Representation and Evolution of Neural Networks

Martin Mandischer
Department of Computer Science VI
University of Dortmund – Germany
e-mail: mam@grete.informatik.uni-dortmund.de

Abstract

An evolutionary approach for developing improved neural network architectures is presented. It is shown that it is possible to use genetic algorithms for the construction of backpropagation networks for real world tasks. Therefore a network representation is developed with certain properties. Results with various application are presented.

1 Introduction

The performance of neural networks highly depends on the architecture of the networks and their parameters. Therefore, determine the architecture of a network (size, structure, connectivity) greatly affects the performance criteria, i.e. learning speed, accuracy of learning, noise resistance and generalization ability. There are only very few mathematical methods to determine the architecture and parameters of networks. Design decisions are normally achieved by rules of thumb and the common approach for designing neural networks is to build a network, test it for the desired function, change structure and parameters if it does not work and repeat this until your evaluation criteria are more or less fulfilled. From a computational and developers viewpoint, this procedure is very expensive and gives no reliable insight into the architecture/performance relation. Recent work of Judd and Lin/Vitter shows that learning in general, as well as choosing an optimal network topology, are NP-complete problems [4, 5]. They also have shown that placing constraints on the topology can help to make learning tractable. This motivates the usage of some evolutionary approach as a heuristic for finding optimal architectures and a network representation which places useful constraints on the architecture.

Several approaches have been made towards the evolutionary design of neural networks [12]. A straight-forward method to determine the architecture of a network has been used Miller et al. The network structure was mapped onto a binary connection matrix where each cell of the matrix determines whether a connection between two units exists or not. A set of matrices were created randomly and the corresponding networks trained. The networks which performed best on the given task, are allowed to produce offsprings for the next generation. New offsprings are created from one or two parent networks by combining or manipulating their connection-matrices. After a fixed number of cycles the best network is chosen [7]. The main problem with this approach is the matrix representation of the network. Incorrect network structure, i.e. feedback connections, and very long codes for larger networks are the result.

A more sophisticated approach has been presented by Harp et al. [3]. They used 'blueprints' for the networks where each network is described by several parameters like, number of layers, layer size, and connections between layers. With this representation it was possible to place constraints on the networks architecture and to reduce the number of incorrect networks. Much less incorrect networks had to be trained an evaluated and the number of evolution cycles needed to get good networks is reduced. Another approach which avoids the representation problem is described by Schiffman et al. To produce the offsprings the best networks are selected and some connections are added or removed from networks and the worst networks are replaced by the offsprings of the best [10].

The next section introduces Genetic Algorithms as a method to search for better networks. Section 3 focuses on a network representation that fulfills some useful properties and the operators to create new networks.

2 Genetic Neural Networks

The basis for evolution is a set of individuals (networks) which establishs a population. The better an individual adapts itself to the given environment, the greater is its chance to survive and produce offsprings. Each individual can be seen as a state in the search space which is, in our case, the space of possible feedforward networks. Each individual is assigned an fitness value which reflects its ability to adapt to the given environment. A genetic algorithms is an iterative procedure, where each iteration is called generation [2]. During each iteration the two natural inspired principles of *selection* and *reproduction* are applied to the population. The selection mechanism determines which individuals are allowed to produce offsprings for the next generation. The probability at which an individual is allowed to reproduce itself and the number of offsprings is based upon its fitness. Supervised learning in networks directly provides a measure of fitness. Therefore we restrict ourselves to the evolution of backpropagation networks. The evolution cycle can now be characterized as follows.

