

Fuzzy C-Means Clustering Based Construction And Training For Second Order RBF Network

Kanishka Tyagi

Department of Electrical Engineering
The University of Texas at Arlington
Arlington, TX, USA, 76010
kanishka.tyagi@mavs.uta.edu

Xun Cai

School of Computer Science and Technology
Shandong University
Jinan, Shandong, P.R.China, 25010
caixunzh@sdu.edu.cn

Michael T. Manry

Department of Electrical Engineering
The University of Texas at Arlington
Arlington, TX, USA, 76010
manry@uta.edu

Abstract—The paper presents a novel two-step approach for constructing and training of optimally weighted Euclidean distance based Radial-Basis Function (RBF) neural network. Unlike other RBF learning algorithms, the proposed paradigms use Fuzzy C-means for initial clustering and optimal learning factors to train the network parameters (i.e. spread parameter and mean vector). We also introduce an optimized weighted Distance Measure (DM) to calculate the activation function. Newton's algorithm is proposed for obtaining multiple optimal learning factor for the network parameters (including weighted DM). Simulation results show that regardless of the input data dimension, the proposed algorithms are a significant improvement in terms of convergence speed, network size and generalization over conventional RBF models which use a single optimal learning factor. The generalization ability of the proposed algorithm is further substantiated by using k-fold validation.

Keywords- *Fuzzy-C means clustering, Hessian Matrix, Newton's Method, Optimal Learning Factor, Orthogonal Least Square.*

I. INTRODUCTION

The Radial Basis function (RBF) network is a three layer supervised feed-forward network [1] used in interpolation, probability density function estimation and approximation of smooth multivariate functions [2]-[7]. The RBF was first introduced as a solution to the real multivariate interpolation problem. The RBF model can approximate any multivariate continuous function arbitrarily well on a compact domain if a sufficient number of radial basis functions are given [4].

Fuzzy clustering has been used in RBF networks for many applications. The model presented in [8] uses it for Traffic status evaluation. Though it has been applied only to the simulated data and real traffic condition is still an "open" problem. Conditional Fuzzy Clustering in the design of RBF networks [9]. The idea in [9] is further extended by using supervised Fuzzy clustering to improve RBF performance in regression task [10]. In [11] Fuzzy Clustering is used for designing a RF network used in Modeling of Respiratory system.

A hybrid learning procedure is proposed in [12] which employs unsupervised clustering algorithm like k-means for determining the center for radial basis function and supervised learning for updating output weights connecting the radial basis

function unit (hidden unit) to the output unit. In [13] a gradient training algorithm for updating all the network parameters (mean vector, spread parameter and output weights) is presented. A novel space filling curves with genetically evolving parameters is proposed in [14].

In order to train the RBF network mostly first order methods are used. Gradient descent learning described in [15], [16] offers a balance between performance and training speed. These networks can be compared with sigmoid hidden unit based feed forward neural networks in [16]-[18]. The combination of steepest descent and Newton's method would seem to be more promising for unconstrained optimization problems [19]. This method is convergent and has a high convergence rate. Since Newton's method for the RBF often has non-positive definite or even singular Hessian, Levenberg-Marquardt (LM) or other methods are used instead. However second order methods do not scale well and suffer from heavy computational cost. Although first order methods scale better, they are sensitive to input means and gain factor [20].

Using other permissible radial basis function apart from Gaussian function has also been explored by constructing reformulated radial basis function neural networks trained with gradient descent algorithm [20]-[22]. But again the convergence and performance of the gradient descent algorithm is strongly affected by the spread parameter and the radial basis function mean vector.

To solve these problems we introduce an interesting family of RBF networks based on Newton's method. The paper is organized as follows: conventional RBF structure and parameter determining methods are reviewed in section 2; In Section 3, the theoretical and mathematical treatment for the proposed family of RBF models is presented. Section 4 discusses the learning parameters for the proposed model. In Section 5, learning algorithms for the proposed model are elaborated; a detailed algorithm for the proposed RBF models is presented in Section 6; the training performance of various RBF models are compared in Section 7, with single Optimal Learning Factor based RBF model on several bench mark and real life datasets.

II. REVIEW OF RADIAL BASIS FUNCTION NETWORK

Without the loss of generality, we restrict ourselves to a three layer fully connected RBF with non linear activation function

A. Basic RBF topology and notation

The structure of the RBF is shown in Fig. 1. The training dataset consist of N_v training patterns $\{\mathbf{x}_p, \mathbf{t}_p\}$ where the p^{th} input vector $\mathbf{x}_p$, and the p^{th} desired output vector $\mathbf{t}_p$ have dimensions N and M respectively. The input units are directly connected to the single hidden layer with N_h hidden nodes. The scalar weight $m_k(n)$ connects the n^{th} input node and the k^{th} hidden node. For the p^{th} training pattern, the k^{th} hidden unit *net function* is

$$net_p(k) = \sum_{n=1}^N (x_p(n) - m_k(n))^2 \quad (1)$$

In order to handle thresholds in the hidden and output layers, the input vectors are augmented by an extra element $x_p(N+1)=1$, so that $\mathbf{x}_p = [x_p(1), x_p(2) \dots, x_p(N+1)]^T$. The size of the hidden layer, N_h , is the number of useful clusters. It should be noted here that N_h is the key factor not only for the performance but also the computational complexity of the network.

