International Journal of Computer and Communication Technology

Volume 3 | Issue 1 Article 11

January 2012

Breast Cancer Detection using Recursive Least Square and Modified Radial Basis Functional Neural Network

Manas Rajan Senapati
Department of computer Science, Gandhi Engineering College, Bhubaneswar, manas_senapati@sify.com

P. K. Routray
Department of computer science and Engineering N M Institute of Engineering and Technology Biju Patnaik University of Technology, India, pravat.routray@gmail.com

Pradipta Kishore Dash
College of Engineering Bhubaneswar, India, pkdash_india@yahoo.com

Recommended Citation
Senapati, Manas Rajan; Routray, P. K.; and Dash, Pradipta Kishore (2012) "Breast Cancer Detection using Recursive Least Square and Modified Radial Basis Functional Neural Network," *International Journal of Computer and Communication Technology*: Vol. 3 : Iss. 1 , Article 11.
DOI: 10.47893/IJCCT.2012.1115
Available at: https://www.interscience.in/ijcct/vol3/iss1/11

Breast Cancer Detection using Recursive Least Square and Modified Radial Basis Functional Neural Network

M.R.Senapati[a], P.K.Routray[b],P.K.Dask[b1]
[,a] Department of computer science and Engineering
Gandhi Engineering College
Biju Patnaik University of Technology, India
manas_senapati@sify.com
[b] Department of computer science and Engineering
N M Institute of Engineering and Technology
Biju Patnaik University of Technology, India
pravat.routray@gmail.com
[b1]S'O'A University, India
pkdash_india@yahoo.com

***Abstract*-- A new approach for classification has been presented in this paper. The proposed technique, Modified Radial Basis Functional Neural Network (MRBFNN) consists of assigning weights between the input layer and the hidden layer of Radial Basis functional Neural Network (RBFNN). The centers of MRBFNN are initialized using Particle swarm Optimization (PSO) and variance and centers are updated using back propagation and both the sets of weights are updated using Recursive Least Square (RLS). Our simulation result is carried out on Wisconsin Breast Cancer (WBC) data set. The results are compared with RBFNN, where the variance and centers are updated using back propagation and weights are updated using Recursive Least Square (RLS) and Kalman Filter. It is found the proposed method provides more accurate result and better classification.**

Keywords: *Radial Basis Functional Neural Networks (RBFNN), Wisconsin Breast Cancer (WBC), Pattern Recognition, Gradient Descent Method, Recursive Least Square, Kalman Filter.*

1. INTRODUCTION

A Radial Basis Functional neural network (RBFNN) is trained to perform a mapping from an m-dimensional input space to an n-dimensional output space. RBFNN's can be used for discrete pattern classification, function approximation, signal processing, control, or any other application, which requires a mapping from an input space to an output space.

An RBFNN consists of the m-dimensional input *x* being passed directly to a hidden layer. Suppose there are *c* neurons in the hidden layer. Each of the *c* neurons in the hidden layer applies an activation function, which is a function of the Euclidean distance between the input and an m-dimensional prototype vector.

Each hidden neuron contains its own prototype vector as a parameter. The output of each hidden neuron is then weighted and passed to the output layer. The outputs of the network consist of sums of the weighted hidden layer neurons. Figure 1 shows a schematic form of an RBFNN network.

It can be seen from the basic architecture, that the design of an RBFNN requires several decisions, including the following:

1. How many neurons will reside in the hidden layer? (i.e., what is the value of the integer c);
2. What are the values of the prototypes (i.e., what are the values of the *v* vectors)?
3. What function will be used at the hidden units (i.e., what is the function g (·))?
4. What weights will be applied between the hidden layer and the output layer?

The performance of an RBFNN network depends on the number and location (in the input space) of the centers, the shape of the RBFNN functions at the hidden neurons, and the method used for determining the network weights. Some researchers have trained RBFNN networks by selecting the centers randomly from the training data[1].

Some have used unsupervised procedures (such as the k-means algorithm) for selecting the RBFNN centers], while others have used supervised procedures for selecting the RBFNN centers [2].

