Salzburg Interest Group on Integrated Systems

siGis-23-2001 October 5, 2001

Evolution of Cubic Spline Activation Functions for Artificial Neural Networks

10th Portuguese Conference on Artificial Intelligence (EPIA–2001)
December 17–20, 2001, Porto, Portugal

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

Roland Schwaiger
rschwaig@cosy.sbg.at

Department of Computer Science
University of Salzburg

Correspondence to:

Helmut Mayer
Universität Salzburg
Institut für Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
AUSTRIA

Telephone: +43-662-8044-6315
FAX: +43-662-8044-611

Evolution of Cubic Spline Activation Functions 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. The most common (or even only) choice of activation functions (AFs) 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 specific activation functions. Biological neural networks 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 functions of generalized multi–layer perceptrons (GMLPs) using the parallel netGEN system to train the evolved architectures. For the evolution of activation functions we employ cubic splines and compare the evolved cubic spline ANNs with evolved sigmoid ANNs on synthetic classification problems which allow conclusions w.r.t. the shape of decision boundaries. Also, an interesting observation concerning Minsky's Paradox is reported.

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 [1]. Moreover, such an MLP is capable of forming an arbitrarily close approximation to any continous nonlinear mapping [2]. 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) [3].

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.

1.1 Related Work

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 [4].

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 [5].

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) [6].

2 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) [7]. In order to speed up the evolutionary process, an arbitrary number of workstations may be employed to train individual ANNs in parallel.

The ANN's genetic blueprint is based on a direct encoding suggested by Miller et al. [8]. The basic structure of the genotype is depicted in Figure 1.

Learn Parameters	Neuron Parameters	Structure Parameters

Fig. 1. The organization of the ANN genotype.

The binary encoded GMLP chromosome contains three main sections. The *Learn Parameters* encode values used for ANN training, the *Neuron Parameters* are used to describe the activation function of each neuron, and the *Structure Parameters* explicitely specify each connection of the network (direct encoding).

Figure 2 shows the details of the learn parameter and the neuron parameter section.

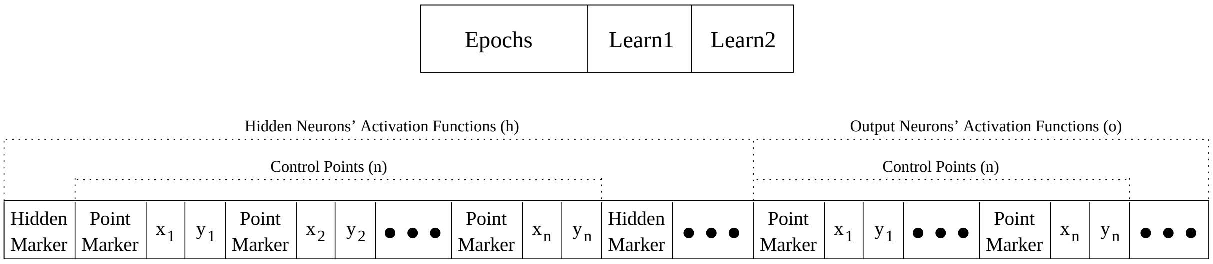

Fig. 2. Details of the learn parameter section (above) and the neuron parameter section (below) of the ANN genotype.

The learn parameter section contains the number of epochs and two learning parameters for the selected training algorithm *Resilient Back–propagation* (RProp) [9]. 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 neuron parameter section are *Markers* – single bases (bits) which are a simple analogue to activators/repressors regulating the expression of wild–type genes. A hidden neuron marker 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.

The activation function for each neuron 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. It should be noted that, if a neuron marker indicates the absence (or pruning) of a particular neuron, all control points for the cubic spline activation function and all connections associated with that neuron become obsolete. As a consequence, most ANN chromosomes contain noncoding regions. When using the logistic AF, only the hidden neuron markers are contained in the neuron parameter section.

