139/2004

Raport Badawczy

Research Report

RB/29/2004

Genetic algorithms in learning methods of associative memory

A. Wilbik

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Janusz Kacprzyk

Warszawa 2004

Genetic algorithms in learning methods of associative memory

December 10, 2004

Anna Wilbik

System Research Institute, Polish Academy of Science, Poland

Abstract: In this paper we analyze the Hopfield neural network model of associative memory that uses evolutionary approach as a learning method. Several types of genetic algorithms will be investigated. In resulting networks quality measures (such as storage capacity, error correcting capabilities and the usage of additional knowledge) will be carefully examined.

The basic criterion for algorithm's evaluation is the storage capacity. The well-known Hebbian rule provides the capacity to be 15% of the number of the neurons. We show that genetic algorithms allow us to improve this result.

Keywords: Genetic Algorithm (GA), Hopfields Model, storage capacity, basin of attraction

1 Introduction

Genetic Algorithms and Neural Networks represent two technologies that enjoy an interest among computer scientists and engineers [20]. These technologies can be used to solve some very challenging problems. A natural question arises as to whether a combination of these two techniques might provide a synergy with more powerful problem solving than either of them alone. Recent years various schemes for combining genetic algorithms and neural networks have been proposed and tested (eg. [3, 16, 18, 21, 24, 25]).

Generally speaking, two main approaches have been proposed: supportive combinations and collaborative ones. Supportive combinations typically involve one of these methods to prepare data for the other. In collaborative combinations usually a genetic algorithm is used to determine the neural network's weight or its topology or both [17, 19].

Back propagation is one of the most popular methods to train neural networks. This method calculates the gradient of the error of the network with respect to the network's modifiable weights. This gradient is almost always then used to find weights that minimize the error. Therefore, using a genetic algorithm instead of, for example, back propagation learning methods does not seem to be successful enough. However, when the gradient information is hard to obtain or not available, genetic algorithms may appear to be promising methods [17].

Inspired by the results from Imada and Araki [6, 7, 8, 9], several experiments will be made. Since Hopfield's model is the basic representative of associative memories, it is chosen to be investigated. Hopfield's model will be trained with several types of genetic algorithms, to check the computational power of genetic algorithms.

The paper is structured as follows. In Section 2 general idea of neural networks and genetic algorithms will be presented. In Section 3 it is presented how evolutionary algorithms may be used to teach neural networks. Results are discussed in Section 5. The paper is completed by the concluding remarks.

2 Neural Networks and Genetic Algorithms

2.1 Neural Networks

Neural network is an information-processing device that consists of a large number of simple non-linear processing modules, connected by elements that have information storage and programming functions. In general, the modules involve four functions: input/output, processing memory, and connections between different modules providing for information flow and control [22].

Neural network is also defined as a network of neurons that are connected through synapses or weights. Each neuron performs a simple calculation that is a function of the activations of the neurons that are connected to it. Through feedback mechanisms and/or the nonlinear output response of neurons, the network as a whole is capable to perform extremely complicated tasks, including universal computation and universal approximation. Three different classes of neural networks are feedforward, feedback, and recursive neural networks, which differ in the degree and type of connectivity that they possess [11].

Neural network can be understood as a network where the nodes correspond to neurons and the arcs correspond to synaptic connections in the biological metaphor — a mathematical model of the brain's neurons. It is also referred to as an artificial neural system, natural intelligence, and neurocomputer [26].

Associative memory is one of the models of neural networks. The goal of associative memory is to learn certain set of patterns and to retrieve a pattern, when a damaged or noised one is given. The cells of associative memory are addressed by their content, not by the physical address [14].

In order to be effectively applied the associative memory should have great capacity, associative retrievability of patterns, especially damaged or noised and speedy learning and retrieving process. With the increasing number of the patterns, the network should deteriorate gradually, not rapidly. It should be flexible in remembering new patterns and association and should have the ability of forgetting the unrequired patterns or unlearn them.

Hopfield Model is one of the basic representatives of recursive networks. It is also called "associative memory", because of its function.