The k^{th} cluster mean $\mathbf{m}_k$ ($k=1, 2 \dots N_h$) is the center of the k^{th} subclass. In Fig. 1, the dotted line between input- hidden unit signifies that instead of weighted sum/sigmoid activation, each hidden unit output O_p is obtained by calculating the ‘closeness’ of the input $\mathbf{x}$ to N_h dimensional parameter $\mathbf{m}_k$ associated with the k^{th} hidden unit. The hidden layer is fully connected to the output weight $w_{oh}(m, k)$ connecting the k^{th} hidden unit’s activation $O_p(k)$ to the m^{th} output $y_p(m)$, which has a linear activation. For the p^{th} pattern, the k^{th} hidden unit output $O_p(k)$ is calculated as a *Gaussian basis function*

$$O_p(k) = e^{-\beta(k)net_p(k)} \quad (2)$$

Here $\beta(k)$ is the *spread parameters* defined as the inverse of the width of the receptive field in the input space for hidden unit k . The mean vector $\mathbf{m}_k$ and spread parameter $\beta(k)$ constitute the conventional parameters of RBF.

Once the hidden units training are complete, the output weights are optimized. The M dimensional output vector $\mathbf{y}_p$ is given by

$$\mathbf{y}_p = \mathbf{W}_{oh} \mathbf{O}_p \quad (3)$$

Training an RBF typically involves minimizing the mean squared error between the desired and the actual network outputs, defined as

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{m=1}^M [t_p(m) - y_p(m)]^2 \quad (4)$$

Here $t_p(m)$ and $y_p(m)$ respectively denote the m^{th} desired and actual outputs for the p^{th} pattern respectively.

B. Classical parameter determining methods and learning algorithms

Broomhead and Lowe [22] proposed a method where the idea is to populate dense regions of the input space with receptive fields. The spread parameter and mean vector need to be found without propagating the output error back through the

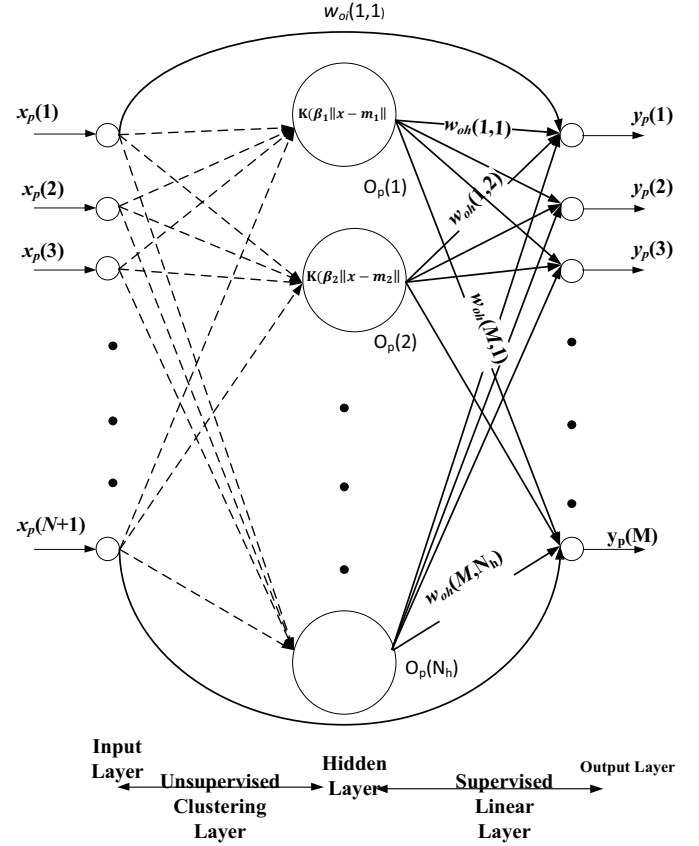


Fig 1: Topology of a fully connected RBF Neural Network

the network. But the disadvantage is for high dimensional space where for ‘ k ’ division along each dimension of an N dimensional input space, we require k^{th} basis function to cover the input space. Later Wasserman [23] presented a heuristic that is useful in determining the spread parameters. In [12] a “good” estimate to represent global average over all Euclidean distance between the centers of each unit to its nearest neighbor is suggested.

For mean vector many methods have been suggested, the most popular being choosing the first sample of the training set, some randomly chosen sample from the training set obtained by clustering or classification method.

In [12] unsupervised learning of mean vectors and spread parameters is employed using relatively small number of RBF clusters. Although clustering can be based on statistical model identification, generally competitive learning based clustering methods are used such as self-organizing map (SOM), learning vector quantization (LVQ), neural gas, Adaptive Resonance Theory (ART), C-means, Fuzzy clustering, Fuzzy C-means clustering and mountain/subtractive clustering. Fuzzy based clustering is used in this paper.

