COEVOLUTION AS AN AUTONOMOUS LEARNING STRATEGY FOR NEUROMODULES

ULF DIECKMANN*

Neurodynamics Group, ATÖ,
Research Center Jülich,
D-52425 Jülich, Germany

We present a framework for the evolution-aided design of neural networks. Our approach combines traditional weight dynamics with a dynamic of the networks' architecture. This allows for the automatic construction of network structures that solve particular problems. We illustrate our method by providing evolutionary solutions to parity problems.

1. The Problem of Network Architecture

The training of artificial neural networks to solve a particular problem usually affects only the weight space of the network. In contrast, the network architecture has to be predefined by the user. For this pretraining choice of network architecture profound skill and experience may be required on the part of the user — an improperly predefined architecture easily renders a learning problem insoluble. Relevant parameters of a network architecture include the choice of feedforward or recurrent structure, of network size and connectivity level, and of the number of layers. The appropriateness of such choices can only be determined posttraining.

Consequently, methods have been devised either for determining upper bounds for network sizes (e.g. the application of Kolmogorov's theorem to 3-layered network structures by Hecht-Nielsen[1,2]) or for automatically pruning neurons from an initial structure in order to adjust network size (e.g. Bartlett's dynamic node architecture learning[3]). However, these methods are restricted to a particular subclass of network architectures involving only feedforward connections.

Here we suggest a general approach allowing for a joint adaptation of neural networks with respect both to weight dynamics and architecture dynamics. All the relevant parameters of network architecture mentioned above are subject to the learning process and thus are adjusted to the specific problem considered. This applies in particular to the crucial alternative of feedforward or recurrent network structure, in fact, the proper choice between these paradigms is automatically made.

*e-mail: u.dieckmann@kfa-juelich.de

2. Evolution of Artificial Neural Networks

Classical learning strategies for neural networks are only capable of altering the weights of connections in a network. But which methods are appropriate to alter dynamically the network architecture with respect to a specific problem? We suggest that this problem of dynamical network architecture is best tackeled in an evolutionary framework[4].

For that purpose we consider a population $p(t)$ of $n(t)$ artificial neural networks undergoing a variation-evaluation-selection loop. This population initially contains networks comprising only the input and output neurons characteristic of the considered problem; in these initial architectures no hidden neurons and no connections are present. The variation part of the variation-evaluation-selection loop then allows for the insertion and deletion of neurons and connections. After evaluating the performance of the networks in the population with respect to the problem considered, a new population is generated from the old population. The number of network copies passed from the old to the new population is dependent on the network's performance — in consequence of this selection process the average performance of the population will either stay the same or increase. After repeated passes through the variation-evaluation-selection loop networks have build up in the population that solve the considered problem.

We now outline in turn the three major steps of the variation-evaluation-selection loop.

1. *Variation*
   The variation operators include insertion and deletion of neurons, insertion and deletion of connections, and alterations of bias and weight terms. The associated operators acting on the ith member of the population p are denoted by N_i^+, N_i^-, C_i^+, C_i^-, N_i^* and C_i^* respectively. A variation pass is then described by

$$V: \quad p(t) \mapsto \prod_{i=1}^{n(t)} C_i^* C_i^+ C_i^- N_i^* N_i^+ N_i^- \; p(t) \quad .$$

   The operators $N_i^{*,+,-}$, $C_i^{*,+,-}$ are of stochastic character. The chance that they will execute their respective action is determined by fixed per-neuron and per-connection probabilities. In a more complex version the variation operator V may also induce the exchange of entire subnetworks between members of the population p.

2. *Evaluation*
   The evaluation operator

$$E: \quad p(t) \mapsto (p(t), e(t))$$

   is problem-specific. In its simplest form the performances $e_i(t)$ of the $n(t)$ networks in the population $p(t)$ are mutually independent. As an example,

the performance of each member of the population could be based on the average error of the actual network output relative to the requested output: $e_i(t) = -\frac{1}{r}\sum_{j=1}^{r} |o_i(t) - o_i|$. Here r denotes the number of input patterns, $o_i(t)$ the resulting output of the network and o_i the requested output. Like in this case the evaluation operator usually will be deterministic; more sophisticated versions can also account for network size and past performance. Furthermore, interactions between members of the population can be defined via an artificial sensomotoric loop opening up the potential for coevolutionary dynamics[5].

3. *Selection*
   Differential survival of the varied members of the population is defined by the selection operator. Possible definitions range from (a) probabilistic survival according to evaluation results to (b) winner-takes-all selection. The selection operator is given by

$$S: \quad (p(t), e(t)) \mapsto \prod_{i=1}^{n(t)} R_i^{\nu(e_i(t))}\, p(t) \quad .$$

   Here R_i is the reproduction operator for the ith network in p. Consequently, $\nu(e_i(t))$ copies of each such network are passed to the new population. In case (a) these integer numbers ν are stochastic variables drawn e.g. from Poisson distributions with mean values larger than 1 if $e_i(t) > \sum_{i=1}^{n(t)} e_i(t)/n(t)$ and mean values smaller than 1 for the other networks in the population p. Case (b) is defined by $\nu(e_i(t)) = n(t)$ if $e_i(t) = \max_i e_i(t)$ and $\nu(e_i(t)) = 0$ otherwise.