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. Technically, the hidden neuron markers are stored in the main diagonal of the connection matrix constructed from the genotype's structure section 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*

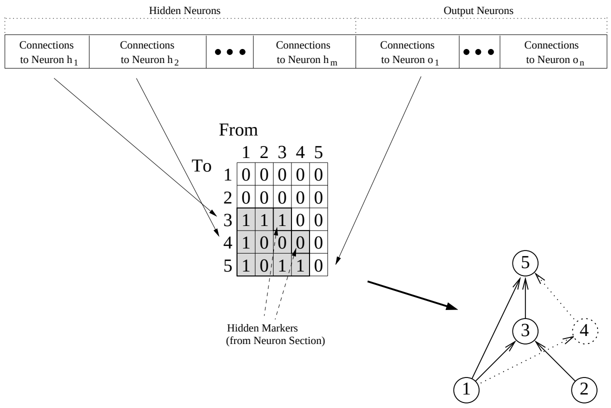

Fig. 3. Genotype/Phenotype mapping of ANN structure.

Pruning, since the system imposes an upper bound on the complexity of the network.

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*

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

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 $\alpha_1 = 0.99$.

2.2 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.

2.3 Cubic Spline Parameters

A cubic spline activation function is described by n control points $(x_i, y_i) \in \mathbb{R}$ 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

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

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 continous 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.

3 Experimental Setup

The experiments have been devised in order to answer two main questions. The first one is simply, if the rather complex organization of the ANN genotype containing genetic information on the neurons' activation function allows the evolutionary process to find any valid and reasonable networks. Thus, like cohorts of ANN researchers, we start out with the very simple but still interesting XOR problem.

The second 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.

For the XOR problem we use a network with two input 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. All pattern sets contain the only four example patterns.

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 an I/O–representation identical to the simple XOR problem. 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 [10]. 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.

4 Experimental Results

For the XOR problem we only evolved networks with cubic spline AFs. We set the maximum number of hidden neurons to three and the sensitivity interval of the neurons' cubic spline AF to $[-10.0, 10.0]$. In order to enable a more detailed fitness assignment we used $\mathcal{E}_m = MSE$ for the model error (Equ. 1). After 20 runs the average MSE of the single best nets of each run was 6.45×10^{-6}. Four of the best nets had a single hidden neuron, while the other 16 ANNs simply consisted of the two linear inputs and the output neuron with a cubic spline AF. These two simple structures are depicted in Figure 4.

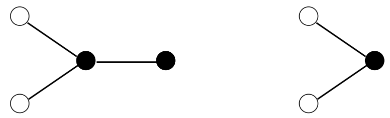

Fig. 4. Only two evolved XOR structures (white/input neuron, black/neuron with cubic spline AF).

The details of two evolved XOR nets are shown in Fig. 5.

When performing the calculations for the four possible input patterns, it can be seen that only a small portion of the sensitivity interval is used. Specifically, the big peak in each AF exceeding the activation interval seems to be useless, however, it could be argued that the steep slope provided by these peaks is profound guidance for the training algorithm.

The most interesting property of these networks is that they seem to contradict the well–known fact that a single perceptron cannot learn the XOR function

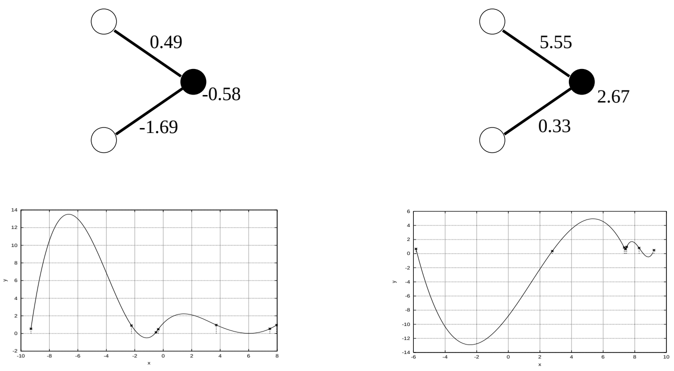

Fig. 5. Evolved XOR nets with weights, bias, and cubic spline activation function.

