Efficiency Aspects of Neural Network Architecture Evolution Using Direct and Indirect Encoding

H. Kwasnicka, M. Paradowski
Department of Computer Science, Wroclaw University of Technology, Poland
E-mail: {halina.kwasnicka, mariusz.paradowski}@pwr.wroc.pl

Abstract

Using a GA as a NN designing tool deals with many aspects. We must decide, among others, about: coding schema, evaluation function, genetic operators, genetic parameters, etc. This paper focuses on an efficiency of NN architecture evolution. We use two main approaches for neural network representation in the form of chromosomes: direct and indirect encoding. Presented research is a part of our wider study of this problem [1, 2]. We present the influence of coding schemata on the possibilities of evolving optimal neural network.

1 Introduction

Genetic Algorithms (GAs) [3] and artificial Neural Networks (NNs) [4] were proposed as the imitation of natural process: biological evolution and real neural systems respectively. Designing architecture of networks causes some problems: How many neurons should be in the network? How they should be organized? How many connections we should establish and between which neurons? A designer of a NN architecture must relay on his experience or on informal heuristics [5].

Designing a NN is a problem of searching for an architecture, which performs some specific task with desired accuracy. This process can be seen as searching the surface defined by networks performance evaluation above the space of all possible neural networks architectures [6, 7, 8, 9, 10, 11]. The coding schema limits a search space for a GA. The smaller the search space is, the easier is to find the optimal, or near optimal network architecture. However, the smaller search space means the small number of different potential NN architectures.

A fitness function, used for evaluation of generated networks, strongly influences obtained results [2]. We developed a fitness function, which try to balance the size and an accuracy of an evaluated network.

2 Neural network representations

There are two major approaches of neural network architecture evolution: *direct encoding* [12] and *indirect encoding* [12, 13]. The main difference lies on a size of a set of representative networks (a size of a search space for a GA).

2.1 Direct encoding

Direct encoding represents neural networks as directional graphs. A very large subset of neural networks can be represented using this method. Direct encoding focuses on the ability to produce neural networks with the highest possible quality. The ability of representing a very large subset of neural networks and very few limitations on the size of the domain causes the time and memory requirements of this method to be very large. A network with n neurons is represented by a matrix and a vector. Matrix E is used to encode information if there are connections between neurons. For feed-forward neural networks only a half of the matrix is used. Vector V is used to encode information if neurons are active. This causes vector V to be a mask for matrix E. Both, vector and matrix encode input layer, hidden layer and output layer neurons and connections between neurons. There are no connections between neurons in input layer. The memory complexity of such approach is $O(n^2)$ for one neural network. An exemplary genotype with direct encoding and a decoded phenotype is shown on fig. 1.

$$V = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1]$$

$$E = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

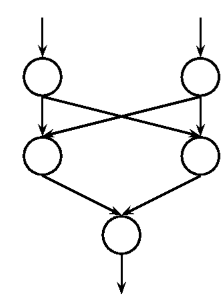

Fig. 1. Direct encoding genotype and the decoded phenotype

2.2 Indirect encoding

Indirect encoding comes as a compromise between time and quality. The most important feature of this approach is the limitation of the size of the domain of search. This causes the method to be very fast, but very often with suboptimal results. Memory requirements of this method are smaller than in direct encoding. Representation of a neural network with n neurons requires $O(n)$ memory. In used approach a neural network architecture is represented as a binary tree with a set of grammar rules for network architecture construction. Every node of the tree contains one of two non-terminal symbols (S and P) or a terminal symbol (E). All leaf nodes of the tree are terminal symbols. Non-terminal symbol S denotes division of the active neuron into two sequentially connected neurons. Non-terminal symbol P denotes division of the active neuron into two parallel neurons. An exemplary genotype with indirect encoding and a decoded phenotype is shown on fig. 2.

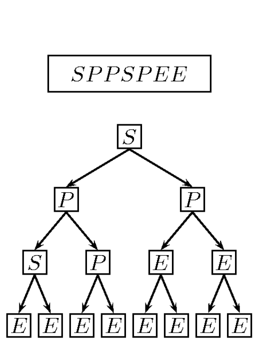

Fig. 2. Indirect encoding genotype and the decoded phenotype

2.3 Quality of a neural network

Quality of a neural network can be measured by two criteria: error rate during processing and size of the network itself. The smaller error, the better are the answers of a network during the processing of the test set. The smaller size, the better are the capabilities of a network to generalize during processing data which have not been seen previously.

