Simultaneous Evolution of Structure and Activation Function Types in Generalized Multi–Layer Perceptrons

HELMUT A. MAYER* and MARC STRAPETZ* and ROMAN FUCHS+
*Department of Computer Science
+Institute of Zoology
University of Salzburg
Jakob–Haringer–Strasse 2, A–5020 Salzburg
AUSTRIA
{ {

Abstract: - The most common (or even only) choice of activation functions for multi–layer perceptrons (MLPs) widely used in research, engineering and business is the logistic function. Among the reasons for this popularity are its boundedness in the unit interval, the function's and its derivative's fast computability, and a number of amenable mathematical properties in the realm of approximation theory. However, considering the huge variety of problem domains MLPs are applied in, it is intriguing to suspect that specific problems call for single or a set of specific activation functions. Also, biological neural networks (BNNs) with their enormous variety of neurons mastering a set of complex tasks may be considered to motivate this hypothesis. We present a number of experiments evolving structure and activation function types (AFTs) of generalized multi–layer perceptrons (GMLPs) using the parallel netGEN system to train the evolved architectures. The number of network parameters subjected to evolution is increased in various steps from learning parameters only for a GMLP of fixed architecture to simultaneous evolution of structure and activation function types. For experimental comparisons we utilize a synthetic and a real–world classification problem, and a chaotic time series prediction task.

Key–Words: - Multi–Layer Perceptrons, Activation Functions, Evolutionary Algorithms, Classification, Prediction

1 Introduction

While many researchers have investigated a variety of methods to improve *Artificial Neural Network* (ANN) performance by optimizing training methods, learn parameters, or network structure, comparably few work has been done towards using activation functions other than the logistic function. It has been shown that a two–hidden–layer MLP with sigmoidal activation function can implement arbitrary convex decision boundaries (Lippmann, 1987). Moreover, such an MLP is capable of forming an arbitrarily close approximation to any continous nonlinear mapping (Cybenko, 1987). However, common to these theorems is that the number of neurons in the layers is at best bounded, but can be extremely large for practical purposes. E.g., for a one–hidden–layer MLP Barron (1993) derived a proportionality of the sum square error (SSE) to the number of neurons according to $SSE \sim \frac{1}{\sqrt{T}}$ with T being the number of neurons (for target functions having a *Fourier* representation) (Barron, 1993).

If we consider the simple example of a rectangular function, it becomes obvious that a rectangular activation function can approximate this target function much easier than a sigmoidal function. But before we set up the experiments let us briefly review related work and the basic computational properties of biological neurons.

1.1 Related work

DasGupta and Schnitger (1993) compared different activation functions in terms of approximation power. The standard sigmoid reaches an approximation power comparable to or better than classes of more established functions investigated in the approximation theory (i.e., splines and polynomials) (DasGupta and Schnitger, 1993).

Jordan (1995) states that the logistic function is a natural representation of the posterior probability in a binary classification problem. However, for layered networks, within the framework of function approximation theory, there is no compelling reason to use the logistic function as compared to any other smooth bounded monotonic nonlinearity (Jordan, 1995).

Liu and Yao (1996) evolved the structure of *Generalized Neural Networks* (GNN) with two different activation function types, namely, sigmoid and Gaussian basis function. Experiments with the *Heart Disease* data set from the UCI machine learning benchmark repository revealed small differences in classification error slightly favoring GNNs over ANNs solely using sigmoid or Gaussian basis activation function (Liu and Yao, 1996).

Sopena et al. (1999) presented a number of experiments (with widely–used benchmark problems) showing that multilayer feed–forward networks with a sine activation function learn two orders of magnitude faster while generalization capacity increases (compared to ANNs with logistic activation function) (Sopena et al., 1999).

1.2 Biological neurons

The computational properties of a single neuron are determined by its morphology, composition of cell membrane and the number and position of *Synapses* (inputs) along the *Dendrites* and its single *Axon* (output). As difference in *Form* means also difference in function, neurons can be classified into several types based on their architecture, e.g. *Pyramidal Cells* exhibiting excitatory functionality or inhibitory *Basket cells*. But even within a main cell type highly variable structures have been observed (Miell, 1989). On the other hand, neurons of different morphology can probably perform the same (or similiar) computational operations, if the number and distribution of synapses levels off the difference in structure. However, considering the enormous complexity of even a single neuron, it is likely that no two neurons in a biological individual are identical w.r.t. computational function. An even higher level of complexity is introduced by the fact that biological neurons are, of course, living structures changing their long– and short–term behavior under the influence of various *Neurotransmitters* (synaptic plasticity) (Shepherd 1994). All these properties contribute to the astonishing plasticity of biological neural networks and could give some important clues for further improvement of ANNs.

