UDC 004.855 (045)
DOI: 10.18372/1990-5548.56.12936

O. I. Chumachenko

ALGORITHM OF NEURON NETWORKS MODIFICATION

National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine
E-mail: chumachenko@tk.kpi.ua

Abstract**—***It is considered a problem of neuron network modification whose topology has been chosen previously as a result of optimization problem solution for given task. The proposed modification algorithm is based on two-stages procedure which consists of genetic algorithm and local algorithm of optimization. The problem of modification is represented as two tasks: the search of optimal neuron network structure and weight coefficients adjustment. For the solution of these two problems it is used two-stages algorithm, in which at the first stage it is applied hybrid multicriteria evolutionary algorithm and at the second stage it is determined values of weight coefficients with help of back propagation error method and method of steepest descent. The determination of optimal values of hidden layers quantity is executed with help of adaptive algorithm of merging and growing.*

Index Terms—Neuron networks; optimization problem; hybrid multicriteria evolutionary algorithm; method of steepest descent; algorithm of merging and growing.

I. INTRODUCTION

The development and implementation of artificial neural networks (ANN) on the basis of advanced technologies is one of the priority directions of development of branches of science and technology in all industrialized countries.

While solving applied problems, in order to increase the accuracy and reduce the complexity, there are problems of finding the optimal topology of the network and, in accordance with it, the structural (determining the number of hidden layers and neurons in them, the interneuron's of separate ANN) and parametric (adjustment of weighting coefficients) optimization.

The main limitations of the known methods and technologies that are being used today are due to the lack of efficiency in solving the problem of learning the ANN, adjusting and adapting to the problem area, processing incomplete and inaccurate source information, interpreting the data and accumulation of expert knowledge of the uniform presentation of information coming from various sources, etc.

The construction of hybrid neural networks, which consist of different types of ANN, each of which is taught by a certain algorithm in a layer, in many cases, can significantly improve the performance of ANN.

Most applications use feed forward ANNs and the back-propagation (BP) learning algorithm [1]. The central issue in using ANNs is to choose their architectures appropriately. A too large architecture may overfit the training data, owing to its excess information processing capability. On the other hand, a too small architecture may underfit the training data, owing to its limited information processing capability. Both overfitting and underfitting cause bad generalizations, an undesirable aspect of using ANNs. It is therefore necessary to design ANNs automatically so that they can solve different problems efficiently.

There have been many attempts in designing ANNs automatically, such as various constructive, pruning, constructive-pruning, and regularization algorithms [2] – [4]. A constructive algorithm adds hidden layers, neurons, and connections to a minimal ANN architecture. A pruning algorithm does the opposite, i.e., it deletes unnecessary hidden layers, neurons, and connections from an oversized ANN. A constructive–pruning algorithm is a hybrid approach that executes a constructive phase first and then a pruning phase. In addition, evolutionary approaches, such as genetic algorithms [5], evolutionary programming [6], [7], and evolution strategies [8], have been us.

II. PROBLEM STATEMENT OF NEURAL NETWORK OPTIMAL MODIFICATION

The finite set is given $J = \{(\mathbf{R}_j, \mathbf{Y}_j)\}$, $j = 1,...,P$ pairs of type "attribute-value", where $\mathbf{R}_j, \mathbf{Y}_j$ are input and output vectors respectively.

It is necessary to optimally modify the basic ANN according to the training sample of this problem (structure and parameters) of the chosen optimal topology according to a sample of this type of task. As the criterion of optimality, a vector criterion is adopted

$$\mathbf{I} = \{I_1(x),\ I_2(x)\},$$

where $I_1(x) = E_{\text{gen}}(x)$ is the generalization error, which determines the magnitude of the error in

