Enhanced Error Correction Algorithm for RBF Neural Networks

Pawel Rozycki and Janusz Kolbusz

University of Information Technology and Management in Rzeszow,
{prozycki,jkolbusz}@wsiz.rzeszow.pl

Abstract. Using RBF units in neural networks are very interesting option that make network more powerful. The paper presents new training algorithm based on second order ErrCor algorithm. The effectiveness of proposed algorithm has been confirmed by several experiments.

Key words: Error Correction, ErrCor, RBF networks, training algorithms

1 Introduction

The rapid development of intelligent computational systems allowed to solve thousands of practical problems using neural networks. Major achievements have been made mainly using architecture MLP (*Multi-Layer Perceptron*), but it turns out that it is also possible with other neural network architectures. Although EBP (*Error Back Propagation*) [1] caused a real breakthrough, it turned out to be a very slow algorithm, not capable of learning other than MLP, compact network architectures. Most visible progress in this field was develop the LM (*Levenberg-Marquardt*) algorithm to train the neural network. This algorithm is able to teach the network by 100 to 1000 times less iterations, but its usage to more complex problems is significantly limited, since the size of the Jacobian matrix is proportional to the number of patterns.

In order to solve more and more complex problems with the use of neuron networks we should thoroughly understand the neural network architecture and its impact on the operation of the system and finally develop appropriate processes of learning these networks. Modification of existing algorithms and development of new algorithms for network learning will allow for faster and more effective network teaching.

Often used networks MLP have limited capabilities[1], but new neural network architectures like BMLP (*Bridged MLP*) [1,2] or DNN (*Dual Neutral Networks*) [2] with the same number of neurons can solve problems up to 100 times more complex [2,3]. Therefore, it can be concluded that the way neurons interconnections in the network is fundamental.

The use of appropriate architecture has a significant impact on the solution of given problem. An example can be FCC (*Fully Connected Cascade*) network architecture. Such a network with 10 neurons can solve the Parity-1023 problem, while the most widely used the MLP architecture network with 10 neurons in the three-tiered, one hidden layer, architecture, is able to solve Parity-9 problem. Thus, moving away from the commonly used architecture MLP, while maintaining the same number of neurons

can increase network capacity, even a hundred times. [2-4]. However, a problem arises in that the currently known network learning algorithms, such as EBP [5], or LM do not deal with such network architectures. LM algorithm is not able to teach other architectures than the MLP, because the size of Jacobian, which must be processed is proportional to the number patterns of learning, which limits LM algorithm for solving network learning a relatively small problems. The only known algorithm that can learn these new architectures is NBN algorithm (Neuron-by-Neuron) [6-8]. It is faster than LM and can be used for all architectures, including BMLP, FCC DNN and MLP, of course, and gives good learning results. However, published in 2012 ISO algorithm [9] and published in 2014 ErrCor (*Error Correction*) algorithm [10].allow to get even better results

2 Enhanced Error Correction Algorithm

2.1 Error Correction Fundamentals

Error Correction is second order LM based algorithm that has been designed for RBF networks where as neurons RBF units with Gaussian activation function defined by (1) are used.

$$\varphi_h(x_p) = exp\left(-\frac{\|x_p - c_h\|^2}{\sigma_h}\right) \tag{1}$$

where: c_h and σ_h are the center and width of RBF unit *h*, respectively. $\|\cdot\|$ represents the computation of Euclidean Norm. The output of such network is given by:

$$O_p = \sum_{h=1}^{H} w_h \varphi_h(x_p) + w_o \tag{2}$$

where: w_h presents the weight on the connection between RBF unit *h* and network output. w_0 is the bias weight of output unit. Note that the RBF networks can be implemented using neurons with sigmoid activation function in MLP architecture [11,12]. The main idea of the ErrCor algorithm is increasing the number of RBF units one by one and adjusting all RBF units in networkby training after adding of each unit. New unit is initially set to compensate largest error in the current error surface and after that all units are trained changing both centers and widths as well as output weights. Details of algorithm can be found in [10].

As can be found in [10, 13] ErrCor algorithm had been successfully used to solve several problems like function approximation, classification or forecasting. The main disadvantage of ErrCor algorithm is long computation time caused mainly by requirement of training of whole network at each iteration.