Discrete Hopfield's Network was built basing on two-state McCulloch-Pitt's neurons. The network consists of N mutually connected neurons. The outputs of neurons are connected back to the inputs of every other processing element except itself. (Figure 1)

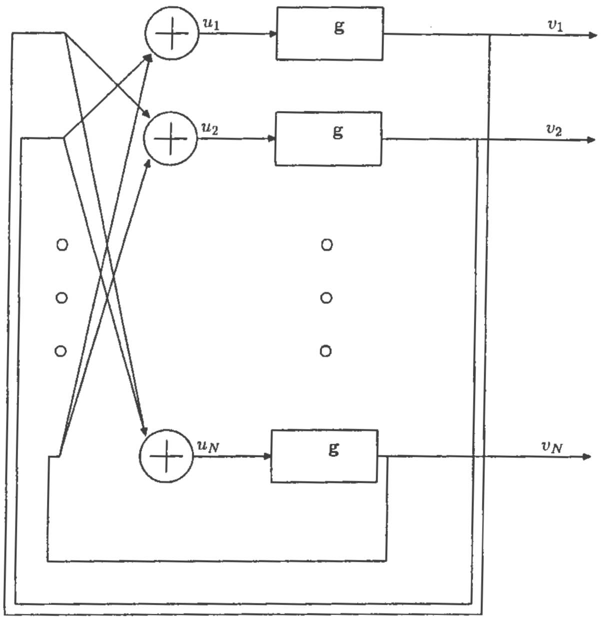

Figure 1: Scheme of the Hopfield Network.

Each neuron computes its input potential as a weighted sum of inputs:

$$u_i(t+1) = \Sigma_{j=1}^{N} w_{ij} v_j(t) + I_i, \;\; i = 1, ..., N$$

The output potential is a nonlinear transformation of the input potential:

$$v_i(t+1) = g(u_i(t+1)), \;\; i = 1, ..., N$$

where $u_i(t)$ is the input potential of the ith neuron at time t, and $v_i(t)$ is the output potential of the ith neuron at the same time. w_{ij} is the weight of the connection among the output of neuron j and input neuron i. I_i is an external potential; it is usually assumed $I_i = 0$.

The network can work either synchronously or asynchronously. In the synchronous mode all the neurons (or only some of them) are updated simultaneously. In the latter case neurons are updated consecutively in some random order [10].

The Hopfield network employs Hebbian learning rule. During the training process a fixed set of selected patterns $\{s_i^\mu\}$ is learned. i is the neuron index $i \in \{1, ..., N\}$, μ is the pattern index $\mu \in \{1, ..., p\}$ and $s_i^\mu \in \{-1, 1\}$ is the value of a particular pattern. It is assumed that there is a fixed number of p patterns that are to be trained. The weights are then calculated according to the following formula called the Hebbian rule:

$$w_{ij} = \begin{cases} \frac{1}{N}\Sigma_{\mu=1}^{p} s_i^\mu s_j^\mu & for\ i \neq j \\ 0 & for\ i = j \end{cases}$$

The factor $\frac{1}{N}$ is a choice of normalization of the weights. It is often convenient to take this factor into account, but essentially it does not affect any results.

The Hebbian rule was based on the idea, how the brain represents the concepts. Hebb postulated that some concepts can be represented by simultaneous stimulation groups of neurons. These groups are constructed during the learning process through the strengthening interplay, in other words weights, between all simultaneously stimulated neurons. The concept can be retrieved if sufficient number of neurons is contemporary stimulated [14].

If the network is updated asynchronously, then the network converges to a fixed point from any initial state vector when the weights matrix is symmetric and the main diagonal is equal to zero. If the model is updated synchronously, then the network eventually converges to either a fixed point or to an oscillation between two states. The maximal number of updating the neurons is equal to $4^{2^{p-1}}$, where p is the number of the patterns.

Hopfield demonstrated that maximum number of patterns that can be stored in the Hopfield model of N nodes before the error in the retrieved pattern becomes serve is around $0,15N$. This number was later evaluated theoretically and is equal to $0,138N$ [4].

