Differentiation of Neuron Types by Evolving Activation Function Templates for Artificial Neural Networks

Helmut A. Mayer and Roland Schwaiger
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
{helmut},{rschwaig}@cosy.sbg.ac.at

Abstract - In this paper we investigate the use of neuron–specific activation functions (AFs) within generalized multi–layer perceptrons (GMLP). We utilize the netGEN system not only to evolve the structure of an artificial neural network (ANN), but also to search for a set of AF templates which are assigned to specific neurons by evolution. This may be seen as a loose analogy to neuron differentiation in biological neural networks (BNNs). While BNNs employ different neuron types in functionally different brain areas, neuron differentiation in ANNs might be useful to increase the adaptability to specific problems. The evolution of AF templates is based on evolving the control points of a cubic spline function, hence non–monotonous AFs of (nearly) arbitrary shape may be generated. We present a number of experiments evolving ANN structure and AF templates using the parallel netGEN system to train the evolved architectures. We compare the evolved cubic spline ANNs with evolved sigmoid ANNs on synthetic classification problems and a time series prediction task so as to assess the benefits of problem–adapted AF templates.

I. 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 [6]. Moreover, such an MLP is capable of forming an arbitrarily close approximation to any continous nonlinear mapping [4]. 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) [1].

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. Consequently, we investigate the evolution of AF templates, as intuitively different problems call for different neuron types having problem–specific AFs.

A. *Related Work*

Liu and Yao (1996) evolved the structure of *Generalized Neural Networks* (GNN) with two different AF templates, 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 [7].

Vecci et al. (1998) introduced the *Adaptive Spline Neural Network* (ASNN), where activation functions are described by *Catmull–Rom* cubic splines. ANN error (including a regularization term) then is not only dependent on the weights, but also on the spline parameters which together with the weights are adapated by backpropagation learning. It has been reported that ASNNs yield a performance comparable to networks using the logistic AF, while using a smaller number of hidden neurons than the conventional network [16].

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) [15].

Our own work along these lines started out with the evolution of neuron types from a predefined set of monotonous AFs [10]. For each neuron an index is evolved which refers to a specific activation function. Tests on three different benchmark problems showed that there is only small room for improvement using the specific set of AF templates. Often the standard sigmoid function has been employed for most neurons in the evolved network which agrees with the choice of most ANN designers.

Coelho et al. (2001) evolved neuron types by utilizing a complete neuron population for each different AF template. They employed eight common AFs including *Gaussian* (non–monotonic) and *Hyperbolic Tangent*, but the logistic AF has not been chosen as an AF template. Results on benchmark problems did not allow a consistent conclusion w.r.t. the neuron types in the best evolved networks. However, it could be observed that in the evolved networks the step function (Heavyside) occurs frequently and the polynomial function $a(x) = x^n$ rarely [3].

II. EVOLUTION OF ANN STRUCTURE AND ACTIVATION FUNCTIONS

The technical platform for the evolution of GMLPs is the netGEN system searching a problem–adapted ANN architecture by means of an *Evolutionary Algorithm* (EA), and training the evolved networks using the *Stuttgart Neural Network Simulator* (SNNS) [17]. In order to speed up the evolutionary process, an arbitrary number of workstations may be employed to train individual ANNs in parallel [5].

The ANN's genetic blueprint is based on a direct encoding suggested by Miller et al. [11]. The basic structure of the genotype is depicted in Figure 1.

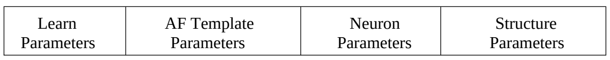

Figure 1: The organization of the ANN genotype.

The binary encoded GMLP chromosome contains four main sections. The *Learn Parameters* encode values used for ANN training, the *Activation Function Template Parameters* are used to describe one or more activation functions, the *Neuron Parameters* indicate the neuron type, and the *Structure Parameters* explicitely specify each connection of the network (direct encoding). Figure 2 shows the details of the learn parameter and the AF template parameter section.

The learn parameter section contains the number of epochs and two learning parameters for the selected training algorithm *Resilient Back–propagation* (RProp) [13]. The evolution of the number of epochs for ANN training is a measure to avoid overfitting with the additional benefit of another critical ANN parameter being discovered by evolution.

The most interesting feature of the presented encoding are *Markers* – single bases (bits) which are a simple analogue to activators/repressors regulating the expression of wild–type genes. A hidden neuron marker (part of the neuron parameters) determines, if the specific neuron associated with it is present in the decoded network. As the number of output neurons is usually strictly dependent on the specific problem the ANN should learn, these neurons are not controlled by markers.