To compute the output weight matrix we can use several training algorithms which are classified into three categories: gradient descent methods, conjugate gradient methods, and Newton related method [24]-[26]. The latter is the basis of much of our research in this paper.

III. PROPOSED RBF NETWORK

In this section we illustrate how the second order based Newton's method can be incorporated in the existing RBF network by modifying the conventional RBF Network described in previous section.

Our proposed family of RBF network introduces a new group of network parameters. Referring to the conventional RBF Neural Network shown in Fig. 1; we have bypass weights $w_{oi}(i, n)$ connecting the n^{th} input to the i^{th} output. Including the bypass weights w_{oi} , the linear output response y_p is now given by

$$\mathbf{y}_p = \mathbf{W}_{oi}\mathbf{x}_p + \mathbf{W}_{oh}\mathbf{O}_p \quad (5)$$

Here the symbols have the usual meaning. The training strategy is a two step procedure, followed in the conventional RBF structure.

A. Phase 1: Initializing the RBF Network

Motivated from the heuristic proposed in [25], we set up the initial parameters for the proposed RBF models as described in following subsection.

1) Setting up initial weighted Distance Measure(DM)

Let elements of vector $\mathbf{c}$, of dimension N , denote weights in a weighted Distance Measure (DM). Let elements of σ represents the standard deviation of input vector of dimension N . Then $\mathbf{c}(n)$ is then defined as :

Then

$$c(n) = \frac{1}{[\sum_{n=1}^N 1/\sigma^2(n)]\sigma^2(n)} \quad (6)$$

The input to each hidden unit is now changed to

$$net_p(k) = \sum_{n=1}^N c(n)(x_p(n) - m_k(n))^2 \quad (7)$$

where $net_p(k)$ is the weighted DM. Note here that the sum is done over N , rather than $N+1$.

2) Setting up initial cluster center using Fuzzy C-means

We use Fuzzy clustering to partition data set into K Fuzzy clusters. The motivation for using Fuzzy clustering is that in several real life applications the data points are on the boundary of two or more classes. Hence they do not belong to any one class. We then assign a membership function which can be seen as a generalization of hard partitions. The Fuzzy C-means clustering algorithm minimizes the cost function J_{fcm} to determine mean vector ' $\mathbf{m}_k$ ' where,

$$J_{fcm} = \sum_{k=1}^K \sum_{p=1}^{N_p} (\varepsilon_{kp})^b \|x_p - m_k\|^2 \quad (8)$$

where ' ε ' is the Fuzzy partition and ' b ' is the free parameter which controls the "bending" of different clusters. If $b > 1$, the cost function allows pattern to belong to multiple cluster. $\mathbf{x}_p$ and $\mathbf{m}_k$ have their usual meaning.

B. Phase 2: Optimization of network parameters

We now propose a procedure to tune the network parameters via Newton's method which uses first and second order derivatives of the parameters w.r.t the error function.

1) Optimizing weighted DM

A typical error function used in the Training of RBF is described in (4). The hidden unit output is given by

$$O_k(p) = e^{-\beta(k) \frac{net_p(k)}{\sum_{n=1}^N (c(n) + d_c(n)) [x_p(n) - m_k(n)]^2}} \quad (9)$$

where $net_p(k)$ is as defined in (7). $c(n)$ is the weighted DM that will be optimized. The output vector $y_p(i)$ is same as described in (6), i.e

$$y_p(i) = \sum_{k=1}^{N_h} w_o(i, k) O_k(p) + \sum_{n=1}^N w_{oi}(i, n) x_p(n) \quad (10)$$

By calculating the 1st order gradient w.r.t. DM we obtain:

$$\begin{aligned} g_c(n) &= -\frac{\partial E}{\partial d_c(n)} \\ &= \frac{2}{N_v} \sum_{p=1}^{N_p} \sum_{i=1}^M [t_p(i) - y_p(i)] \frac{\partial y_p(i)}{\partial d_c(n)} \end{aligned} \quad (11)$$

where

$$\frac{\partial y_p(i)}{\partial d_c(n)} = -\sum_{k=1}^{N_h} w_o(i, k) O_p(k) (-\beta(k)) [x_p(n) - m_k(n)]^2 \quad (12)$$

calculating the Hessian matrix element we get:

$$\begin{aligned} h_c(u, v) &= -\frac{\partial^2 E}{\partial d_c(u) \partial d_c(v)} \\ &= \frac{2}{N_v} \sum_{p=1}^{N_p} \left[\sum_{i=1}^M \frac{\partial y_p(i)}{\partial d_c(u)} \frac{\partial y_p(i)}{\partial d_c(v)} \right] \end{aligned} \quad (13)$$

One realization that occurs now is that of the following linear equations

$$\mathbf{H}_c \mathbf{d}_c = \mathbf{g}_c \quad (14)$$

This is solved using orthogonal least square (OLS). For n^{th} input unit, $c(n)$ is then changed according to

$$c(n) \leftarrow c(n) + d_c(n) \quad (15)$$

Hence solving (15) we get the optimal d_c via OLS.