(Minsky's Paradox [11]). In a strict sense a perceptron is defined having a threshold AF, but even a logistic AF does not resolve Minsky's Paradox. While these functions have the desirable non–linearity, they are monotonous in contrast to the non–monotonous evolved cubic spline AFs. In all the reported aftermath of Minsky's work, it seems ironic that a simple triangle activation function would have resolved the paradox. Recently, Rueda and Oommen (2000) reported on this observation in the context of statistical pattern recognition proposing pairwise linear classifiers [12]. A non–monotonous triangle AF being a simplification of the sinusoidally shaped parts of the cubic spline AFs in Fig. 5 transforms the ANN into a pairwise linear classifier.

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

cXOR							
AF	Structure		Validation Set		Test Set		
	Hidden	Connections	Acc	StdDev	Acc	StdDev	Best
Sigmoid	2.80	10.70	0.9878	0.0053	0.9679	0.0065	0.9752
Spline	2.50	9.70	0.9780	0.0092	0.9583	0.0132	0.9767

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

Again, we set the maximum number of hidden neurons to three and the sensitivity interval of the cubic splines to $[-10.0, 10.0]$. The results slightly favor the sigmoid networks, however, the best overall ANN found is a cubic spline net (Fig. 6), even though the search space of the latter is much larger. While the genotype for the network with logistic AF consists of 39 bases, the cubic spline net chromosome comprises 538 bases.

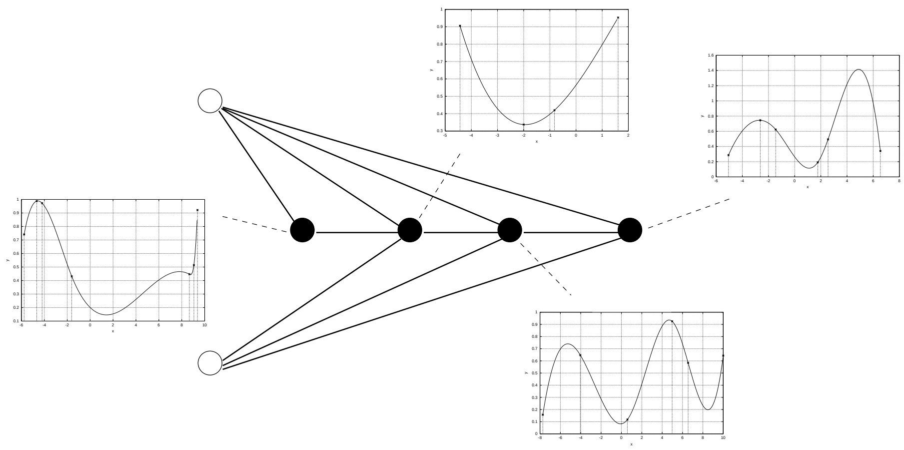

Fig. 6. *cXOR* – Overall best ANN with cubic spline AFs.

In Fig. 7 the output neuron value for the complete unit square by the best sigmoidal and the best cubic spline ANN is shown.

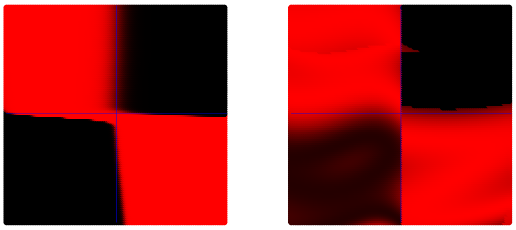

Fig. 7. *cXOR* – Output neuron activation for unit square of the best evolved sigmoid ANN (left) and the best evolved spline ANN(right)(darker = lower value).

Although, the classification results of both nets are nearly identical, it can be seen that the spline 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.

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

TwoSpirals							
AF	Structure		Validation Set		Test Set		
	Hidden	Connections	Acc	StdDev	Acc	StdDev	Best
Standard	16.00	64.00	0.6173	0.0281	0.6150	0.0303	0.6701
Sigmoid	14.90	130.70	0.8541	0.0273	0.8379	0.0327	0.8918
Spline	10.15	58.20	0.7090	0.0385	0.6791	0.0502	0.7887

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).

sigmoid network 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 evolved sigmoidal nets is clearly superior to the spline networks, but the difference in search space size (234 bases sigmoid, 2682 bases spline) might be the main reason for these results. In order to gather some evidence for this explanation, we started a single run evolving spline nets for 500 generations (instead of 50) and found a network (9 hidden, 36 connections) with an accuracy of 0.8711 and 0.8969 on validation and test set, respectively. As this single run took approx. 60 hours on 22 Linux PCs (200–800MHz Pentiums), we are working on a substantial speedup of the evolutionary process by using (very) early stopping techniques for ANN training.

