Electrical Engineering  
School of Computing and Engineering  
University of Missouri at Kansas City  
Kansas City, Missouri

### ***ABSTRACT***

One of the most important issues for computational methods is their time complexity. This paper introduces a temporal MDL (minimum description length) policy for evolving neural networks based on their execution time on the hosting hardware. Temporal MDL implements an adaptive selection pressure based on the actual processing time of the evolving solutions and thus favors creation of faster, more compact networks for the given data. Temporal MDL reduces the time complexity directly and the network size indirectly. The latter helps generalization and reduces model variance, making the temporal MDL a viable candidate for regularization. This methodology is especially important for time-critical applications. Mackey-Glass time series prediction results are presented for evolutionary distributed time lag neural networks with temporal MDL to demonstrate the above stated capabilities.

### **INTRODUCTION**

Bias-variance dilemma is an important issue in neural network design. Besides validation-based early stopping, regularization is an effective method for helping networks' generalization capabilities by penalizing more complex solutions and reducing unwanted variance. This is especially important when the training data is scarce and one does not have the luxury of setting aside a part of data for validation-based early stopping, which is usually the case for many real world applications (Haykin, 1999).

Speaking of real world applications, time is usually one of the most important factors in computing systems, especially for time-critical applications. Furthermore, temporal agility has been mentioned as an indicator of machine intelligence (Kurzweil, 2000). Traditional regularization techniques such as Akaike information criterion (Box et al, 1994) do not deal with the actual time complexity. Here we introduce a practical minimum description length procedure in the context of evolutionary neural networks that will address this issue.

### **TEMPORAL MINIMUM DESCRIPTION LENGTH AND NEURAL NETWORKS**

Our hypothesis is as follows: since the actual training time for a neural network on a given computational platform is directly related to its size, then penalizing each network not only for its error but also its time complexity should reduce its space complexity which has been reported by other researchers as a successful form of regularization (Principe et al, 2000; Sathyanarayan and Kumar, 1996; Hansen and Yu, 2001). This favoring of parsimony through selection pressure on both error and actual computation

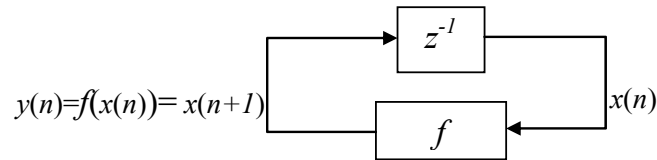
time will produce data-driven, lean, and agile solutions tailored to the given computing environment.

To further explain the above concept, let's consider the two-part definition of the Minimum Description Length, or MDL. According to Rissanen, the best model for description of given data is one that yields the minimum length (expressed in bits in terms of digital computing) for both the model and the data encoded with it. Mathematically, this can be expressed as finding a model (or hypothesis)  $H$  in order to minimize (Grünwald, 2005)

$$L(H) + L(D|H) \quad (1)$$

where  $L(H)$  is the code length of the model describing our data  $D$  and  $L(D|H)$  is the description of  $D$  given the model  $H$ . In this sense, MDL is equating learning with compression since if the data's structure is known, a much shorter description of data (e.g. a computer program) maybe be used to represent it.