2.2 Genetic Algorithms

The basic ideas of Genetic Algorithm (GA) were originally proposed by Holland [5]. GA is inspired by the mechanism of natural selection where stronger individuals are more likely to win in a competing environment.

Genetic algorithms are probabilistic algorithms in which a population of individuals is generated. GA presumes that the potential solution of any problem is an individual and can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary form.

Every genetic algorithm aims to find the best solution, so every individual needs to be evaluated. For this purpose, so called fitness function are used. A positive value, generally known as fitness value, is used to reflect the degree of "goodness" of the chromosome for the problem which would be highly related with its objective value [13].

New population is created by transforming the best-evaluated individuals and the new solutions are obtained [15]. Using a genetic evolution, the fitter chromosome has a tendency to yield offspring of better quality, which means a better solution of any problem.

Genetic algorithms involve continuous adaptation of population for the environment. Unfortunately, there is no guarantee that the process of adaptation terminates with providing the most desirable individual. However under some conditions, it is possible to prove that the probability of success increases and aims asymptotically to 1 [2].

Artificial evolution is occasionally named as "method of the last chance". If the knowledge about the problem suffices to shape an effective solution than this solution, it is more efficient than the usage of any evolutionary algorithm. Otherwise, the evolutionary approach may be used [2].

3 Using evolutionary approach to train neural networks

The existence of unequivocal attractors, towards which converges the network's state, allows us to use the net as an associative memory. The shape of the surface of the energy function, inter alia position of attractors and the shape of basin of attraction, depends on values of the weights between the neurons. Therefore it is sufficient to value the weights so that every pattern becomes an attractor point and the applicable trough is deep and wide enough to ensure the proper error correcting capabilities.

During evolution one can distinguish four main operations: initialization, fitness evaluation for every individual, checking if the termination condition is fulfilled and generating a new population (Figure 2).

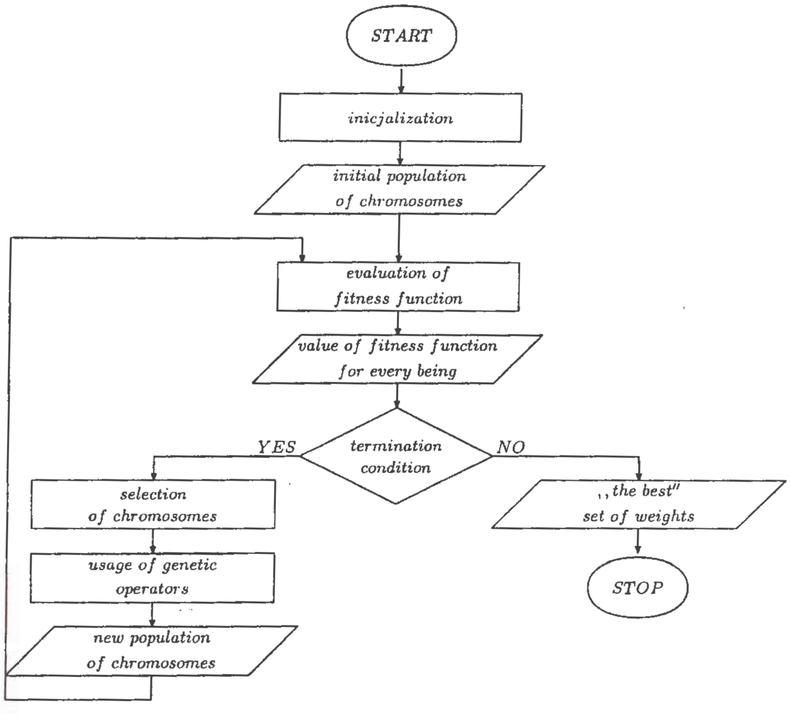
START
inicjalization
initial population of chromosomes
evaluation of fitness function
value of fitness function for every being
termination condition
YES
NO
selection of chromosomes
usage of genetic operators
new population of chromosomes
,,the best" set of weights
STOP

Figure 2: Scheme of the evolution's process.