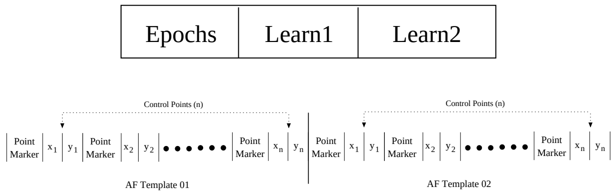

Figure 2: Details of the learn parameter section (above) and the activation function template parameter section (below) of the ANN genotype.

The activation function template is composed by a maximum number of n control points of a cubic spline. In order to also allow fewer control points, each control point is "regulated" by a point marker. If a point marker indicates the absence of the corresponding point, the control point is not used for cubic spline calculations to construct the AF template. Similarly, an active neuron marker prunes the neuron it controls, and all connections associated with that neuron become obsolete. As a consequence, most ANN chromosomes contain noncoding regions [8]. When using the single logistic AF template, only the learn and structure parameters are evolved.

If a neuron is expressed, the neuron type is given by an AF index in the neuron parameters. Evidently, when a single AF template is evolved the size of the bit field encoding the index is 0.

ANN structure is represented by a linearized binary adjacency matrix. As in this work we are only concerned with feed–forward architectures, all elements of the upper triangle matrix must be 0, hence they are not included in the ANN genotype. Due to this linearization we are able to use the standard 2–point crossover operator for recombination. When constructing the ANN phenotype the AF index (0, if neuron not expressed) is stored in the main diagonal of the connection matrix as shown in Fig. 3 for an exemplary network.

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.

A. *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

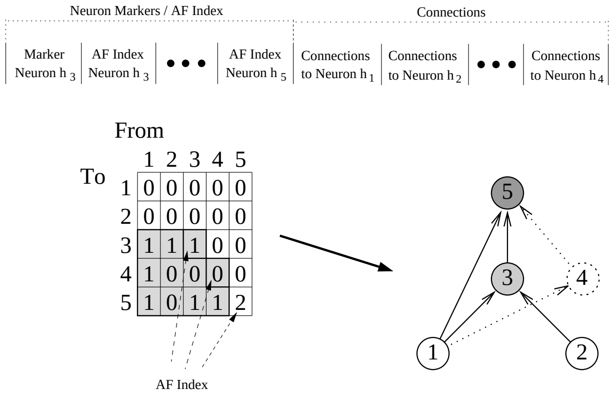

Figure 3: Genotype/Phenotype mapping of ANN structure.

Complexity Fitness

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

with $\alpha_1 + \alpha_2 = 1.0$, and $\mathcal{E}_m$ being the model error. Earlier work showed that α_2 in the range of $0.001 - 0.01$ is sufficient to guide the evolution towards ANNs of low complexity. In effect the complexity term acts as a "tie–breaker" between different ANNs with (nearly) identical model error, where the less complex network receives a slightly better fitness. In this work we set $\alpha_1 = 0.99$.

B. *EA and ANN Parameters*

The following EA and ANN parameters have been used with all the experiments in this paper:
EA Parameters: Population Size = 50, Generations = 50, Crossover Probability $p_c = 0.6$, Mutation Probability $p_m = 0.005$, Crossover = 2–Point, Selection Method = Binary Tournament.
ANN Parameters: Network Topology = Generalized Multi–Layer Perceptron, Activation Function (hidden and output neurons) = Sigmoid or evolved Cubic Splines, 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.

C. *Cubic Spline Parameters*

A cubic spline activation function is described by n control points $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$ where $i = 1, \ldots, n$. These n points define $n-1$ intervalls on the x-axis denoted by $x \in [x_i, x_{i+1}]$ with $x_{i+1} > x_i$. In each of these intervalls a function $f_i(x)$ is defined by

$$f_i(x) = f_i(x_i) + a_i(x - x_i) + b_i(x - x_i)^2 + c_i(x - x_i)^3 \quad (2)$$

with constants $a_i, b_i, c_i \in \mathbb{R}$. Demanding equality of the function value, and the first and second derivative at the interval borders the constants can be determined for each interval yielding a continuous and differentiable function composed of a number of cubic splines.

With respect to the ANN genotype (Fig. 2) the maximum number of control points n_c has to be set prior to the evolutionary run. Also, the x–, and y–range of the cubic spline activation function has to be fixed to specific intervals $[x_{min}, x_{max}]$ (sensitivity interval) and $[a_{min}, a_{max}]$ (activation interval), respectively. However, within these bounds the evolutionary process may freely float.