Now consider the neural network arrangement of Fig. 1, where the network is trained to predict the next sample of a sequence given input's current value. It is usually a better practice to implement a tapped delay line (a sequence of delay elements accepting an input signal at one end and yielding delayed versions of it in-betweens) instead of the single step delay element shown in Fig. 1 to provide more of the signal history to the neural network for better prediction. Either way, with proper initial conditions and provided that the neural network is successfully trained on prediction of  $x(n)$ , one can claim that the network  $f$  has learned (modeled) the data sequence  $x(n)$  as it can regenerate it. That is, a successful predictor neural network for a data sequence can be considered as a computational model of that data. In terms of Eq. (1), the neural network model length is  $L(H)$  (e.g. the number of bits for the corresponding computer program). The part of  $x(n)$  not correctly represented by  $f$  constitutes a residue (error) that needs to be transmitted for proper reconstruction of  $x(n)$ . Here we represent this error by  $E$ .  $E$  is obviously dependent on our neural network model  $L(D|H)$ , as more complex models have more degrees of freedom and thus can yield better fits to more complex functions. One can also think of a kernel-based machine's learning capacity vs. the size of its hidden layer as explained by Cover's theorem (Cover, 1965). Here we refer to  $L(D|H)$  by  $M$ . Our goal is to minimize, or at least arbitrarily reduce, both  $M$  and  $E$  simultaneously through a time-dependent mechanism. Note that since a neural network's error  $E$  is influenced not only by the network size but also its architecture,  $E$  also takes into account the so called *parametric complexity* as indicated in the *refined MDL* (Grünwald, 2005).



**Fig. 1 A neural network  $f$  in a sequence generating configuration.**

*Temporal MDL:* Consider an evolutionary framework for breeding series predictor neural networks, where the training time of the whole population is curtailed by the time complexity of smaller networks in each generation. This idea is based on the need for a time-conscious regularization and the observation that the computation complexity of neural networks using gradient decent algorithms is a function of the number of their connection weights. More specifically, assume the actual physical time spent on the instruction  $i$  to be  $f(i)$ , where  $i \in \{\text{hosting machine instruction subset used in the weight update}\}$ . Here we presumed a simple Single Instruction, Single Data (SISD) digital computing hosting machine. Moreover, assume the total number executions for each instruction  $i$  for a weight update (given the training algorithm) to be  $N(i)$ . The time spent on updating each weight then will be  $\sum f(i)N(i)$ . As for the whole network weight adaptation time per epoch, we can write

$$C_t = N_w \sum_i f(i)N(i) \quad (2)$$

where  $N_w$  is the total number of network weights. Since  $N(i)$  is constant for the chosen training algorithm and  $f(i)$  is fixed for the given simple hosting hardware, one can deduct that under ideal conditions  $C_t$ , the time complexity of the neural network under study, is proportional to the number of connection weights  $N_w$ . We said ideally as these calculations assume a basic serial machine and do not take into account factors such as cache memory size and parallel execution of independent instructions that may alter  $C_t$  as given in Eq. (2) for modern computer architectures. In such case, for a given computing platform in steady state running no simultaneous major other routines but the evolutionary neural networks,  $C_t$  will generally have a nonlinear relation with  $N_w$  given by

$$C_t = f_t(N_w) \quad (3)$$

Where  $f_t$  is a monotonically increasing function of  $N_w$ . The condition for the evolutionary neural networks to be the only substantial running process is mentioned because the host computing machine and its resources can be considered as the electronic ecosystem (Ray, 1991) for the competing evolutionary neural networks and we do not want other unrelated processes to introduce noise into this evolutionary landscape, where the available CPU resources and eventually execution time play a vital role.

Going back to the two-part definition of MDL in Eq. (1), we want to ideally minimize, or for practical purposes adequately reduce both the error  $E$  and the model description length  $M$  (or  $N_w$ , see below); where the latter according to Eqs. (2) and (3) is directly related to the network time complexity. During the following calculations, we assume that a network can be described (i.e. uniquely defined) by its connection weights and thus we consider the description length to be a function of  $N_w$ . For the sake of simplicity, here we equate  $M$  with  $N_w$ <sup>1</sup>; which is also reported in literature (Hinton and Camp, 1993). Note that the influence of structure in learning capabilities of the network is reflected in  $E$ . We desire to simultaneously decrease both model bias and time

---

<sup>1</sup> To be more accurate, one should add a small overhead to  $N_w$  for a description length that describes the topology of the network as well. For instance, weight descriptors can be accompanied by indices that denote their source and destination nodes.

complexity so that the latter indirectly reduces model length. This way one approaches MDL through the more practical parameter  $C_t$ . This mechanism is implemented for evolving neural networks under two pressure mechanisms: a) a fitness score that is inversely proportional to the network error, and b) a temporal pressure that favors the faster (i.e. more compact) networks.