The evolution of the population p is then generated by repeated application of the mapping

$$p(t) \mapsto SEV\ p(t)$$

on the initial population p(0).

3. Learning the Parity Problem

We illustrate our method by presenting evolutionary solutions to the two parity-4 problems. Based on the activity of n input neurons, the even-(odd-)parity-n problem amounts to activating a single output neuron if and only if the number of active inputs is even (odd). We employ parity-n as a benchmark for our approach since the parity functions are the hardest binary functions to learn (via random search, as demonstrated e.g. by Koza[6]).

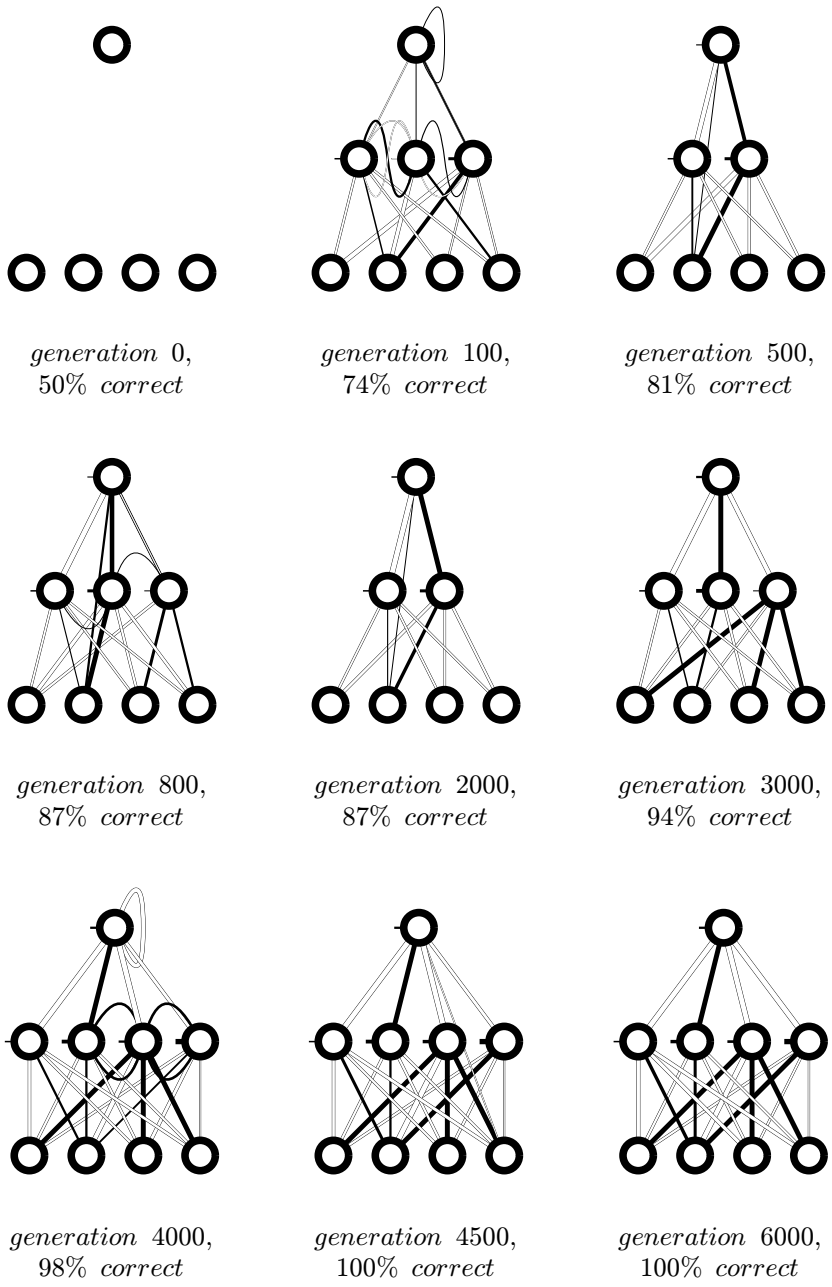

Fig. 1. Evolution of the best-performing network for the odd-parity-4 problem. Neurons are depicted by circles, connections by curves. Excitatory connections are shown in white, inhibitory connections in black, the thickness of curves measures connection strength. Horizontal bars at neurons indicate thresholds.

