

# Non-smooth regularization in radial artificial neural networks

V N Krutikov<sup>1</sup>, L A Kazakovtsev<sup>2</sup> and V L Kazakovtsev<sup>3,4</sup>

<sup>1</sup> Kemerovo State University, 6 Krasnaya street, Kemerovo, 650000, Russia

<sup>2</sup> Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russia

<sup>3</sup> Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, 49, Kronverksky Av., Saint Petersburg, 197101, Russia

<sup>4</sup> E-mail: levk@bk.ru

**Abstract.** We propose a new approach to the approximation of an arbitrary function by the sum of radial basis functions using non-smooth regularization method combined with a new algorithm in determining the initial approximation which ensures an acceptable quality of approximation uniformly over the data area.

## 1. Introduction

Construction of artificial neural networks (ANN) [1-3] is a dynamically developing field of the computational intelligence theory. The most important ANN applications [4] are: approximation [5, 6], classification, pattern recognition [7], forecasting [8], identification and evaluation etc. Often, we can use approximal formulation [9] of the problem for some simulation, identification and signal processing problems. In this case, the ANN can be a universal approximator [5] of a function of several variables that realizes a non-linear function  $y = f(x)$ , where  $x$  is the input vector and  $y$  is the realized function. A.N. Kolmogorov [10] obtained theoretical results on the representation of continuous functions of several variables in a form of a superposition of continuous functions. In this paper, we present efficient algorithms for training neural networks with neurons in the form of radial basis functions (RBF) [11-13].

In radial networks (RBF networks), [11] hidden neurons realize radial basis functions  $\varphi(x) = \varphi(\|x - c\|)$ . They vary radially around the selected center  $c$  and taking non-zero values only in the vicinity of this center [14]. A role of the hidden neuron is to reflect the radial space around a single given point or cluster of points. The superposition of signals coming from all hidden neurons, which is performed by the output neuron, allows us to obtain a reflection of the whole multidimensional space. The mathematical basis of the functioning of radial networks is Cover's theorem on the recognizability of images [15]. The structure of a typical radial network includes an input layer to which signals described by the input vector  $x$  are fed, a hidden layer with neurons of radial type and an output layer consisting of one or more linear neurons. So, the function of the output neuron is the weighted summation of hidden neuron signals.

Having an excessive number of neurons, it is possible to reflect the data being approximated with an arbitrarily high accuracy by an interpolating RBF network [16]. However, the network becomes sensitive to interference and irregularities accompanying the training samples and has low generalizing properties. To increase the generalizability of the network, the number of neurons should be reduced



When we consider the problem of approximation of an arbitrary function by the RBF network, there are following issues: specifying the correct initial values of the parameters of radial functions [1, 13], determining the number of neurons [1] needed for approximation and the learning rate of the network. If the initial values of the centers of the radial functions are incorrectly determined, clusters of neurons maybe formed. Such clusters remain in the further training of the network, and part of the domain of definition of the approximated function may remain uncovered by the neurons. This will lead to a deterioration in the approximating properties of the network. This is typical even for a uniform initial distribution of neurons [1].

Despite the mentioned difficulties, the RBF networks can be applied in various fields, for example, in the solution of the equations of mathematical physics [2, 15]. In [2,17], to eliminate the problems described above, authors use restrictions on the changes in the parameters of neurons at the learning stages. This leads to a significant decrease in the convergence rate and the inability to implement the processes of reducing the mathematical description of the network (for example, by removing redundant neurons). The other problem of this method is the restriction on the use of algorithms: it is not possible to apply the set of fast-converging minimization methods [12].

In this paper, we propose an algorithm for finding the initial approximation of a RBF network with an excessive number of neurons, such that an acceptable initial quality of approximation is provided uniformly over the data range. Moreover, the quality of the approximation in certain data ranges does not deteriorate with further training. As well as the network training procedure using regularization for the displacement of redundant neurons and protection from retraining.