Initialization is a method of creating the first population of individuals. Generally, a random initialization is used, but we can use an initial solution (if such exists). For example, in Hopfield Model such a solution is determined by Hebbian rule. The population is created by adding small perturbations to this solution.

Fitness is a real function, which is usually maximized. It features verifying, and therefore determines the algorithm. Improperly defined fitness function may lead to a domination of a solution (or a group of similar solutions) too early, eliminating the role of operators that might lead out the process from local minima. Nevertheless, if the function selects the population indifferently, the algorithm may loose the convergence to the optimal solution. In case of bipolar activation function of a neuron in output layer, fitness function may quantify the number of output signals coherent to expected ones.

Important part of generating new population is a selection process. It decides about the selecting the individuals, which form a new population or which children do it. New individuals are created by means of mutation and recombination operator. Recombination from selected two parent chromosomes creates a new chromosome, which has some features of each parent. Mutation modifies randomly genes in chromosome.

The termination condition depends on the particular application. Relatively easiest case is the situation, in which the optimal fitness value is known. For example, while training the neural network, it is the zero value of the error. We often limit the number of generations to avoid the infinite loop.

There are several reasons suggesting that it may be advantageous to apply genetic algorithms to the neural network weight optimization problem. Genetic algorithms have the potential to produce a global search and thereby to avoid local minima. Also, it may be advantageous to apply genetic algorithms to problems where gradient information is difficult or costly to obtain [20].

Genetic algorithms may be used to many different classes of neural networks. They are efficient with Feed Forward Neural Networks and with Recursive Neural Networks in supervised as well as in unsupervised learning. The essence of this independence from the topology of Neural Networks results from adequate defining the fitness function and coding the information for the specified application [10].

4 Algorithms

The lack of strict rules in genetic algorithms enable to created many sorts of these algorithms.

Since different algorithms leads to different results, it seems natural to state some quality criterions: We will consider the following:

- Storage capacity — the maximal number of patterns, that are remembered as fixed points by the network.
- Velocity of learning — number of generation, in which the solution was found.
- Degree of symmetry — a parameter of the form:

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} wji}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^2},$$

  This rate was introduced by Krauth [12]. Many researchers belief that the maximal number of connections and 100% symmetry of the weight matrix disable to achieve its maximal storage capacity. Other claims that doe to asymmetry of the weights matrix, the number of false attractors increases. Many genetic algorithms used for training neural network increases the storage capacity of the network. The higher storage capacity may result from reduction of the condition of full symmetry and pruning of the connection.

- 0 rate — the number of pruned connections. This indicator is defined as quotient of number of 0 weights to the number of all connections.

- Basin of attraction — the aim of the associative memory is to retrieve correct patterns and correcting of the errors of noised patterns. Therefore the basin of attraction is worth considering.

  The basin of attraction is the region around a memory in which all states are attracted to the memory within a prescribed time [1]. There is no analytical way to study basin of attraction except drawing a graph which shows the number of successes out of many trials against the number of input errors.

- If the optimal solution was not found, it may be worth checking which of the patterns were remembered correctly and how big is the error for the others.

- Parameters of genetic algorithms — numbers of individuals, number of parents and probability of mutation. These parameters are defined by the user.

In all algorithms presented below, we will use the same fitness function in order to facilitate comparison. Various methods of evaluating networks in algorithms could support various individuals.

Firstly, for each pattern μ its fixed point s^μ is calculated. The neurons are updated asynchronously to avoid problems with the oscillation. The fitness function is calculated according to the formula:

$$f = \frac{1}{Np} \Sigma_{\mu=1}^{p} \Sigma_{j=1}^{N} x_j^\mu s_j^\mu \tag{1}$$

where x_j^μ is a value of j th element of the μ th pattern.

The maximum value of this function is 1.0. This means that all patterns were remembered as fixed points. Value smaller than 1.0 informs that not all patterns became attractors.

4.1 Genetic Algorithm

Genetic Algorithm uses the solution found by Hebbian rule and modifies it. The scheme of the algorithm is presented on Figure 3.

