Salzburg Interest Group on Integrated Systems

siGis-003-1996 July 4, 1996

Lean Artificial Neural Networks
Regularization Helps Evolution

2nd Nordic Workshop on Genetic Algorithms
August 19–23, 1996, Vaasa, Finland

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

Reinhold Huber
reini@cosy.sbg.ac.at

Roland Schwaiger
rschwaig@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg

Correspondence to:

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

Lean Artificial Neural Networks
Regularization Helps Evolution

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

Reinhold Huber
reini@cosy.sbg.ac.at

Roland Schwaiger
rschwaig@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg, Jakob–Haringer–Straße 2
A–5020 Salzburg, Austria

Abstract

A main criterion for the accuracy of solutions of an Artificial Neural Network (ANN) for classification tasks is the architecture. In order to find problem-adapted topologies of ANNs, we adopted the evolutionary approach to ANN design by employing a Genetic Algorithm (GA) to evolve ANNs which are represented using a direct encoding method. As ANNs of low complexity show better generalization capabilities than more complex networks, we incorporated a regularization term into the fitness function which together with the problem representation is determining the preferred regions of the search space the GA will focus on. Especially, we investigated two different regularization terms proposed in literature and experimented with various degrees of impact on the fitness function. The parallel netGEN system which has been implemented by the authors is generating problem–adapted Feed–Forward ANNs being trained by Error–Back–Propagation. Empirical results on a real world problem taken from the PROBEN1 ANN benchmark suite are presented.

1 Introduction

ANN design is a key issue in ANN research. The architecture of the ANN has direct impact on training time, convergence and generalization ability. ANN architecture or complexity is determined by the number of neurons, number of connections, and their connectivity. It has been found that ANNs with lower complexity, i. e. small number of neurons, low connectivity, show a better generalization performance than more complex networks [RWL94] which tend to lose generalization ability, i.e. *overfitting*.

Various methods have been suggested for the automatic construction of ANN topologies. Among these are *Network Growing* (e.g. Cascade Correlation [FL90]), *Network Pruning* (e.g. OBS [HSW93], OBD [CDS90]), and *Evolutionary Design* [Yao93] of networks. Using the evolutionary design approach the topology search is governed by a fitness function. With our direct encoding scheme implemented in the netGEN system which is an extension to the representation suggested by *Miller et al.* [MTH89] the fitness is evaluated as the ANN performance on the validation set. netGEN genotypes comprise *Neuron Markers* [HMS95] which indicate the absence or presence of a maximum preset number of hidden neurons, hereby, increasing the variability of the neuron number and the space of ANN topologies. This procedure could be labeled as *Evolutionary Pruning*, since the algorithm imposes an upper bound on the complexity of the network. In order to combat overfitting we also encoded the number of training epochs with the additional benefit of another ANN parameter being set automatically.

2 Complexity Regularization and Fitness Functions

2.1 Complexity Regularization

Given the problem of relating given inputs to unknown outputs based on experience – examplified through known input-output mappings – we are interested in minimizing the overall "*error*" of our model, i.e. artificial neural network. Usually this error is given by ([RHW86]):

$$l_1: \qquad \mathcal{E}_s = \frac{1}{NP}\sum_{i=1}^{N}\sum_{j=1}^{P}|d_{ij}-o_{ij}|, \qquad l_2: \qquad \mathcal{E}_s = \frac{1}{NP}\sum_{i=1}^{N}\sum_{j=1}^{P}(d_{ij}-o_{ij})^2\,,$$

where $\mathcal{E}_s$ stands for the standard error measure and P denotes the number of input-output training examples, N the number of output units, d_{ij} the desired output, and o_{ij} the actual output for unit i of pattern j.

Other free parameters of the network such as weights (number, magnitude), biases (magnitude), units (number) are not assessed by this error measure. It has been shown that these free parameters have a direct influence on the generalization ability of neural nets [KH92]. To include these parameters in the error function we have to add another term to $\mathcal{E}_s$, i.e.

$$\mathcal{E} = \alpha_s \mathcal{E}_s + \alpha_c \mathcal{E}_c, \tag{1}$$

where $\mathcal{E}_c$ is called a *Complexity Regularization Term* [Hay94]. The constants α_s and α_c emphasize both error measures, accordingly.