solving the task; $I_2(x) = S(x)$ is the complexity of the neural network (the number equal to the total number of computational operations needed to calculate the source vector **Y** by the input vector **R**, which depends only on the topology of the network, which is determined by the vector **X** or number of cross-connections); $x = (s, p, q, w)^T$; s is the number of hidden layers; $p = \{p_i\}$, $i = \overline{1,s}$ is the number of neurons in hidden layers; p_i is the number of neurons in the ith hidden layer; $q = \{q_{ir,if}\}$ are cross-connections; $i \neq j$, i, j are numbers of layers, $i, j = \overline{1,s}$; r, f are numbers of the neurons in the hidden layers i and j respectively; $r = \overline{1,p_i}$, $f = \overline{1,p_j}$, where p_i and p_j are number of neurons in the layers i and j respectively;

$$q_{ir,if} = \begin{cases} 1 & \text{cross-connction,} \\ 0 & \text{no cross-connction,} \end{cases}$$

$w = \{w_{ij,k}\}$, $i = \overline{1,s}$; $j = \overline{1,p_j}$, $k = \overline{1,g_{ij}}$. are values of weight coefficients; g_{ij} is the number of inputs of the ith neuron and jth layer.

III. Analysis of Effectiveness of Evolutionary Algorithm

The results obtained by the author testify to the low percentage of experiments in which the value of the extremum with given accuracy was obtained for the hybrid genetic algorithm (HGA) [9], methods of random gradient search, in case of optimization of multi-extremum functions. However, under unimodal functions optimization, there is a good approximation of the found extremum to the true extrumums of all methods, and with the optimization of multi-extreme ones – for methods (HGA) and random searches.

The reason for this is the property HGA and method of random search, which is associated with the rapid localization of the extreme existence zone.

Gradient algorithms is characterized by a consistent study of the search zone which allows in most experiments find a local extremum with a given accuracy, but it's not suitable for the search of a global extremum. To contrast to the method of random search in HGA, issue a mechanism of directed motion to extremum due to implementation in the algorithm of natural selection, that's why HGA gives a greater percentage of the localization of the global extremum.

Thus, HGA on one side is time consuming enough, requires setting of certain parameters values by user, determination of optimal control parameters set, so that evolutionary processes can balance the search and using in case of good quality solutions finding (for example, if the speed of crossover and mutation are chosen too high, then a significant part of search space has been investigated, but there is a high probability of good solutions loss, inability to use existing solutions), reveals the inability highly likely to find the exact value of extremum under neuron networks training, and, on the other hand, it enables to localize the region of global extremum existence.

According to the conclusions it's expedient to create a two-stage optimization algorithm: the genetic algorithm will be the most effective procedure at the initial stage of the solution finding, on which the existence region of the global extremum is determined; the second stage of the search will be connected to a refinement of the minimum based on the local optimization algorithm. The analysis of the results showed that the use of the local optimization algorithm on the last iteration can improve the accuracy of the extremum finding for both single-extremity and multi-extremity functions. The development and implementation of this algorithm is discussed below. The use of two-stage algorithm for optimizing the training procedure ANN will allow simultaneously to solve two tasks: to increase the rate of convergence of the algorithm due to the properties of the genetic algorithm, to investigate the entire search space as a whole and increase the accuracy of the extremum finding through the use of effective methods of local optimization.

As stated above, the problem of optimal modification of the neural network consists of two subtasks: the search of optimal structure (the number of hidden layers and neurons in them and the cross-connections between neurons) and adjusting the weighting coefficients (subtask of parametric optimization). To solve both subtasks a two-stage optimization algorithm is used: in first stage, which use hybrid multicriteria evolution algorithm, with help of which the zone of finding the optimal structure and weight coefficients of ANN is localized. At second stage the optimal values of ANN weight coefficients determination is executed on the basis of back propagation error and steepest descent (stochastic gradient descent) methods use and the determination of the optimal values of hidden layers and the number of neurons in them will be realized by means of an adaptive algorithm of merging and growing, when for each neuron of hidden layer the significance will be calculated and in case when its value is less than threshold then these neurons will be labeled with the subsequent calculation of the correlation coefficients of these

neurons with unlabeled neurons of this layer. Each labeled hidden neuron will be merged with its most correlated from this layer unmarked analogue. In case if the learning error in the process of the merging performing operations stops decreasing, an operation of growing is performed, i.e. adding one neuron to this hidden layer, and so on.