At the beginning a constant matrix W_0 is created, $W_0 \in w_{ij}^0$,
$i = 1, ..., N$ and $j = 1, ..., N$. The values of the elements w_{ij}^0 are defined by Hebbian rule in two different ways. Its values can be either integers or they can be limited to the set $\{-1, 0, 1\}$ by the signum function.
Then the first population of individuals is created. Every being is described by a vector C of length N. The elements of the vector c_k can accept values $\{-1, 0, 1\}$. The weight matrix for every individual is computed as a product of corresponding elements of matrix W_0 and the vector C: $w_{ij} = w_{ij}^0 c(iN+j)$.

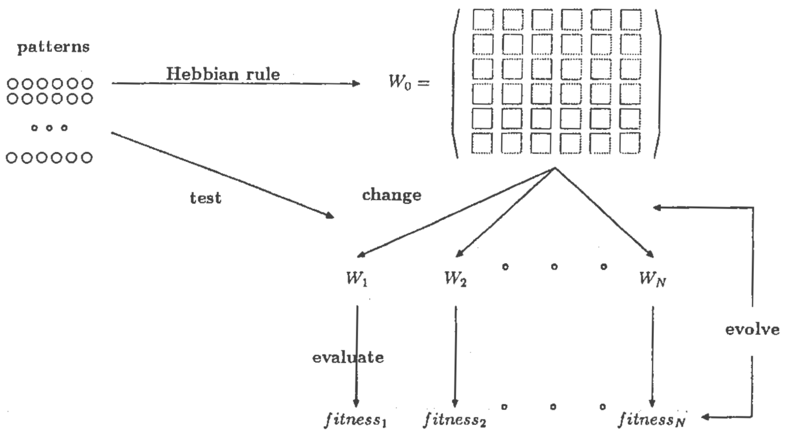

Figure 3: Scheme of the Genetic Algorithm.

Furthermore each network obtained in such a way, is evaluated with fitness function described by (1).

New individuals are created by uniform crossover and following mutation: $(-1) \longrightarrow (0)$, $(0) \longrightarrow (1)$, $(1) \longrightarrow (-1)$.

New generations are created until the optimal solution is found or number of generations exceeds the previously defined value.

If the values of matrix W_0 were integers, then the storage capacity is 27% of neurons' number. The matrix had poor error correcting capabilities. A mistake at one bite is corrected in 90% of the cases. The weight matrix has a high symmetry degree and only few connections are pruned.

In the other case, when values of the matrix W_0 are limited, the storage capacity is only 18%. The basin of attraction is smaller then for a network trained only with Hebbian rule, even if the weights of connections are limited in the same way. As in the previous experiment the weight matrix has a high degree of symmetry and the value of 0 rate is small.

4.2 Baldwin Effect Algorithm

In biology, the "Baldwin effect" is the result of the interaction of evolution with learning by individual animals over their lifetime. It turns out that individual learning tends to enhance evolutionary learning at the species level. The effect is named after J. Mark Baldwin, an American naturalist who described it in 1896 [23].

In machine learning the Baldwin effect means that it is possible to improve the results of an evolutionary algorithm by applying it to systems that learn by themselves.

Matrix W_0 is created randomly, $w_{ij}^0 \in \{-1,\ 1\}$. It remains unchanged during the evolution. Every chromosome consists of N^2 values from $\{-1,\ 0,\ 1\}$. The weights are calculated as

$$w_{ij}^k = e^k(iN + j)w_{ij}^0 \; i, j = 0, ..., N - 1$$

. Every matrix is taught with Hebbian rule:

$$w_{ij}' = w_{ij} + \lambda \Sigma_{\mu=1}^{p} x_i^\mu x_{i \neq j}^\mu$$

Then every trained network is evaluated with fitness function described by (1).

New individuals are created through uniform crossover of the chromosomes and cycle mutation: $(-1) \longrightarrow (0),\ (0) \longrightarrow (1),\ (1) \longrightarrow (-1)$.

Same as in previous algorithm new generations are created until the optimal solution is found or number of generations exceeds the previously defined value.