The above equation can also be seen in the light of the old intuitive principle of *Occam's Razor* which according to [WRH91] was formally introduced by *Rissanen* [Ris89] and *Cheeseman* [Che90] using the information theoretic *Minimum Description Length (MDL) criterion* which reflects the trade-off between residual error and model complexity and determines the most probable model by minimizing $\mathcal{E}$ in equation (1).

The first regularization term examined was proposed by Hinton [Hin86], where C_{total} refers to all the weights in the network. The second term was suggested by Rumelhart [HP89][1]:

$$\text{Hinton:} \qquad \mathcal{E}_c = \sum_{i \in C_{total}} w_i^2, \qquad \text{Rumelhart:} \qquad \mathcal{E}_c = \sum_{i \in C_{total}} \frac{w_i^2}{1 + w_i^2}.$$

2.2 Fitness Functions

Harp & Samad [HS91] introduced a general fitness function for ANNs

$$\mathcal{F} = \sum_{k=1}^{m} \alpha_k \Phi_k \left(\mathcal{E}_k \right),$$

where m is the number of freely adjustable parameters, α_k are the weights of the various error measures of the different parameters denoted by $\mathcal{E}_k$, and the Φ_k transform the error measures. For the problem of finding an appropriate network topology for classification problems we established a fitness function which allows to minimize learning error and complexity. Hence, we take $\mathcal{E}_c$ and $\mathcal{E}_s$ and scale each of them such that they fall in the interval $[0, 1]$. Thus the *Composite Fitness Function* is the sum of a *Model Fitness* $\mathcal{F}_1$ and a *Complexity Fitness* $\mathcal{F}_2$ and looks like,

$$\mathcal{F} = \mathcal{F}_1(\mathcal{E}_s) + \mathcal{F}_2(\mathcal{E}_c),$$

$$\mathcal{F} = \alpha_1 \frac{1}{1 + \mathcal{E}_s} + \alpha_2 \frac{1}{1 + \mathcal{E}_c},$$

where

$$\alpha_1 + \alpha_2 = 1,$$

with the model fitness coefficient α_1 and the complexity fitness coefficient α_2. Note that $\mathcal{E}_s$ does not necessarily have to be derived from the actual output neuron activations, but could also be given by $\mathcal{E}_s = \frac{e_v}{n_v}$, with e_v being the number of misclassified patterns and n_v the total number of patterns in the validation set. This error measure could be of benefit, because the error measures l_1 and l_2 described above are biasing the fitness function and will partially mislead the GA when overfitting of the training data occurs, i.e. $\mathcal{E}_s$ decreases, but at the same time the generalization capabilities of the ANN become worse.

In this paper we investigate the impact of different regularization parameters and various values of α_i on the topology and the classification accuracy of the ANN on selected problems.

[1] Weigend et al. [WRH91] modified the regularization term by dividing the weights by a preset constant w_0.

3 The netGEN System

The netGEN system [HMS95] has been designed for an arbitrary amount of workstations employing the *Parallel Virtual Machine (PVM)* library [GBD$^+$94] and is currently running on a network cluster of DEC AXP workstations (9 machines) in order to train individual ANNs in parallel. It consists of three main components, the Simple Genetic Algorithm (SGA)[SGE91], the Genotype/Phenotype Mapper (GPM), and the Neural Network Manager (NNM) as shown in figure 1 .