The method of neural network evaluation used in all experiments has been presented in details in [2, 1]. Most of the experiments have been performed using data sets, especially HOUSING, IRIS and ZOO problems [14]. It is important to notice that used method allows to create neural networks with connections between more than one layer. A two phased evolution is used. The task of the first phase is to receive networks with small error rates, the task of the second phase is to reduce the size of neural networks and to keep the error rates small enough. The fitness function value consists of two elements: an error rate factor and a network size factor.

$$f = \begin{cases} f_t(1+f_s) & if \quad f_t > f_t^{min} \\ f_t \quad if & f_t \le f_t^{min} \end{cases} \tag{1}$$

Size factor f_s of a neural network is a function of number of connections between neurons. Error rate factor f_t is a function of error values on every output of a network during testing process. f_t^{min} is the maximum acceptable error factor. Detailed calculation of all factors have been presented in [2]. Evolved neural networks architectures are trained using Backpropagation method or by evolutionary approach.

3 Results of experiments

The difference of quality between direct and indirect encoding strongly depends on the discussed problem. The biggest difference is seen if the input vector is large. The natural ability of direct encoding of removing less or non important input features gives this method a big advantage. In case of such problems it has proven to be much more effective than the indirect approach in the aspect of quality. Indirect encoding is able to give very good solutions in very few generations, but evolved networks size is larger than in direct encoding (fig. 3).

Neural network evolved and trained for solving XOR problem is shown on fig. 1. This network architecture has been evolved using direct encoding and trained using evolutionary approach. It is impossible to receive this neural network using indirect encoding due it is outside the domain of search.

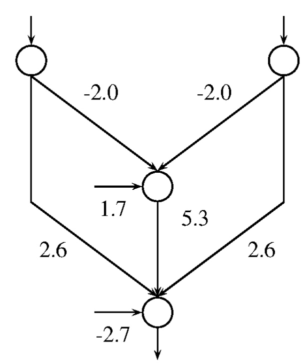

Fig. 3. Optimal neural network for XOR problem

Fig. 4. shows a neural network evolved for solving Parity of a number problem. It is evolved using direct encoding and trained using Backpropagation method. The smallest neural network for this problem evolved with

indirect encoding contains 6 connections between neurons. This example shows the abilities of direct encoding for neural network pruning.

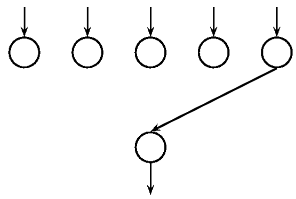

Fig. 4. Optimal neural network for Parity of a number problem

Fig. 6. shows the evolution results for the IRIS problem. In the experiment direct and indirect encoding give almost identical result. The neural network received using direct encoding is smaller only by 2 connection and evolution process took many more generations than for indirect encoding.

Exemplary results for ZOO, HOUSING and XOR problems using direct encoding have been shown on fig. 7. Evolution process requires a lot of computation time, but received neural networks are often very small and very well pruned. All neural networks have been trained using Backpropagation method.

4 Summary

Efficiency of direct encoding in generation of neural networks with very good quality has been proven by many experiments. A large part of experiments led to generation of very small networks with very good quality.

4.1 A size of networks and computation time

The biggest advantage of direct encoding is the ability of adding and removing single connections between neurons. The evolution process using direct encoding requires very long computation time (fig. 7.).

In every experiment indirect encoding has proven not to be able to reach global optimum. The optimal neural networks are outside the domain of search of this method. Received networks are often up to 6 times larger (fig. 6.) than in direct encoding. Indirect encoding has given very good neural networks in very few generations. It takes many more generations to reach the same solution by direct encoding.

Direct encoding requires much more computation time but is very effective in giving very small, very well pruned neural networks. Indirect encoding gives good results in very few generations, but it is not able to proceed further with optimization due to its limitations on the domain of search.

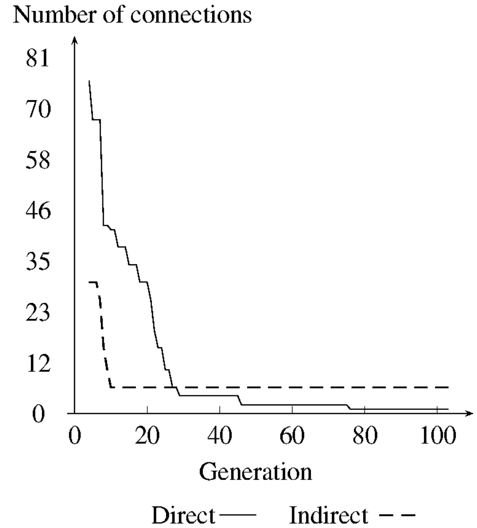

Fig. 5. Size of neural network in direct and indirect encoding, Parity of a number problem