We choose the Gauss function with the diagonal covariance matrix as a function of activation of the neuron. The RBF network centers are selected by some clustering algorithm. This makes it possible to place the neurons evenly across the entire data area. This idea is not new [1]. We suggest determining the network parameters at fixed centers using the usual least squares method in the next step of the search for an initial approximation. At the very first stage, with the use of regularization, this allows us to build a network with an acceptable initial approximation uniformly over the data area, even in the case of an excessive number of network parameters compared to the number of data vectors. This stage ensures that during the subsequent training of the network, after removal of the fixation on the parameters of the centers, all data areas will remain covered by neurons, and their movement will only improve the quality of the approximation.

We offer the basic procedure for learning the RBF network built on the principle of sequential learning tasks with non-smooth regularization, which allows us to suppress excess neurons and smooth out the consequences of redundant network description. After each cycle of training, we removed neurons having little effect on the quality of the approximation. We chose the final model from a set of obtained models with the number of parameters that is less than the number of data vectors by the criterion of minimum mean square error.

We used a relaxation subgradient method with a double-trace correction of the metric matrices (SMDM) [18] to train the network. It is equivalent to N. Z. Shor's  $r$ -algorithm [19] when we excluding the space compression operation. These class methods [20, 21], as shown in [22], can be as fast-converging as quasi-Newtonians. However, they are also efficient for minimizing non-smooth functions which is necessary for implementing the studied ANN training schemes. The advantage of subgradient methods is the use of rough one-dimensional minimization in the iteration. This does not lead to shortening of the descent steps and thus positively affects the overcoming of small local minima and the absence of premature stops of the algorithm due to the insensitivity of the function to small changes in parameters at small search steps.

## 2. The approximation by an artificial neural RBF network problem

Let us consider an approximation problem:

$$w^* = \arg \min_w E(\alpha, w, D), \quad (1)$$

$$E(\alpha, w, D) = \sum_{x, y \in D} (y - f(x, w))^2 + \sum_{i=1}^k \alpha_i R_i(w),$$

where  $D = \{(x^i, y_i) \mid x^i \in R^p, y_i \in R^1, i = 1, \dots, N\}$  are observational data,  $R_i(w)$  are different types of regularizes,  $\alpha_i$  are regularize parameters,  $f(x, w)$  is the approximating function,  $x \in R^p$  is the data vector,  $w \in R^n$  is the vector of adjustable parameters,  $p$  and  $n$  are its dimensions. In solving such problems various methods of regularization are used which are closely related to the form of the approximating function.

The following minimization algorithm will be used in this paper to solve the problem (1). Let it be required to solve the problem of minimizing a differentiable function  $f(x)$ ,  $x \in R^n$ , where  $R^n$  is finite-dimensional Euclidean space. Let  $(x, y)$  be the scalar product of vectors,  $\|x\| = \sqrt{(x, x)}$  be the vector norm. For an arbitrary symmetric strictly positive definite matrix  $H$  of size  $n \times n$ , we use designation  $H > 0$ .

The successive approximations of the subgradient SMDM algorithm [18] on some iteration  $k$  for an exact one-dimensional descent are constructed by the equations:  $x_{k+1} = x_k + \gamma_k r_k$ ,  $r_k = -H_k g_k$ ,  $H_k > 0$ ,  $\gamma_k = \arg \min_{\gamma > 0} f(x_k + \gamma r_k)$ . Here,  $x_0$  is the specified initial point,  $H_0 > 0$  is specified initial matrix and matrices  $H_k > 0$  are calculated by formulas:

$$H_{k+1} = H_k - (1 - \frac{1}{\alpha^2}) \frac{H_k y_k y_k^T H_k^T}{(y_k, H_k y_k)} - (1 - \frac{1}{\beta^2}) \frac{H_k p_k p_k^T H_k^T}{(p_k, H_k p_k)},$$

$$y_k = g_{k+1} - g_k, p_k = g_{k+1} + t_k y_k, t_k = \frac{(H_k y_k, g_{k+1})}{(p_k, H_k p_k)},$$

$\alpha > 1$ ,  $\beta \in (0, 1]$ ,  $\alpha \cdot \beta > 1$ , where coefficient  $t_k$  is calculated from the orthogonality condition  $(y_k, H_k p_k) = 0$ . Here and below,  $g$ ,  $g(x)$  is a subgradient from the subgradient set  $\partial f(x)$  of function  $f(x)$ ,  $g_k = g(x_k)$ .