2.2 Enhanced ErrCor

Long computation time depends on many factors. One of the most important is number of patterns used in training and long training of whole network after adding of next

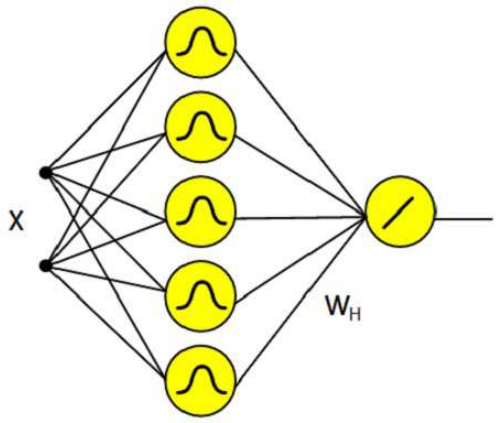

Fig. 1. RBF network architecture

RBF unit. In order of improve this process we suggest the following modifica-tions of ErrCor algorithm:

- after adding new RBF unit only this new unit is trained using LM-based method used in ErrCor algorithm [10] and after that all output weights are justified using regression;
- after added N new RBF units whole network is trained using the same LM-based method used in ErrCor algorithm where N is arbitrary assigned value.

Such modification allow to shortened training process because critical whole training process is limited to cases when N new units are added to network. In the other cases the training is much faster because in fact trained is only one RBF unit and regression is quite small absorbing process.

Pseudo code of the enhanced ErrCor algorithm is shown below. Changes to original ErrCor algorithm [10] are bolded.

Enhanced ErrCor pseudo code

```
evaluate error of each pattern;
while 1
    C = pattern with biggest error;
    add a new RBF unit with center = C;
    if N new RBF units are added
        train the whole network using LM-based method;
    else
        train only one new added RBF unit using LM-based method;
        adjust output weights for whole network by regression
    end
    evaluate error of each pattern;
```

```
    calculate SSE = Sum of Squared Errors;
    if SSE < desired SSE
           break;
    end;
end
```

In the next section some experimental results for this approach is presented.

3 Experiments Results

To confirm suggested approach several experiments for different approximation benchmark functions and training parameters have been prepared. The following functions have been selected: Peaks function, Second Shaffer function and Shwefel function. In the next three subsections the ErrCor algorithm and the Enhanced ErrCor algorithm have been used to solve approximation problem of mentioned functions. In all experiments 900 training patterns and 3481 testing patterns have been generated. For such prepared data series experiments have been done with different values of parameter N and compared to results achieved using original ErrCor algorithm. To prepare experiments Matlab 2009b software with Windows 7 64 on Intel Core i5-M560 CPU and 8GB platform was used.

3.1 Shwefel Function

First experiment was prepared for Shwefel function given by

$$z\left(x,y\right)=2*418.9829-xsin\left(\sqrt{|x|}\right)-ysin\left(\sqrt{|y|}\right) \tag{3}$$

shown in Figure 2.

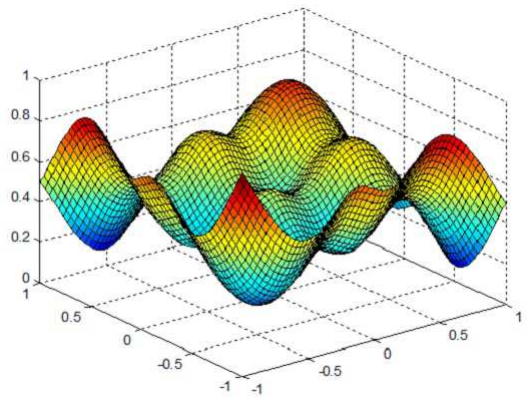

Fig. 2. Surface of normalized Shwefel function.

Results achieved for Shwefel function are shown in Table 3. Result for original ErrCor that can be treated as a reference is denoted as OrgErrCor. Parameter N means the number of units that are added to network between full training. The case when training process is done without full network training is denoted as X in column N. The RMSE is Root Mean Square Error given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(out_T - out_E\right)^2}{n}} \tag{4}$$

where out_T is the output of trained network and out_E is expected value and *n* is the number of patterns.