Several training methods separate the tasks of prototype determination and weight optimization for classification and rule generation. This trend probably arose because of the quick training that could result from the separation of the two tasks. In fact, one of the primary contributors to the popularity of RBFNN networks was probably their fast training times as compared to gradient descent training (including back propagation) shown in Figure 1, it can be

seen that once the prototypes are fixed and the hidden layer function *g(·)* is known, the network is linear in the weight parameters *w*. At that point training the network becomes a quick and easy task that can be solved via linear least squares. (This is similar to the popularity of the optimal interpolative net that is due in large part to the efficient non- iterative learning algorithms that are available [3,4].

Training methods that separate the tasks of prototype determination and weight optimization often do not use the input—output data from the training set for the selection of the prototypes. For instance, the random selection method and the k-means algorithm result in prototypes that are completely independent of the input—output data from the training set. Although this results in fast training, it clearly does not take full advantage of the information contained in the training set.

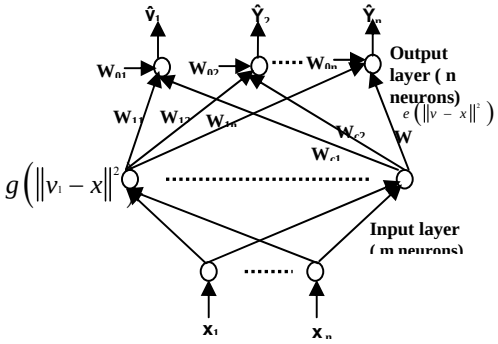

Fig. 1: Radial Basis Functional Network

Gradient descent training of RBFNN networks has proven to be much more effective than more conventional methods[2]. However gradient descent training can be computationally expensive. This paper extends the results of [2] and formulates a training method for RBFNN's based on Recursive Least Square. This new method proves to be quicker than gradient descent while still providing performance at the same level of effectiveness.

Training a neural network is, in general, a challenging nonlinear optimization problem. Various derivative-based methods have been used to train neural networks, including gradient descent [2], Kalman .Filtering [5, 6], and the well-known back-propagation [7]. Derivative-free methods, including genetic programming [8-10] and simulated annealing [11] have also been used to train neural networks.

Derivative-free methods have the advantage that they do not require the derivative of the objective function with respect to the neural network parameters. They are more robust than derivative-based methods with respect to finding a global minimum and with respect to their applicability to a wide range of objective functions and neural network architectures. However, they typically tend to converge more slowly than derivative-based methods. Derivative-based methods have the advent age of fast convergence, but they tend to converge to local minima. In addition, due to their dependence on analytical derivatives, they are limited to specific objective functions and specific types of neural network architectures.

2. INTERPRETATION OF RADIAL BASIS FUNCTIONAL NEURAL NETWORK

The multi layered feed forward network (MFN) is the most widely used neural network model for pattern classification applications. This is because the topology of the MFN allows it to generate internal representations tailored to classify the input regions that may be either disjointed or intersecting. The hidden layer nodes in the MFN can form hyper planes to partition the input space into various regions and the output nodes can select and combine the regions that belong to the same class. Back propagation (BP) is the most widely used training algorithm for the MFN's. Recently researchers have begun to examine the use of Radial Basis Function neural networks (RBFNN) for pattern Recognition problems due to a number of drawbacks of BP-trained networks. Although a BP network produces decision surfaces that effectively separate training examples of different classes, this does not necessarily result in the most plausible or robust classifier. The decision surfaces of BP networks may not take on any intuitive shapes because regions of the input space not occupied by training data are classified arbitrarily, not according to proximity to training data. In addition, BP networks have no mechanism to detect that a case to be classified has fallen into a region with no training data. This is a serious drawback since the power system operates within a wide range of system and fault conditions.

The RBFNN consists of an input layer made up of source nodes and a hidden layer of a sufficiently high dimension. The output layer supplies the response of the network to the activation patterns applied to the input layer. The nodes within each layer are fully connected to the previous layer as shown in the Figure 1. The input variables are each assigned to a node in the input layer and pass directly to the hidden layer without weights. The hidden nodes, or units, contain the radial basis functions (RBFNN's) and are represented by the bell-shaped curve in the hidden nodes as shown in the Fig 1.