2) Optimizing "width" (spread parameter)

Using the same error function as in (4), for p^{th} pattern and k^{th} hidden unit.

$$O_p(k) = e^{(-\beta(k) + d_\beta(k)) \frac{net_p(k)}{\sum_{n=1}^N c(n) [x_p(n) - m_k(n)]^2}} \quad (16)$$

here $d_\beta(k)$ is the learning factor for $\beta(k)$ in each k^{th} hidden unit. For i^{th} output we have,

$$y_p(i) = \sum_{k=1}^{N_h} w_o(i, k) e^{-\frac{\text{net}_p(k)}{\sum_{n=1}^N c(n)[x_p(n) - m_k(n)]^2}} \quad (17)$$

We follow the same approach as in the previous subsection. After calculating $g_\beta(k) = -\frac{\partial E}{\partial d_\beta(k)}$ and Hessian matrix element $h_\beta(u, v) = -\frac{\partial^2 E}{\partial d_\beta(u) \partial d_\beta(v)}$ we obtain

$$\mathbf{H}_\beta \mathbf{d}_\beta = \mathbf{g}_\beta \quad (18)$$

Solving this again via OLS, $\beta(k)$ is then changed according to

$$\beta(k) \leftarrow \beta(k) + d_\beta(k) \quad (19)$$

3) Optimizing mean vector

In this algorithm we are changing only the mean vectors holding other parameters constant.

$$O_k(p) = e^{-\frac{\text{net}_p(k)}{\sum_{n=1}^N c(n)[x_p(n) - (m_k(n) + z_k \cdot g_{mean}(k, n)]^2}} \quad (20)$$

For the p^{th} pattern, the i^{th} output unit can be written as

$$y_p(i) = \sum_{k=1}^{N_h} w_o(i, k) e^{-\frac{\text{net}_p(k)}{\sum_{n=1}^N c(n)[x_p(n) - (m_k(n) + z_k \cdot g_{mean}(k, n)]^2}} \quad (21)$$

Unlike the previous gradients, here we calculate the gradient matrix element $g_{mean}(k, n) = -\frac{\partial E}{\partial m_k(n)}$. In [27] multiple optimal learning factor algorithms (MOLF) is shown to train better than OWO-BP. We therefore update the mean vector $\mathbf{m}_k$ based on its own learning factor z_k which is optimal for the k^{th} hidden unit.

The basic idea for MOLF is that while updating the mean vector for every iteration, instead of using a single optimal learning factor 'z' we use a vector $\mathbf{z}$ of optimal learning factor called Multiple OLF (MOLF). Each element of $\mathbf{z}$ is for each mean vector. We now derive the expression for the MOLF as used in updating mean vector. Using (5), we calculate $g_z(k) = -\frac{\partial E}{\partial z_k}$ and $h_z(u, v) = \frac{\partial^2 E}{\partial z(u) \partial z(v)}$. The Gauss-Newton update guarantees that $\mathbf{H}_z$ is non-negative definite. Given the negative gradient vector $\mathbf{g}_z = \left[-\frac{\partial E}{\partial z_1}, -\frac{\partial E}{\partial z_2}, \dots, -\frac{\partial E}{\partial z_{N_h}} \right]^T$ and the Hessian $\mathbf{H}_z$, we minimize E with respect to the vector $\mathbf{z}$ by solving linear equation:

$$\mathbf{H}_z \mathbf{z} = \mathbf{g}_z \quad (22)$$

Thus we get

$$\mathbf{z} = \mathbf{H}_z^{-1} \mathbf{g}_z \quad (23)$$

During each iteration, the k^{th} mean vector is updated as

$$m_k(n) \leftarrow m_k(n) + z_k \cdot g_{mean}(k, n) \quad (24)$$

Here the MOLF procedure has been used to train the mean vectors. It should also be mentioned here that if MOLF fails then we can combine the optimal learning factors into a single optimal learning factor.

IV. COMBINED LEARNING ALGORITHMS

In this section we describe the construction of advance learning algorithm from the previously described algorithms.

A. Output weight optimization (OWO)

In each iteration of our proposed two stage algorithms we optimize the output weight matrix using a technique called output weight optimization (OWO). Since the outputs have linear activations, finding the weights connected to the outputs is equivalent to solving a system of linear equations. If $\mathbf{W}_o$ denotes the output weight matrix, equating $\frac{\partial E}{\partial \mathbf{W}_o} = \mathbf{0}$ leads to

$$\mathbf{C}_a = \mathbf{W}_o \mathbf{R}_a^T \quad (25)$$

Where

$$\begin{cases} \mathbf{C}_a = \frac{1}{N_v} \sum_{p=1}^{N_v} y_p \bar{x}_p^T \\ \mathbf{R}_a = \frac{1}{N_v} \sum_{p=1}^{N_v} \bar{x}_p \bar{x}_p^T \end{cases} \quad (26)$$

Since the equations in (25) are often ill conditioned [28][29], we solve them using OLS[30] which is equivalent to using the QR decomposition [31].

