Neural Networks Learning Methods for Image Processing Applications

JIRI STASTNY*, VLADISLAV SKORPIL**
* Department of Automation and Computer Science,
** Department of Telekommunications,
Brno University of Technology,
Purkynova 118, 612 00 Brno,
CZECH REPUBLIC,
http://www.vutbr.cz/

Abstract: - This paper deals of the problem of the image processing by using artificial neural networks. The MLP neural network was used. We compared results obtained by a using of different learning algorithms – the classical back-propagation algorithm (BP) and the genetic algorithm (GA). The real technological scene for image processing was simulated with digitization of two-dimensional pictures.

Key-Words: - Neural Networks, Genetic Algorithms, Image Processing, Pattern Recognition

1 Introduction

Back-propagation algorithm is based on minimization of neural network energy given with the formula (SSE):

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - d_i)^2 \tag{1}$$

where n means the number of network outputs, y_i means the i-th output and d_i means the i-th output required. 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. The network is supposed to be made up of neurons the behaviour of which is described with the formula:

$$y = S\left(\sum_{i=1}^{N} w_i x_i + \Theta\right) \tag{2}$$

where the output nonlinear function S is defined with the formula:

$$S(j) = \frac{1}{1+e^{-gj}} \tag{3}$$

where g determines the curve steepness in an origin of coordinates. Input and output values are supposed to be in the range <0, 1>.

In the following formulas parameter o denotes output layer, h hidden layer and i,j are indexes. Then y_i^h means i-th neuron output of the hidden layer and w_{ij}^o means weight connecting i-th neuron of the output layer and j-th neuron of the previous hidden layer.

The following steps can describe the appurtenant back-propagation algorithm:

1. **Initialization**. All the weights in the network are randomly set at values in the recommended range <–0.3, 0.3>.

2. **Pattern submitting**. A chosen pattern from the training set is put in network inputs. Then outputs of particular neurons are computed under relations (2) and (3).

3. **Comparison**. This step contains the computation of the neural network energy under relation (1) and the error for the output layer under the relation:

$$d_i^o = (d_i - y_i^o)y_i^o g\,(1 - y_i^o) \tag{4}$$

4. **Back-propagation of an error and weight modification**. The values

$$\Delta w_{ij}^l(t) = h d_i^l(t) y_j^{l-1}(t) + a\,\Delta w_{ij}^l(t-1) \tag{5}$$

$$\Delta\Theta_i^l(t) = h d_i^l(t) + a\,\Delta\Theta_i^l(t-1) \tag{6}$$

are computed for all neurons in the layer. Under the relation:

$$d_i^{h-1} = y_i^{h-1}(1-y_i^{h-1})\sum_{k=1}^{n} w_{ki}^h d_k^h \quad (7)$$

an error is back-propagated in the layer nearer. Then weights are modified:

$$w_{ij}^l(t+1) = w_{ij}^l(t) + \Delta w_{ij}^l(t) \quad (8)$$

$$\Theta_i^l(t+1) = \Theta_i^l(t) + \Delta\Theta_i^l(t) \quad (9)$$

This step is applied to all the layers of the network.

5. **Termination of pattern selection from the training set**. Another pattern from the training set is chosen and the step number 2 follows until all patterns were submitted.

6. **Termination of learning process**. The algorithm ends when the neural network energy in last computation has been less then the criterion selected.

The 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).

2 General schematic of the genetic algorithm (GA) used

The applied genetic algorithm operates as follows:

1. **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.

2. **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.

3. 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. (See Tab. 6).

4. Calculation of the value of object function

Neural network error function SSE is used as the object function over all models [4]:

$$E = \frac{1}{2} \sum_{q=1}^{p} \sum_{i=1}^{n} (y_i^q - d_i^q)^2 \qquad (10)$$

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.

4) 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.

6) 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).

7) 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 Problem Solution

The real technological scene was simulated with digitization of two-dimensional pictures of five real objects (see Fig. 1). The classical image preprocessing was used. Then every pattern was described by the flag vector [5]. Flag vectors have been submitted to network and arm lengths have been transformed to values from interval < 0, 1 >.

Fig. 1 – The scene used for testing

3.1 Evaluation

Optimal solution is such weights configuration, for which is the network possible to recognize all samples into 100%.

Algorithms were tested on the computer with the processor AMD Athlon XP 1700+ and 512 MB RAM.

Back-propagation:
In the Table 1 and 2 are results with Back-Propagation algorithm.