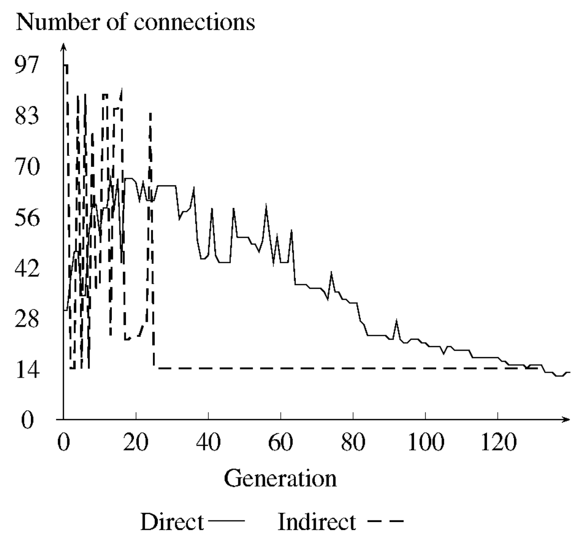

Fig. 6. Size of neural network in direct and indirect encoding, IRIS problem

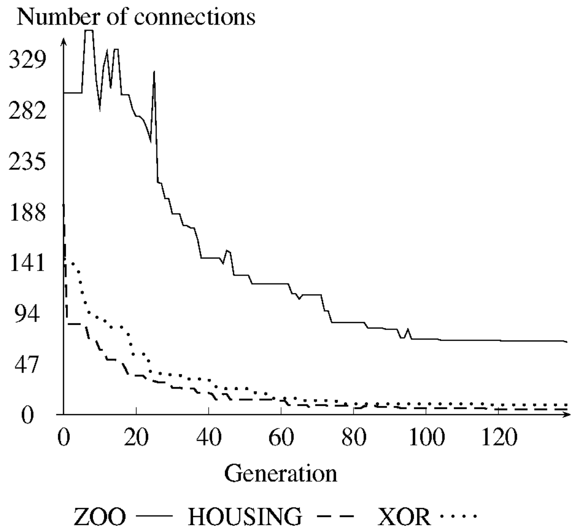

Fig. 7. Size of neural network in direct encoding

4.2 Size of the domain of search

Let us denote n as number of neurons in hidden layers, k as the size of input vector and l as the size of output vector. Used indirect encoding stores a neural network as a binary tree with two possible non-terminal symbols in each tree node. This gives the size of the domain equal to $2^n - 1$. Direct encoding uses a matrix (and a vector which is a mask) to encode a neural network and has the size of the domain equal to $2^{\sum_{x=1}^{n+l}(k+x-1)}$.

4.3 Further research

The further research will focus on combining the advantages of these two methods. Parallel process of evolution with two both types of encoding will be presented. Once, a network with indirect encoding is proven to be effective enough, it will be converted to a network with direct encoding and further evolved in the aspect of its quality optimization.

Other important problem concerns a learning method. As it is mentioned above, direct coding enables generation of small sized networks. Although, such a small network must learns its task. Our experiments showed that using Backpropagation method, a small network could not be able to learn desired task. Surprisingly, a GA can learn such a small network. An example is a XOR problem (see fig. 1). Backpropagation method starting from a set of random weights (from the range covering the proper values of weight) is not able to learn a network XOR problem while a GA does it. It seems that the optimum is very narrow and Backpropagation methods leads to the local optimum.

References

[1] Paradowski M. (2004) Evolution of neural network Architecture Using Genetic Algorithms (in Polish), Master Thesis, Wroclaw University of Technology

[2] Kwasnicka H., Paradowski M. (2004) Selection Pressure and an Efficiency of Neural Network Architecture Evolving, Lecture Notes on Artificial Intelligence, Springer-Verlag

[3] Goldberg D.E. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc.

[4] Tadeusiewicz R. (1993) Neural networks (in Polish), Akademicka Oficyna Wydawnicza RM

[5] Yao X. (1999) Evolving Artificial Neural Networks, School of Computer Science, The University of Birmingham

[6] Balakrishnan K., Honavar V. (1995) Properties of Genetic Representations of Neural Architectures, Proc. of the World Congress on Neural Networks (WCNN'95)

[7] Bornholdt S., Graudenz D. (1992) General Asymmetric Neural Networks and Structure Design by Genetic Algorithms, Neural Networks, Vol. 5, pp. 327-334

[8] Stanley K.O., Bryant B.D., Miikkulainen R (2003) Evolving Adaptive Neural Networks with and without Adaptive Synapses, 2003 IEEE Congress on Evolutionary Computation (CEC-2003)

[9] Chang S.O., Okurama A.M. (2004) Impedance-Reflecting Teleoperation with a Real-Time Evolving Neural Network Controller

[10] Garcia-Pedrajas N., Ortiz-Boyer D., Hervas-Martinez C. (2004) An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization, Preprint submittet to Elsevier Science

[11] Dewri R. (2002) Evolutionary Neural Networks: Design metodologies, AI depot, http://ai-depot.com

[12] Whitley D. (1995) Genetic Algorithms and Neural Networks, Genetic Algorithms in Engineering and Computer Science

[13] Gruau F. (1994) Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm, Universite Claude Bernard, Lyon

[14] Perchelt L. (1994) Proben1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules, Technical Report 21/94