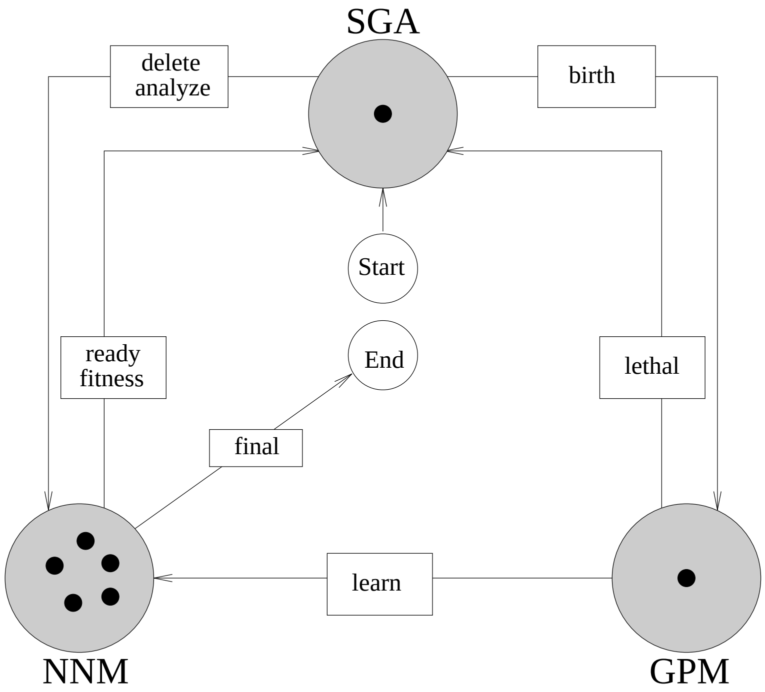

Figure 1: The netGEN system with the Simple Genetic Algorithm (SGA), the Genotype/Phenotype Mapper (GPM), and a group of Neural Network Managers (NNM).

SGA generates a genotype for each ANN which is processed by the GPM module mapping the ANN genotype to a phenotype adapted to the Neural Network Simulator [ZMV$^+$94]. The mapping module's task is to prepare valid configuration files and to dismiss ANN topologies which cannot be processed by netGEN. If the topology is valid, a command, e.g. *learn*, is passed to the NNM module. The NNM interfaces to the ANN simulator, the network is trained on a training set for the specific number of training epochs and, the fitness of each ANN is evaluated over a validation set and passed back to the SGA. With the parallel implementation a set of NNM processes runs on different machines and/or different processors.

Furthermore, we achieve *implicit load balancing* which means that more ANNs are passed to the faster machines within the PVM. A simple time–out mechanism prevents the system from deadlock when waiting for results from crashed machines. Performance reducing situations at the end of a generation due to overloaded machines are handled by *bench penalties* (i.e. the overloaded machine(s) take(s) a rest for a number of generations).

4 ANN Genotype Representation

Out of numerous existing genotype representations of an ANN [BH95], we adopted a *direct encoding* method introduced by *Miller et al.* [MTH89]. Their original encoding - in the following called *Miller–Matrix* (MM) - uses an adjacency matrix to represent the topology of an ANN. Each entry $w_{ij} \in \{0, L\}$ (0 . . . no connection, L . . . learnable connection) in the matrix at position (i, j) indicates the absence or presence of a directed connection between unit j and unit i. Further, they added a column for the biases of the units.

In our *Modified–Miller–Matrix (MMM)* encoding scheme [HMS95] we dropped the bias column and added *Neuron Markers* in the main diagonal (i,i) of the MMM, indicating the absence or presence of a particular neuron i. As a consequence, the only topology parameters which have to be set are the number of input and output neurons (depending on the problem) and the maximum number of neurons.

In order to combat overfitting [RF95], we also encoded the number of training epochs with the further benefit of another ANN parameter being set automatically.

In addition to our modifications of the MM encoding scheme, the GA does not operate on the matrix representation, but on the concatenation of the rows of the non–zero part of the MMM's lower triangle matrix representing the possible connections. All MMM entries clamped to zero need not be processed by the GA, hence, they are not encoded in the genotype.

The general form of the genotype looks like:

$$e_1 e_2 \ldots e_{n_1} m_1 m_2 \ldots m_{n_2} x_1 x_2 \ldots x_{n_3},$$

where

1. $e_i \in \{0,1\}, i = 1, \ldots, n_1$ is the binary representation of the number of epochs,

2. $m_i \in \{0,1\}, i = 1, \ldots, n_2$ are the neuron markers, where $m_i = 0$ indicates the absence and $m_i = 1$ indicates the presence of a neuron and

3. $x_i \in \{0,1\}, i = 1, \ldots, n_3$ indicate the absence/presence of a connection. The x_i represent possible connections (non–zero entries in the MMM). Reflexive connections, connections within input and output layer, and connections from unit i to unit j with $i < j$, i. e. constant zero entries in the MMM, are not encoded.