2.1 RBFNN Algorithm:

This section describes how we used an RBFNN network to classify the data sets. RBFNN used here has an input layer, a hidden layer consisting of Gaussian node function, an output layer, and a set of weights, W to connect the hidden layer and output layer. We denote x to be the input vector to the network, where x = (x_1, x_2, x_3, ……x_D), and D is the embedding dimension. We call o the ANN output vector, where o = (o_1, o_2, o_3 , …. o_n) T is the number of out put nodes. We have P training patterns. The RBFNN classification problem is to approximate the mapping from the set of inputs,

x = {x(1), x(2), ….., x(P)}, …….(1)

to the set of outputs,

o={o(1),o(2),o(3), ……., o(P)} ……..(2)

For an input vector x(t) , the output of j^{th} output node produced by an RBFNN is given by

$$o_j(t) = \sum_{i=1}^{mtot} w_{ij}\phi_i(t) = \sum_{i=1}^{mtot} w_{ij} e^{-\frac{\|x(t)-c_i\|}{2c_i}} \quad (3)$$

Where C_i is the center of the i^{th} hidden node,
σ_i is the width of the i^{th} center, and mtot is the total number of hidden nodes. Using vector notation, let m = (m1(t),m2(t),…..,mtot(t)) and w_j = ($w1_j$, $w2_j$, ….., wmto(tj)) and RBFNN output can be written as o_j = w_j*T(t) .

The cost function of the network for the jth output is then calculated as e = (d – o_j) where d = desired output. The RBFNN classifier contains four sets of parameters that have to be learned form the examples. They are the centers, ci(t), number of centers mtot, variances σ_I, and weights wij,. We denote all the RBFNN's centers by Cwhole. In our implementation of RBFNN, classes do not share centers. Each of these sets of centers is trained with a separate PSO clustering run.

Once the RBFNN centers are initialized by PSO then the weights are updated according to the following:

$w_{ij}(t+1) = w_{ij}(t)+2e\varnothing_i$ (t)

The centers are then updated according to the following:

$$t+1) = c(t) + \frac{2ew_{ij}\varnothing_i(x_i - c_{ij})}{\sigma_i^2} \quad (4)$$

The width associated with the k^{th} center is adjusted as

$$\sigma_k(i) = \sqrt{1/Na\sum\|c_k(i)-c_j(i)\|^2} \quad (5)$$

There are several reasons for using an RBFNN in our classification problem. First many neural networks require nonlinear optimization for training.

The second reason for employing a RBFNN classifier is that the internal representation of training data of an RBFNN is intuitive. Each RBFNN center approximates a cluster of training of data vectors that are close each other in Euclidean space. When a vector is input to the RBFNN, the center near to that vector becomes strongly activated, in turn activating certain output nodes.

The hypothesis space implanted by these learning machines is constituted by functions of the form

$$f(x,w,v) = \sum_{i=1}^{m} w_k \varnothing_k(x,v_k) + w_0 \quad (6)$$

The nonlinear activation function Øk expresses the similarity between any input pattern x and the center v_k by means of a distance measure. Each function $Ø_k$ defines a region in the input space (receptive field) on which the neuron produces a appreciable activation value. If the common case when the Gaussian function is used, the center C_k of the function $\varnothing_k$ defines the prototype of input cluster k and the variance $Ø_k$ the size of the covered region in the input space.

The rule extraction method for RBFNN derives descriptions in the form of ellipsoid. Initially, assigning each input pattern to their closest center of RBFNN node according to the Euclidean distance function a partition of the input space is made. When assigning a pattern to its closest center, this one will be assigned to the RBFNN node that will give the maximum activation value for that pattern. From these partitions the ellipsoid are constructed. Next, a class label is assigned for each center of RBFNN units. Output value of the RBFNN network for each center is used in order to determine this class label. Then, for each node an ellipsoid with the associated partition data is constructed. Once determined the ellipsoid, they are transferred to rules. This procedure will generate a rule by each node.

2.2 Modified Radial Basis Functional Neural Network **:**

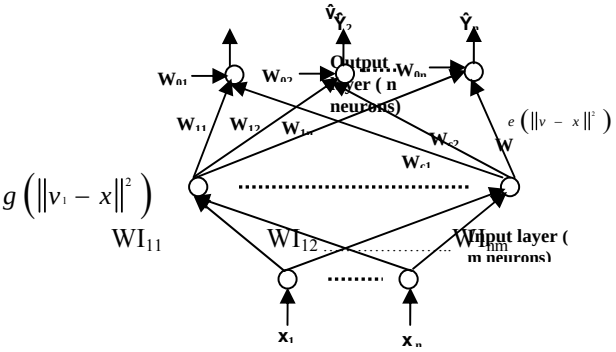

Fig. 2: Modified Radial Basis Functional

Modified Radial Basis Functional Neural Network is same as that of RBFNN with an exception that weights are assigned between neurons in the input layer and the

neurons in the hidden layer (fig.2). The net input to the neurons in the hidden layer is calculated as $i = \sum \bar{x} * \bar{w}$ and the out put is given as equation 3. Where i is the neuron number, x is the input to the network and w is the weights between input layer and hidden layer

The centers are updated using the equation 4 and the variances are updated using the equation 5.