B. Proposed family of algorithms

In this subsection we combine OWO with the optimization approaches of III (B). In our algorithms we combine a multi-stage training algorithm which can be denote as F[N(param1), param2] where 'F' represents Fuzzy c-means clustering and 'N' represents Newton's method applied on param1, while param2 are the parameter(s) which have been set during net control and have not been modified or "tuned" during the training process. We also combine all three parameter optimizations where we update weighted DM from (15) and use (19) and (24) to update beta and mean vector. This largest combined algorithms is denotes as F[N(c, β , m.)].

During our investigation we compared our algorithms with RBF models using single OLF (SOLF). Here we discuss the single OLF case for updating spread parameters. Similar equations can be derived for the single OLF for mean vector case. Similar to the proposed algorithms we denote F[SOLF(param1), param2] where SOLF represents that single OLD has been used for updating the parameters, param1 and param2. We combine both and form a RBF model that will be compared later with the rest of the family of algorithms.

Given the error function as defined in (5) we have,

$$y_p(i) = \sum_{k=1}^{N_h} w_o(i, k) e^{-(\beta(k) + z \cdot g_{\beta(k)}) \text{net}_p(k)} + \sum_{n=1}^N w_{oi}(i, n) x_p(n) \quad (27)$$

Note the similarity in (18) and (23). Instead of using two optimizing parameter, we optimize the performance by using a single parameter $d_{\beta}(k)$ in (18). By using $g_{\beta}(u) = -\frac{\partial E}{\partial \beta(k)}$, we calculate $\frac{\partial E}{\partial z}$. The Gauss-Newton approximation of the second partial and 2nd order Taylor series leads to,

$$z = -\frac{\partial E / \partial z}{\partial^2 E / \partial z^2} \quad (28)$$

Instead of using a single optimal learning factor for updating all the parameters, we use Newton's method to estimate a vector of optimal learning factors z . One of our investigation aims is to show this novel method to update the parameters to be better than the single optimal learning case

It should be mentioned here that during an iteration, at one time, one parameter is updated followed by OWO and then the next parameter is updated. From a philosophical point of view we modify the two step optimization approach which in turn is similar to EM algorithm where the E denotes expectation as the first step and then in the second step maximization denoted by M.

V. ALGORITHM FOR TRAINING THE PROPOSED RBF NETWORK

We follow a two step hybrid learning procedure to train the proposed RBF Network. Contrary to [12] [24], our proposed

algorithm efficiently uses Newton Method and Backpropagation algorithm to optimize the network parameters in every epoch. Fig. 2 shows the various stages

VI. NUMERICAL RESULTS

We present the results for the proposed family of algorithms discussed. All algorithm were implemented in MATLAB script. By using different benchmark and real-life dataset we judge the performance of the algorithms by analyzing the effect of applying Newton's method on network parameters. The generalization ability of each algorithm is later shown by using k -fold validation procedure all the data set used for the simulation is available from the Image Processing and Neural Networks Lab repository [32] and UCI Machine Learning Repository [33].

For all the figures discussed here, ' B ' denotes the spread parameters β ; m_k as the mean vectors and c is the weighted DM.

For better comparison, initial clusters, initial parameters value and the output synaptic weights of all algorithms are initialized by the same way so that the MSE of the first training iteration is same. The RBF architecture for training each dataset has different number of hidden units, different number of input units and output units. Here, the number of training iterations is fixed as $N_{it} = 50$. In all the data set, the inputs have been normalized to be zero mean and unit variance. The family of networks analyzed, are based on Newton's method with multiple OLF and compared with conventional network based on single OLF for β and m_k . During our investigation we take two highly correlated ($\rho > 0.8$) and other two as least correlated ($\rho < 0.2$). Here ρ is the correlation coefficient. By doing so, it is found that the performance of the algorithm is severely affected by the data set correlation.