This enables the use of the standard genetic operator *2–Point–Crossover* [Gol89] which has a more global effect on the bit string representation (and therefore, the ANN topology) than the MM crossover operator exchanging rows or columns of two individuals. The mutation operator used in netGEN is the standard bit–flip with mutation probability p_m.

For the GA selection phase we chose *Binary Tournament Selection*, for it showed mostly better results than *Proportionate Selection Methods*. First analytical steps, possibly confirming this observation, have been reported recently [BT95].

4.1 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 $p_m = 0.005$, Crossover = 2–Point, Selection Method = Binary Tournament.

ANN Parameters: Network Topology = Feed–Forward, Activation Function (all neurons) = Sigmoid, Output Function (all neurons) = Identity, Training = Back–Propagation, Learning Parameter $\eta = 0.2$, Momentum Term = None, Number of Epochs = Evolutionary (max. 1000).

5 Benchmark Problem

The PROBEN1 benchmark suite of real-world ANN problems was chosen to present empirical results [Pre94]. Table 1 shows the parameters of the discussed data sets of problem *Glass* and the initial number of hidden units. This classification problem is known to be a "tough" one, as the number of patterns in the data sets is small and two of the 9 input values hardly have any statistical correlation with the result [Pre94]. The 9 input values represent the percent content of 8 different chemical elements of glass splinters and their refractive index. The 6 classes represented by the output neurons are float processed building windows, non float processed building windows, vehicle windows, containers, tableware, and head lamps. The study of classification of types of glass was motivated by criminological investigation.

Data Set Name	Input	Output	Hidden	Training	Test	Validation
Glass	9	6	10	107	54	53

Table 1: Parameters for the data sets and the network topology.

6 Discussion

Table 2 reports the parameters and results of the best nets for problem *Glass* using three different fitness functions without complexity regularization. It can be seen that the neuron markers itself lead to less complex networks with at least equal classification performance and that the fitness function (v_{acc}) adopting the classification accuracy on the validation set generates the most accurate nets. We believe that with this fitness function together with the encoding of training epochs the impact of overfitting is reduced, as the fitness measure emphasizes classification performance in contrast to modeling the validation set data (sub)–optimally.

Table 3 displays the results for fitness function l_1 with the two complexity regularization terms. While the Hinton term improves classification accuracy with very small complexity reduction, the Rumelhart term helps to find less complex nets without improving classification accuracy. It is interesting to see that there seems to be a small intervall in the α_2 values where the fitness function abruptely starts to be dominated by complexity regularization and the number of training epochs tends towards zero. The latter can be explained by the fact that the initial weights are randomly selected between -1.0 and 1.0, i.e. a net with high regularity. Back–propagation immediately shifts the weights towards irregularity, thus, in this case creating ANNs with lower fitness and driving the number of training epochs towards zero. Clearly, the setting of the complexity regularization coefficient α_2 depends on the ratio of standard fitness to complexity fitness.

Table 4 presents complexitiy regularization with fitness function l_2. Only very small changes in complexity and classification accuracy can be observed, but again the "complexity threshold" from that on nets are merely evolved according to their complexity appears.

Table 5 shows the results utilizing fitness function a_v in combination with the two regularization terms. Here the reduction of complexity is quite impressive when compared to table 2. Especially, with the Rumelhart term the number of hidden neurons drops from 7 to 1 and the number of weights from 130 to 40 (approximate values), while the accuracy on the test set remains equal for $\alpha_2 \leq 0.3$. Generally, increasing impact of complexity regularization reduces accuracy, though, the best net discovered with all evolutionary runs had $\alpha_2 = 0.7$. Surprisingly, this net with a test set accuracy of 0.7547 given by 13 misclassifications of 53 test patterns has no hidden neurons. In [Pre94] the best net is reported with a $9-8-6$ fully connected topology and an accuracy of 0.6793 (17 misclassifications of 53 test patterns). [2] The given topology has been found by a "manual" search with human experience.