The weights between input layer and the hidden layer as well as hidden layer and output layer of the RBFNN classifier can be trained using the linear recursive least square (RLS) algorithm. The RLS is employed here since it has a much faster rate of convergence compared to gradient search and least mean square (LMS) algorithms.

$$P(i-1)\varphi T(i)$$

$$k(i)= \lambda+P(i-1)\ \varphi T(i) \quad (7)$$

$$w(j)=wj(i-1)+k(i)[dj(i)-wj(i-1)\ \varphi T(i)] \quad (8)$$

$$P(i) = \frac{1}{\lambda}[P(i-1) - k(i)\varphi(i)\, p(i-1)] \quad (9)$$

where λ is real number between 0 and 1,$P(0)=a^{-1}I$, and a is a small positive number and $w_j(0)=0$.

The computational steps involved in implementing of MRBFNN for fault classification are:

1. for each class c, initialise the centers using Particle swarm Optimization mc =minit (initialization);
2. train the MRBFNN Centers and spreads using Error Back Propagation
3. train the MRBFNN Weights (Between input layer & hidden layer and hidden layer & output layer) using RLS
4. add einc centers to Nc classes with highest output, to get a new m, then go to step 2;
5. the RBFNN is used with the one with the current m. The learning rate of the RBFNN is 0.1 and the center and the weights are updated in every iteration that is by new training input to the RBFNN.

3. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a population based stochastic search process, modeled after the social behavior of a bird flock[12-15]. The algorithm maintains a population of particles, where each particle represents a potential solution to an optimization problem.

In the context of PSO, a swarm refers to a number of potential solutions to the optimization problem, where each potential solution is referred to as a particle. The aim of the PSO is to find the particle position that results in the best evaluation of a given fitness (objective) function. Each particle represents a position in N_d dimensional space, and is "flown" through this multi-dimensional search space, adjusting its position towards both

- The particle's best position found thus far, and
- The best position in the neighborhood of that particle.

Each particle I maintains the following information :

- x_i : The *current position* of the particle.
- v_i : The *current velocity* of the particle.
- y_i : The *personal best position* of the particle.

Using the above notation, a particle's position is adjusted according to

$$v_{i,k}(t+1)=wv_{i,k}(t)+c1r1,k(t)\,(y_i,k(t)-x_i,k(t))+c2r2,k(t)(\hat{y}\,k(t)-x_i,k(t)) \quad (10)$$

$$x_i(t+1)=x_i(t)+v_i(t+1) \quad (11)$$

where w is the inertia weight c_1 and c_2 are the acceleration constants, $r_{1,j}(t)$, $r_{1,j}(t) \sim U(0,1)$, and k=1, … ., N_d. The velocity is thus calculated based on three contributions: 1) a fraction of the previous velocity, 2) the cognitive component which is a function of the distance of the particle from its personal best position, and 3) the social component which is a function of the distance of the particle from the best particle found thus far (i.e; the best of the personal bests).

The personal best position of the particle is calculated as

$$y_i = \begin{cases} y_i & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases} \quad (12)$$

Two basic approaches to PSO exists based on the interpretation of the neighborhood of particles. Equation (10) reflects the *gbest* version of PSO where, for each particle, the neighborhood is simply the entire swarm. The social component then causes particles to be drawn toward the best particle in the swarm. In the *lbest* PSO model, the swarm is divided into overlapping neighborhoods, and the best particle of each neighborhood is determined. For the *lbest* PSO model, the social component of equation(10) changes to

$$c_2 r_{2,k}(t)(y_k(t) - x_{i,k}(t)) \quad (13)$$

where $\hat{y}_j$ is the best particle in the neighborhood of the i^{th} particle. The PSO is usually executed with repeated application of equations (10) and (11) until a specified number of iterations has been exceeded. Alternatively,

the algorithm can be terminated when the velocity updates are close to zero over a number of iterations.