For all experiments we set the number of control points $n_c = 8$ and the activation interval $[a_{min}, a_{max}] = [0.0, 1.0]$. The sensitivity interval depends on the specific problem.

III. EXPERIMENTAL SETUP

The main question to be adressed is the performance of a cubic spline network in comparison to a GMLP with conventional, sigmoidal AFs. Intuitively, the decision boundaries in classification problems could be modeled more easily by cubic splines of great plasticity than by the "frozen" logistic function. Thus, we choose the simple continous XOR problem with piece–wise linear decision boundaries, and the synthetic, but nevertheless hard *Two Spirals* problem with complex, nonlinear decision boundaries.

During the evolutionary process the ANN structure is trained on a training set and evaluated on a validation set (ANN fitness). Finally, the performance of the evolved best net is measured on a test set. Each evolutionary run is repeated 20 times.

The continous XOR (cXOR) problem is defined by a division of the unit square into four squares of identical area being separated by the lines $x = 0.5$ and $y = 0.5$. The lower left and the upper right square are labeled with binary value 1, the two remaining squares represent the class with binary value 0. The corresponding ANN has two inputs and one output neuron. The binary output value 1 is defined by an activation of the output neuron $a_o > 0.5$, otherwise the output is mapped to 0. The training set is composed of 100 randomly selected points in the unit square, the validation set contains 200 random patterns, and the test set is made of 10,000 equally spaced points covering the complete unit square.

The task in the *Two Spirals* problem 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 [2]. Training, validation, and test set comprise 194 patterns each. The basic ANN structure is defined by two input neurons and two output neurons (one for each spiral, winner takes all).

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 order to not only experiment with classification problems, we also present results on a time series prediction task. 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 $x(t+84)$ [2]. The original 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). For all experiments the identity AF 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

$$\mathcal{E}_m = \sqrt{\frac{\sum_{i=1}^{N} (x_i - o_i)^2}{\sum_{i=1}^{N} (x_i - \overline{x})^2}}, \tag{3}$$

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. It should be noted that the time series results have only qualitative character, as we present results of single runs (which took approx. 20 hours each, on a cluster of 17 Pentium PCs varying from 200–800 MHz CPUs).

IV. EXPERIMENTAL RESULTS

After extending the netGEN system with the required functionality we performed some runs with the simple XOR problem so as to test the new software components. Surprisingly, when evolving AF templates for the XOR net, we mostly arrived at networks without any hidden neurons (two inputs, one output with cubic spline AF). This seems to contradict the well–known fact that a single perceptron cannot learn the XOR function (Minsky's Paradox [12]). In a strict sense our evolved XOR net is not a perceptron which is defined having a threshold AF, but even a logistic AF does not do the trick. While these functions have the desirable non–linearity, they are monotonous in contrast to the non–monotonous evolved cubic spline AFs. A non–monotonous triangle AF being a simplification of the evolved cubic spline AFs in our experiments enables a single neuron to solve the XOR problem. Recently, Rueda and Oommen (2000) reported on this observation in the context of statistical pattern recognition proposing pairwise linear classifiers [14].

In Table 1 the results for the continous XOR (cXOR) problem are presented.

We set the maximum number of hidden neurons to three and the sensitivity interval of the cubic splines to $[-10.0, 10.0]$. The number of AF templates in the evolutionary runs is set to 1, 2, and 4, yielding ANN genotypes with 175, 315, and 591 bases, respectively, while the usage of the single logistic AF template resulted in a chromosome with 39 bases. The results vary only slightly, however each best cubic spline network outperforms the evolved sigmoid net, even though the search space is larger for the former. It can also be observed that the average number of neurons (#H) and connections (#C) is decreasing with the number of AF templates, which could be interpreted as a consequence of increased computational power introduced by the AF templates. Figure 4 shows the structure and the corresponding AFs of the best evolved networks.

cXOR					
AF	Structure		Test Set		
	#H	#C	Acc	StdDev	Best
Sigmoid	2.80	10.70	0.9679	0.0065	0.9752
1Spline	2.85	10.95	0.9680	0.0079	0.9801
2Spline	2.75	10.45	0.9664	0.0118	0.9912
4Spline	2.55	9.60	0.9600	0.0134	0.9775

Table 1: *cXOR* – Structure parameters, classification accuracy (Acc) with standard deviation (StdDev) and overall best performance of evolved ANNs (averaged on 20 runs).