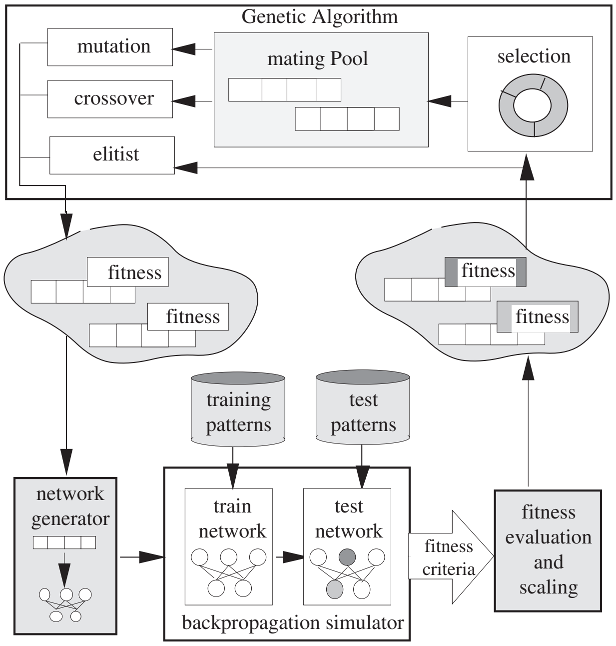

Figure 1: evolution of networks

The use of GA implies a genotypic representation of the individuals. This allows the genetic operators to modify them without using knowledge about the individuals structure. Starting from a randomly generated initial population of genotypic networks a network-generator builds the networks from the genotyp. The backpropagation networks can be interpreted by the network simulator. In the simulator, all networks are trained on the same training data (environment) and their performance on the test-data is evaluated. The simulator delivers several quality criteria (mean-square-error, training time, size of the network) to a device for fitness evaluation and scaling. This device calculates the actual fitness value for each network according to its performance and relatively to the fitness of all other networks. To prohibit that networks with very high fitness values dominate the population in the first generations and to compensate low fitness diversity at the end of the search, we used the sigma-scaling method [9] to rescale the fitness of all networks. The fitness is calculated by several weighted factors:

$$\text{fitness} := \begin{array}{c} w_1 \times error + w_2 \times connections \\ + \\ w_3 \times units + w_4 \times \quad epochs \end{array}$$

With the fitness every network has its expected number of offsprings assigned. The roulette wheel selection method [2] chooses networks with a probability according to their relative fitness, until the population size is reached. The chosen networks are put in the mating pool and allowed to reproduce themselves. Reproduction is achieved by several genetic operators like mutation and crossover which ensure that offsprings are new but somehow related to their parent networks. To prevent the best network from being manipulated by the operators it is automatically put into the new generation. This evolution cycle is iterated for a fixed number of times. During evolution the actual population, best networks and some statistical measures to evaluate the process are collected and saved.

3 Networkrepresentation and Genetic Operators

Weiß distinguished between low- and high-level representations [12]. A *low-level representation* specifies exactly each network parameter (each connection, unit etc.). This causes a large search space for the GA, so that the number of iterations increases dramatically. In contrast a *high-level representation* is an abstract description of the networks where connection patterns and parameter to determine the number and distribution of units characterize the networks structure. The search space is much smaller and networks can be prestructured with the restriction that some architectures are excluded.

The works of Harp et al. and Miller et al. presented representations of both types which are not able to insure the correctness of coded networks [3, 7]. Our representation of backpropagation networks allows only correct networks which prevents incorrect networks from participating in the evolution cycle and saves useless training. Therefore the population size and the number of iterations can be reduced to a number that makes the problem tractable. Excluding incorrect networks and prestructuring gives some evidence that first generation networks can solve the given problem and genetic search can focus on finding improved networks. Additionally training time of the networks can significantly be reduced if layers are sparsely connected rather than fully. The following representation allows only correct networks and the genetic operators do not produce incorrect networks.

The network is specified by describing its main components. A *network-parameter-area* specifies learning rate and momentum term for all connections. Each of the layers has its own layer-area specifying the number of unit in the layer and the connections to other layers. To insure input and output for every unit and therefore correct networks we distinguish projective and receptive connections.