3.1 PSO Clustering :

In the context of clustering, a single particle represents the Nc cluster centroid vectors. That is, each particle xi is constructed as follows:

$x_i = (m_i1, \ldots, m_ij, \ldots, m_iNc)$ (14)

where mij refers to the j-th cluster centroid vector of the i-th particle in the cluster Cij. Therefore, a swarm represents a number of candidate clustering for the current data vectors. The fitness of particles is easily measured as the quantization error,

$$c_2 r_{2,k}(t)(y_k(t) - x_{i,k}(t)) \quad (13)$$

Where d is defined in equation,

$$d(Z_p, m_j) = \sqrt{\sum_{k=1}^{N_d} (Z_{pk} - m_{jk})^2} \quad (15)$$

where k subscripts the dimension.

$$mj = \frac{1}{nj} \sum_{\forall Zp \in Cj} Zp$$

and | Cij | is the number of data vectors belonging to the cluster Cij, ie; the frequency of that cluster.

This section first presents the standard gbest PSO for clustering data into a given number of clusters and then shows the PSO algorithm can be used to improve the performance of Radial basis functional Neural Network (RBFNN) for classification.

3.1.1 gbest PSO clustering Algorithm

Using the standard gbest PSO, data vectors can be clustered as follows :

1. Initialize each particle to contain Nc randomly selected cluster centroids.
2. For t = 1 to tmax do
   a) For each particle i do
   b) For each data vector Zp
      i) calculate the Euclidean distance d(Zp, mij) to all cluster centroids Cij
      ii) Assign Zp to cluster Cij, such that d(Zp, mij) = min∀c=1, ..., Nc {d(Zp, mic)}
      iii) calculate the fitness function using (6)
   c) Update the global best and local best positions
   d) Update the cluster centroids using equations (10) and (11). where tmax is the maximum number of iterations.

4. DISCUSSION

In order to evaluate the performance of the algorithm, we carried out a two fold experiment with WBC data set with the center initialized by PSO. The result shows if both the set of weights of MRBFNN are optimized using RLS then the performance i.e. the percentage of Classification is better as compared to optimizing the same using Kalman Filter and RLS Table.13. The algorithms associated to the extraction method were simulated using MATLAB v6.5.

4.1 Simulation Environment.

We tested the algorithms of the previous sections with Wisconsin breast cancer (WBC) data set by optimizing the weights of RBFNN using RLS, Kalman Filter. The weights of the MRBFNN are optimized using RLS.

4.2 WBC Dataset

The WBC training set contains 400 exemplars and the test set containing 299 exemplars for a total of 699 exemplars. The input data were normalized by replacing each feature value x by x= (x − μx) / σx where μx and σx denote the sample mean and standard deviation of this feature over the entire data set. The networks are trained to respond with the target value y_{ik} =1, and y_{jk} = 0. j □ i, when presented with an input vector x_k from the i^{th} category.

The MATLAB m-files were used to generate the simulation results presented in this section. The training algorithms were initialized with prototype vectors randomly selected from the input data on a two fold basis and with the weight matrix W set to 1 and □ initialized to random values.

4.3 Simulation Results

4.3.1 Tabular Data:

The results of centers obtained by from our simulation studies are shown in tables. Table-1 , Table- 5, and Table-8 shows the centers of WBC obtained from MRBFNN, RBFNN using RLS and Kalman Filter respectively. Table-2, Table-3 shows the weights obtained from MRBFNN and Table-6, and Table-9, shows the weights obtained form RLS and Kalman Filter respectively. Table-4, Table-7, and Table-10, shows the variances obtained.