The justification for the latter item is as follows: for a fixed training time, leaner and faster networks that are not too small for the task at hand, will have the advantage of training for more epochs and thus will have a better chance of survival. Generally speaking, assume following function subjected to minimization under the above two pressure mechanisms<sup>2</sup>

$$f_{MDL}(E, M) \quad \frac{\partial f_{MDL}}{\partial E} > 0, \frac{\partial f_{MDL}}{\partial M} > 0 \quad (4)$$

However, because of the conflicting requirements for simultaneous minimization of  $E$  and  $M$  in neural network design (i.e. the bias-variance dilemma), one needs to find an operating point that is an acceptable compromise. Note that minimization of  $f_{MDL}$  will also provide an answer to the two part definition of MDL, albeit not exactly the one found under the simple summation of  $E$  and  $M$ , i.e.  $f_{MDL}(E, M) = M + E$  in Eq. (1). In a time-constrained training scenario for our evolutionary neural networks, the selection routine may first seem to only penalize the model bias  $E$  as its inverse is used for fitness evaluation. However, besides the network architecture, weight contents, and network size  $N_w$ ,  $E$  is inversely (and nonlinearly) proportional to the number of training epochs  $N_{epochs}$  that the network is allowed during its gradient descent learning. That is

$$E = f_1\left(\frac{1}{N_{epochs}}, \frac{1}{N_w}, \overline{W}_0, A\right) \quad (5)$$

where  $f_1$  is a monotonically non-decreasing function of its first two variables  $N_w^{-1}$  and  $N_{epochs}^{-1}$ . The other two variables are  $\overline{W}_0$  (initial weights, continuous in  $\mathfrak{R}^{N_w}$ ) and  $A$  (representing architecture, a member of discrete space of feasible network configurations). Equation (5) simply states that given the architecture and initial conditions, bigger networks as well as those with more epochs of gradient descent training can provide a better fit to the training data. Now if each network's training time during each generation  $G$  is limited to  $t_G$ , then  $N_{epochs}$  will be given by

$$N_{epochs} = \left\lceil \frac{t_G}{C_t} \right\rceil \quad (6)$$

From Eqs. (2) and (3) and their discussions we know that  $C_t$  in neural networks is proportional to the description length  $M$ . Thus, considering Eqs. (3), (5), and (6), during each generation  $G$  we will have

---

<sup>2</sup> Since  $M$  is a natural number, one can either interpolate in between  $M$  values in Eq. (4) to introduce continuity or use discrete derivatives utilized for sampled data in digital signal and image processing.

$$E = f_1 \left( \left[ \frac{t_G}{f_i(N_w)} \right]^{-1}, N_w^{-1}, \vec{W}_0, A \right) \quad (7)$$

By replacing  $N_w$  with  $M$  and for a given configuration we can write

$$E = f_2 \left( M, \frac{1}{M} \right) \quad (8)$$

Where  $f_2(x,y)$  is a non-decreasing function of its variables, that is  $\partial f_2 / \partial x \geq 0$  and  $\partial f_2 / \partial y \geq 0$  (please see the definitions of  $f_1$  and  $f_i$ ). Equation (8) reveals two mechanisms with opposing requirements, which exposes the bias-variance dilemma under the aforementioned time-constrained evolution of neural networks. First, reducing the error  $E$  (model bias) requires a bigger network for better data fitting as expressed by the inverse relationship of  $E$  to  $M$  via the second variable in Eq. (8). However, at the same time, reduction of  $E$  requires more epochs of gradient descent training which in turn requires faster and thus smaller networks, as expressed by direct relationship of  $E$  to  $M$  in Eq. (8) via the first variable.