When we take a closer look at the results of table 5, it seems that the drastic complexity reduction can be achieved with very small values of α_2. Therefore, we further experimented with very small values of α_2 increasing from $\alpha_2 = 0.0001$ (figure 2). Even with these small values complexity does not increase and the accuracy remains on a high level (around 0.7). This behavior can be explained with the synergy of the fitness function a_v and the tournament selection used with netGEN. Without regularization high quality nets take over in the population having exactly the same numerical fitness value (if the number of misclassifications is equal). The slightest impact of complexity regularization by very small values of α_2 immediately favors low complexity nets because of tournament selection which selects the fittest individual in the tournament without regard to the numerical fitness difference of individuals. With practical population sizes fitness proportionate selection methods would not preferably select low complexity nets.

[2] In [Pre94] a different training algorithm (*RPROP* [RB93]) and output neurons with linear activation function have been used.

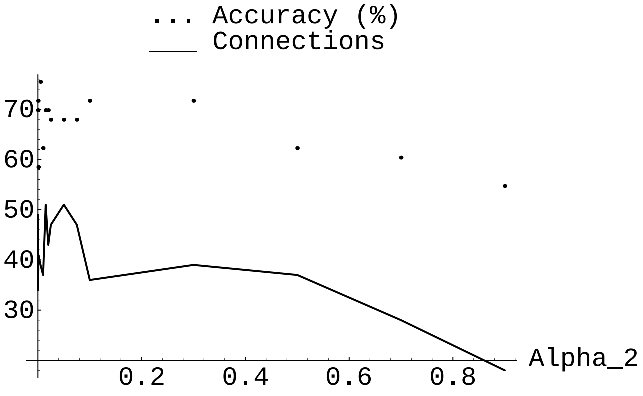

Figure 2: The dependencies of the test set classification accuracy and the number of connections on α_2 for fitness function a_v with Rumelhart regularization

Glass										
Fitness Type	α_1	α_2	$\mathcal{F}$	**Hidden Units**	**Connections**	**Weights** min	max	**Epochs**	**Accuracy** Valid. Set	**Accuracy** Test Set
No Regularization, Without Neuron Markers										
l_1	1.0	0.0	0.0260	10	214	-10.19	10.47	970	0.7222	0.6415
l_2	1.0	0.0	0.0390	10	184	-8.41	7.50	687	0.7037	0.6793
a_v	1.0	0.0	0.7593	10	137	-8.26	7.57	417	0.7593	0.7170
No Regularization, With Neuron Markers										
l_1	1.0	0.0	0.0260	9	179	-13.50	13.56	998	0.7037	0.6604
l_2	1.0	0.0	0.0394	7	117	-9.79	12.47	821	0.7222	0.6793
a_v	1.0	0.0	0.7407	7	131	-10.15	10.35	928	0.7407	0.7170

Table 2: Experimental results for problem *Glass* for three fitness functions without regularization and with/without neuron markers (α_1: Model Fitness Coefficient, α_2: Complexity Fitness Coefficient, $\mathcal{F}$: Composite Fitness Function)

7 Outlook

We have presented empirical evidence that ANNs of low complexity are able to achieve similar classification performance as compared to networks of higher complexity. Both regularization terms studied led to ANNs with fewer connections, though, the Rumelhart term has a more continuous impact on the complexity. The numerical values of the complexity regularization coefficients depend on the classification problem, the ratio of Model Fitness (ANN fitness function) to Complexity Fitness (regularization term) , the GA selection scheme, and remain to be studied in more detail. Future research will include the evolution of the α_i, thus, setting these parameters to (sub)–optimal values automatically.

8 Acknowledgements

We wish to thank the *Austrian Center for Parallel Computation* (ACPC), Group Salzburg, for making available a cluster of DEC AXP workstations where the netGEN system has been ported to. Many thanks to two anonymous referees who helped to clarify important issues with their very constructive remarks.