The scheme of the algorithm is presented in Figure 4.

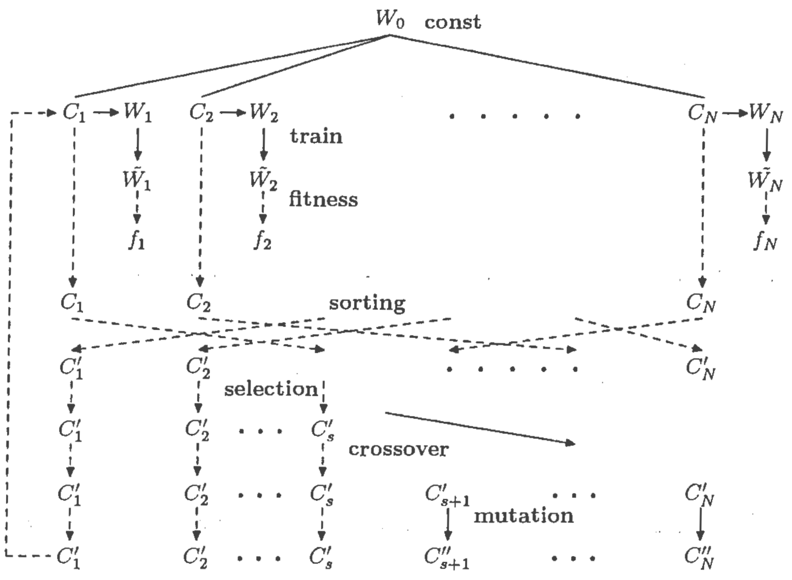

Figure 4: Scheme of the Baldwin Effect Algorithm.

The obtained storage capacity is 16% of neurons' number. The network is not resistant on errors. The degree of symmetry range from 0.3 till 0.6, so the matrix is asymmetric. There are not many connections with 0 weight.

4.3 Algorithm with Diploid Chromosome

In genetic algorithms a possible solution is usually represented as a one-dimensional array. However, an alternative way of representing solution can be used.

A genotype consists of two chromosomes, each of length $\frac{N(N-1)}{2}$. One of them represents the weights from upper right matrix's triangle while the other form lower left triangle. The elements on the main diagonal are always 0, therefore there is no use to represent them in genotype.

The scheme of the information representation is presented in Figure 5.

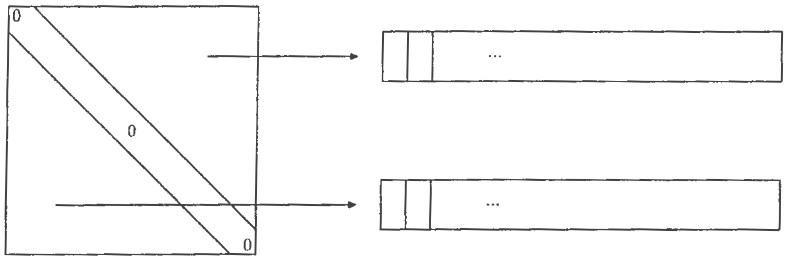

Figure 5: Scheme of the information representation in Algorithm with Diploid Chromosome.

At the beginning a constant matrix W_0 is created, $W_0 \in w_{ij}^0$, $i = 1, ..., N$ and $j = 1, ..., N$. The values of the elements w_{ij}^0 are integers and are defined by Hebbian rule.

The weight matrix is computed as:

$$\tilde{w}_{ij} = w_{ij}^0 c_k^1 \quad for\ i = 1, ..., N,\ j = i+1, ..., N$$

$$\tilde{w}_{ji} = w_{ji}^0 c_k^2 \quad for\ j = 1, ..., N-1,\ i = j+1, ..., N$$

$$k = \frac{(2N-1)(i+1)}{2} - 2N + j$$

The obtained network is then evaluated with fitness function (1).

Different form of representation of the solution requires different crossover operator. First, uniform crossover is operated on the pair of chromosomes of each parent and so a gamete is created. Two gametes from different parents form a single offspring.