The presented algorithm is similar to quasi-Newtonian methods [22, 24]. It allows us to solve the problems of smooth and non-smooth optimization efficiently and we will use it in the scheme of constructing the RBF network.

In the approximation problem, the RBF network requires to train a neural network of the following form according to data  $D$  (to estimate its unknown parameters  $w$ )

$$f(x, w) = w_0^{(2)} + \sum_{i=1}^m w_i^{(2)} \varphi(s_i), \quad (2)$$

$$\varphi(s_i) = \exp(-s_i), \quad (3)$$

$$s_i = \sum_{j=1}^p (w_{ij}^{(1)} (x_j - w_{ij}^{(0)}))^2, \quad i = 1, 2, \dots, m$$

where  $x_j$  are components of vector  $x \in R^p$ ,  $w = ((w_i^{(2)}, i = 0, \dots, m), (w_{ij}^{(1)}, j = 0, \dots, p, i = 1, \dots, m), (w_{ij}^{(0)}, j = 0, \dots, p, i = 1, \dots, m))$  is a set of

unknown parameters which must be estimated by the least squares method (1),  $\varphi(s)$  is a function of neuron activation and  $m$  is number of neurons. Set of parameters  $(w_{ij}^{(0)}, j = 0, \dots, p, i = 1, \dots, m)$  is a set of centers of neurons. The other set  $(w_{ij}^{(1)}, j = 0, \dots, p, i = 1, \dots, m)$  is used to scale variables. Here, scaling is used in the numerator, not in the denominator, unlike the standard RBF network entry. This prevents the issues of dividing by zero and the effects of denominator tends to zero. To solve the optimization problems (1), we use the relaxation subgradient method of SMDM.

Regularization method of Tikhonov [25] was proposed in 1963. The method of ridge regression was proposed as a further development by A. Hoerl and R. Kennard [26]. The stability of the obtained solutions has been improved by introducing a regularizing term in the minimized functional [27]. The main idea of regularization is to stabilize the solution with the help of some auxiliary non-negative function that carries a priori information about the solution. The most general form of a priori information is the assumption of the smoothness of the function of the desired mapping (the solution of the reconstruction problem) in the sense that the same input signal corresponds to the same output.

In the case of approximation, the linear model uses the Tikhonov regularizer [25]  $R_2(w) = \sum_{i=1}^n w_i^2$ . There are parameters of the linear part of the model. In [28] a non-smooth regularizer  $R_\gamma(w) = \sum_{i=1}^n (|w_i| + \varepsilon)^\gamma$  ( $\varepsilon = 10^{-6}$ ,  $\gamma = 0.7$ ) was used. It leads to the suppression of the small components of the vector  $w$  when we use  $R_\gamma$ . This property of  $R_\gamma$  allows to reduce weak components to zero, unessential for the description of data. In problem (1), we will use regularization to suppress non-essential for describing the neuron elements of the network:  $R_\gamma(w) = \sum_{i=1}^m (|w_i^{(2)}| + \varepsilon)^\gamma$  ( $\varepsilon = 10^{-6}$ ,  $\gamma = 0.7$ ).

Regularization  $R_\gamma(w)$  is turned out to be quite efficient for small dimensionality of data and the condition that the number of neurons and their location corresponds to the complexity of the function being approximated. The addition of  $R_\gamma(w)$  leads to a decrease in the coefficients of neurons, which automatically causes the expansion of the working areas of the neurons and their coordinated interaction.