IV. Adaptive algorithm for Merging and Growing

Proposed adaptive algorithm for merging and growing is a modifying version of [8] and can be represented as follows.

Step 1) Create an initial ANN architecture consisting of m hidden layers. The number of neurons in the input and output layers is the same as the number of inputs and outputs of a given problem, respectively. The number of neurons *M* in the hidden layer is generated at random. Initialize all connection weights of the ANN uniformly at random within a small range.

Step 2) Initialize an epoch counter $\mu_{ij} = 0$, (*i* is the number of hidden layer; *j* is the number of hidden layer neurons, $i = \overline{1, n_1}$; $j = \overline{1, n_{2i}}$; n_1 is the quantity of hidden layers; n_{2i} is the quantity of neurons in *i* hidden layer) for each hidden neuron h_{ij}. This counter is used to count the number of epochs for which a hidden neuron is trained so far.

The number of epochs τ is specified by the user.

Step 3) Train the ANN using a two-stage learning optimization algorithm on the training set for a fixed number of epochs.

Step 4) Increment the epoch counter as follows $i = \overline{1, n_1}$, $j = \overline{1, n_{2i}}$,

$$\mu_{ij_k} = \mu_{ij_k} + \tau, \qquad (1)$$

where *k* is the number of epochs, $k = \overline{1, N}$; *N* is the quantity of epochs where *N* is the number of hidden neurons in the existing ANN architecture. Initially, *N* and *M* are the same.

Step 5) Evaluate each ANN in accordance with a predetermined fitness function on the validation set, if the termination criterion is satisfied, then stop the evolution process and go to step 6. In the opposite case, the evolutionary process continues and go to step 15.

Step 6) Compute the error of the ANN on the validation set. If the termination criterion is satisfied, stop the training process, and the current network architecture is the final ANN. Otherwise, continue.

Step 7) Remove the label of hidden neurons, if there exists, and compute the significance $\eta_{ij}, i = \overline{1, n_1}, j = \overline{1, n_{2i}}$ of each hidden neuron $\eta_{ij} = \frac{\sigma_{ij}}{\sqrt[3]{\mu_{ij}}}$, where σ_{ij} is the standard deviation, which is computed based on the outputs h_{ij}, for the examples in the training set.

Step 8) If the significance of one or more hidden neurons is less than the threshold value *h** (defined by the user), label these neurons with *S* and continue. Otherwise, go to *step 13*.

Step9) Compute the correlation between each *S*-labeled hidden neuron and other unlabeled hidden neurons in the ANN.

Step 10) Merge each *S*-labeled hidden neuron with its most correlated unlabeled counterpart. It is assumed here that the *S*-labeled hidden neuron is not only less significant but also redundant. Thus, AMGA produces one new hidden neuron by merging the *S*-labeled hidden neuron with its unlabeled counterpart. This new neuron does not contain any label *S*, and AMGA initializes a new epoch counter with zero for it.

Step11) Repeat the training of a modified ANN, which is obtained after the merging of hidden neurons, until its previous level of error is reached. If a modified ANN is able to reach its previous level of error, continue. Otherwise, restore the unmodified ANN and go to *step 13*.

Step 12) Update the epoch counter for each hidden neuron of the modified ANN go to *step 6*. The epoch counter is updated as follows:

$$\mu_{ij_k} = \mu_{ij_k} + \tau_r, \quad i = 1, 2, ..., N,$$

where τ_r is the number of epochs for which the modified ANN is retrained after the merge operation.

Step 13) Check the neuron addition criterion that monitors the progress of the training error reduction. If this criterion is satisfied, continue. Otherwise, go to *Step 3* for further training. It is assumed here that, since the merge operation is found unsuccessful (or cannot be applied) and the neuron addition criterion is not satisfied, the performance of the ANN can only be improved by training.