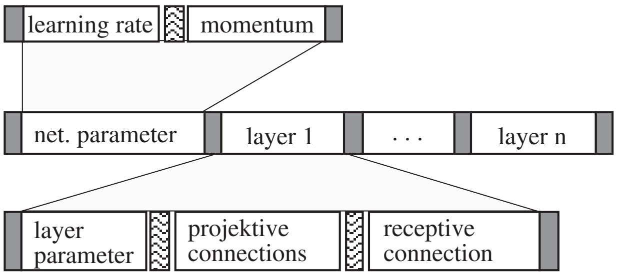

Figure 2: network representation

The receptive connections connect the actual layer with one or more of the preceding layers. Each units receives connections from at least one unit in a precedent layer. The projective connections are used to insure each unit an output and are therefore connected to the next layer. The structure of the network can evolve freely because of the variable number of connection areas.

The connection pattern between two layers is motivated by neurophysiological receptive fields and determined by two parameters. A *radius parameter* specifies the connection radius for each unit of the layer. The size of the radius is given in percent relative to the size of the destination layer. To determine the beginning of a of receptive field an offset relative to the position of the actual unit is added. The second parameter determines the density of the connections within the receptive field and the percentage of units which are connected to an actual unit.

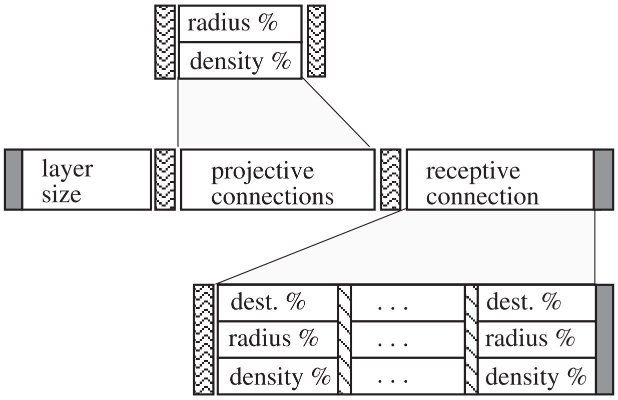

Figure 3: layer representation

Addressing *destination layers* is difficult because of the variable number of layers and connection areas. As a solution relative addresses are used. The projective connections are always related to the following layer, whereas the receptive connections are related to one or more preceding layers. A destination parameter gives the percentage of layers which will be skipped before connections to a preceding layer are established.

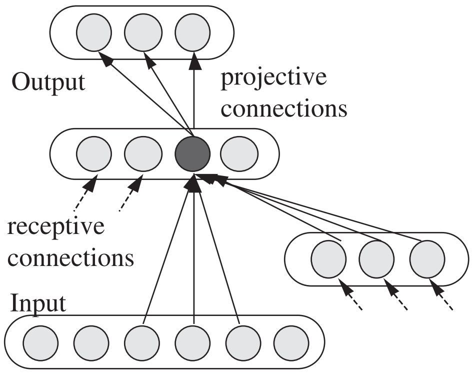

Figure 4: type of connections

The figure illustrates the principles of projective and receptive connections.

This network representation requires a modification of the classical genetic operators. Starting with the idea of mutation should introduce and maintain genetic diversity and crossover should inherit features during reproduction, operators are developed which work on this high level representation and obey the underlying ideas.

Three operators are used: relative mutation, one- and two-point crossover. The *mutation-operator* is applied to those networks in the mating pool which are chosen with the probability given by the mutation-rate ($0 \leq P_m \leq 1$). After a network has been selected for mutation, one of the following parameters in the representation are either increased or decreased: learning-rate, momentum-term, size of one randomly selected layer, radius or density of a randomly selected connection-area. Each of the parameters has an upper bound, a lower bound and a maximum change-rate to prevent changes from being to radical. Learning rate and momentum-term are allowed to change at most 50% from their total range ($0 \leq lr \leq 16$), mutation of all other parameters can vary within 10% of the total range.

