NEURAL NETWORKS AND BACK PROPAGATION ALGORITHM

[1] Vladislav Skorpil, [2] Jiri Stastny

[1] Department of Telekommunications, [2] Department of Automation and Computer Science,
Brno University of Technology, Purkynova 118, 612 00 Brno,
CZECH REPUBLIC, ++420 541 149 212, skorpil@feec.vutbr.cz

This paper describes our research about neural networks and back propagation algorithm. Back-propagation algorithm is based on minimization of neural network Back-propagation algorithm is an iterative method where the network gets from an initial non-learned state to the full learned one. The learning algorithm of back-propagation is essentially an optimization method being able to find weight coefficients and thresholds for the given neural network and training set.

Keywords: Neural networks, Back propagation algorithm

1. INTRODUCTION

The following steps can describe the appurtenant back-propagation algorithm:
a) Initialization.
b) Pattern submitting.
c) Comparison.
d) Back-propagation of an error and weight modification. The values
e) Termination of pattern selection from the training set.
f) Termination of learning process.

2. PROBLEM STATEMENT

The General schematic of the genetic algorithm (GA) is considered to be a stochastic heuristic (or meta-heuristic) method. Genetic algorithms are inspired by adaptive and evolutionary mechanisms of live organisms. The best use of GA can be found in solving multidimensional optimisation problems, for which analytical solutions are unknown (or extremely complex) and efficient numerical methods are also not known.

Commonly used genetic algorithms do not copy the natural process precisely. The classical version of GA uses three genetic operators – reproduction, crossover and mutation. There are many ways how to implement genetic algorithm. Many differences can be observed in the strategy of the parent selection, the form of genes, the realization of crossover operator, the replacement scheme etc. One of the biggest disadvantages is a tendency of GA to reach some local extreme. In this case GA is often not able to leave this local extreme in consequence of the low variability of members of population. The interesting way how to increase the variability is using of the death operator [7].

Every member of the population has the additional information – age. A simple counter incremented in all GA iterations represents the age. If the age of any member

of population reaches the preset lifetime limit, this member "dies" and is immediately replaced with a new randomly generated member. While new population member is created, its age is set to zero. The age is not mutated nor crossed over. This version of GA was used for minimization of the neural network energy [1].

The applied genetic algorithm operates as follows:

a) Generating the initial population. The initialization of all bits of all chromosomes in initial generation is random, using the generator of random numbers, which is a standard feature of the C++ Builder 5 development environment. The Gray code is used to encode the chromosome bits.

b) Ageing. It only shows up in the variant with limited length of life. All the individuals in the population have their age incremented and if it exceeds a set limit (it is also possible to set, implicitly, 10), the element is removed from the population and a new element is randomly generated in its place.

c) Mutation. Two methods of mutation are used in the program:

- classical method
- back-propagation method

In the classical method of mutation all the chromosome bits are successively scanned and mutated with a certain small probability p_{mut} . In the case of long chromosomes (of the order of tens of thousands of bits), however, this procedure proved to be tto slow. It was therefore replaced by another method, which yields the same but substantially faster results: $v = p_{mut} * n$ are chosen randomly from the chromosome and then mutated.

In the back-propagation method of mutation the Back-propagation algorithm is used as the operator. Weights are decoded from the chromosome and set in the neural network and then, depending on the assignment, several cycles of Back-propagation are perfomed. The adjusted weights are then encoded back in the chromosome bit A disadvantage of the method is the great computation complexity.

d) Calculation of the value of object function

Neural network error function SSE is used as the object function over all models [4]. GA performs the minimization of this error function. The quality of an individual in the population is calculated as follows:

A neural network with the respective configuration (which is invariant and designed prior to starting the GA) is formed

Weights are decoded from the binary chromosome and set in the neural network.