Table 1. Results for Shwefel function.

N	***Training time***	***Training RMSE***	***Testing RMSE***
OrgErrCor	246.6722	0.0038543	0.0038299
2	215.7920	0.0032388	0.0031717
3	148.7899	0.0030517	0.0029844
4	88.0451	0.0052154	0.0051204
5	75.5889	0.0053222	0.0051949
6	59.7885	0.0052533	0.0051889
7	51.9710	0.0057257	0.0057030
8	66.9600	0.0079042	0.0077081
9	60.3580	0.0054597	0.0053507
10	51.2228	0.0059679	0.0058742
15	43.1061	0.0105723	0.0104105
30	36.2405	0.0497498	0.0498103
X	21.1715	0.0747471	0.0732735

As shown in Table 1 training time decreases with increased value of N. This is ob-vious because frequency of full training, that is the most time consuming part of training process is smaller for higher N. More important is that values of testing and training RMSE for small values of N (2 and 3) are better than these achieved with original ErrCor, and for higher value of N are only slightly worse. Note that results for N=10 are only 53% worse but achieved almost 5 times faster.

3.2 Second Shaffer Function

The second experiment have been done for Second Shaffer function. This function is given by

$$z\left(x, y\right) = 0.5 + \frac{sin^2\left(x^2 - y^2\right) - 0.5}{\left[1 + 0.001\left(x^2 - y^2\right)\right]^2} \tag{5}$$

shown in Figure 3.

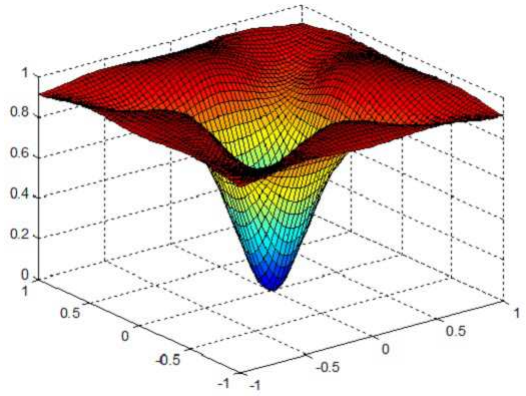

Fig. 3. Surface of normalized Second Shaffer function.

Results achieved with Enhanced Error Correction algorithm is shown in Table 2. Similarly like for Shwefel function training time decreases with N while RMSE is relatively are close to or even lower than for original ErrCor.

Figure 4 show the training process that show changes of training and testing RMSE during training process. Training RMSE is drown as blue circles and testing RMSE is drawn as a red stars. As can be observed in the case of Enhanced ErrCor with N = 2 reaches similar result as ErrCor but is able to obtain same results faster and using less neurons.

Table 2. Results for Second Shaffer function.

N	***Training time***	***Training RMSE***	***Testing RMSE***
OrgErrCor	262.1465	0.0008487	0.0046744
2	160.7057	0.0005260	0.0046379
3	140.3324	0.0005305	0.0046374
4	123.2476	0.0008320	0.0051382
5	106.8903	0.0014932	0.0047826
6	50.9329	0.0018604	0.0049701
7	44.1457	0.0011056	0.0047119
8	43.9627	0.0010136	0.0046913
9	28.2079	0.0018110	0.0049119
10	57.9185	0.0010418	0.0047068
15	21.7188	0.0017827	0.0049438
30	17.5358	0.0093627	0.0103736
X	14.4285	0.0093627	0.0103736