Additionally a gen can be mutated:

$$(-1) \longrightarrow (0), \ (0) \longrightarrow (1), \ (1) \longrightarrow (-1)$$

The algorithm will stop creating new generations if the optimal solution is found or number of generation exceeds previously defined value.

The storage capacity of the network trained with this algorithm is 24% of neurons' number. The network can cope with maximum with 2 errors on satisfactory level. The weight matrix is almost symmetrical and has a few pruned connections.

5 Comparison of obtained results

When the algorithms are initialized with zero matrixes or randomly, basins of attraction of the networks are definitely smaller, then the basin of the network learned only by Hebbian rule. However, there is a method that allows us to improve correcting abilities of the network so that they are comparable with ones obtained from Hebbian rule. This method requires some modification of fitness function. Its value is no longer a number that informs if all patterns became fixed points, but also it consists information about the correcting abilities of the network.

If the initial population uses Hebbian rule to define the values of the weights, the basin of attraction improves itself while evolution processes.

The degree of symmetry depends on the way of initialization. The value of this parameter has a higher value if initial weight's matrix is computed according to the Hebbian rule. If first population is generated randomly, or all weights are 0, then obtained solutions are asymmetric.

The number of pruned connections remains usually at a very low level, only few percent of all connections. Only when the weights of all connection are 0 in the first generation, the 0 rate has a higher value.

Apart from rare cases for *Genetic Algorithm*, complements of all patterns are remembered as fixed points.

The influence of parameters, which affect the evolution and are defined by the user, is also investigated. They are numbers of the population, probability of mutation and the number of parents in population.

The numbers of individuals in population affect most velocity of finding the optimal solution. The bigger is the population the more points of the space is checked in every generation. Therefore it is more likely to find solution in generation with smaller ordinal number. In most tests no influence of the values of this parameter on the quality of solution is observed.

Another important parameter is the number of parents in the population, or better, the rate between numbers of parents and all population. It has significance not only for the velocity of learning, but also for its quality. When the number of parents is too small, the children may be not enough diverse. However, if there are too many parents, the improvement may be too slow. From the observations it appears that the optimal results are obtained if 40% of all individuals are parents.

A parameter that causes most problems is the mutation probability. Its value should be defined very carefully and the optimal value may be different for each algorithm. Mutation causes random changes to the population. If the value of this parameter is too small, the evolution may never leave out a local minimum. On the other side, too big value prevents from convergence.

6 Conclusions

In this paper a few genetic algorithms used to train neural networks, were presented and discussed. Tests that were carried out showed that genetic algorithms may be used to train the neural networks.

Obviously, there is no guarantee that the optimal solution will be found. However, the user does not obtain only one possible solution, but a whole set. In popular hybrid applications solutions found by the genetic algorithms are further trained with different methods if they are not optimal.

In presented algorithms a Hebbian rule was used. It is said that the storage capacity of the network learned by this rule is equal to 15% of numbers of neurons. However, there are many other learning rules for Hopfield's network. For example, a network learned with pseudo-inverse rule can remember N line-depended patterns, where N is total number of neurons. There is a possibility that if a different rule would be used instead of the Hebbian rule, the network could remember more patterns as fixed points.

Finally, we are thinking of using the evolutionary algorithms to train

Hopfield model of associative memory. A usage of different learning rules is also considered.

References

[1] D. J. Amit (1989). *Modeling Brain Function — The world of attractor neural networks*, Cambridge University Press.

[2] J. Arabas (2001). *Wykłady z algorytmów ewolucyjnych*, WNT, Warszawa (in Polish).

[3] S. Bornholdt and D. Grudenz (1991). *General asymmetric neural networks and structure design by genetic algorithms*, Deutsches Electronen-Synchrotron.

[4] B. Borowik (2002) *Pamięci asocjacyjne*, MIKOM (in Polish).

[5] J. H. Holland (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press.

[6] A. Imada and K. Araki (1995). Mutually Connected Neural Network Can Learn Some Patterns by Means of GA, Proceedings of the World Congress on Neural Networks *WCNN'95*, Washington DC, USA, vol.1, pp. 803-806.