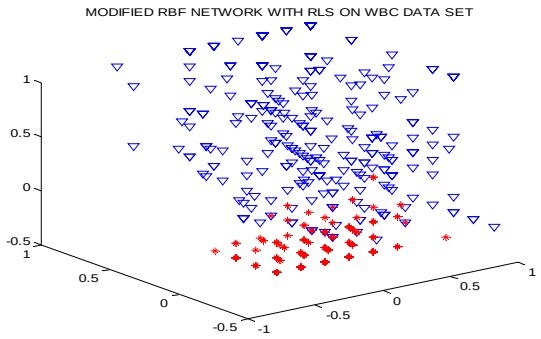

Fig 5: Classification using MRBFNN and RLS

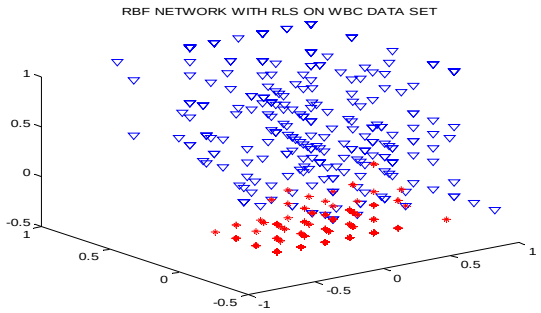

Fig6: Classification using RBFNN and RLS

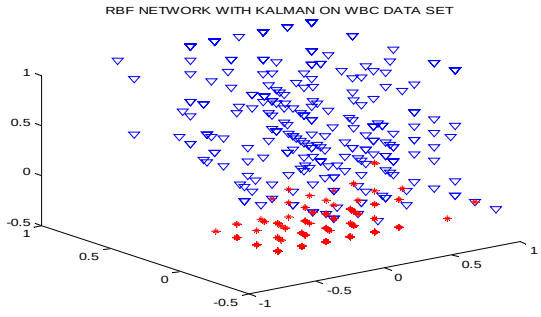

Fig7: Classification using RBFNN and Kalman Filter

Table 1: Centers obtained from MRBFNN

W	2.5200	2.5200	2.5200	2.5200	2.5200
	2.5200	2.5200	2.5200	2.5200	
B					
	1.5038	1.5038	1.5038	1.5038	1.5038
C	1.5038	1.5038	1.5038	1.5038	

Table 2: Weights between input layer and hidden layer obtained from MRBFNN

W	4.4475	4.4475	4.4475	4.4475	4.4475
	4.4475	4.4475	4.4475	4.4475	
B					
	102.1668	102.1668	102.1668	102.1668	102.1668
C	102.1668	102.1668	102.1668	102.1668	

Table 3: Weights between hidden layer and output layer obtained from MRBFNN

3.4475	100.1310
3.4475	100.1310

Table 4: Variances obtained from MRBFNN

2.1557
0.3765

Rule for classification of WBC data sets using MRBFNN

if (oo(r,1)>=0.9508 & oo(r,1)<=1.1540) &
(oo(r,2)>=0.1350 & oo(r,2)<=0.3999)
then class Benign

if (oo(r,1)>=1.1562 & oo(r,1)<=2.2055) &
(oo(r,2)>=0.4070 & oo(r,2)<=15.484)
then class Malignant

Table 5: Centers obtained from RBFNN and RLS

W	1.2352	1.2352	1.2352	1.2352	1.2352
B	1.2352	1.2352	1.2352	1.2352	
C					
	1.6426	1.6426	1.6426	1.6426	1.6426
	1.6426	1.6426	1.6426	1.6426	

Table 6: Weights obtained from RBFNN and RLS

1.6426	1.6426
5.7481	5.7481

Table 7: Variances obtained from RBFNN and RLS

7.8622
8.6432

Rule for classification of WBC data sets using RBFNN and RLS

if (oo(r,1)>=8.5288 & oo(r,1)<=8.9768) &
(oo(r,2)>=13.2243 & oo(r,2)<=13.8661)
then class Benign;

if (oo(r,1)>=8.9937 & oo(r,1)<=10.6680) &
(oo(r,2)>=13.8850 & oo(r,2)<=16.2742)
then class Malignant;

Table 8: Centers obtained from RBFNN and Kalman Filter

W	1.1905	1.1905	1.1905	1.1905	1.1905
	1.1905	1.1905	1.1905	1.1905	
B					
	1.0019	1.0019	1.0019	1.0019	1.0019
C	1.0019	1.0019	1.0019	1.0019	

Table 9: Weights obtained from RBFNN and Kalman Filter

0.9849	0.9981
1.1905	1.0019

Table 10: Variances obtained from RBFNN and Kalman Filter

7.8622
8.6432

Rule for classification of WBC data sets using Kalman Filter

if (oo(r,1)>=1.6277 & oo(r,1)<=1.7129) &
(oo(r,2)>=1.5870 & oo(r,2)<=1.6614)
then class Benign
if (oo(r,1)>=1.7163 & oo(r,1)<=2.0338) &
(oo(r,2)>=1.6651 & oo(r,2)<=1.9274)
then class Malignant;

Table 11: Percentage of Classification

	% of Accuracy
MODIFIED RBFNN	98.1402
RBFNN and RLS	97.1388
RBFNN and Kalman Filter	96.4235

Table-11 shows the percentage of classification of respective techniques for WBC datasets.