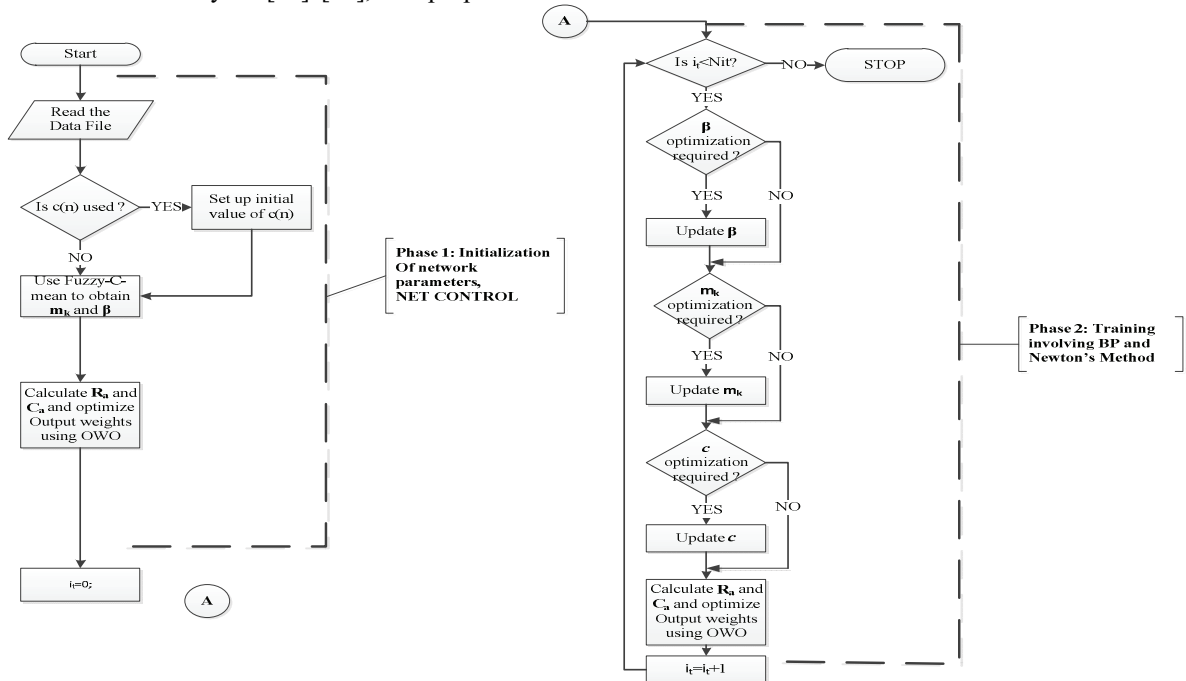


Figure 2: Algorithm for training the proposed family of RBF Network

A. Scattering Parameter Data Set

The training data file contains 1768 patterns. The inputs consist of 8 theoretical values of scattering parameters. The outputs are its 7 corresponding heights, lengths and permittivity. [32] ‘twod.tra’ is a highly correlated input data set. We trained all the RBF models with $N_h = 20$. In Fig. 3, the average mean square error (MSE) for training is plotted versus the number of iterations (N_{it}) for each algorithm. From

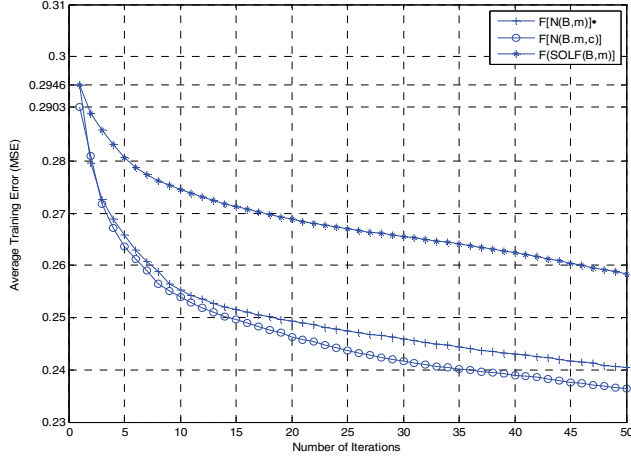


Figure 3 Performance of proposed family of algorithm with single OLF based RBF on twod

The plots we deduce that the network with Newton’s method applied on all parameters gives the best performance applied

B. Radar Scattering Data Set

This training set also called ‘oh7’, contains polarization at different angles along with the corresponding surface height, surface correlation length, and volumetric soil moisture content [32].

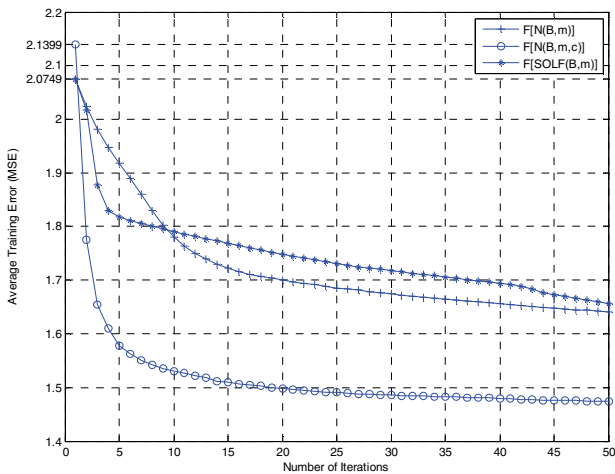


Figure 4 Performance of proposed family of algorithm with single OLF based RBF on oh7

Here not only the input values but also one of the output values is highly correlated with the input values unlike ‘twod’. We train all the RBF models with $N_h = 20$. In Fig. 4, the MSE for training is plotted versus N_{it} for each algorithm. The plot shows that the RBF model where we apply Newton’s method on all parameters perform best.

C. Inverse Matrix Data Set

Also called ‘Mattn’, it is an uncorrelated dataset with 2000 training patterns obtained from the inversion of random two-by-two matrices [32]. Each pattern consists of 4 input features and 4 output features. For this data set, all the RBF models are trained with $N_h = 15$. In Fig. 5, MSE for training is plotted versus N_{it} for each algorithm. This data set is uncorrelated and we observe that the model where Newton is applied on β and m_k performs at par with the model where it is applied on all the parameters.