One may wish to obtain the optimal complexity  $M^*$  by setting the derivative of Eq. (8) equal to zero<sup>3</sup> to get

$$M^* = \sqrt{\frac{f_{2,y} \left( M^*, \frac{1}{M^*} \right)}{f_{2,x} \left( M^*, \frac{1}{M^*} \right)}} \quad (9)$$

and solve for  $M^*$  (say through iteration) using (9), where  $f_{2,x}$  is the partial derivative of  $f_2$  with respect to its first variable and  $f_{2,y}$  is the partial derivative of  $f_2$  with respect to its second variable. The aforementioned equations explain the bias-variance dilemma under temporal MDL, however,  $f_2$  is not analytically known and thus we leave the job of finding a satisfactory, near-optimal temporal MDL operating point the evolutionary process.

## IMPLEMENTATION

A general evolutionary temporal neural network framework or GETnet (Derakhshani, 2005a, Derakhshani, 2005b) was used to show the effects of temporal MDL. GETnet evolves general nonlinear recurrent neural networks with distributed delay elements and can model arbitrary dynamic systems (Seidl and Lorenz, 1991; Siegelmann and Sontag, 1995) with super Turing capabilities (Siegelmann et al, 1997). Based on the given tagged data, the evolutionary process of GETnet finds proper network topology, size, and weights through an evolutionary mixture of deterministic and stochastic searches in connection weight, connection delay, and network configuration spaces. GETnet first generates a random population of temporal neural networks with both feed

---

<sup>3</sup> Again we are assuming continuity in  $M$  and an eventual rounding of the result.

forward and recurrent connections with various path delays. These individuals go through the evolution loop to create the fittest solution. The ontogenetic component of learning of each individual is implemented by a time-constrained gradient descent (i.e. temporal MDL) and the phylogenetic component is implemented by adaptive weight, delay, connection, and node mutations. Mutation-only evolution strategies' technique (Schwefel, 1981) with Gaussian mutations and evolvable standard deviation was used. No crossover was implemented as it can destroy the distributed knowledge of a neural network (Yao, 1999). Scaled Conjugate Gradient (SCG) (Moller, 1993) was used for ontogenetic learning. Among other advantages, this method has better performance on sharp error surface valleys that may be produced by compact networks evolved under temporal MDL. The survival chance of each individual (i.e. temporal neural network) in a generation is calculated as the inverse of its mean squared error on the validation data (can be partially or completely composed of the training data) after partial time-limited SCG training used in conjunction with a roulette wheel selection scheme. After exiting the evolution loop, the best evolved network is fully trained and then fed the test data. Below we explain the details of the temporal pressure implementation as it is the cornerstone of temporal MDL in this algorithm.

During each generation, individuals are sorted in descending order of their size,  $N_w$ . Next, a representative member from the smaller half of this sorted population (say the network found on the third quartile slot) is chosen. This representative of the more compact sector of the population is then trained for a number of epochs that is heuristically known to be adequate for a rather significant reduction in MSE (e.g. 5 epochs for SCG algorithm on Mackey-Glass data, a hyper-parameter of this method). A multiple-run average of this time is then calculated and set as the training time limit for all other networks in the generation. The averaging is necessary for a more objective evaluation of  $E$  as each offspring in GETnet acquires its starting point partially from its parent and an adaptive noise is added to each transmitted weight. This partial, time limited training will favor faster, more compact networks since they can train for more epochs in the given time limit, provided that the smaller networks are structurally capable learning the task at hand. If not (consider the analogy of a one layer neural network trying to solve the XOR problem for instance), then the partial training of the bigger networks will outperform that of the nonfunctioning smaller networks and thus the minimum training time increases to the advantage of the bigger networks and the incapable smaller networks will become extinct.

## SIMULATIONS

All the following simulations were written in MATLAB 6.5 and its neural network toolbox v 4 and were executed on a PC with single 2 GHz Athlon XP 2400+ processor.