The *one-point-crossover* operator exchanges layers between two networks. Networks are again chosen from the mating pool with a probability given by a crossover-rate ($0 \leq P_{c1} \leq 1$). If the representation is viewed a linear ordering of layers, a randomly chosen number (0.100) specifies the crossing-point at which layers are exchanged. The *two-point-crossover* operator exchanges layers between two randomly chosen crossing-points p_1, p_2. Networks are again chosen from the mating pool with a probability given by the crossover-rate ($0 \leq P_{c2} \leq 1$). The following figure illustrates crossover.

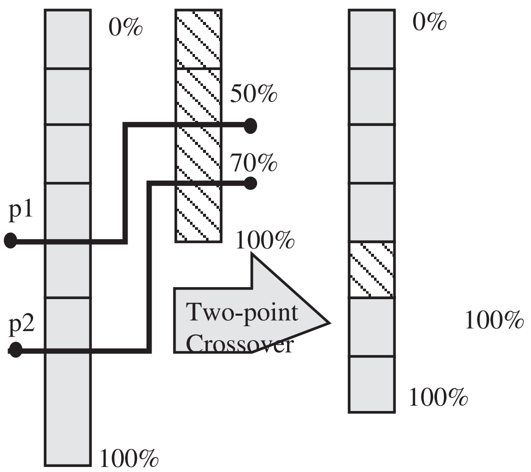

Figure 5: two-point crossover

Due to the relative addressing of destination layers and the usage of percentual parameters instead of fixed values, layers can be exchanged between networks without loosing correctness. It can been shown that the network representation is closed under the given operators and therefore only correct networks participate in the evolution [5].

4 Results

To evaluate our approach we selected four applications for backpropagation networks. The first task is to discriminate edges and corners. Training data consisted of 70 patterns, where each pattern is given as 9x9 binary matrix, which displays either an edge or a corner. A more difficult task is to discriminate 490 handwritten digits from 6 different writers which are given by 8x11 matrices each showing one of 10 possible digits. The third application consists of 279 vectors with 112 real-valued components which encode control-knowledge to improve proofs in PROLOG [11]. It has been proven [13] that feedforward networks are capable of arbitrary exact function approximation. Despite this theoretical result, Fox showed that it is difficult for a great number of architectures to approximate a Mexican hat function [1]. Therefore we used 841 vectors with x, y, z coordinates building a 3-dimensional Mexican hat function. We developed some evaluation criteria for the

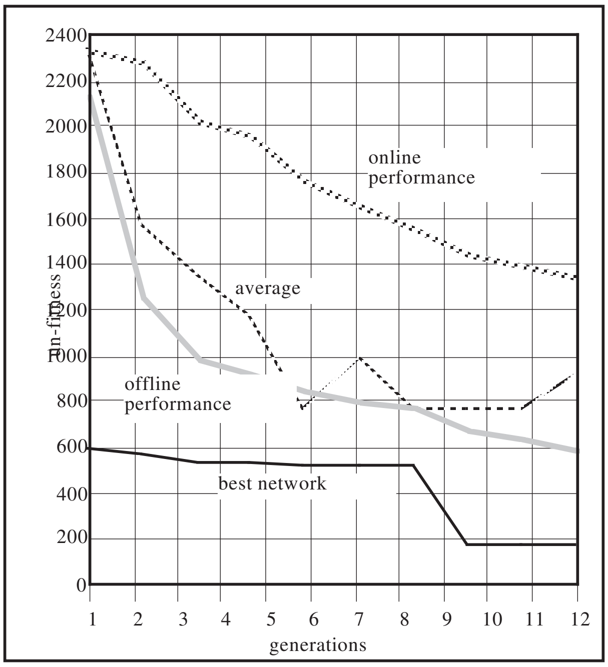

Figure 6: evolution process for function approx.

networks and the evolution process itself. The functionality of our approach is evaluated by its capability to evolve networks which solve a given task and its reliability to reproduce this results with a high probability. A second criteria measures the quality of the evolved networks by evaluating its performance on the given task and a comparison with standard architectures on the same tasks. For all tasks the GA discovered architectures which are capable of solving the given task better or at least as well as all other architectures.