2 Evolution of Structure and Activation Function Types

The technical platform for the evolution of GMLPs 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* – single bases 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. For the evolution of AFTs each neuron marker is followed by a sequence of bases coding for an index to an AFT. If a neuron marker indicates the absence (or pruning) of a particular neuron i, the AFT and all connections associated with a specific neuron become obsolete. As a consequence, most ANN chromosomes contain noncoding regions (Mayer, 1998).

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.

2.1 ANN fitness function

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*

$$\mathcal{F} = \alpha_1 \frac{1}{1+\mathcal{E}_m} + \alpha_2 \frac{1}{1+\mathcal{E}_c} \qquad (1)$$

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. Hence, we set $\alpha_2 = 0.01$.

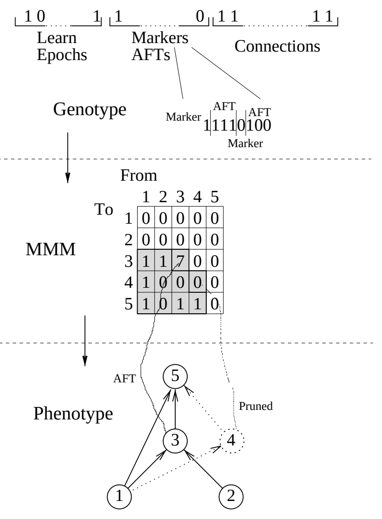

Figure 1: ANN encoding scheme.

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 $p_m = \frac{1}{l}$ ($l \ldots$chromsome length), Crossover = 2–Point, Selection Method = Binary Tournament.
ANN Parameters: Network Topology = Generalized Multi–Layer Perceptron, Activation Function (hidden neurons) = AFT (Table 1) / (output neurons) = AFT, Output Function (all neurons) = Identity, Training = RProp, Learning Parameters Δ_0 (max 1.0), Δ_{max} (max 50.0), Number of Training Epochs (max 1000) = Evolutionary.

3 Experimental Setup

The various AFTs used in this paper are listed in Table 1 as implemented in the *Stuttgart Neural Network Simulator* (SNNS) (Zell et al., 1994) we are employing for ANN training. Note that this is only the subset of differentiable AFTs required for any type of back–propagation training. We have also tested *Simulated Annealing* as training regime which allows non–differentiable AFTs as well. However, the computational cost was unrelated to ANN performance which usually was slightly worse compared to RProp training.

In an effort to separate the influence of ANN structure and AFTs, we devised the following experiments:

- `fixedNet` – For the test problems a standard network given in the literature is adopted, the AFT for the hidden neurons is set to the default type `Logistic`, and only the RProp learn parameters Δ_0 and Δ_{max} and the number of epochs are evolved.
- `fixedTop` – The network topology given by the standard net remains unaltered, while AFTs and learn parameters are subjected to evolution.
- `fixedAct` – ANN structure and learn parameters evolve with the default AFT `Logistic` utilized for all hidden neurons.
- `fixedNon` – All the ANN parameters – structure, AFTs, and learn parameters evolve.

Name	Function
`Identity`	$a_j = net_j$
`IdentityPlusBias`	$a_j = net_j + \Theta_j$
`BSB`	$a_j = net_j \times \Theta_j$
`Elliot`	$a_j = \frac{net_j + \Theta_j}{1+\lvert net_j+\Theta_j\rvert}$
`TanH`	$a_j = \tanh(net_j + \Theta_j)$
`TanH_Xdiv2`	$a_j = \tanh(\frac{net_j}{2} + \Theta_j)$
`Logistic`	$a_j = \frac{1}{1+e^{-(net_j+\Theta_j)}}$

Table 1: Types of activation functions used for evolution (net_j / neuron input, Θ_j / bias, and a_j / activation).

For a rough description of the ANN structure we use the number of hidden neurons and the total number of connections. The prefix `st` is used for a standard net, and `ev` identifies an evolved net. Hence, `ev`5-80 would represent an evolved GMLP with 5 hidden neurons and 80 connections. The maximum number of hidden neurons for the evolution of ANN structure is always identical to the number of hidden neurons in the standard net.

3.1 Test problems

For each of the three test problems, we use a training set (ANN teaching), a validation set (ANN fitness), and a test set (ANN performance) with all data sets being pairwise disjoint.

The *Two Spirals* problem is a synthetic, but nevertheless hard problem. The task is to discriminate between two sets of points which lie on two distinct spirals in the plane. These spirals coil three times around the origin and around one another (CMU-Repository, 1993). Training, validation, and test set comprise 194 patterns each. The standard net is an MLP with one hidden layer with two input neurons, and two output neurons (one for each spiral, winner takes all) and 16 hidden neurons (`st16-64`).

