Genetic Algorithm and Neural Network

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 describes application of Genetic algorithm (GA) for design of network configuration and for learning of neural network. Design of network configuration is the first area for GA exercise in relation to neural network. The number of neurons in network and placement to the layers has big influence over effectivity of whole system. If we are able to formulate quality classification of designed network from standpoint of topology, we can use GA for design of suitable network configuration. The second area (learning of neural network) consists in using of advantages of GA toward learning of neural networks. In this case GA looks for acceptable setting of network weights so, to make specified transformation – it practices minimalization of its mistake function. The Genetic algorithm 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 Genetic algorithm 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.

Key-Words: Genetic algorithm, Fitness function, Neural network, Back-propagation method

1. Introduction

Commonly used Genetic algorithms (GA) do not copy the natural process precisely. The classical version of Genetic algorithm uses three genetic operators – reproduction, crossover and mutation [1]. 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 Genetic algorithm to reach some local extreme. In this case Genetic algorithm is often not able to leave this local extreme in consequence of the low variability of members of population [6]. The interesting way how to increase the variability is using of the D-operator. Every member of the population has the additional information – age. A simple counter incremented in all Genetic algorithm 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.

2. Neural Network

Used neural network with radial basis RBFN (the Radial Basis Function Network) is type of single-direction multilayer network [3], [5]. 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. This neural network can be used for wide scale of problems, because it is able to approach arbitrary function and its training is quicker than for MLP (the Multi Layer Perceptron neural network). Quicker learning is given by this that RBFN has only two layers with weight and every layer may be determined sequential. The biggest problem of this network is to determine optimal network size for solution of given problem [2], [3], [7].

3. General Schematic of the Genetic Algorithm

The applied Genetic algorithm operates as follows [1]:

3.1 Generating of 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.

3.2 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.

3.3 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 performed. The adjusted weights are then encoded back in the chromosome bit A disadvantage of the method is the great computation complexity.

3.4 Calculation

Calculation of the value of object function. Neural network error function is used as the object function over all models :

$$E = \frac{1}{2} \sum_{q=1}^{p} \sum_{i=1}^{n} (y_i^q - d_i^q)^2 \qquad (3.1)$$

Genetic algorithm 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 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 error function over all models. This error represents the chromosome quality.
-

3.5 Sorting of upward population

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.

3.6 Crossing

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

3.7 Finalization

Return to the step 2 if the finalization condition is not realized. If it is realized then the end of Genetic algorithm is made. They obtain two finalization variation (or their combination):

- maximal number of iterations
- quality of the best solution, smaller then entered

4 Implementation of Genetic Algorithm

4.1 Initialization of Genetic algorithm

We have population in of the number of 100 chromosomes (randomly generated)) at the beginning. We preset engaged chromosomes longevity for each of them and calculate their value of fitness function. We sort out them according to sizes of fitness uplink , by this we set the best chromosome on the first position. It was used classical algorithm QuickSort for sorting.

4.2 Calculation of new generation

We create the following (new) generation so, that at first we expire all chromosomes and recognize its age for each of them, if it equals to zero. If it is true, we generate new chromosome on its place and we calculate its fitness. It is necessary again to order the whole population, the best chromosome to be forever on the first place.

Further it follows crossing. Two parents from whole population are selected at first, the crossing is made. Two new descendants originate and we calculate for them fitness value. We advance so further, that the number of chromosomes, from which we select, is decrease about 8 (in principle we leave out last 8 chromosome). This way we advance so long until it is not the number of new chromosomes (descendants) equals one quarter of original population. On the last step the size of population, from which parents for crossing will be selected, equals

to 8, it means, that it will be selected from the first 8 (the best) chromosomes.

It is necessary to ensure at selection of parents, so as to both parents are different. For all chromosomes which do not cross are their ages reduced about one. We insert according to fitness magnitude individual descendants to the original population, whereby we create the population, which as size one and quarter of original population (100 + 24 = 124) chromosomes. The last (the worst) quarter (24 chromosomes) will be cancelled. We obtain by this new population, which has again original number of chromosomes and includes old and new chromosomes.

The last step of generation of new population is mutation of all chromosomes in population. We calculate fitness value of all chromosomes in the end and we sort them uplink by fitness size. After this sorting we get on the first position chromosome with the best fitness value of just at the immediate iteration.

4.3 Pseudocode of algorithm

```
NumberIter          // ordered number of iterations
SizePopul           // ordered size of chromosomes
QuartPop            // one quarter of original population
Selection           // size of population, from which are selected chromosomes for crossing

// algorithm initialization
GeneratePopul();                    // it generates population of chromosomes with given size
CalculFitnessChromosome();          // it calculates fitness for every chromosome
SortePopulChromosome ();            // it sorts population uplink by fitness size
// own body of algorithm
for(i=0;i<NumberIter;i++)
{
        Select=SizePopul;
        for(j=0;j<SizePopul;j++)
        {
            if(Chromosome[j].Age()<0)
            {
        Chromosome[j].Generate();
        Chromosome[j].CalculFitness();
        SortePopul();
                };
};
for(k=0;k<(QurtPop/2))      // selection of parents and crossing
{
       R1=rand()%Selection;
       R2=rand()%Selection;
       CrossChromosome(R1,R2);
       Descendant[2*k].CalculFitness();
       Descendant [2*k+1]. CalculFitness()
        Selection=Selection-(2*4);
};
For all disinterested rossing_Reduce_Age();
SortePopulDescendant();
InsertPopulDescendantsBetweenChromosomes();
for(j=0;j<SizePopul;j++)
{
        Chromosome[j].Mutation();  // mutation of chromosome
};
CalculFitnessChromosome();
SortePopulChromosome();
};
return Chromosome[0];               // chromosome with the best fitness value
```

5. Conclusion

It is possible to submit on the basis of tests with Genetic algorithm of neural networks learning, that standard setting of Genetic algorithm (GA) demonstrates at crushing majority to learn neural network for diagnostic of one object completely on 100%

and in time multiple shorter than classical methods of learning. At higher of the number of learned standards however GA already did not give good results (did not find optimal solution). Calculation time is to be shown the biggest restriction for GA using. It turned out further, so on-learning of network depends very much on the network topology. GA could find its exercise for example in the area of network configuration design.

While the Genetic algorithm (GA) was used, the results and the learning time highly depends on Genetic algorithm parameters setting. The increasing of reliability and decreasing of the learning time of Genetic algorithm using limited lifetime were observed. The best results were obtained by Genetic algorithm using the D-operator and not using sexual reproduction.

Acknowledgement

This research was supported by the grants:

MSM 0021630529 Intelligent systems in automation (Research design of Brno University of Technology)

MSM 6215648904/03 Research design of Mendel University of Agriculture and Forestry in Brno

No 102/07/1503 Advanced Optimizing the design of Communication Systems via Neural Networks. The Grant Agency of the Czech Republic (GACR)

Grant 1884/2007/F1/a "Innovation of computer networks participation in high-speed communication" (grant of the Czech Ministry of Education, Youth and Sports)

Grant 1889/2007/F1/a „ Repair of digital exchanges education in the course Access and Transport Networks" (grant of the Czech Ministry of Education, Youth and Sports)

No MSM 0021630513 Research design of Brno University of Technology " Electronic communication systems and new generation of technology (ELKOM)"

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)

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] 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.

[8] VESTENICKY,P.: Optimazition of Selected RFID System Parameters. In Proceedings of AEEE 3, Vol2, pp. 113-114, 2004

[9] KRBILOVA,I.-VESTENICKY,P.: Use of Intelligent Network Services. In Proc. Of ITS, RTT, CTU Prague, 2004