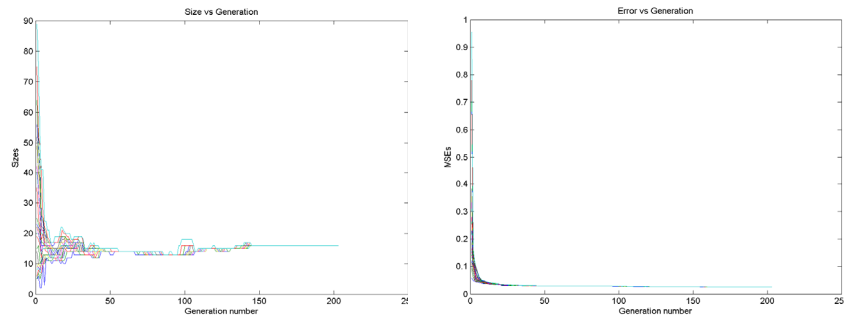
GETnet with temporal MDL was used for Mackey-Glass time series prediction. The series itself is defined by the following differential equation (Mackey and Glass, 1977; de Menezes and dos Santos, 2000)

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{(1+x(t-\tau))^{10}} - 0.1x(t) \quad (10)$$

With  $x(0)=1.2$  and  $\tau=17$  (MG17), Mackey-Glass becomes a chaotic, pseudo-periodic, and non-convergent series. The simulations here are for 6 and 36-step predictions on this series. Initial population size in both cases was chosen to be 25, as larger populations on a single processor PC make this simulation very slow. The discrete-time MG17 series was obtained from benchmark collection of temporal data for FIRnet (Wan WWW Ref).

*Case 1, six-step prediction:* In this experiment, samples 1-500 of a 1500 point MG17 series were used for training. Samples 1-1000 were used for calculation of network error (that is both seen and unseen data were used to estimate both generalization and trainability), and samples 1001-1500 were left for testing. After 203 generations, the best evolved network, i.e. the one with the lowest network error, produced a training mean squared error (MSE) of 0.0052 and a test MSE of 0.0054. The small difference between the training error and the error for the unseen test data shows the success of temporal MDL in minimizing the model variance by reducing the model size. The average training time for the ancestor of this network is 22.543 seconds, whereas the same time for the evolved network is 6.269 seconds. This indicates a 3.6 fold improvement of  $C_t$  under temporal MDL. Figure 2 (left) shows the MSEs of the evolving networks, and on the right it shows their evolving sizes,  $N_w$ . Fig. 3 visualizes the effect of temporal MDL on the time complexities of this evolving population further by contrasting the histograms of training times of individuals in generations 1 through 25 (left) vs. generations 176 through 200 (right).

*Case 2, thirty six-step prediction:* This simulation is similar to the previous one but for a thirty six-step MG-17 prediction. Here the best evolved network produced a training MSE of 0.0077 and a test MSE of 0.0114. The average training time for the ancestor of this network was 53.778 seconds, whereas the same time for the evolved network was 4.401 seconds. This indicates a 12.2 fold improvement of  $C_t$  under temporal MDL. Figure 4 shows the MSEs of evolving networks (left) and their sizes,  $N_w$  (right). Figure 5 visualizes the effect of temporal MDL on time complexities of the evolving population further by depicting the histograms of training times of individuals in generations 1 through 43 (left) vs. generations 130 through 172 (right).