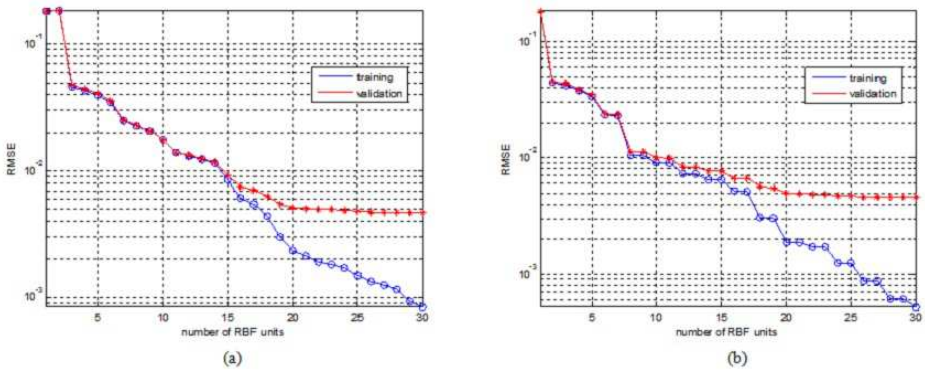

Fig. 4. Training process for approximation of Second Shaffer function with: (a) original ErrCor algorithm, (b) Enhanced ErrCor (N=2)

3.3 Peaks Function

The last experiment with described Enhanced Error Correction algorithm has been used for approximation of Peaks function given by:

$$\begin{aligned} z(x,y) = &-\tfrac{1}{30}\ e^{\left(-1-6x-9x^2-9y^2\right)}+ \\ &-\left(0.6x-27x^3-243y^5\right)\ e^{\left(-9x^2-9y^2\right)} \\ &+\left(0.3-1.8x+2.7x^2\right)\ e^{\left(-1-6y-9x^2-9x^2\right)} \end{aligned} \tag{6}$$

and shown in Figure 5.

Results achieved for this function in the same way like for previous functions are shown in Table 3. Unfortunately, they are not so clear like for previous functions. While training time decreases with N in the same time RMSE values increase.

Examples of training process by original ErrCor and Enhanced ErrCor with N=3 is presented in Figure 6. As can be observed the full network training is seen as a rapid RMSE decreasing while adding and training of one RBF unit initially produces similar effect but later does not decrease RMSE. In the case when N value is higher than maximal number of units in the network the training is limited to adding new units and training then one-by-one without full network training. Such training process is shown in Figure 7. Note that starting from 14^{th} unit added to network RMSE values are not decreasing. This is because each new unit added to network is localized ac-cording to the pattern with the highest error and in these case each new unit, starting from 14^{th}, is initially localized in the same place.

4 Conclusions

Achieved results confirm effectiveness of suggested method for improvement Er-ror Correction algorithm that is currently one of the most powerful for training RBF networks. Proposed modification allows to reduce training time in most cases without

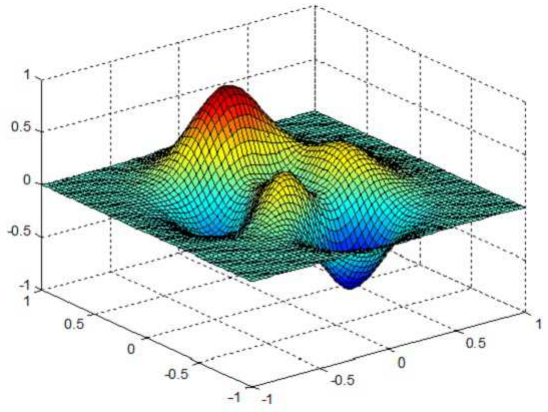

Fig. 5. Surface of Peaks function.

Table 3. Results for Peaks function.

N	***Training time***	***Training RMSE***	***Testing RMSE***
OrgErrCor	710.2437	0.0000071	0.0000071
2	418.5150	0.0000234	0.0000234
3	255.5103	0.0000580	0.0000580
4	137.8621	0.0001049	0.0024100
5	75.8098	0.0015600	0.0018230
6	72.7661	0.0079510	0.0080396
7	136.9426	0.0001695	0.0001710
8	98.7397	0.0024247	0.0024510
9	28.9673	0.0191530	0.0231557
10	25.4868	0.0107400	0.0108643
15	54.5841	0.0014544	0.0014681
30	29.8234	0.0464481	0.0470313
X	17.1136	0.0464481	0.0470313