The *Glass* problem is taken from the PROBEN1 benchmark suite of real–world problems (Prechelt, 1994). The nine input values represent the percent content of eight different chemical elements of glass splinters and their refractive index. The six classes represented by the output neurons are float processed building windows, non float processed building windows, vehicle windows, containers, tableware, and head lamps. The training, validation, and test sets contain 107, 53, and 54 patterns, respectively. The standard net is a $9-8-6$ MLP (`st8-120`).

For these classification problems the model error in the fitness function is simply given by $\mathcal{E}_m = \frac{e_v}{n_v}$, where e_v is the number of misclassifications on the validation set, and n_v its respective size. In addition to the classification accuracy we use the statistical measure *Coefficient of Agreement* κ which gives the improvement over random assignment of patterns to classes. $\kappa = 0.0$ would reveal a classifier performance equal to random assignment, while $\kappa = 1.0$ indicates perfect classification.

The *Mackey–Glass* time series is generated by a differential delay equation. The four input neurons represent the points in time $(x(t-18), x(t-12), x(t-6), x(t))$, and the network is trained to predict the future value $x(t+84)$. (CMU-Repository, 1993). 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 and the next 2000 patterns for the validation set (test set unchanged). The standard net is a $4-10-10-1$ MLP (`st20-150`). For all experiments the AFT `Identity` has been assigned to the single output neuron.

We define the model error for prediction tasks as the normalized root mean square (NRMS) on the validation set

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

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 $i = 1 \ldots N$. For additional assessment of the evolved networks the absolute maximal prediction error $|\epsilon_{max}|$ for a single point in time is given.

4 Experimental Results

In Table 2 the results for the various experiments with the Two Spirals problem are presented.

Two Spirals				
RProp	Validation Set		Test Set	
	Acc	Kappa	Acc	Kappa
fixedNet	0.7010	0.4021	0.7010	0.4021
fixedTop	0.6701	0.3402	0.6598	0.3196
fixedAct	0.8608	0.7216	0.8814	0.7629
fixedNon	0.8505	0.7010	0.8351	0.6701

Table 2: *Two Spirals* – Classification accuracy (Acc) and coefficient of agreement (Kappa) for various ANN parameters subjected to evolution.

The most noteable fact is the large difference in classification accuracy for experiments using `st16-64` and evolved ANN architectures This behavior can be explained by looking at the evolved nets `ev14-122` (`fixedAct`) and `ev16-154` (`fixedNon`). Due to the GMLP structure containing short–cut connections the number of connections is roughly twice the number of the standard net which is comparable to structures reported at (CMU-Repository, 1993). As the maximal number of epochs (1000) is rather small compared to an overview of results presented in (Sopena et al., 1999), we trained the final networks for up to 20,000 epochs, but performance sometimes even decreased. ANN structures and specific training regimes to reach perfect classification have been reported, but our main goal was to investigate the influence of a variety of AFTs.

It can be seen (Table 2) that the usage of AFTs different from `Logistic` in `fixedTop` and `fixedNon` slightly decrease performance. Interestingly, out of all hidden and output neurons of the evolved networks only two AFTs are not `Logistic`. Experiment `fixedTop` generated the standard net with one `TanH` and one `TanH_Xdiv2`, while `fixedNon` yielded the net `ev14-122` with one `TanH` and one `Identity`. These results indicate that of all AFTs used in this work `Logistic` is the best choice for the Two Spirals problem.

Table 3 gives an overview on evolution results for the Glass problem.

In these experiments the evolution of AFTs gives a slight advantage over networks with the default AFT.

Glass				
RProp	Validation Set		Test Set	
	Acc	Kappa	Acc	Kappa
fixedNet	0.7407	0.6365	0.6038	0.4151
fixedTop	0.7593	0.6640	0.6604	0.5005
fixedAct	0.7778	0.6891	0.6604	0.5023
fixedNon	0.7778	0.6910	0.6981	0.5717

Table 3: *Glass* – Classification accuracy (Acc) and coefficient of agreement (Kappa) for various ANN parameters subjected to evolution.

Mackey–Glass				
RProp	Validation Set		Test Set	
	NRMS	$\|\epsilon_{max}\|$	NRMS	$\|\epsilon_{max}\|$
fixedNet	0.2270	0.1621	0.2321	0.1511
fixedTop	0.2463	0.1874	0.2380	0.1400
fixedAct	0.2124	0.1778	0.2109	0.1793
fixedNon	0.2021	0.2412	0.2008	0.2313

Table 4: *Mackey–Glass* – Normalized root mean square (NRMS) and absolute maximum prediction error $|\epsilon_{max}|$ for various ANN parameters subjected to evolution.

Also, in comparison to `st8`-120 the evolved networks of `fixedAct` (`ev`5-79) and `fixedNon` (`ev`6-105) are smaller. The architecture of the latter is depicted in Figure 2.