In Fig. 8 the classification of the unit square by the best evolved ANNs (Table 2) is depicted.

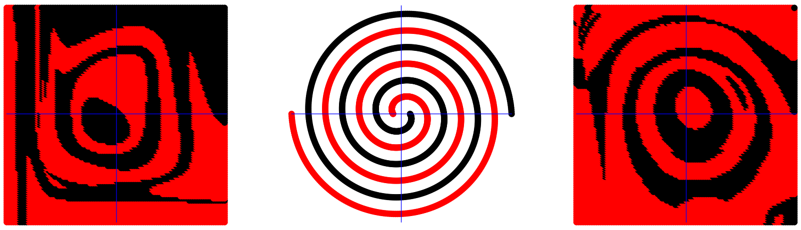

Fig. 8. *TwoSpirals* – Unit square classification by best evolved sigmoid ANN (left) and best evolved spline ANN (right), and the test set (middle).

Both evolved ANNs generate non–contigous subspaces for the two classes which make misclassifications inevitable. Though, the spline net has lower classification accuracy, the decision boundaries are of a shape closer resembling the spirals than those of the sigmoid net. However, the spirals are only defined in the areas shown in the middle figure (Fig.8), and these areas are classified more accurately by the sigmoid net. A specific source of error for the spline net is located in the upper right quadrant, where a black area is inside the inmost grey ring (falsely) covering a part of the grey spiral. As indicated above, further evolution of the complex cubic spline genes could eliminate such artefacts.

5 Summary

We have reported on experiments evolving structure and activation functions of generalized multi–layer perceptrons. We compared evolution of ANNs with the widely used logistic activation function in each hidden and output neuron to networks with cubic spline activation functions. Each neuron in the spline net can have a different nonlinear, non–monotonous activation function 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 the more complex two spirals problem favored the evolved sigmoid nets. However, the one order of magnitude larger genotype of the spline networks could be the main reason for this result. 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 [5].

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. Lippmann, R.P.: An introduction to computing with neural nets. IEEE Acoustics, Speech and Signal Processing **4** (1987) 4–22
2. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems **2** (1987) 303–314
3. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory **39** (1993) 930–944
4. Liu, Y., Yao, X.: Evolutionary Design of Artificial Neural Networks with Different Nodes. In: Proceedings of the Third IEEE International Conference on Evolutionary Computation. (1996) 570–675
5. Vecci, L., Piazza, F., Uncini, A.: Learning and approximation capabilities of adaptive spline activation function neural networks. Neural Networks (1998) 259–270

6. Sopena, J.M., Romero, E., Alquézar, R.: 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)
7. Zell, A., Mamier, G., Vogt, M., Mach, N., Huebner, R., Herrmann, K.U., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeogiou, A., Doering, S., Posselt, D.: SNNS Stuttgart Neural Network Simulator, User Manual. University of Stuttgart (1994)
8. Miller, G.F., Todd, P.M., Hegde, S.U.: Designing neural networks using genetic algorithms. In Schaffer, J.D., ed.: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, Philips Laboratories, Morgan Kaufman Publishers, Inc. (1989) 379–384
9. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA (1993)
10. CMU-Repository: ftp://ftp.cs.cmu.edu/afs/cs.cmu.edu/ project/connect/bench/. WWW Repository, Carnegie Mellon University (1993)
11. Minsky, M.L., Papert, S.A.: Perceptrons: Introduction to Computational Geometry. Expanded edn. MIT Press (1988)
12. Rueda, L., Oommen, B.J.: The Foundational Theory of Optimal Bayesian Pairwise Linear Classifiers. In: Proceedings of Joint IAPR International Workshops SSPR 2000 and SPR 2000 (LNCS 1876), Springer (2000) 581–590