4. CONCLUSION

An efficient Pattern Recognition and rule extraction technique using Recursive Least square approximation and Modified Radial Basis Functional Neural Networks (MRBFNN) is presented in this paper. Particle Swarm Optimization [12-15] is used to find the initial centers of Wisconsin Breast Cancer (WBC). After the centers have been initialized RBFNN is used to update the centers and the variances and the weights of MRBFNN are updated using Recursive Least Square approximation. The Classification result is given in Table 13 shows the effectiveness of MRBFNN. The trained network is capable of providing better Classification with comparison to training RBFNN network using Kalman Filter and Recursive Least Square approximation. Further research could focus on the application of different training methods to train MRBFNN. This technique can be applied to large problems to obtain experimental verification of the computational results can be included as a future work.

5 ACKNOWLEDGEMENT :

We would like to acknowledge the encouragement and support given by Prof. S. N. Dehury, Fakir Mohan University, Orissa, India. He had been very kind and patient while suggesting the outlines of this work.

6. REFERENCES

[1] D.Broomhead, D.Lowe, "Multivariable Functional Interpolation and adaptive networks", complex systems, v2, 1998, pp. 321-355.

[2]. N. Karayiannis, "Reformulated radial basis neural networks trained by gradient descent", IEEE Transactions on Neural Networks, Volume:10, Issue: 3, 1999 pp. 657-671.

[3]. D. Simon, "Distributed fault tolerance in optimal interpolative nets",. IEEE Transaction on Neural Networks, Volume: 12, Issue: 6, 2001 pp. 1348-1357.

[4]. S. Sin, R. DeFigueiredo, " Efficient learning procedures for optimal interpolative nets", Neural Networks, Volume:6 , Issue:1, 1993 pp. 6 99-113.

[5]. John Sum, Chi-sing Leung, Gilbert H. Young, and Wing-kay Kan,. "Kalman Filtering Method in Neural-Network, Training and Pruning", IEEE Transactions On Neural Networks,. Volume: 10, Issue:1, 1999, pp. 161-166.

[6]. Y. Zhang, X. Li, "A fast U-D factorization-based learning algorithm with applications to nonlinear system modeling and identification", IEEE Transaction on Neural Networks, Volume: 10, Issue: 4, 1999, pp. 930 - 938.

[7]. Duro, R.J.; Reyes, J.S, "Discrete-time back propagation for training synaptic delay-based artificial neural networks", IEEE Transaction on Neural Networks, Volume: 10, Issue: 4, 1999, pp. 779-789.

[8]. S. Chen, Y. Wu, B. Luk,"Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks", IEEE Transaction on Neural Networks, Volume: 10, 1999,Issue: 5, pp. 1239-1243.

[9]. Bahram G. Kermani, Mark W. White, H. Tory Nagle, "Feature Extraction By Genetic Algorithms for Neural Networks in Breast Cancer Classification", IEEE, 1997, pp.831-832.

[10]. McKay B.; Wills, M.J.; Hidden, H.G.; Montague, G.A. and Barton, G.W. March, "Identification of industrial processes using genetic programming". Proceedings of International Conference on Identification in Engineering Systems, 1996, pp.328-337.

[11]. S. Kirkpatrick, Cl. Gelatt, M. Vecchi, "Optimization by simulated annealing", Volume: 220, no: 4598, 1983, pp. 671 - 680.

[12]. J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis" In *Proceedings of the* IEEE Congress on Evolutionary Computation (CEC), 2000, pp.1507-1512.

[13]. J Kennedy, RC Eberhart, Y.Shi. "Swarm Intelligence", International Journal of Computer Research, 2002, pp.434-452 .

[14]. Xiaohui Hu, Yuhui Shi, and Russ Eberhart, "Recent advances in particle swarm". In Proceedings of IEEE Congress on Evolutionary Computation (CEC), 2004. pp. 90-97.

[15]. .M.R.Senapati, I.Vijaya, P.K.Dash, "Rule Extraction from Radial Basis Functional Neural Networks by using Particle Swarm Optimization." Journal of computer science, Science Publication 3 (8), 2007. pp.592-599