### 3. Optimization algorithm for finding the initial approximation

The working regions of neurons  $\varphi(s)$  have the character of ellipsoidal clusters and vary only in some neighborhood of  $s = 0$ , and in other regions are close to their asymptotes. When the initial parameters in the minimization problem (1) are arbitrarily set, the working areas of neurons often cover only a part of the approximation region, or go beyond it, forming local minima, the exit from which cannot be performed by the methods of local changes of the current approximation. The location of neurons at points with a high concentration of data also does not ensure the preservation of this position, since with further training neurons can leave initially specified regions. Therefore, additional binding of the neuron working areas is required through the training of ANNs at fixed centers. In this case the neuron will be able to leave its region only if the region already has the accuracy of approaching the data.

In the next algorithm, it is proposed to find the ANN approximation. The parameters of the neurons with a fixed position of the neurons working regions with help of given centers  $c_i \in R^P$ ,  $i = 1, 2, \dots, m$ , in the approximation region  $x \in R^P$  which is determined by the data. In this case in (3) we use equation:

$$\varphi(s_i) = \exp(-s_i), \quad (4)$$

$$s_i = \sum_{j=1}^p (w_{ij}^{(1)}(x_j - c_{ij}))^2, \quad i = 1, 2, \dots, m,$$

where  $x_j$  are components of vector  $x \in R^P$  and parameters of network will be:  $w = ((w_i^{(2)}, i = 0, \dots, m), (w_{ij}^{(1)}, j = 0, \dots, p, i = 1, \dots, m))$ .

Centers  $c_i$  can be found by some clustering algorithms  $x^i \in R^P, i = 1, \dots, N$ . This is also useful for arranging neurons in areas with high data density. In this paper, we use the maximin algorithm [29] where the two most distant data points are selected as the first two centers. Every new center is chosen as data point  $x^i$ , distance from this point to the nearest known center is the maximum.

Algorithm for determining the initial approximation of ANN:

1. Set data  $D$ , number of neurons  $m < N$ . Choose regularizer and its parameters in (1).
2. Set the centers on working areas of neurons  $c_i \in R^P, i = 1, 2, \dots, m$  on data  $D$ .
3. Choose the initial approximation of ANN parameters (2-3) in accordance with (4).
4. Find unknown parameters by solving the problem (1) for ANN (2-3) in form (4).
5. Return to the original description of the network in the form (2-3) by entering parameters

$$\begin{aligned} w &= ((w_i^{(2)}, i = 0, \dots, m), \\ &(w_{ij}^{(1)}, j = 0, \dots, p, i = 1, \dots, m), \\ &(w_{ij}^{(0)} = c_{ij}, j = 0, \dots, p, i = 1, \dots, m)) \end{aligned} \quad (5)$$

Step 4 of the algorithm ensures that the data area will be covered by the working areas of neurons. In their allocated area, each neuron will provide some approximation quality, which is retained when returning (5) to the form (2-3) and can only improve with further training.

#### 4. Algorithm for ANN training in approximation problems

The technology of the non-smooth regularization method consists in the non-informative variables elimination of the linear model by means of a sequence of alternating steps in the ranking of features by means of non-smooth regularization in terms of their significance and removing the first insignificant features according to a certain criterion. To suppress excess variables, preliminary training should be done by minimizing  $w$  with a mean squared error using a non-smooth smoothing functional.

In case of small data dimension  $x \in R^P$ , for example in solving equations of mathematical physics [4, 13], we can focus on choosing the optimal number of neurons, removing redundant, not deleting variables inside neurons. In the following algorithm an excessive number of neurons is specified and regularization is performed only by parameters  $w_i^{(2)}, i = 1, \dots, m$ , from (2) using regularizer  $R\gamma$ .

Algorithm for training approximating ANN in the absence of interference.