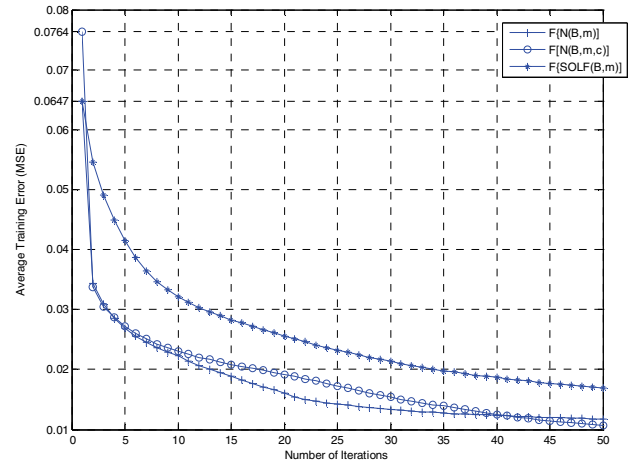


Figure 5 Performance of proposed family of algorithm with single OLF based RBF on mattn

D. Concrete Compressive Strength Data Set

“Concrete” is an un-correlated dataset used to approximate the nonlinear function of age and ingredients of concrete compressive strength [33]. With a total number of patterns as 1030, this dataset consists of 8 inputs and 1 output and $N_h = 20$. This dataset is also least correlated and we see the same trend as we see in mattn.tra. The performance of the model with β and m_k optimization is close to the model that have weighted DM optimization.

E. K-Fold Crossing Validation

We use the k-fold crossing validation procedure to show the generalization abilities of various RBF models. Choosing $k=10$ so for each dataset, we split it randomly into 10 non-overlapping parts of equal size, and use 9 parts of total data for training and leave the remaining one part for testing. This procedure was repeated till we have exhausted all 10 combinations. Then, by training all these combinational datasets, we got the average of training errors. Also, the validation error of each dataset was obtained by averaging all corresponding testing errors on every testing dataset. The training MSEs and test MSEs of k-fold crossing validation on these four datasets are listed as Table 1.

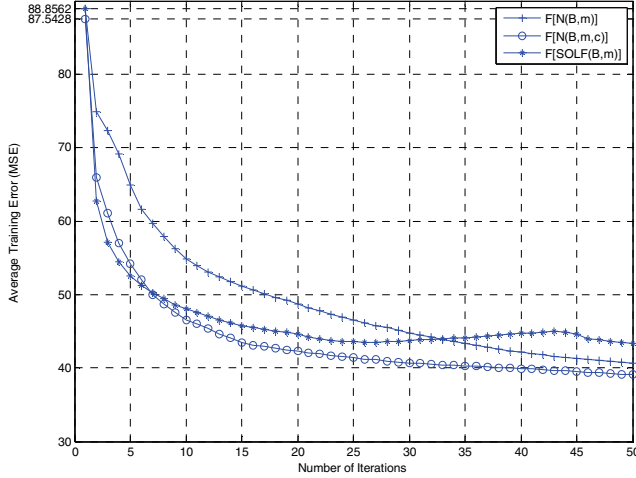


Figure 6 Performance of proposed family of algorithm with single OLF based RBF on concrete

TABLE 1: COMPARISON OF K-FOLD VALIDATION ON DATASET

Data Set	Error	F[N(B,m)]	F[N(B,m,c)]	F[SOLF(B,m)]
Twod	Validation	0.2456	0.2363	0.2587
	Testing	0.2519	0.2433	0.2713
Oh7	Validation	1.6547	1.4738	1.6847
	Testing	1.6843	1.4916	1.7214
mattm	Validation	0.0118	0.0106	0.0168
	Testing	0.0215	0.0204	0.0181
concrete	Validation	40.5882	39.9854	43.1598
	Testing	41.0689	40.6538	43.2811

VII. CONCLUSION AND FUTURE WORK

The conclusion that we make from our experimental results is that the used of Newton's method on all three parameters helps significantly in improving the performance of RBF networks. Newton's method effect is more pronounced in correlated datasets. Training of the DM weights significantly improves the RBF network. From the experiments we conclude that for un-correlated datasets, optimizing $\mathbf{m}_k$ through SOLF has no effects. We have seen that the MOLF training of the $\mathbf{m}_k$ is more effective than the training of β . The experimental results further bolster this conclusion.

REFERENCES

- [1] Kumar,S., *Neural Network: a classroom approach*, International ed. McGraw Hill Press, 2005, pp 304-314.
- [2] Medgassy,P. 1961. *Decomposition of superposition of distributed functions*, Hungarian academy of Sciences, Budapest.
- [3] Micchelli, C.A.1986. *Interpolation of scattered data: Distance and conditionally positive definite functions*, Constructive Approximations, vol.2, pp. 11-22.
- [4] Powell, M.J.D 1987. *Radial Basis functions for multivariate interpolation: a review*, in *algorithms for the approximation of Functions and Data*. J.C.Mason and M.G.Cox eds., Clarendon Press, Oxford, England.
- [5] Duda, R.O., and Hart, P.E.1973. *Pattern Classification and Scene Analysis*, Wiley, New York.