Glass										
Fitness Type	α_1	α_2	$\mathcal{F}$	**Hidden Units**	**Connections**	**Weights**		**Epochs**	**Accuracy** Valid. Set	**Accuracy** Test Set
						min	max			
Regularization Hinton, With Neuron Markers										
l_1	0.9	0.1	0.0239	9	175	-13.18	11.70	977	0.7037	0.7359
l_1	0.85	0.15	0.0211	9	154	-9.70	11.81	978	0.7222	0.7170
l_1	0.8	0.2	0.0208	9	166	-12.12	10.78	982	0.7222	0.6981
l_1	0.75	0.25	0.0192	9	179	-13.67	10.60	969	0.7222	0.6981
l_1	0.7	0.3	0.0310	0	18	-0.88	0.99	0	0.4259	0.5283
Regularization Rumelhart, With Neuron Markers										
l_1	0.9	0.1	0.0245	6	149	-7.16	10.05	645	0.6667	0.6604
l_1	0.85	0.15	0.0265	6	149	-7.16	10.05	645	0.6667	0.6604
l_1	0.8	0.2	0.0353	0	13	-1.56	0.97	1	0.2963	0.3585
l_1	0.75	0.25	0.0391	1	15	-0.88	1.00	0	0.2037	0.3208
l_1	0.7	0.3	0.0481	0	14	-0.88	0.97	0	0.1296	0.1887

Table 3: Experimental results for problem *Glass* for fitness function l_1 with two types of regularization and with neuron markers (α_1: Model Fitness Coefficient, α_2: Complexity Fitness Coefficient, $\mathcal{F}$: Composite Fitness Function)

Glass										
Fitness Type	α_1	α_2	$\mathcal{F}$	**Hidden Units**	**Connections**	**Weights**		**Epochs**	**Accuracy** Valid. Set	**Accuracy** Test Set
						min	max			
Regularization Hinton, With Neuron Markers										
l_2	0.9	0.1	0.0349	6	118	-11.41	8.78	794	0.7222	0.6793
l_2	0.85	0.15	0.0330	5	104	-11.37	9.32	758	0.7222	0.6793
l_2	0.8	0.2	0.0319	6	112	-11.67	8.58	799	0.7222	0.6415
l_2	0.75	0.25	0.0293	5	102	-10.56	10.11	767	0.7037	0.6038
l_2	0.7	0.3	0.0317	0	25	-1.73	0.97	4	0.3519	0.3208
Regularization Rumelhart, With Neuron Markers										
l_2	0.9	0.1	0.0366	6	130	-9.14	12.49	895	0.7222	0.6981
l_2	0.85	0.15	0.0366	6	149	-7.16	10.05	645	0.6667	0.6604
l_2	0.8	0.2	0.0426	0	19	-1.47	0.97	1	0.3519	0.4906
l_2	0.75	0.25	0.0526	0	11	-1.76	0.97	2	0.3333	0.3962
l_2	0.7	0.3	0.0492	1	15	-2.21	0.97	4	0.3519	0.3962

Table 4: Experimental results for problem *Glass* for fitness function l_2 with two types of regularization and with neuron markers (α_1: Model Fitness Coefficient, α_2: Complexity Fitness Coefficient, $\mathcal{F}$: Composite Fitness Function)

Glass										
Fitness Type	α_1	α_2	$\mathcal{F}$	**Hidden Units**	**Connections**	**Weights**		**Epochs**	**Accuracy** Valid. Set	**Accuracy** Test Set
						min	max			
Regularization Hinton, With Neuron Markers										
a_v	0.9	0.1	0.6835	3	49	-10.16	12.11	707	0.7593	0.6793
a_v	0.7	0.3	0.5193	2	41	-9.05	6.10	397	0.7407	0.5660
a_v	0.5	0.5	0.3720	3	50	-8.30	5.32	286	0.7407	0.5849
a_v	0.3	0.7	0.2239	0	38	-11.57	10.48	520	0.7407	0.7547
a_v	0.1	0.9	0.0820	2	51	-1.82	2.57	30	0.6667	0.6415
Regularization Rumelhart, With Neuron Markers										
a_v	0.9	0.1	0.6865	1	36	-11.09	13.37	741	0.7593	0.7170
a_v	0.7	0.3	0.5396	2	39	-12.68	12.31	742	0.7593	0.7170
a_v	0.5	0.5	0.3863	1	37	-6.88	7.64	346	0.7407	0.6226
a_v	0.3	0.7	0.2518	1	28	-8.76	10.85	239	0.7222	0.6038
a_v	0.1	0.9	0.1382	0	18	-3.03	0.97	28	0.6296	0.5472