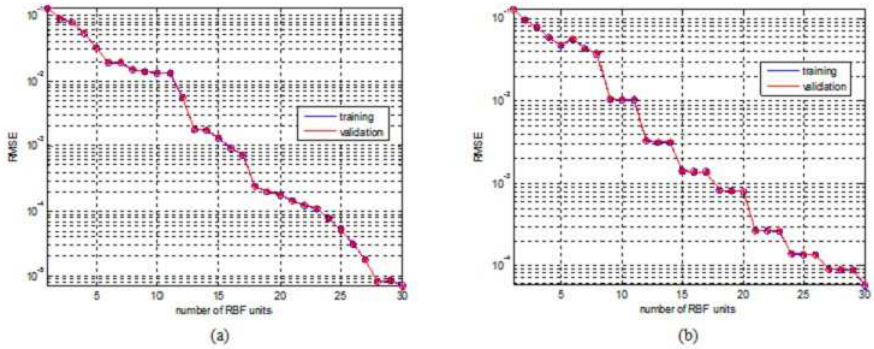

Fig. 6. Training process for approximation of Peaks function with: (a) original ErrCor algorithm, (b) Enhanced ErrCor (N=3).

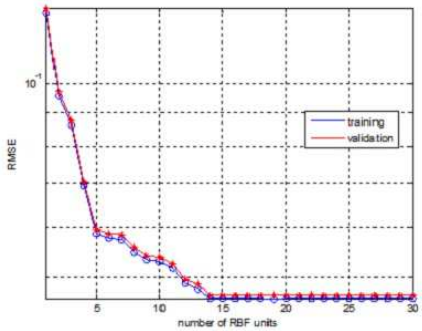

Fig. 7. Training process for approximation of Peaks function with Enhanced ErrCor without full network training.

losses of low training and testing errors. Further work will be focused on improvement of proposed algorithm by correction of method for selection of initial localization for new RBF units and on applying described algorithm for wider spectrum of functions and real world classification datasets from UCI Machine Learning Repository.

References

1. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533-536, 1986.
2. S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture". In D. S. Touretzky (ed.) Advances in Neural Information Processing Systems 2 . Morgan Kaufmann, San Mateo, CA, 1990, pp. 524-532
3. K. L. Lang, M.J. Witbrock, "Learning to Tell Two Spirals Apart". Proceedings of the 1988 Connectionists Models Summer School, Morgan Kaufman.
4. B. M.Wilamowski, "Challenges in Applications of Computational Intelligence in Industrial Electronics", IEEE International Symposium on Industrial Electronics (ISIE 2010), Jul 04-07, 2010, pp. 15-22.
5. Y. Bengio, "Learning deep architectures for AI". Foundations and Trends in Machine Learning, 2(1), 1-127. Also published as a book. Now Publishers, 2009.
6. D. C. Ciresan, U. Meier, L.M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition", CoRR, 2010.
7. B. M. Wilamowski and H. Yu, "Neural Network Learning Without Backpropagation," IEEE Trans. on Neural Networks, vol. 21, no.11, pp1793-1803, Nov. 2010.
8. P. J. Werbos, "Back-propagation: Past and Future". Proceeding of International Conference on Neural Networks, San Diego, CA, 1, 343-354, 1988.
9. H. Yu, T. Xie, J. Hewlett, P. Rozycki, B. Wilamowski, "Fast and Efficient Second Order Method for Training Radial Basis Function Networks", IEEE Transactions on Neural Networks, 2012, Vol. 24, Issue: 4, pp. 609-619
10. H. Yu, P. Reiner, T. Xie, T. Bartczak, B. Wilamowski, "An Incremental Design of Radial Basis Function Networks", IEEE Transactions on Neural Networks and Learning Systems, vol 25, No. 10, Oct 2014, pp. 1793-1803.
11. B. M. Wilamowski, R. C. Jaeger, "Implementation of RBF type networks by MLP networks", IEEE International Conference on Neural Networks (ICNN 96), pp. 1670-1675
12. X. Wu, B.M. Wilamowski "Advantage analysis of sigmoid based RBF networks". In: Proceedings of the 17th IEEE International Conference on Intelligent Engineering Sys-tems (INES'13). 2013. p. 243-248.
13. C. Cecati, J. Kolbusz, P. Rozycki, P. Siano, B. Wilamowski, "A Novel RBF Training Algorithm for Short-Term Electric Load Forecasting and Comparative Studies", IEEE Trans. on Ind. Electronics, Eearly Access, 2015.