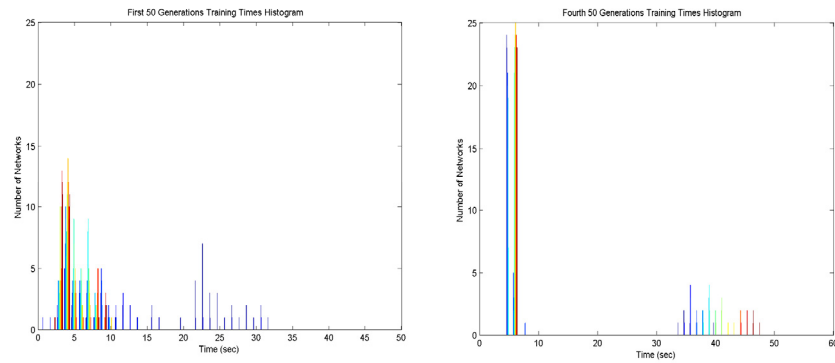


**Fig. 2 MG17 six-step prediction. Left: MSEs of evolving networks. Right: their sizes.**

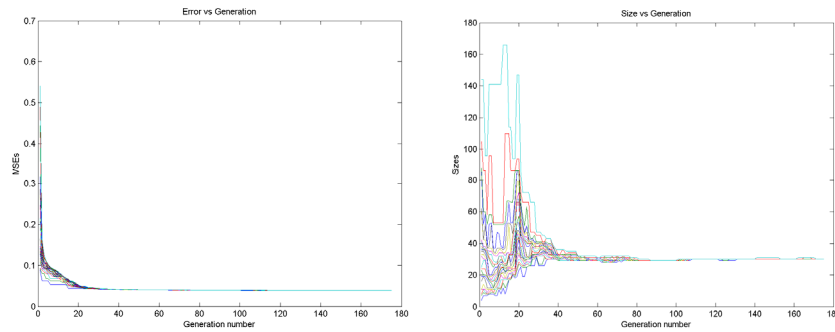
## CONCLUSIONS

MDL is an embodiment of the well known *Occam's Razor* principle. It provides an answer to the bias-variance dilemma, and has been used by different researchers as a means to regularize neural networks (Zemel, 1995). Here we introduced a practical,

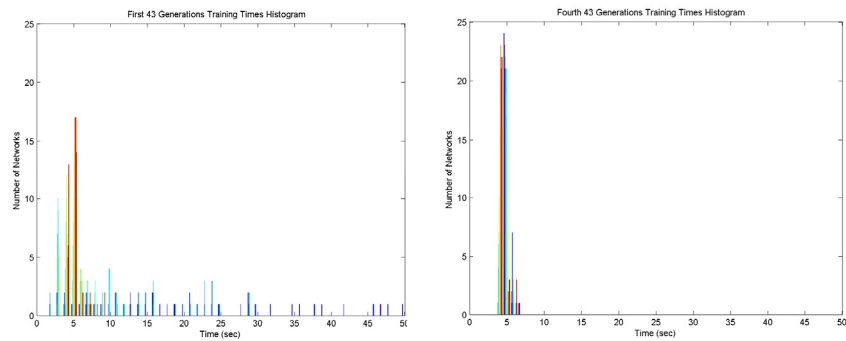
temporal version of MDL that reduces the size and error of evolutionary neural networks through reduction of actual execution time, and thus makes a better use of the given computational real state by breeding lean and fast solutions within an ecology that is made of the hosting hardware and the other competing solutions. In this sense, these-



**Fig. 3 Histogram of evolving network training times, MG17 six-step prediction, generations 1-25 (left) and 176-200 (right).**



**Fig. 4 MG17 thirty six-step prediction. Left: MSEs of evolving networks. Right: their sizes.**



**Fig. 5 Histogram of evolving network training times, MG17 thirty six-step prediction, generations 1-43 (left) and 130-172 (right).**

results are comparable to those of evolving machine codes in Tierra artificial life system (Ray, 1991; Ray, 1994). The parsimony of neural networks obtained under temporal MDL is especially important when the training data points are scarce, which is usually the case for the real world problems. It should be noted that the principle of temporal MDL can be applied to the evolution of other time-critical statistical models and can be explored as future work.