Step 14) Add one neuron to the current ANN architecture and go to *Step 3*. Since the error of the ANN does not reduce significantly after training and the merge operation is found unsuccessful (or cannot be applied), the performance of the ANN can only be improved by adding hidden neurons h_{ij}. The hidden neuron is added by splitting an existing hidden neuron of the ANN. The splitting operation produces two new hidden neurons from an existing

hidden neuron. The epoch counters are initialized by dividing μ_i into two, where μ_i is the number of epochs for which the existing hidden neuron is trained so far.

Step 15) Choose a ANN for reproducing and evolutionary operations.

Step 16) Apply evolutionary operators such as crossover and / or mutation to ANN architectures and weight coefficients for obtaining "offspring".

Step 17) Obtain a new general sample of "parents" and "descendant" for the next generation, then go to *step 3*.

V. Results

The developed algorithm permits dynamically prune or add hidden neurons at different stages of the training process. This means that two-stages algorithm applies the merge or add operation when the criteria for these operations were satisfied during training. It is agreed with the results, received in [8].

The application of two-stages algorithm was considered under creation of deep belief networks.

Deep belief networks can be used to solve a wide range of tasks (classification, forecasting, etc.). For faster results, deep belief networks were used to solve a prediction task based on a minimalistic sample.

The kin sample (Kinematics of Robot Arm) describes the kinematics of a robot manipulator with multiple links. Among the existing variants of this sample, the 8mn version (8 parts of the manipulator) is used in the work. This option is considered highly nonlinear and moderately noisy.

The results of proposed algorithm are shown on Fig. 1.

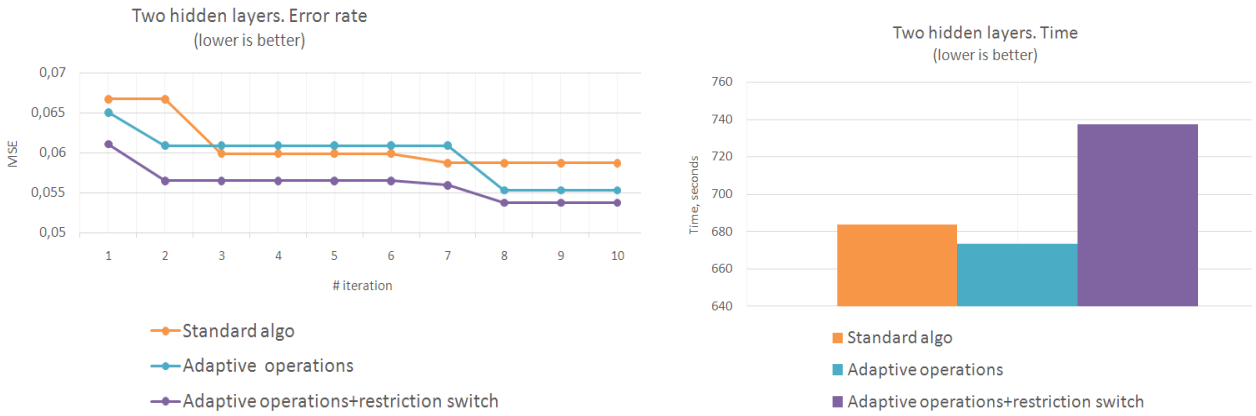

Fig. 1. Results of proposed algorithm functioning

VI. Conclusion

A new method for modifying a neural network has been developed, in which the adaptation of the SNM parameters (weight coefficients, architecture) is carried out in two stages in order to increase the efficiency of the solution of the problem (increase of accuracy and reduction of solving time): in the first stage a hybrid multicriteria evolutionary algorithm is used, and on the second - for more accurate determination of the number of neurons in hidden layers an adaptive algorithm of merging and growing is used, the weighting coefficients are specified by the method steepest descent.

This approach permits to design compact ANN architectures with good generalization ability.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, vol. I, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[2] T. Y. Kwok and D. Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 630–645, May 1997.

[3] R. Reed, "Pruning algorithms – A survey," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 740–747, Sep. 1993.