Figure 6 gives an example of the evolutions progress. A population of 10 networks evolves for 12 generations and each network is trained for 500 epochs. The online-performance is the averaged fitness over all generation and the offline-performance averages the best fitness values of all generations.

At the end of the evolution process stands a network with a performance much better than that of comparable standard networks. By standard networks we mean a fully connected input-hidden-output architecture.

For the other applications a population size between 10 and 50 networks and 10 to 100 generations has been typical and yields similar results. To show the reliability of our approach we repeated one experiment 50 times and gained a statistical certainty of 99% that other experiments result in a best network which is within a 5% range from the average best network.

After the successful evolution process we compared the best evolved network for each task to a standard network with the same number of units and a standard network with the same number of connections and, as far as known, results from other publications. The following charts summerizes our results on the applications.

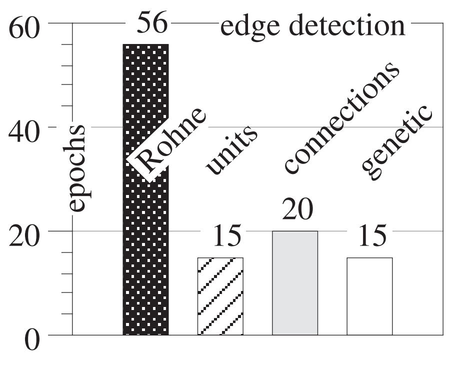

Figure 7: results from edge detection

On the edge detection task the genetic network with 81-45-2 units and 821 connections performed better than a standard network 81-10-2 with 830 connections and as good as a 81-45-2 network with 3735 connections. The classification capability is 100% for all networks. A network (88-52-10) for digit dis-

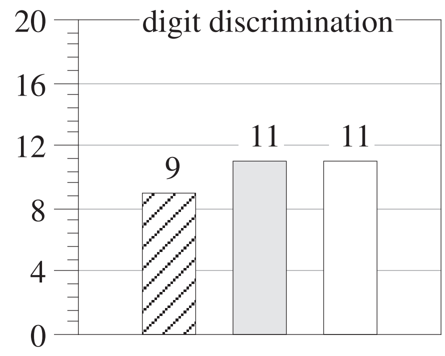

Figure 8: results from digit discrimination

crimination with 2671 connections performed as good as the standard network with the same number of connections and slightly worse than a network with twice as much connections. Classification capability is more than 97% for all networks. Results in learn-

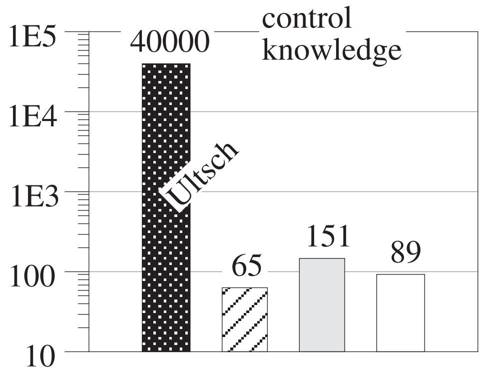

Figure 9: results from proof optimization

ing control-knowledge are comparable to the digit task. A network (112-36-7) with 2103 connections performed better than the standard network with the same number of connections and slightly worse than a network (112-36-7) with twice as much connections. Compared to Ultsch et al. [11] where less than 50% of the control knowledge has been learned, our networks achieved more then 90%. The most astonishing result we gained from the function approximation. We varied the number of hidden units in the standard net-