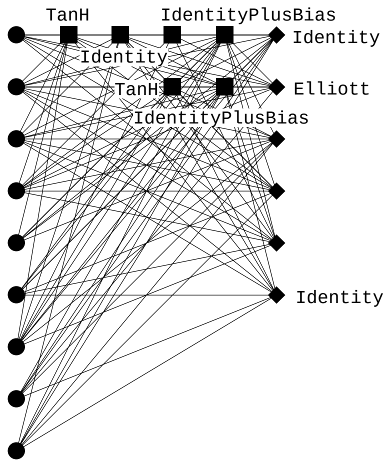

Figure 2: *Glass* – Architecture of the best GMLP evolved in experiment `fixedNon` (unlabeled AFTs are `Logistic`).

The results of GMLPs evolved to predict the Mackey–Glass time series are presented in Table 4.

Generally, the prediction error for the various experiments varies only slightly, but again the best result was found in experiment `fixedNon` (`ev`19-207), when structure and AFTs evolve simultaneously. As with the Two Spirals problem the default AFT `Logistic` seems to be the right choice for the Mackey–Glass prediction. Experiment `fixedTop` yielded an architecture with eight hidden neurons (out of 20) having AFTs different from the default AFT (this may be a reason for the worse prediction compared to `fixedNet`), whereas `fixedNon` evolved to an architecture with two `Identity` AFTs and one `TanH_Xdiv2` (all others `Logistic`). The occurrence of an `Identity` might indicate that the specific neuron could be pruned, because linear behavior can be solely modeled by the products of neuron output and weight and the sum of neuron inputs.

5 Summary

We have reported on experiments evolving structure and activation function types (AFTs) of generalized multi–layer perceptrons utilizing a synthetic and a real–world classification problem, and a chaotic time series prediction task. Tough, these first results demonstrate that the use of different AFTs in an ANN could improve its performance, it also shows that the logistic activation function used in the vast majority of MLP applications is a good (if not the best) choice. As a matter of fact, the consensus in usage of the logistic activation function can be also viewed as the result of an evolutionary process in the scientific community.

In this work we used a set of conventional differentiable, monotonic activation functions for the evolution of GMLP architecture, however, recent interest in non–monotic activation functions has provided first encouraging results in learning speed and ANN performance (Sopena et al., 1999). The use of non–monotonic (periodic) activation functions could again open paths back to biological neural networks, where some hypotheses predict the existence of a “pulse generator” periodically activating (and hereby synchronizing) brain areas.

A natural continuation of this work will be the evolution of complete activation functions based on a general model, e.g., splines, instead of using predefined AFTs. A first step in this direction will be the evolution of parameters of the logistic function so as to adapt the steepness of the function for each single neuron in the network. Moreover, as indicated above, the usage of non–monotonic, periodic functions for evolutionary design of ANN architectures will be investigated.

6 Acknowledgments

We would like to thank our student Karl Fürlinger who developed the ANN simulation tool we used to create the network graphs [1].

References

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39:930–944.

CMU-Repository (1993). ftp://ftp.cs.cmu.edu/afs/cs.cmu.edu/ project/connect/bench/. WWW Repository, Carnegie Mellon University.

Cybenko, G. (1987). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

DasGupta, B. and Schnitger, G. (1993). The Power of Approximating: A Comparison of Activation Functions. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems 5*, pages 615–622, San Mateo, CA. Morgan Kaufmann Publishers.

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, 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.

Jordan, M. I. (1995). Why the logistic function? A tutorial discussion on probabilities and neural networks. Computational Cognitive Science Technical Report 9503, Massachusetts Institute of Technology.

[1]The simulation tool *AnnEx* can be downloaded from `http://www.cosy.sbg.ac.at/~kfuer/AnnEx/`

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Processing*, 4(2):4–22.

Liu, Y. and Yao, X. (1996). Evolutionary Design of Artificial Neural Networks with Different Nodes. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 570–675.

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

Miell, C. (1989). The diversity of neuronal properties. In Durbin, R., editor, *The Computing Neuron*. Addison–Wesley.

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.

Prechelt, L. (1994). PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report TR 21/94, Universität Karlsruhe, Fakultät für Informatik, 76128 Karlsruhe, Germany.

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.

Sopena, J. M., Romero, E., and Alquézar, R. (1999). Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *Proceedings of the 9th International Conference on Artificial Neural Networks*.

Zell, A., Mamier, G., Vogt, M., Mach, N., Huebner, R., Herrmann, K.-U., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeogiou, A., Doering, S., and Posselt, D. (1994). *SNNS Stuttgart Neural Network Simulator, User Manual*. University of Stuttgart.