All the models from the training set (see) are successively conveyed to the neural network input. The response of neural network to the input data is calculated and the difference between the actual and the required value is used to evaluate the SSE error over all models. This error represents the chromosome quality.

e) Upward population sorting. We are looking for the function minimum so that a chromosome with the least object function value will be in the first place. The Quicksort algorithm, which is very effective, is used for sorting; it can therefore be expected that the necessity of sorting will not affect the speed of algorithm negatively.

f) Crrossing. Uniform crossing is used – every bit of descendatnt is with the probabillity 0.5 taken from one of the parents. N^* = N/2 of descendants is made by the crossing (one half of the population).

g) Finalization. Return to the step 2 if the finalization condition is not realized. If it is realized then end of GA transaction. There are exist two finalization variation (or their combination)

-maximal number of iteration

-the quality of the best solution, smaller then entered

3. RESULTS

This network belongs to the most recent neural networks. It is a type of forward multi-layer network with counter-propagation of signal and with teacher learning. The network has two layers, with different types of neurons in each layer. Its advantage is mainly the speed of learning.

The structure of this two-layer network is similar to that of the Back-Propagation type of network but the function of output neurons must be linear and the transfer functions of hidden neurons are the so-called Radial Basis Functions – hence the name of the network. The characteristic feature of these functions is that they either decrease monotonically or increase in the direction from their centre point. Excerpt for the input layer, which only serves the purpose of handing over values, an RBF network has an RBF layer (hidden layer) and an output layer formed by perceptrons.

3.1. RBF neural network learning

The training set is formed by the input-output pair. RBF network learning is divided into two stages:

1. RBF layer neurons learning (prototypes learning). In the first stage, prototype C and sigma are determined for each RBF neuron. For example the algorithms for cluster analysis are used. To speed up this stage, non-adaptive methods can also be used such as uniform or random distribution of RBF neuron centres over the input space.

2. Output layer neurons learning. The objective of the second stage of learning is to determine the weights of input neurons, for example the least square method or gradient algorithms can be used.

Prototypes learning. First the number of clusters in input data is estimated, the pertinence function of model m to the cluster is defined and the coordinates of all p vectors Cp being the centres of clusters are estimated.

Steps of K-Means algorithm:

- Initialize RBF neuron centres C in random.
- Calculate m() for all samples from the training set.
- Calculate new centres C as the average of all samples that pertained to centre k by the pertinence function.
- Terminate if m() does not change, otherwise continue with point 2.

Specially for classification, the RBF network is simplified by refraining from searching for clusters, which have to be searched when approximating. The centres

(prototypes) of neurons are set so that RBF neurons are represented by model clusters for the best.

Description of implementation. Each neuron in the radial basis layer will give on the output the value that depends on how close the input vector is to each of the weight vectors of the given neurons. Thus the RBF neurons whose weight vectors are a bit different from input vector p have an output of about zero. On the contrary, the RBF neuron whose weight vector is close to the input vector will have a value of about 1. Individual neuron layers have the form of one-dimensional array. The weight matrix is in the form of two-dimensional array, where the index gives the number of neurons being connected. It is necessary to enter the number of RBF neurons n for one category. The principal computation methods are:

- calculatePrototype() – by using the algorithm chosen (the K-means algorithm is used in the program) it will calculate the weight values of prototype C. On the output of RBF neurons the first 1…n neurons will calculate the output for the first category, n+1…2n for the second category, etc.
- calculate_sigma() – after calculating the weight values of prototypes, the size of the sphere of influence will be calculated for each RBF neuron.
- calculate_h() – it will calculate the outputs of RBF neurons. The radial basis function reaches maximum of 1 when there is 0 on the input.The RBF neuron thus operates as an indicator that produces 1 always when the input vector is identical to its weight vector.
- calculateOutputRBF() – on the output each RBF neuron contains the (in principle) percentage agreement with the input model. For the output neuron the value of the neuron which most agrees with the network input (maximum value) is chosen from n RBF neurons in the category given.

RBF networks tend to have more neurons than a comparable network with back propagation. This is because neurons with output function in the shape of sigmoid can have outputs over a large area of input space whereas RBF neurons respond only to relatively small areas of the input space. The result is that the larger the input space is the more RBF neurons are necessary. But the design of RBF networks takes considerably less time than training multi-layer network with back propagation.

