Salzburg Interest Group on Integrated Systems

siGis-001-1996 January 16, 1996

On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks

World Congress on Neural Networks
September 15–20, 1996, San Diego, California

Reinhold Huber
reini@cosy.sbg.ac.at

Roland Schwaiger
rschwaig@cosy.sbg.ac.at

Helmut A. Mayer
helmut@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

On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks

Reinhold Huber
reini@cosy.sbg.ac.at

Roland Schwaiger
rschwaig@cosy.sbg.ac.at

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

Corresponding author: Reinhold Huber

Presenting author: Helmut Mayer

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

Telephone:
Reinhold Huber: +43-662-8044-6324
Helmut Mayer: +43-662-8044-6315
Roland Schwaiger: +43-662-8044-6339

Fax:
+43-662-8044-611

Choice of technical session:
Session 16: Evolutionary/Genetic/Annealing Algorithms

Oral presentation preferred

Audio-visual requirements:
Overhead projector

On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks

Reinhold Huber
reini@cosy.sbg.ac.at

Roland Schwaiger
rschwaig@cosy.sbg.ac.at

Helmut A. Mayer
helmut@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 Artificial Neural Networks (ANNs) 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. The role of fitness functions used by the GA is investigated, especially, the impact of a fitness function expressing both, the learning error and a toplogy-dependent regularization term, is studied. A parallel system – the 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. While networks of high complexity tend to lose generalization ability, i.e. *overfitting*, networks of too low complexity are not able to *memorize* input patterns.

Various methods have been suggested for the automatic construction of ANN topologies. Among these are *Network Growing* (OBS [HSW93], OBD [CDS90], etc.), *Network Pruning* (Cascade Correlation [FL90], etc.), and *Evolutionary Design* 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 & Todd* [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. In order to combat overfitting we also encoded the number of training epochs with the additional benefit of another ANN parameter set automatically.

2 Regularization and Fitness Functions

2.1 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.

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

where $\mathcal{E}_c$ is called a complexity or regularization term [Hay94]. The constants α_s and α_c emphasize both error measures, accordingly.

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]:

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

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 irregularity. 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 generic fitness function looks like,

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

where $\alpha_1 + \alpha_2 = 1$. In the following we analyze the impact of different regularization parameters and various values of α on the topology and the classification accuracy of the ANN.

3 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 set *cancer* and the initial number of hidden units.

Data Set Name	Input	Output	Hidden	Training	Test	Validation
cancer	9	2	9	350	174	175

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

4 Discussion

Table 2 reports the parameters and results of the best nets for problem cancer. The ANN classification accuracy remains on a good level in a very narrow range for different settings of the α_i, except for the case when choosing α_2 too large, causing an unproportional high influence of the regularization term on the

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

fitness value. As we could observe in all of our previous work, e.g. [HMS95], evolution tends to decrease the number of connections which is enhanced by the use of neuron markers. With the additional use of regularization parameters the excess weights not only assume values close to zero [Hay94], but disappear, especially, when using the Rumelhart term. Both terms implicitely have an influence on the number of connections, but the numerical value of the Hinton term is very small for a large number of weights and has little effect on the overall–fitness (with the exception of $\alpha_2 = 1.0$). The number of training epochs is close or equal to zero for fitness functions focusing on regularization. This 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.

5 Outlook

We have shown 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. Future research will include the evolution of the α_i, thus, setting these parameters to (sub)–optimal values automatically.

6 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.

References

[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.

[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.

[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.

[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.

[WRH91] A.S. Weigend, D.E. Rumelhart, and B.A. Huberman. Generalization by weight eliminationwith application to fore casting. In D. Touretzky, editor, *Advances in Neural Information Processing Systems, Vol. 3*, volume 3, pages 875–882. Morgan Kaufmann, 1991.