Table 5: Experimental results for problem *Glass* for fitness function a_v with two types of regularization and with neuron markers (α_1: Model Fitness Coefficient, α_2: Complexity Fitness Coefficient, $\mathcal{F}$: Composite Fitness Function)

References

[BH95] Karthik Balakrishnan and Vasant Honavar. Evolutionary Design of Neural Architectures – A Preliminary Taxonomy and Guide to Literature. Technical Report CS TR #95-01, Iowa State University, Department of Computer Science, Ames, Iowa 50011–1040, U.S.A., January 1995.

[BT95] Tobias Blickle and Lothar Thiele. A Mathematical Analysis of Tournament Selection. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the 6th International Conference (ICGA95)*, San Mateo, California, 1995. Morgan Kaufman Publishers, Inc.

[CDS90] Y.L. Le Cun, J.S. Denker, and S.A. Solla. Optimal Brain Damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems, Vol. 2*, volume 2, pages 599–605. Morgan Kaufmann, 1990.

[Che90] Peter C. Cheeseman. Genetic Synthesis of Neural Network Architectures. In J. Shrager and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*, page 73. Morgan Kaufmann, 1990.

[FL90] S.E. Fahlman and C. Lebiere. The Cascade-Correlation Learning Architecture. In D. Touretzky, editor, *Advances in Neural Information Processing Systems, Vol. 2*, volume 2, pages 524–532. Morgan Kaufmann, 1990.

[GBD+94] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Laboratory, 1994.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, 1989. (ISBN 0-201-15767-5).

[Hay94] Simon Haykin. *Neural Networks. A Comprehensive Foundation.* MacMillan, 1994.

[Hin86] G.E. Hinton. Learning Distributed Representations of Concepts. *Proceeding of the Eighth Annual Conference of the Cognitive Science Society*, 1986.

[HMS95] 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.

[HP89] S.J. Hanson and L.Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D. Touretzky, editor, *Advances in Neural Information Processing Systems, Vol. 1*, volume 1, pages 177–185. Morgan Kaufmann, 1989.

[HS91] S.A. Harp and T. Samad. Genetic Synthesis of Neural Network Architectures. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 202–221. Van Nostrand Reinhold, 1991.

[HSW93] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal Brain Surgeon and General Network Pruning. In *Proceeding IEEE International Conference on Neural Networks, San Francisco*, volume 1, pages 293–299, 1993.

[KH92] A. Krogh and J.A. Hertz. A Simple Weight Decay can Improve Generalization. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems, Vol. 4*, volume 4, pages 950–957. Morgan Kaufmann, 1992.

[MTH89] 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.

[Pre94] Lutz Prechelt. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report TR 21/94, Universität Karlsruhe, Fakultät für Informatik, 76128 Karlsruhe, Germany, September 1994.

[RB93] 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.

[RF95] Paul L. Rosin and Freddy Fierens. Improving Neural Network Generalisation. In *Proceedings of IGARSS'95, Firenze, Italy*, July 1995.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representations by Error Propagation. In David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundations*, pages 318–364. MIT Press, 1986.

[Ris89] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, 1989.

[RWL94] David E. Rumelhart, Bernard Widrow, and Michael A. Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, March 1994.

[SGE91] Robert E. Smith, David E. Goldberg, and Jeff A. Earickson. SGA-C: A C–Language Implementation of a Simple Genetic Algorithm. TCGA Report 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, Department of Engineering Mechanics, Tuscaloosa, AL 35487, May 1991.

[WRH91] A.S. Weigend, D.E. Rumelhart, and B.A. Huberman. Generalization by Weight–Elimination with Application to Forecasting. In D. Touretzky, editor, *Advances in Neural Information Processing Systems, Vol. 3*, volume 3, pages 875–882. Morgan Kaufmann, 1991.

[Yao93] Xin Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.

[ZMV+94] 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.