4. CONCLUSION

Recognition with the aid of neural network is suitable where high-speed classification with randomly rotated objects is required and where we need to tolerate some differences between learned etalons and classified objects. The network was able to classify correctly 100% models and at the same time to recognize correctly even slightly damaged models. As the number of radial basis neurons is comparable the input space size and problem complexity, RBF networks can be larger than back-propagation networks. Back propagation algorithm presented very good results at classification. The network recognized all the patterns submitted. The learning using the BPx method (extended Back propagation algorithm) was a little slower, but it can be succesfully used for networks with lower number of neurons. Radial Basis

Function networks can be designed very quickly. The time necessary for network learning was very little.

5. ACKNOWLEDGMENTS

No MSM 0021630513 Research design of Brno University of Technology " Electronic communication systems and new generation of technology (ELKOM)"
Grant 2789/2006/C/b "Design of large electronic presentation of a different forms and branches of study at electrotechnic faculty" (grant of the Czech Ministry of Education, Youth and Sports)
Grant 2788/2006/F1/b „ Design of new course Services of telecommunication networks" (grant of the Czech Ministry of Education, Youth and Sports)
No CEZ: J22/98: 261100009 Nontraditional methods for investigating complex and vague systems (Research design of Brno University of Technology)
2E06034 Lifetime education and professional preparation in the field of telematics, teleinformatics and transport telematics (grant of the Czech Ministry of Education, Youth and Sports)

6. REFERENCES

[1] Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addisson-Wesley, 1989.

[2] Lim, T. – Loh, W. Y. – Snih, Y.: A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. 1999. http://www.stat.wisc.edu/~limt/mach1317.pdf

[3] Miehie, D. – Spiegelhalter, D. J. – Taylor, C. C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood, NY, 1994.

[4] Ripley, B. D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (United Kingdom), 1996.

[5] Pavlidis, T.: Algorithms for Graphics and Image Processing. Bell Laboratories, Computer Science Press, 1982.

[6] Wong, K. CH.: A new diploid scheme and dominance change mechanism for non-stationary function optimization. In Proceedings of the Sixth International Conference On genetic Algorithms, Pittsburgh, USA, 15. – 19. July 1995

[7] Popela, P.: An Object-Oriented Approach to Multistage Stochastic Programming: Models and Algorithms. Ph.D. Thesis, UK MFF Praha, 1998.

[8] Ošmera, P. – Roupec, J. – Matoušek, R.: Genetic Algorithms with Sexual Reproduction. In Proceedings of World Multiconference on Systemics, Cybernetics and Informatics SCI 2000, Orlando (Florida, USA), 2000, pp. 306–311.

[9] Sarle, W. S.: Neural Networks and Statistical Models. Proceedings of the Nineteenth Annual SAS Users Group International Conference, Cary, NC: SAS Institute,1994, pp 1538-1550.

[10] Galko, M., Krbilová, I., Vestenický, P.: Blocking Probability Influence on the Single-Channel Service System Operation with Various Input Flows. Proceedings of TRANSCOM'97 Volume 2, University of Žilina, Žilina 1997, pages 37-40

[13] Tomašov, P., Krbilová, I., Muzikářová, Ľ., Vestenický, P.: Operation Control of Railway 1Telecommunication Network. Proceedings of 10th International Conference TEMPT'97, Sofia 1997

[12] Vestenický, P., Krbilová, I.: Perspectives of Information Networks Development. Sborník přednášek celostátní konference s mezinárodní účastí TELEKOMUNIKACE '98. VUT, Brno 1998

[13] Krbilová, I., Vestenický, P.: Development Trends of Internet Services in Next Millennium. Zborník prednášok konferencie s medzinárodnou účasťou SLOVENSKO A INFORMAČNÁ SPOLOČNOSŤ. Slovenská elektrotechnická spoločnosť, Stupava 1998

[14] Bubeníková, E., Vestenický, P.: Principles Of The Intranet Information System Creation. ELEKTRO'99 Conference Proceedings, section Information & Safety Systems. University of Žilina, Žilina 1999, pages 77-81