α_1	α_2	$\mathcal{F}$	# hu	# c	w			e	a
					$\sum \lvert w \rvert / C_{total}$	min	max		
Fitness: l_2, Regularization: *Hinton*, Neuron Markers: *ON*									
1.0	0.0	0.452242	7	69	2.098406	-7.467010	7.415960	618	0.965714
0.9	0.1	0.407160	7	69	2.098406	-7.467010	7.415960	618	0.965714
0.7	0.3	0.379237	4	48	3.101606	-8.336510	11.979530	982	0.954286
0.5	0.5	0.190832	4	40	2.994844	-8.670150	10.656380	996	0.971429
0.3	0.7	0.110343	5	62	2.425412	-7.773090	10.725960	986	0.965714
0.1	0.9	0.029707	6	79	0.860844	-2.066690	2.207090	36	0.982857
0.0	1.0	0.138841	1	11	0.448396	-0.799710	0.968830	0	0.314286
Fitness: l_2, Regularization: *Rumelhart*, Neuron Markers: *ON*									
0.9	0.1	0.480004	4	48	3.213799	-7.397520	11.831200	988	0.971429
0.7	0.3	0.231823	5	50	2.486367	-7.410350	7.983690	603	0.971429
0.5	0.5	0.225149	5	49	2.585741	-8.160010	7.622570	979	0.982857
0.3	0.7	0.113894	1	12	2.060456	-7.426330	6.725070	124	0.977143
0.1	0.9	0.192861	0	9	0.450610	-0.799710	0.968830	0	0.737143
0.0	1.0	0.116231	0	9	0.899945	-1.771740	2.292100	2	0.971429
Fitness: l_2, Regularization: *Hinton*, Neuron Markers: *OFF*									
1.0	0.0	0.319948	9	104	1.647463	-4.434680	6.726260	557	0.965714
0.9	0.1	0.311811	9	78	2.334906	-8.610520	8.006980	849	0.965714
0.7	0.3	0.210277	9	90	1.631857	-7.009340	6.553670	631	0.971429
0.5	0.5	0.313690	9	77	1.959996	-8.429780	7.972270	817	0.965714
0.3	0.7	0.197922	9	78	2.232136	-9.767600	7.749640	913	0.971429
0.1	0.9	0.029473	9	112	0.725830	-2.863760	2.298470	28	0.977143
0.0	1.0	0.052623	6	43	0.465714	-0.965500	0.994490	0	0.314286
Fitness: l_2, Regularization: *Rumelhart*, Neuron Markers: *OFF*									
0.9	0.1	0.342882	9	98	1.870474	-7.272400	5.105050	752	0.971429
0.7	0.3	0.386372	9	77	2.424402	-9.141730	8.428380	881	0.971429
0.5	0.5	0.176040	8	53	2.345912	-7.458370	9.844840	786	0.971429
0.3	0.7	0.104538	3	20	2.083025	-7.071000	8.302060	213	0.971429
0.1	0.9	0.162190	2	11	0.465084	-0.799710	0.968830	0	0.737143
0.0	1.0	0.178878	2	11	0.465084	-0.799710	0.968830	0	0.308571
Fitness: l_1, Regularization: *Hinton*, Neuron Markers: *ON*									
1.0	0.0	0.163961	6	64	2.685373	-6.961870	6.873810	936	0.971429
0.9	0.1	0.158118	6	64	2.630755	-7.530180	7.480870	778	0.971429
0.7	0.3	0.124712	6	68	2.028423	-7.602490	7.708420	807	0.971429
0.5	0.5	0.090339	6	65	2.743030	-8.899630	8.555040	869	0.971429
0.3	0.7	0.052502	5	53	2.571612	-6.559660	7.824400	954	0.982857
0.1	0.9	0.132823	0	10	0.448829	-0.799710	0.968830	0	0.302857
0.0	1.0	0.138841	1	11	0.448396	-0.799710	0.968830	0	0.314286
Fitness: l_1, Regularization: *Rumelhart*, Neuron Markers: *ON*									
0.9	0.1	0.145022	6	59	2.701894	-8.678320	7.388910	924	0.971429
0.7	0.3	0.105509	3	29	2.651527	-9.012010	9.840510	940	0.977143
0.5	0.5	0.101549	4	39	3.624948	-10.193230	9.131520	956	0.977143
0.3	0.7	0.070495	0	10	3.340573	-11.285990	9.609410	994	0.965714
0.1	0.9	0.192263	0	9	0.450610	-0.799710	0.968830	0	0.737143
0.0	1.0	0.116231	0	9	0.899945	-1.771740	2.292100	2	0.971429
Fitness: l_1, Regularization: *Hinton*, Neuron Markers: *OFF*									
1.0	0.0	0.267309	8	67	2.886354	-8.576090	7.346700	964	0.965714
0.9	0.1	0.240668	8	67	2.886354	-8.576090	7.346700	964	0.965714
0.7	0.3	0.140246	9	89	2.676674	-7.983270	6.924730	962	0.971429
0.5	0.5	0.076618	8	77	2.041707	-5.999670	5.604530	791	0.971429
0.3	0.7	0.050617	9	85	1.790945	-7.150770	5.543870	800	0.977143
0.1	0.9	0.061904	4	34	0.449992	-0.965500	0.994490	0	0.114286
0.0	1.0	0.052623	6	43	0.465714	-0.965500	0.994490	0	0.314286
Fitness: l_1, Regularization: *Rumelhart*, Neuron Markers: *OFF*									
0.9	0.1	0.173717	8	78	2.137446	-10.383490	6.402010	932	0.971429
0.7	0.3	0.109616	7	68	1.982141	-7.081490	5.086890	808	0.977143
0.5	0.5	0.104886	6	59	2.383264	-8.410130	6.963340	811	0.971429
0.3	0.7	0.059250	3	17	2.884702	-9.574580	8.505390	787	0.971429
0.1	0.9	0.161566	2	11	0.465084	-0.799710	0.968830	0	0.308571
0.0	1.0	0.178878	2	11	0.465084	-0.799710	0.968830	0	0.308571

Table 2: The abbreviations stand for: (α_i: $i = 1, 2$) = weights for the fitness function $\mathcal{F}$; **hu** = hidden units; **c** = connections; **w** = weights; **e** = epochs, evolutionary determined; **a** = accuracy (validation set).