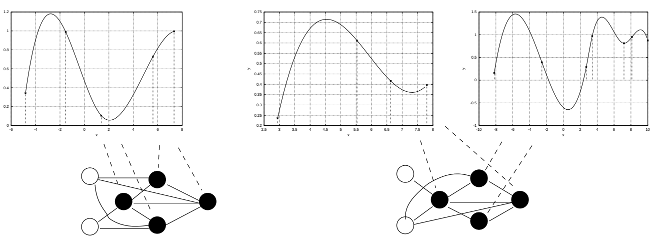

Figure 4: *cXOR* – Overall best ANNs 1Spline (left) and 2Spline (right) with corresponding AF templates.

In Fig. 5 the output neuron values for the complete unit square generated by the best sigmoidal and the best cubic spline ANNs are shown.

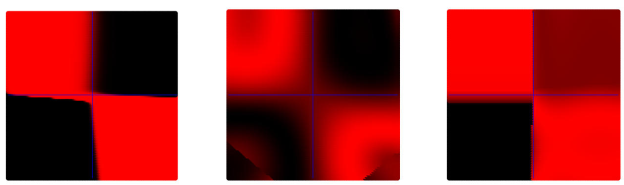

Figure 5: *cXOR* – Output neuron activation for unit square of the best evolved ANNs, Sigmoid (left), 1Spline (middle), and 2Spline (right)(darker = lower value).

Clearly, it can be seen that the decision boundaries of the spline nets are formed by more complex structures. The 2Spline net has found a very sharp and accurate decision boundary for $x = 0.5$, and a more fuzzy boundary at $y = 0.5$. The sigmoid net exhibits uniform activations (due to monotonicity of the AF), but has slight problems to accurately model the vertical and horizontal decision boundaries.

The results of GMLPs evolved for the Two Spirals problem are presented in Table 2.

TwoSpirals					
AF	Structure		Test Set		
	#H	#C	Acc	StdDev	Best
Standard	16.00	64.00	0.6150	0.0303	0.6701
Sigmoid	14.90	130.70	0.8379	0.0327	0.8918
1Spline	13.10	89.30	0.9327	0.0218	0.9639
2Spline	12.84	85.90	0.9235	0.0260	0.9639
4Spline	12.30	81.45	0.8768	0.0646	0.9536

Table 2: *TwoSpirals* – Structure parameters, classification accuracy (Acc) with standard deviation (StdDev) and overall best performance of evolved and standard (not evolved) ANNs (averaged on 20 runs).

The upper bound of hidden neurons has been set to 16, and the sensitivity interval of cubic splines to $[-100.0, 100.0]$. In addition to the evolutionary runs we present results (20 different initial sets of random weights) of a standard sigmoid network (not evolved) with one hidden layer (16 neurons) for which the evolved learn parameters of the best evolved sigmoid net have been used.

Here the performance of the spline nets is considerably superior to the sigmoidal networks. It also demonstrates the potential of evolutionary networks, when comparing the networks with only logistic AFs (Standard w/out, Sigmoid with evolution of structure).

Again, conclusions very similar to those of the cXOR problem can be drawn. Increasing complexity of the genotype leads to a decrease in network performance, and the size of the network w.r.t. number of neurons and connections decreases with increasing number of neuron types (AF templates).

In Fig. 6 the classification of the unit square by the best evolved ANNs (Table 2) is depicted.

The evolved sigmoid ANN generates pronounced non–contigous subspaces for the two classes which make misclassifications inevitable. The shape of the decision boundaries of the evolved spline net much more resemble the two spirals. There are only two small spots, where the red spiral (middle of upper right quadrant) and the black spiral (left end of x–axis) are disrupted.

In Table 3 a summary of the results on the Mackey–Glass problem is given, where the maximal number of hidden neurons has been fixed at 20, and the sensitivity interval of cubic splines at $[-100.0, 100.0]$.

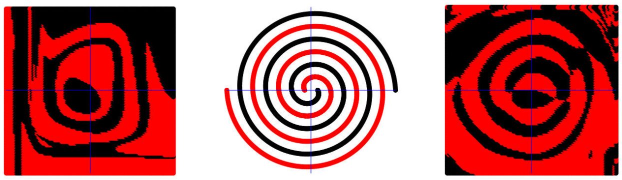

Figure 6: *TwoSpirals* – Unit square classification by best evolved sigmoid ANN (left) and best evolved 2spline ANN (right), and the test set (middle).