[7] A. Imada and K. Araki (1997). Baldwin Effect on the Evolution of Associative Memory, Proceedings of the 3rd International Conference on

Artificial Neural Networks and Genetic Algorithms, Norwich, England, Springer–Verlag, pp. 354-358.

[8] A. Imada and K. Araki (1997). Random Perturbations to Hebbian Synapses of Associative Memory using a Genetic Algorithm, Proceedings of International Work-Conference on Artificial and Natural Neural Networks *IWANN'97*, Canary Islands, Spain, Lecture Notes in Computer Science 1240, Springer—Verlag, pp. 398-407.

[9] A. Imada and K. Araki (1997). Evolution of Associative Memory using Diploid Chromosomes, Proceedings of the Workshop on the Evolutionary Computation (10th Australian Joint Conference on Artificial Inteligence), Perth, Australia, pp. 24-36.

[10] J. Korbicz, A. Obuchowicz, and D. Uciński (1994). *Sztuczne sieci neuronowe, podstawy i zastosowania*, Akademicka Oficyna Wydawnicza, Warszawa (in Polish).

[11] R. A. Kosiński (2002). *Sztuczne sieci neuronowe — dynamika nieliniowa i chaos*, WNT, Warszawa (in Polish).

[12] W. Krauth, J. P. Nadal, M. Mezard (1988). The Roles of Stability and Symmetry in the Dynamics of Neural Networks, Journal of Physics A: Math Gen. 21, pp. 2995-3011.

[13] K. F. Man, K. S. Tang and S. Kwong (1999). *Genetic Algorithms - Concepts and Designs* Springer-Verlag London Limited.

[14] J. Mańdziuk (2000). *Sieci neuronowe typu Hopfielda, Teoria i przykłady zastosowań*, Akademicka Oficyna Wydawnicza EXIT (in Polish).

[15] Z. Michalewicz (1996). *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT, Warszawa (in Polish).

[16] D. J. Montana and L. Davis (1989). Training feedforward neural networks using genetic algorithms, Proceedings of eleventh international joint conference on artificial intelligence, San Mateo, pp. 762-767.

[17] M. Nałęcz, W. Duch, J. Korbicz, L. Rutkowski, and R. Tadeusiewicz (eds) (2000). *Biocybernetyka i inżynieria biomedyczna - Sieci neuronowe*, t. 6, Akademicka Oficyna Wydawnicza EXIT, Warszawa (in Polish).

[18] K. E. Nygard and N. Kadaba (1990). Modular neural networks and distributed adaptive search for traveling salesman algorithm, Proceedings of the SPIE, 1294, pp. 442-445.

[19] D. Rutkowska, M. Piliński, L. Rutkowski (1997). *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, PWN, Warszawa (in Polish).

[20] J. D. Schaffer, D. Whitley, L. J. Eshelman (1992). *Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art*, International Workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, Maryland, pp. 1-37.

[21] K. Suzuki and Y. Kakazu (1991). An approach to the analysis of the basins of associative memory model using genetic algorithms, In R. K. Belew and L. B. Booker (eds), Fourth international conference on genetic algorithm, San Mateo, pp. 539-546.

[22] R. Tadeusiewicz (1998). *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydawnicza, Warszawa (in Polish).

[23] P. D. Turney (1996). Myths and Legends of the Baldwin Effect, Proceedings of the Workshop on Evolutionary Computation and Machine Learning (13th International Conference on Machine Learning) , Bari, Italy, pp. 135-142.

[24] D. Whitley (1989) Applying genetic algorithms to neural network learning, Proceedings of the Seventh Conference of the Society of Artificial Intelligence and Simulation of Behaviour, Sussex, England, pp. 137-144.

[25] A. P. Wieland (1991). Evolving neural network controllers for unstable systems, IEEE international joint conference on neural networks, Seattle, pp. II667-II673.

[26] J. Żurada, M. Barski, W. Jędruch (1996). *Sztuczne sieci neuronowe — podstawy teorii i zastosowania*, Wydawnictwo Naukowe PWN, Warszawa (in Polish).