[4] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, Mar. 1995.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[6] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[7] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE Press, 1995.

[8] Md. Monirul Islam, Md. Abdus Sattar, Md. Faijul Amin, Xin Yao, *Fellow, IEEE*, and Kazuyuki Murase, "A New Adaptive Merging and Growing Algorithmfor Designing Artificial Neural Networks,"

IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, vol. 39, no. 3, June 2009, pp. 705–709.

[9] V. M. Sineglazov, O. I. Chumachenko, and D. Koval, "Improvement of the Hybrid Genetic Algorithm for the Deep Neural Networks Synthesis," IV International Scientific and Practical Conference "Computing Intellect" (Kyiv, May 16-18, 2017), pp. 142–143.

Received February 09, 2018

Chumachenko Olena. Candidate of Science (Engineering). Assosiate Professor.
Technical Cybernetic Department, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine.
Education: Georgian Politechnic Institute, Tbilisi, Georgia, (1980).
Research area: system analysis, artificial neuron networks.
Publications: more than 60 papers.
E-mail: chumachenko@tk.kpi.ua

О. І. Чумаченко. Алгоритм спрощення гібридних нейронних мереж

Розглянуто задачу модифікації нейронної мережі, топологія якої була обрана раніше в результаті вирішення оптимізаційної проблеми. Запропонований алгоритм вирішення базується на двоетапній процедурі, яка складається з генетичного і локального алгоритму оптимізації. Проблему модифікації представлено у вигляді двох задач: пошук оптимальної структури нейронної мережі і налаштування вагових коефіцієнтів. Для вирішення цих двох завдань використано двоетапний алгоритм, у якому на першому етапі застосовується гібридний багатокритеріальний еволюційний алгоритм, а на другому етапі визначаються значення вагових коефіцієнтів за допомогою методу зворотного поширення помилки і методу найшвидшого спуску. Визначення оптимальної кількості прихованих шарів виконується за допомогою адаптивного алгоритму об'єднання і нарощування.

Ключові слова: нейронні мережі; проблема оптимізації; гібридний багатокритеріальний еволюційний алгоритм; метод найшвидшого спуску; алгоритм об'єднання і нарощування.

Чумаченко Олена Іллівна. Кандидат технічних наук. Доцент.
Кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна.
Освіта: Грузинський політехнічний інститут, Тбілісі, Грузія, (1980).
Напрям наукової діяльності: системний аналіз, штучні нейронні мережі.
Кількість публікацій: більше 60 наукових робіт.
E-mail: chumachenko@tk.kpi.ua

Е. И. Чумаченко. Алгоритм упрощения гибридных нейронных сетей

Рассмотрена задача модификации нейронной сети, топология которой была выбрана ранее в результате решения оптимизационной проблемы. Предложенный алгоритм решения основан на двухэтапной процедуре, которая состоит из генетического и локального алгоритма оптимизации. Проблема модификации предоставлена в виде двух задач: поиск оптимальной структуры нейронной сети и настройки весовых коэффициентов. Для решения этих двух задач использован двухэтапный алгоритм, в котором на первом этапе применяется гибридный многокритериальный эволюционный алгоритм, а на втором этапе определяются значения весовых коэффициентов с помощью метода обратного распространения ошибки и метода наискорейшего спуска. Определение оптимального количества скрытых слоев выполняется с помощью адаптивного алгоритма объединения и наращивания.

Ключевые слова: нейронные сети; проблема оптимизации; гибридный многокритериальный эволюционный алгоритм; метод наискорейшего спуска; алгоритм объединения и наращивания.

Чумаченко Елена Ильинична. Кандидат технических наук. Доцент.
Кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт им. Игоря Сикорского», Киев, Украина.
Образование: Грузинский политехнический институт, Тбилиси, Грузия, (1980).
Направление научной деятельности: системный анализ, искусственные нейронные сети.
Количество публикаций: более 60 научных работ.
E-mail: chumachenko@tk.kpi.ua