Mackey–Glass				
AF	Validation Set		Test Set	
	NRMS	$\lvert\epsilon_{max}\rvert$	NRMS	$\lvert\epsilon_{max}\rvert$
Sigmoid	0.1975	0.1565	0.1996	0.1422
1Spline	0.1534	0.1590	0.1543	0.1539
2Spline	0.1712	0.2210	0.1744	0.1991
4Spline	0.1864	0.1812	0.1866	0.1661

Table 3: Normalized root mean square (NRMS) and absolute maximum prediction error $|\epsilon_{max}|$ of ANN predictors for the Mackey–Glass time series.

Though, the results are gathered only from single runs, the spline ANNs might also be beneficial to this time series prediction problem. All evolved spline networks exhibited superior performance (measured by NRMS) than the overall best network (NRMS = 0.1868 on the test set) reported in previous work, where we coevolved ANN structure (with sigmoidal neurons) and training data [9].

V. SUMMARY

We have investigated the evolution of structure and activation function templates for generalized multi–layer perceptrons. Each neuron in the network is associated with a specific neuron type whose differentiation is based on the AF templates. Both, AF shape and the association of a neuron to a specific shape are subjected to evolution. We compared evolution of ANNs with the widely used logistic activation function in each hidden and output neuron to networks with AF templates based on cubic splines. The evolution of cubic spline AFs generates different nonlinear, non–monotonous activation functions of (nearly) arbitrary shape. The non–monotonicity of these activation functions led the way to solve the XOR problem with only a single neuron which is impossible by using monotonous activation functions.

Further experiments with more complex problems clearly showed the benefits of the cubic spline AF templates. By inspecting the decision boundaries of the two types of networks the potential of the spline nets to model linear and circular boundaries of class subspaces with great accuracy have been demonstrated.

Also, with the two spirals problem the number of hidden neurons and connections of a spline network was found to be considerably smaller than that of a sigmoid network. This is in accordance to reports on the adaptive spline neural networks presented in [16].

Continuing work will explore the use of cubic splines with improved locality, as a single mutation in a cubic spline gene can dramatically change the shape of the whole activation function. More advanced classes of cubic splines avoid this behavior, and a change of a single control point will result in only a local change of the cubic spline in the neighborhood of this point. Moreover, techniques to decrease the training time of ANNs are currently investigated so as to decrease the running time of network evolution employing the parallel netGEN system.

References

[1] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39:930–944, 1993.

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

[3] André Coelho, Daniel Weingaertner, and Fernando J. von Zuben. Evolving Heteregeneous Neural Networks for Classification Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 266–273, San Francisco, July 2001. Morgan Kaufmann.

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

[5] Reinhold Huber, Helmut A. Mayer, and Roland Schwaiger. 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*, August 1995.

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

[7] Yong Liu and Xin Yao. Evolutionary Design of Artificial Neural Networks with Different Nodes. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 570–675, 1996.

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

[9] Helmut A. Mayer and Roland Schwaiger. Evolutionary and Coevolutionary Approaches to Time Series Prediction Using Generalized Multi-Layer Perceptrons. In *Congress on Evolutionary Computation*, pages 275–280. IEEE, July 1999.

[10] Helmut A. Mayer, Marc Strapetz, and Roman Fuchs. Simultaneous Evolution of Structure and Activation Function Types in Generalized Multi-Layer Perceptrons. In *Proceedings of the 2001 WSES International Conference on Neural Networks and Applications*, pages 349–354. WSES Press, February 2001.

[11] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing neural networks using genetic algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[12] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: Introduction to Computational Geometry*. MIT Press, Expanded edition, 1988.

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

[14] L. Rueda and B. John Oommen. The Foundational Theory of Optimal Bayesian Pairwise Linear Classifiers. In *Proceedings of Joint IAPR International Workshops SSPR 2000 and SPR 2000 (LNCS 1876)*, pages 581–590. Springer, 2000.

[15] Josep M. Sopena, Enrique Romero, and René Alquézar. 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*, 1999.

[16] L. Vecci, F. Piazza, and A. Uncini. Learning and approximation capabilities of adaptive spline activation function neural networks. *Neural Networks*, (11):259–270, September 1998.

[17] Andreas Zell, Guenter Mamier, Michael Vogt, Niels Mach, Ralf Huebner, Kai-Uwe Herrmann, Tobias Soyez, Michael Schmalzl, Tilman Sommer, Artemis Hatzigeogiou, Sven Doering, and Dietmar Posselt. *SNNS Stuttgart Neural Network Simulator, User Manual*. University of Stuttgart, 1994.