1. Initialize data  $D$ , number of neurons  $m < N$ . Choose regularizer  $R\gamma(w)$  and parameter  $\alpha$  for SNP algorithm and training ANN algorithm (1).

2. Find the initial approximation of ANN  $W_0$  using the SNP algorithm.

3. For  $k=1, 2, \dots, m-1$  do:

- 3.1.  $w^k = \arg \min_w E(\alpha, w^{k-1}, D)$ . Calculate the value of the root-mean-square

$$\text{deviation } S_k = S(D, f_k) = \sum_{x, y \in D} (y - f(x, w^k))^2 / N.$$

- 3.2. Consistently, one by one to remove the neurons, providing a minimal value of  $S(D, f)$  after the removal, which does not exceed the value of  $S_k$  by more than a certain number of percentages.

3.3. If there was no removal of neurons in Step 3.2 then remove one of the neurons leading to the smallest increase in the indicator  $S(D, f)$ .

4. Choose ANN  $f(x, w^k)$  as the final model with  $n$  parameters. Here,  $n \leq N$ , and ANN has the lowest value of the index  $S_k$ .

## 5. Experimental results

The following functions were taken for numerical investigation of the obtained algorithm:

$$f_1 = 3(1 - x_1)^2 \exp(-x_1^2 - (x_2 + 1)^2) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \exp(-x_1^2 - x_2^2) - \frac{1}{3} \exp(-(x_1 + 1)^2 - x_2^2),$$

$$f_2 = \frac{1}{2} \sin(\pi x_1^2) \cdot \sin(2\pi x_2).$$

For function  $f_2$  testing variables vary in range  $0 \leq x_1 \leq 1$ ,  $0 \leq x_2 \leq 1$  and in range  $-3 \leq x_1 \leq 3$ ,  $-3 \leq x_2 \leq 3$  for function  $f_1$ . The choice of function  $f_2$  is explained by the problems in approximation of harmonic functions. Results are presented in Table 1. The results of approximation by known methods for comparison are taken from [1], approximation results of other functions were received using the Surfer 6.0 program, each time an algorithm was selected with the best approximation results. We used the following designations:  $\Delta$  is maximum deviation, we measure it on a test sample when  $N=1000$ .  $S = S(D, f_k) = \sum_{x, y \in D} (y - f(x, w^k))^2 / N$ . In Table 1,  $m$  is number of neurons,  $E = \sqrt{S}$  is standard deviation,  $N$  is number of initial data vectors.

**Table 1.** Results of computational experiments.

Function	Known results	Obtained result
$f_1$	$\Delta_0=0.06$ , $N=625$ , $m=36$	$\Delta_0=0.0035$ , $E=0.00061$ , $N=150$ , $m=19$
$f_2$	$\Delta_0=0.15$ , $N=500$ , $m=41$	$\Delta_0=0.013$ , $E=0.0016$ , $N=150$ , $m=28$

Here,  $\Delta_0$  is maximum deviation of approximating solution from the true value of the function,  $E$  is standard deviation,  $N$  is amount of training data,  $m$  is number of radial network neurons using for approximation.

The resulting algorithm can approximate functions of many variables, unlike the software package Surfer 6.0. The quality of approximation of the developed algorithm exceeds the known results for all functions. The quality of the approximation of the functions  $f_1$  and  $f_2$  exceeds the known results. Less data was used to train the network. The number of neurons necessary for approximation also turned out to be less than in known results.

## 6. Conclusion

In the process of network training, non-smooth regularization allows you to remove unnecessary network neurons, as well as expand the areas of approximation for each neuron. This allows each neuron to take its place in the domain of determining the approximated function and to exclude duplication of neurons improving the quality of the approximation. Comparison with known learning algorithms shows that the new algorithm provides a higher quality of approximation, even with a smaller amount of data.



