Analysis of Algorithms for Radial Basis Function Neural Network

Jiri Stastny[1], Vladislav Skorpil[2]

[1] Department of Automation and Computer Science,
[2] Department of Telekommunications,
Brno University of Technology,
Purkynova 118, 612 00 Brno, Czech Republic,
stastny@fme.vutbr.cz, skorpil@feec.vutbr.cz

Abstract. This paper describes the analysis of algorithms for the hidden layer construction of network and for learning of the Radial Basis Function neural Network (RBFN). We compared results obtained by using of learning algorithms LMS (Least Mean Square) and Gradient Algorithms (GA) and results are obtained by using of algorithms APC-III and K-means for hidden layer contruction of neural network. The principles and algorithms given below have been used in an application for object classification that was developed at Brno University of Technology. This solution is suitable for the research of personal wireless communications and similar systems.

Keywords: Radial basis function, Learning algorithm, Neuron, Hidden layer.

1 Introduction

The Radial Basis Function Network (RBFN) belongs to the most recent neural networks. It is a type of single-direction multilayer network and forward multi-layer network with counter-propagation of signal. This 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 MLP (the Multi Layer Perceptron Neural Network) 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. Except for the input layer which only serves the purpose of handing over values an RBFN has a hidden layer (RBF) and an output layer is 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 neural network. Quicker learning is given by this, that RBFN has only two layers with weight and every layer may be determined sequential.

2 Creation of Hidden Layer RBFN

Problems at creation of RBFN consist on determination of the number of neurons in hidden layer, on determination of the middles of these neurones and on determination of the neurones width. Powerful method for determination of the number and quality of neurons of hidden layer is the algorithm APC-III. This single-pass associating algorithm unlike other uses constant radial.

2.1 Algorithm APC-III

C: the number of neurons
c_j: the middle of j-th neuron
n_j: the number of samples in j-th neuron
d_{ij}: the distance between x_i and j-th neuron

```
{
    C=1; c1 ← x1; n1 = 1;
    for(i = 2; i =< P; i++)              // for every pattern from training set
    {
        for(i = 1; i =< C; i++)          // for every neuron
        {
            calculate dij;
            if(dij =< R0)                // insert xi into j-th neuron
            {
                cj = (cjnj + xi)/(nj + 1);
                nj = nj + 1;
                break;
            };
        };
        if(xi is not in any neuron)      // create new neuron
        {
            C = C + 1;
            cc ← xi;
            nc = 1;
        };
    };
```

2.2 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.

2.3 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.

Algorithm APC-III is very powerful at creation of hidden layer of neural network. That is why it is enough to use this algorithm on input training set only once. That is great advantage against K-means algorithm. In addition APC-III is headed to creation of reasonable of the number of neurons and to determination of the radial of neurons on basis of training set distribution.

3 Determination of the Width of the Area (of Neuron)

For hidden layer it is necessary to determine the width of the single areas. The transient neurons function uses this width for calculation. In created testing program are used two variants of the areas width determination.

a) all areas have the same width R_0

b) every neuron has its own width

3.1 Algorithm of Calculation of the Own Width of the Area (of Neuron)

Algorithm searches to every area the nearest one such, to this area belongs to the other pattern and at the same time it was the nearest to calculated area. By the help of distance of middles of these two areas is determined the width of the area like multiple of those distance. Like the multiplier is used the *distance coefficient*, which is adjusted in the programme and which specifies on how many per cent of those distance will be set-up the width of this area.

3.2 Pseudo-code of the Algorithm

Input: the distance coefficient of the neuron hidden layer
Output: the widths of neurons
rk: the distance coefficient
C: the number of neurons
c_j: the middle of j-th neuron
v_j: the pattern allocated to j-th neuron
D_{ij}: the distance between i-th and j-th neurons
dst: the minimal distance between i-th and j-th neurons
σ_j: the width of j-th neuron

```
{
    for(i = 1; i =< C; i++)     // for every neuron
    {
        for(j = 1; j =< C; j++)     // pro every neuron
        {
            if(i == j)
            {       // the distance of neuron to itself is zero
                    Dij = -1;
                    continue;
            };
            if(vi == vj
            {       // the distance of neuron with the same patterns is
        omitted
                    Dij = -1;
                    continue;
            };
            Dij = ||ci − cj||;
            Dji = Dij;
        };
    };
```

```
for(i = 1; i =< C; i++)     // for every neuron
{
        //  location of minimal distance will be performed
        //  it is necessary at calculation to ignore items, which have value –1
        dst = min_{j=1..c}(D_ij);
        // we calculate the width of neuron
        σ_i = dst(rk/100);
};
};
```

The distances between middles of areas are placed into the matrix of distances. This matrix must have on diagonal zero distances, that have be good for better filtering out on the minimization substituted by the value -1. Likewise the distances among the neurons with the same associated pattern are eliminated and in lieu of distance is into the matrix inserted the value -1. The distance coefficient is inserted in percents (in front of using is divided by 100).

Connection of hidden layer to output layer of neurons is the last step on network creation. These layers are connected by the system every with everyone from the other layer. The programme then randomly sets scales and thresholds of outgoing layer of the neural network. Thereby it is creation of the Radial Basis Function Network (RBFN) finished. The next phase is learning of neuron network.

4 Learning of RBFN

The learning process consist on precept of given network to answer correctly to engaged training set. As the hidden layer was in this network represented so-called areas and the middles of the areas are fast given to it, the learning process oversimplifies only to setting of scales and thresholds of the output layer. Gradient method and Last Mean Square (LMS) method were tested for learning of neuron network.

The gradient method (GA) uses relations derived for outgoing layer for algorithm Back-propagation (BP). On difference from the method BP this method optimizes only scales and thresholds of outgoing layer.