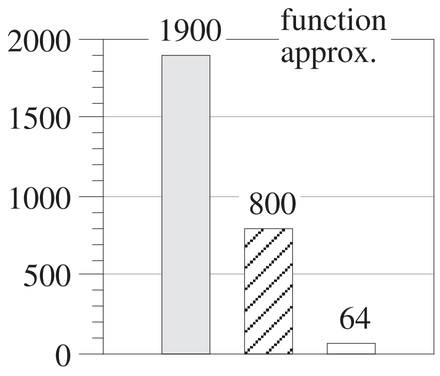

Figure 10: results from function approximation application

works between 40 and 200 but never get below 800 epochs. The genetic network (2-97-1) with 590 connections performed more than 12 times better than the standard network (2-200-1) with 600 connections.

The above results give strong evidence that the underlying structure is essential for the performance of the network. Backpropagation networks can efficiently be developed by GAs and perform better than standard networks. All experiments have been run on a SPARC-station within a couple of hours up to two days.

5 Conclusion

The successful application of GAs to determine the architecture of neural networks shows that an automated design of networks is possible. The resulting networks proved to be better than standard architectures. The best evolutionary networks for all four application solved the tasks correct. Nearly every genetic network showed its superiority over networks with a comparable number of connection, units and learning epochs.

Applied to the function approximation task the genetic network needed 12 times less learning time and showed better approximation quality than standard networks [6]. These results give strong evidence that the underlying structure is essential for the performance of the network. An emerging question in this context is how the structure influences the learning process and how we can gain knowledge about this structure/performance relation from superior genetic neural networks.

References

[1] D. Fox, V. Heinze, K. Möller, S. Thrun, and G. Veenker. Learning by error-driven decomposition. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pages 207–212, Amsterdam, June 1991. North–Holland.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[3] S. A. Harp, T. Samad, and A. Guha. Designing application-specific neural networks using the genetic algorithm. In D. S. Touretzky, editor, *Proceedings of IEEE conference on Neural Information Processing Systems*, volume 2, pages 447–454, San Mateo, 1990. (Denver 1989), Morgan Kaufmann.

[4] J. S. Judd. *Neural Network Design and the Complexity of Learning.* MIT Press - Bradford Book, Cambridge, MA, 1990.

[5] H. J. Lin and J. S. Vitter. Complexity issues in learning by neural networks. Technical Report TR-CS-90-01, Department of Computer Science, Brown University, Providence, RI, 1990.

[6] M. Mandischer. Genetische Algorithmen zur Optimierung Konnektionistischer Modelle. Master's thesis, Department of Computer Science, University of Dortmund, Dortmund, PO BOX 500500, February 1992.

[7] G. F. Miller, P. M. Todd, and S. U. Hegde. Designing neural networks using genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo, 1989. (Arlington 1989), Morgan Kaufmann.

[8] T. Rohne. Künstliche neuronale netze zur eckendetektion in digitalisierten grauwertbildern. Master's thesis, University of Dortmund, Department of Computer Science X, Dortmund, FRG, 1991.

[9] N. Schaudolph and R. K. Belew. Dynamic parameter encoding for genetic algorithms. Technical Report CSE-TR#CS90-175, University of California, San Diego, CA, 1990.

[10] W. Schiffmann, J. Merten, and W. Randolf. Performance evaluation of evolutionary created neural network topologies. In *First International*

Workshop on Parallel Problem Solving from Nature, pages A–III:1–11, Dortmund, FRG, 1990. University of Dortmund, Department of Computer Science X.

[11] A. Ultsch, R. Hannuschka, U. Hartmann, M. Mandischer, and V. Weber. Optimizing symbolic proofs with connectionist models. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pages 585–590, Amsterdam, June 1991. North–Holland.

[12] G. Weiß. Combining neural and evolutionary learning: Aspects and approaches. Technical Report FKI-132-90, Technische Universität München, 1990.

[13] H. White and K. Hornik. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *Proceedings of the International Joint Conference on Neural Networks*. (San Diego, CA), IEEE, 1989.