## References

- [1] Osowski S 2000 *Sieci neuronowe do przetwarzaniain formacji* (Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej)
- [2] Schmidhuber J 2015 Deep Learning in Neural Networks: An Overview *Neural Networks* **61** 85–117
- [3] Haykin S 2016 *Neural networks* (Moscow: Williams)
- [4] Galushkin A I 2000 *Neuron networks theory* (Moscow: IPRJR)
- [5] Park J and Sandberg I W 1991 Universal Approximation Using Radial-Basis-Function Networks *Neural Computation* **3(2)** 246-57
- [6] Fasshauer G F 2007 *Meshfree Approximation Methods with MATLAB* (New Jersey: World Scientific Publishing Company)
- [7] Bishop C M 1996 *Neural Networks for Pattern Recognition* (Oxford: Oxford University Press)
- [8] Benghanem M 2010 *Al-Madinah Energy* **35** 3751-62
- [9] Wendland H 2010 *Scattered Data Approximation* (Cambridge: Cambridge University Press)
- [10] Kolmogorov A N 1958 On the functions of several variables representation in the form of a functions of fewer variables superposition *Mathematics, its teaching, applications and history* **2(3)** 41-61
- [11] Broomhead D S and Lowe D 1988 *Multivariable functional interpolation and adaptive networks Complex Systems* **2** 321-55
- [12] Xie T, Yu H, Hewlett J, Rozycki P and Wilamowski B 2012 Fast and Efficient Second-Order Method for Training Radial Basis Function Networks *IEEE transactions on neural networks and learning systems* **23(4)** 609-19
- [13] Gorbachenko V I and Jukov M V 2017 Boundary value problems of mathematical physics by means of radial basis functions networks *Journal of Computational Mathematics and Mathematical Physics* **57(1)** 133-43
- [14] Buhmann M D 2003 *Radial Basis Functions: Theory and Implementations* (Cambridge: Cambridge University)
- [15] Cover T 1965 *IEEE Trans. Electronic Computers* **14** 326-34
- [16] Yee P V and Haykin S 2001 *Regularized Radial Basis Function Networks* (JohnWiley)
- [17] Know O J and Bang S Y 1994 A design method of fault tolerant neural networks *Proc. ICONIP'94* pp 396-400
- [18] Krutikov V N and Gorskaya T A 2009 A set of relaxation subgradient methods with two-trace metric matrices correction *Economics and math methods* **45(4)** pp 37-80
- [19] Shor N Z *Methods for minimizing nondifferentiable functions and their applications* (Kiev: Naukovadumka) p 200
- [20] Krutikov V N and Vershin J N 2014 Subgradient method of minimization with correction of descent vectors on the basis of pairs of training relations *Vestnik KemGU* **1-1(57)** 46-54
- [21] Nurminskii E A and Tien D 2014 Method of Conjugate Subgradients with Constrained Memory *Automation and Remote Control* **75(4)** 646-56
- [22] Chen J S and Hu H Y 2008 Radial basis collocation method and quasi-newton iteration for nonlinear elliptic problems *Numer. Meth. for Partial Differential Equations* **24(3)** 991-1017
- [23] Krutikov V N and Petrova T V 2003 *Economics and math methods* **39(1)** 106-19
- [24] Dennis J E and Schnabel R B 1983 *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Englewood, NJ: Prentice-Hall)
- [25] Tihonov A N and Arsenin V 1986 *Methods for solving ill-posed problems* (Moscow: Nauka)
- [26] Hoerl A E and Kennard R W 1970 *Technometrics* **3(12)** 55
- [27] Bjorkstrom A 2001 *Ridge regression and inverse problems* (Stockholm: Stockholm University)
- [28] Krutikov V N and Arishev D V 2004 *Vestnik KemGU* **3(7)** 124-9
- [29] Gilev S E and Gorban A N 1996 On completeness of the class of functions computable by neural networks *Proc. of the World Congress on Neural Networks (WCNN'96)* (Sept 15-18 1996 San Diego, CA, Lawrens Erlbaum Accociates) pp 984-91