- [6] Speecht, D.F. 1990. *Probabilistic neural networks*, Neural Networks, vol 3, pp.109-118.
- [7] Poggio, T., and Girosi, F. 1989. *A theory of Networks for Approximation and Learning*. A.I. Memo 1140, MIT, Cambridge.
- [8] Liu,X., Sun, D., Chang, Y., and Peng, Z., "Traffic Status Evaluation Based on Fuzzy Clustering and RBF Neural Network," *Intl. Conference on Fuzzy Systems and Knowledge Discovery*, 2010., pp 1405-1409.
- [9] Pedrycz, W., "Conditional Fuzzy Clustering in the Design of Radial Basis Function Neural Networks," *IEEE Trans. Neural Network*, vol.9, no.4, pp.601-612, Jul.1998.
- [10] Staiano, A., Tagliaferri, R., and Pedrycz, W., "Improving RBF Networks performance in regression tasks by means of a supervised Fuzzy clustering," *Neurocomputing*, pp 1570-1581, 2006.
- [11] Madedda, K., Kanae, S., Yang, Z., and Wada, K., "Design of RBF Network Based on Fuzzy Clustering Method for Modelling of Respiratory System", *Advances in Neural Networks*, vol. 3973/2006.
- [12] Moddy,J.E., and Darken,C.J., "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol.1, pp.281-294, 1989.
- [13] Cha, I., and Kassam,S.A., "Interference cancellation using radial basis function networks," *Signal Processing*, vol.47, pp.247-268, 1995
- [14] Whitehead, B.A., and Chaote,T.D., "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Trans. Neural Networks*, vol.5, pp. 15-23, Jan. 1994.
- [15] Karayiannis, N.B., "Gradient descent learning of radial basis neural networks," in *Proc.1997 IEEE Int. conf. Neural Networks*, vol.3, Houston, TX, June 9-12, 1997, pp.1815-1820.
- [16] Karayiannis, N.B., "Learning algorithms for reformulated radial basis neural networks," in *Proc.1998 Int. Joint Conf. Neural Networks*, Anchorage, AK, 1998, pp.2230-2235.
- [17] Karayiannis, N.B., "Reformulated radial basis neural networks trained by gradient descent," *IEEE Trans. Neural Networks*, vol.10, pp.657-671, May 1999.
- [18] Karayiannis, N.B., and Behnke,S., 2000 "New radial basis neural networks and their application in a large-scale handwritten digit recognition problem," in *Recent Advances in Artificial Neural Networks: Design and Applications*, L. C. Jain and A. M. Fanelli, Eds. Boca Raton, FL: CRC, pp. 39-94.
- [19] Shi, Y., Globally convergent algorithms for unconstrained optimization, *Computational Optimization and Application* vol.16, pp 295-308, 2000.
- [20] Malalur,S., Manry,M., "Feed-Forward Network Training Using Optimal Input Gains," *International Conference on Neural Networks*, Atlanata, pp. 1953-1960, Jun 2009.
- [21] Cover, T.M., "Geometrical and Statistical Properties of systems of linear inequalities with application in pattern recognition", *IEEE Transaction on Electronic Computers*, Vol. EC-14, pp 326-334, 1965.
- [22] Broomhead, D.S. and Lowe, D. (1988), *Multivariate functional interpolation and adaptive networks*, Complex Systems, vol.2, pp. 321-355.
- [23] Wasserman, P.D., *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, New York, 1993.
- [24] Haykin, S., *Neural Network: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1999.
- [25] Fletcher, R., *Practical Methods of Optimization*, 2nd ed. Chichester, NY: John Wiley & Sons, 1987.
- [26] Moller, M., *Efficient training of feed forward neural Networks*, Ph.D. dissertation Aarhus University, Denmark, 1997.
- [27] Malalur, S., and Manry,M., "Multiple optimal learning factors for feed-forward networks," *Proc. of SPIE: Independent Component Analyses, Wavelets, Neural Networks, Biosystems, and Nanoengineering VIII*, Orlando Florida, vol. 7703 pp. 77030F-1 – 77030F-12, April 7-9, 2010.
- [28] Saarein,S., Bramley, R., and Cybenko, G., "Ill-conditioning in neural network training problems." *SIAM Journal on Scientific Computing*, vol.14,pp 693-714, 1993.

- [29] Smagt, P., and Hirzinger, G., *Solving the ill-conditioning in neural networks learning*, ser. *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science 1524, G.Orr and K.R.Muller, Eds, Springer Verlag, 1998.
- [30] Hadzer et al., "Improved Singular Value Decomposition by Using Neural Network," in *Proc. Of IEEE International Conference on Neural Networks*, 1995, Perth, WA Australia, Nov, 1995, pp.438-442.
- [31] Press, W.H.et al., *Numerical Recipes*. NBey York: CAmbirdge University Press, 1986.
- [32] http://www-ee.uta.edu/cewb/ip/training_data_files.htm
- [33] <http://archive.ics.uci.edu/ml/>