## REFERENCES

- Box, G.E.P., Jenkins G.M., and Reinsel G.C., 1994, "Time Series Analysis: Forecasting and Control," Third edition, Prentice Hall, Upper Saddle River, NJ.
- Cover T., 1965, "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition," *IEEE Transaction on Electronic Computers*, pp. 326-334, 1965.
- de Menezes M. A., and dos Santos R. M. Z., 2000, "The Onset of Mackey-Glass Leukemia at the Edge of Chaos," *International Journal of Modern Physics C*, Vol. 11, No. 8.
- Derakhshani R., 2005a, "GETnet: A General Framework for Evolutionary Temporal Neural Networks," To appear in *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*, Montreal, Canada.
- Derakhshani R., 2005b, "Spoof-Proofing Fingerprint Systems Using Evolutionary Time-Delay Neural Networks" *Proceedings of IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety*, Orlando, FL.
- Grünwald P.D., 2005, "Minimum Description Length Tutorial," In P. Grünwald, I.J. Myung, M. Pitt (editors). *Advances in Minimum Description Length: Theory and Applications*. MIT Press, April 2005. Preprint online at <http://homepages.cwi.nl/~pdg/ftp/mdlintro.pdf>.
- Hansen M. H., and Yu B., 2001, "Model Selection and the Principle of Minimum Description Length," *Journal of the American Statistical Association*, Vol. 96, No. 454, pp. 746-774.
- Haykin S., 1999, "Neural Networks, A Comprehensive Foundation, 2nd Edition, Prentice Hall, Upper Saddle River, NJ.
- Hinton, G. E., and van Camp, D., 1993, "Keeping Neural Networks Simple by Minimizing the Description Length of the Weights," *Proceedings of the 6th Annual Workshop on Computer Learning Theory*, pp. 5-13. ACM Press, New York, NY. Online at <http://citeseer.ist.psu.edu/hinton93keeping.html>
- Kurzweil R., 2000, "The Age of Spiritual Machines: When Computers Exceed Human Intelligence," Penguin, New York.
- Mackey, M. C., and Glass, L., 1977, "Oscillation and Chaos in Physiological Control Systems," *Science*, Vol. 197, pp 287-289.
- Moller M., 1993, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, Vol. 6, No. 4, pp 525-533.
- Principe J., Euliano N., and Lefebvre W., 2000, "Neural and Adaptive Systems: Fundamentals Through Simulations," Wiley, New Yourk.
- Ray, T. S. 1991, "Evolution and Optimization of Digital Organisms," Billingsley K. R., E. Derohanes, H. Brown, III [eds.], *Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers*, Athens, GA, 30602: The Baldwin Press, University of Georgia. pp. 489-531.
- Ray, T. S., 1994, "An Evolutionary Approach to Synthetic Biology: Zen and the Art of Creating Life," *Artificial Life*, Vol. 1, pp. 179 – 210, The MIT Press. Reprinted In: Langton, C. G. [ed.], *Artificial Life*, an overview, The MIT Press, 1995.
- Rissanen J., 1989, "Statistical Inquiry," Singapore, World Scientific.
- Sathyanarayan S. R., and Kumar C., 1996, "Evolving Recurrent Bilinear Perceptrons for Time Series Prediction," *ASME Press Series on Intelligent Engineering Systems through Artificial Neural Networks*, ANNIE-1996 Proceedings, St. Louis, Missouri.

- Schwefel H. P., 1981, "Numerical Optimization of Computer Models," John Wiley & Sons, New York.
- Seidl D.R., and Lorenz D., 1991, "A Structure by Which a Recurrent Neural Network can Approximate a Nonlinear Dynamic System," *Proceedings of International Joint Conference on Neural Networks*, Vol. 2, pp. 709-714.
- Siegelmann H. T., and Sontag E. D., 1995, "On the Computational Power of Neural Networks," *Journal of Computer Systems Science and Engineering*, Vol. 50, No. 1, pp. 132-150.
- Siegelmann, H.T., Horne, B.G., Giles, C.L., 1997, "Computational Capabilities of Recurrent NARX Neural Networks," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Vol.27, Issue 2, pp 208-215.
- Wan, E., World Wide Web: <http://www.cse.ogi.edu/~ericwan/data.html>
- Yao, X., 1999, "Evolving Artificial Neural Networks," *Proceedings of IEEE*, September, 87 (9) pp. 1423-1447.
- Zemel R. S., 1995, "Minimum Description Length Analysis," *The Hand Book of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, Massachusetts, The MIT press.