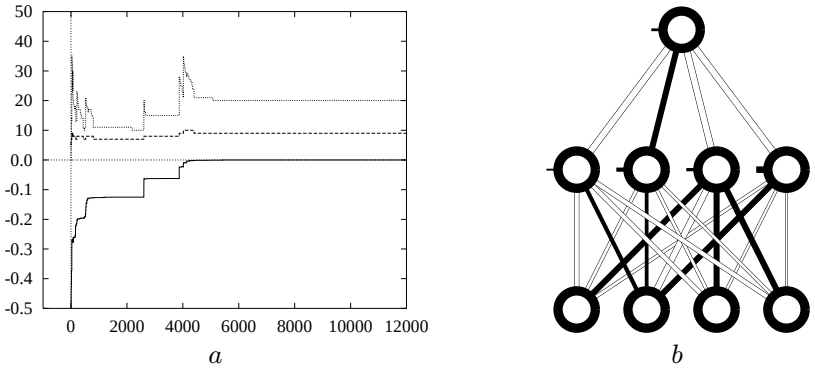

Fig. 2. (a) Time-evolution of network performance (continuous curve), network size (broken curve) and number of connections (dotted curve) of the best-performing network for the odd-parity-4 problem. (b) Evolutionary solution to the odd-parity-4 problem.

In Figure 1 we show the evolution of networks for the odd-parity-4 problem. The pictures indicate for several generations structure and performance of the best-performing networks of the population. An increase in the number of hidden neurons and connections typically is followed by a pruning phase, in which the network structures exhibit a reduction of complexity while conserving performance. This is demonstrated in Figure 2a, where the time-evolution of network performance, network size and number of connections is given. The resulting network structure is depicted again in Figure 2b.

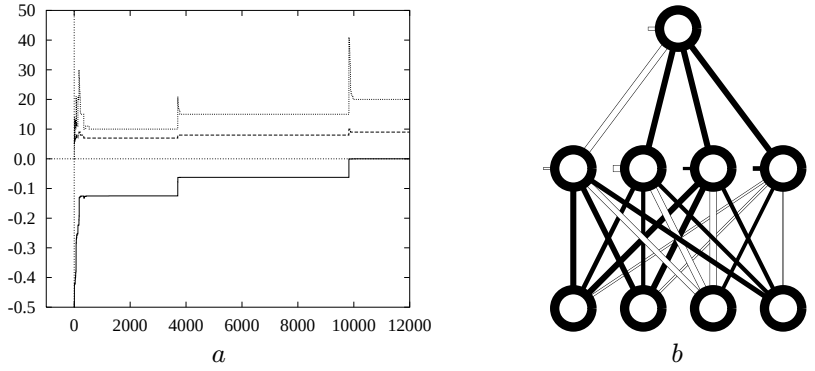

Fig. 3. (a) Time-evolution of network performance (continuous curve), network size (broken curve) and number of connections (dotted curve) of the best-performing network for the even-parity-4 problem. (b) Evolutionary solution to the even-parity-4 problem.

Figure 3a and 3b give the same information as to the evolution of networks for the even-parity-4 problem. Notice that the time required to make the error of the network essentially vanish is longer than in Figure 2. This result underpinns an observation made by Koza[6] that even-parity problems are slightly harder to learn than odd-parity problems.

4. Conclusions

We have presented a framework for handling dynamical network architectures. Our method is based on evolutionary mechanisms and is capable of automatically constructing artificial neural networks that solve specific problems. In particular, we have shown in this paper successful applications of our method to the two parity-4 problems.

In contrast to classical genetic algorithms[7,8] the intricate question of how to represent the neural networks in terms of single, fixed-length strings of symbols does not arise in our approach. Since our represention of the networks in the population is based on variable-length lists of neurons and connections, there is no artificial limit to the complexity of the evolving networks.

Two further advantages of an evolutionary approach to network learning should by mentioned. First, evolutionary methods provide a natural answer to the credit-assignment problem[9] as infrequent or low-quality evaluative feedbacks are no obstacle in applying this method. Second, traditional learning methods are only capable of training feedworward networks; in contrast, evolutionary algorithms can be applied both to feedforward and recurrent networks.

Finally we contend that — regarding the largely unsolved problem of finding a suitable network architecture for a given problem — a shift of emphasis from weight dynamics towards architecture dynamics is inevitable. For this purpose we suggest as a powerful and flexible tool the evolution-aided design of neural networks.

Acknowledgements

I would like to thank Frank Pasemann for many interesting and fruitful discussions on modular neurodynamics.

References

1. R. Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem", *IJCNN International Conference on Neural Networks*, **1**, 11–14, IEEE, New York, 1987
2. R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, Reading Massachusetts, 1990
3. E.B. Bartlett, "Dynamic Node Architecture Learning: An Information Theoretic Approach", *Neural Networks*, **7**, 129–140, 1994
4. W.M. Spears, K.A. De Jong, T. Bäck, D.B. Fogel, H. de Garis, "An Overview of Evolutionary Computation", in: P.B. Brazdil (ed.), *Proc. European Conference on Machine Learning*, pp. 442–459, Springer, New York, 1993
5. D.J. Futuyma and M. Slatkin (eds.), *Coevolution*, Sinauer Associates, Sunderland Massachusetts, 1983
6. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, 1992
7. J.H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, 1975
8. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading Massachusetts, 1989
9. A.G. Barto, R.S. Sutton, C.W. Anderson. "Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems", *IEEE Transactions on Systems, Man, and Cybernetics*, **13**, 834–846, 1983