Learning of two models by BP:

Number of Layers	Time [s]	Error SSE over all models	Whole preception [%]
2	4,487	0,000032	100
2	3,976	0,000042	100
2	3,565	0,000116	100
1	3,545	0,000041	100
1	2,924	0,000047	100

Tab. 1: 2 models, 5000 iterations, sequency selestion of models

Learning of five models by BP:

Number of Layers	Time [s]	Error SSE over all models	Whole preception [%]
2	7,751	0,000040	100
2	5,938	0,000075	100
2	6,029	0,000091	99
1	5,638	0,000070	99
1	3,956	0,000077	99

Tab. 2: 5 models, 5000 iterations, sequency selestion of models

Genetic algorithm

Learning of two models by GA
Neural network parameters: 1 secret layer, 70 neurons GA parameters: population 500 of individuals, mutation probability 0.05, 11 bits for weight, life time 10 cycle, finalization for the error smaller then 0.01, or 500 of iterations

Number of Iterations	Time [h:mm:ss]	Error SSE over all models	Preception of the output [%]
6	0:00:23,0	0,000039	100
8	0:00:32,9	0,001978	94
2	0:00:09,5	0,001053	100
1	0:00:04,7	0,000712	100
17	0:00:64,6	0,008621	97
2	0:00:08,7	0,000356	97
6	0:00:23,0	0,006595	88
1	0:00:04,9	0,000604	100
2	0:00:09,7	0,000460	100
3	0:00:14,0	0,001251	95

Tab. 3 Network learning results by Genetic algorithm, 2 models

Learning of three models by GA:

Neural network parameters: 1 secret layer, 70 neurons GA parameters: population 500 of individuals, mutation probability 0.05, 11 bits for weight, life time 10 cycle, finalization for the error smaller then 0.08, or 500 of iterations

Number of Measuring	Number of Measuring [h:mm:ss]	Error SSE over all Models	Preception of the output 1 [%]	Preception of the output 2 [%]	Preception of the output 3 [%]
402	0:29:29	0,056189	87	88	86
500	0:36:07	0,084470	66	91	100
102	0:11:50	0,075513	90	63	100
304	0:16:18	0,062479	10	92	85
366	0:22:47	0,073709	87	89	87
5	0:00:19	0,003813	97	93	100
484	0:25:57	0,057401	84	88	78
500	0:27:01	0,099898	100	87	87
500	0:26:59	0,209757	55	81	73
500	0:26:58	0,175896	86	94	100
366,3	0:22:22	0,089913	85,2	86,6	89,6

Tab. 4 Network learning results by Genetic algorithm, 3 samples, population 500 chromosome

Number of Measuring	Number of Iterations	Time of Measuring [h:mm:ss]	Time of Measuring	Preception of the output 1 [%]	Preception of the output 2 [%]	Preception of the output 3 [%]
1	200	0:29:13	0,118626	61	71	78
2	200	0:29:37	0,148604	100	96	100
3	200	0:29:45	0,115807	95	70	66
4	32	0:05:04	0,015287	100	100	85
5	200	0:30:27	0,213459	0	91	97
Average	166,4	0:24:49	0,122357	71,2	85,6	85,2

Tab. 5 Network learning results by Genetic algorithm, 3 samples, population 1200 chromosome

Learning of four models by GA:)

Neural network parameters: 1 secret layer, 70 neurons GA parameters: population 500 of individuals, mutation probability 0.05, 11 bits for weight, life time 10 cycle, finalization for the error smaller then 0.1, or 1200 of iterations

Number of Measuring	Number of Iterations	Time of Measuring [h:mm:ss]	Time of Measuring	Preception of the output 1 [%]	Preception of the output 2 [%]	Preception of the output 3 [%]
1	1200	1:11:02	0,642201	0	94	90
2	1200	1:11:02	0,281050	72	87	72
3	1200	1:11:04	0,431428	93	0	100
4	1200	1:11:49	0,712210	55	95	100
5	1200	1:11:45	0,530850	0	89	92
Average	1200	1:11:20	0,519548	44	73	90,8

4 Conclusion

Back-propagation algorithm presented very good results at classification. The network recognized all the patterns submitted. The learning using the BPx method was a little slower, but it can be succesfully used for networks with lower number of neurons. While the GA was used, the results and the learning time highly depends on GA parameters setting. The increasing of reliability and decreasing of the learning time of GA using limited lifetime were observed. The best results were obtained by GA using the death operator and not using sexual reproduction.

Acknowledgments:

This paper has been supported by the research project CEZ: J22/98: 26100009 Non-Traditional Methods for Investigating Complex and Vague Systems of Brno University of Technology and by the grants:

No 102/03/0434 Limits for broad-band signal transmission on the twisted pairs and other system co-existence The Grant Agency of the Czech Republic (GACR),

No CZ 400011(CEZ 262200011) Research of communication systems and technologies (Research design)

Grant 2811 Introduction of advanced transport network technology into teaching (grant of the Czech Ministry of Education, Youth and Sports)

Grant 3112 Teaching Innovation for the Last Mile Data Transmission (grant of the Czech Ministry of Education, Youth and Sports)

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 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] Roupec, J.: *Vývoj genetického algoritmu pro optimalizaci parametru fuzzy regulátoru*. Ph.D. Thesis, VUT v Brne, 2001.

[8] 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.