Method Least Mean Square (LMS) tries to find optimal scaled vector for general middle quadratic error of the network. This scaled vector is given by normal division:

$$w = \left(H^T H\right)^{-1} H^T y$$

where **w** is scale vector, H is suggestion matrix $H_{ij} = h_j\left(x_i\right)$ and **y** is vector of outgoing values. This method in contrast to of others ones uses like transient function of outgoing neurons layer in lieu of sigmoid the linear function.

5 Programme Solution, Parameter Setting

At learning programme requires setting of following values:

a) The values common for all types of learning

- *The number of iteration on one etalon:* designates how many times will be submitted pattern set on learning of the network on the input. The optimal value for given to application was 5000.
- *The diagnosis accuracy:* designates, to what degree of accuracy the network will learn and subsequently will also diagnostic presented patterns. At setting of high value network will diagnostic high accurate, it is so errors in recognition will be almost zero, but the disadvantage will be inability of the network to generalize, which makes no-diagnosis of the damaged patterns. On the contrary at setting of low value the error count will grow at realising. The recommended value is 80%.
- *The minimal error:* past its over-fulfilment, i.e. if the error of learned network will be less than minimal error, reaches to stopping of calculation before reaching of given of the number of iterations. This election is recommended to use only while using of a large number of iterations.
- *The method of determination of the number of neurons:* here it is possible to choose the method of creation of hidden layer. The recommended setting is algorithm APC-III.
- *The coefficient* α*:* **t**his coefficient is the main creation parameter for RBFN network. The coefficient is exploited to calculation of the radius R_0, by the help of it subsequently algorithm APC-III.
- *Option of the radius:* this option causes, so after creation of hidden layer oneself its own width calculates for every neuron in this layer.
- *The coefficient of the distance of the middles:* designates in percents how many from the distances of neurons will be the width of neuron.

b) The values of parameter for gradient method

- *The learning speed:* set the step size degree of learning. The recommended value is 0.5.
- *The adaptive step of learning:* it is possible to set adaptive (variable) step of learning
- *The maximum step:* designates the initial step of learning of the neural network. The recommended value is to the extend of 1 until 1.5.
- *The exponent of the curve of the learning step:* designates by the help of what curve will be the actual step of learning extracted. Whereby higher number, the speed of learning will be thereby moved towards finding of learning process, where it will decline very quickly.

The number of outgoing neurons forms dynamically, according to the number of learned patterns. The outgoing vector is after rounding binary and indicates the class of classified subject. For example at the number of classes 5 can the outgoing vector shapes $[1,0,0,0,0]$, which indicates first-class inside of order.

The values of all scales and thresholds on the outgoing layer are at the beginning randomly initialized to values from the interval $\langle -0.1;0.1 \rangle$.

The current total mean square error and information on activities, that the programme just performs, displays along the learning.

6 Conclusion

The LMS learning method quickly navigates to aim. In contrast to the gradient method it needs only one iteration for all patterns together. LMS has also better characteristics on downsizing of a number of neurons in the hidden layer. The gradient method is not very suitable for learning of this network (in light of learning time). Learning of LMS is thanks to using algorithm APC-III reduced to learning of outgoing layer. The best results arranged the neuron network RBFN with the algorithm APC-III and with the learning method LMS (Least Mean Square).
Radial Basis Function networks can be designed very quickly. The time necessary for network learning was very little. 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 MLP networks. 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.

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. Miehie, D., Spiegelhalter, D. J., Taylor, C. C.: *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, NY, 1994.

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. De Juan, Ch.: Contour Recognition Problem, [online]. 2001. Dostupné z: <www.cc.gatech.edu/classes/ay2000/cs7495_fall/participants/cnd/ps3/ps3.html

4. Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addisson-Wesley, 1989.

5. Nilsson, N. J.: *Principles of Artificial Intelligence*. Springer Verlag, Berlin, 1982.

6. Ripley, B. D.: *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge (United Kingdom), 1996.

7. Pavlidis, T.: *Algorithms for Graphics and Image Processing.* Bell Laboratories, Computer Science Press, 1982.

8. 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

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. Šnorek, M., Jiřina, M.: *Neuronové sítě a neuropočítače*. ČVUT, Praha,1998. ISBN 80-01-01455-X.

11. Šťastný, J., Škorpil, V.: *Neural Networks Learning Methods Comparison*. International Journal WSEAS Transactions on on Circuits and Systems, Issue 4, Volume 4, April 2005, ISSN 1109-2734, pp. 325-330.

12. Vestenický,P. The Prediction Properties of Kalman Filter in Proceedings of TRANSCOM '95. UT&C, Zilina, pp. 243-248, 1995

13. Krbilová, I. and Vestenický,P.: Forgalomszabályozásés szolgáltatásminöség" Magyar távközlés 7, Vol.. 6/96, pp. 32-33, 1996

14. Vestenický, P. The Functions of ATM Interfaces in Proc. of DDECS '97 Conference Proceedings. VSB Technical University, Ostrava, pp. 186-191, 1997

15. Bubeníková, E. and Vestenický, P. Principles Of The Intranet Information System Creation in Proc. of ELEKTRO'99 Conference Proceedings, section Information & Safety Systems. University of Žilina, pp. 77-81, 1999

16. Vestenický, P. Optimization of Selected RFID System Parameters in Proc. of. AEEE 3, Vol. 2, pp. 113-114, 2004

17. Krbilová, I. and Vestenický, P. Use of Intelligent Network Services in Proc. of ITS. RTT , CTU Prague, 2004

18. Vestenický, M. and Vestenický, P. Evolutionary Algorithms in Design of Switched Capacitors Circuit in Proc. of International Workshop „Digital Technologies 2004“. Slovak Electrical Society and University of Zilina, pp.34-37, 2004