Chapter 6

Ensemble Based Lasso Ridge Radial Basis Function Network

In this chapter, we propose Lasso-Ridge Radial Basis Function Network (LR-RBFN) and Ensemble LR-RBFN strategies in diagnosis of breast cancer. The proposed networks are deployed on the Wisconsin Breast Cancer (WBC) data set, and comparative analysis is carried out. An overview of the Radial Basis Function Network is provided in Section 6.1. The methodology of using a Radial Basis Function Network and k-means clustering is detailed in Section 6.2. The Lasso and Ridge Regularization is covered in detail in Section 6.3. Proposed LR- RBFN is explained in detail in Section 6.4. Proposed Ensemble Learning is covered in Section 6.5. A Novel Modified Gaussian Kernel is discussed in Section 6.6. Method evaluation of proposed Ensemble RBFN is presented in the Section 6.7. The Section 6.8 discussion focuses on the Simulation Results and Conclusions.

6.1 Introduction

Autonomous detection and diagnosis systems built on top of Machine Learning have shown promising results in terms of accuracy and speed. Breast cancer analysis methods have come a long way in the last decade. Multiple automatic classification strategies have been deployed in recent years. The outcomes you get from using various approaches will differ. Concerns, such as the necessity for the creation of more effective methods, must still be addressed. To enhance the classification of breast cancer illnesses, an unique Lasso-Ridge Radial Basis Neural Network (LR-RBFN) and Ensemble LR-RBFN is introduced.

Yuehui Chen et. al. proposed Hierarchical RBF (HRBF) that can be built and expanded according to a set of instructions. Using the same cutting-edge artificial methods, the authors compared their proposed optimised HRBF network to alternatives networks like NN Flexible Neural Tree and RBFN and achived 96.84% classification accuracy [28]. Aye Mya Thandar and Myo Kay Khaing offered a consistency-based method for selecting traits to use in order to narrow down the available options [117]. The proposed method also reduced irrelevant features on breast cancer, lymphography, sick-thyroids, and hepatitis, enhancing the RBFN's performance. In testing, they had an accuracy of 85.19% when using the proposed consistency-based RBF. Alex Alexandridis and Eva Chondrodima proposes method for training RBF classifiers using RBF Neural Networks and Non-Symmetric Fuzzy Means (NSFM) training, with evolutionary simulated annealing (ESA) to optimise the RBF models [8]. The WBCD and WDBC data sets were used to test the effectiveness of their suggested method. It was asserted that their proposed method required less time to calculate than SVMs. In order to compare the efficacy of RBFNN and BPN (Back Propagation Neural Network) strategies and SVM, S. Vijayalakshmi and J. Priyadarshini and Sanaz Mojrian et. al. conducted experiments on the WBCD, WDBC, and WPBC databases [125] [81]. This new system, called ELM-RBF, combines Extreme Learning Machine with the RBF model and it achieves an accuracy of 99.72% during training, 99.23% during testing, and 95.69% during validation, respectively. For the purpose of WBCD classification in breast cancer, Roguia Siouda and Mohamed Nemissi proposed an optimised RBF-NN [111]. Propagation networks and PSO + K-mean networks were compared. They said their accuracy in identifying breast cancer was 97.82%.Vincent F. Adegoke et. al. and Lavanya Doddipalli have successfully analysed and tested their models with WBCD, including the Adaptive Neuro Fuzzy Inference System (ANFIS), Artificial Neural Networks, Recurrent Back-propagation Fuzzy Networks and Back-propagation Neural Networks [5] [33].

6.2 Methodology

6.2.1 Radial Basis Function Network

A Radial Basis Function Network (RBFN) is a type of Artificial Neural Network in the field of mathematical modelling that use Gaussian Function as activation function. As a nonlinear classifier based on supervised machine learning, RBFNs are a specific sort of ANN. They were initially proposed in a study that was published in 1988 by Broomhead and Lowe, who were both researchers at the Royal Signals and Radar Establishment [22] [23] [109]. The network's output is a linear mixture of the input Gaussian's and neuron parameters. RBFN have a wide variety of applications, some of which include the approximation of functions, the prediction of time series, the classification of data and the control of systems. RBF network is based on Cover's theorem.

Unlike other types of neural networks, RBFNs are designed differently from the ground up. Many layers are used in ANN's architecture and non-linearity is introduced through the repeated application of nonlinear activation functions. On the other hand, an RBF network needs only three layers to function properly: an input layer, a single hidden layer with non-linear activation function and a linear output layer. The input layer is not a calculation layer; rather, it only accepts the input data and sends it on to the RBF network's hidden layer. The processing that takes place within the hidden layer of an RBF network is considerably different from that of the majority of neural networks because of the Cover's theorem [31]; this is where the RBF network gets its impressive level of performance. The prediction task, which may include classification or regression is carried out by the output layer. The paradigm of high-dimensional feature transformation, as stated by Cover's theorem, is supported by the fact that it enhances classification by linear separation when features are translated from low-dimensional to high-dimensional spaces [31].

- **Cover's theorem:**
  In 1965, Thomas cover introduced Cover's theorem based on the separability of patterns which is stated as below [31]:
  "A complex pattern classification problem which is non-linearly separable in low-dimensional space can be more likely to be linearly separable in a high-dimensional space, provided that the space is not densely populated."

The RBF network is feed forward neural network consisting of three layers, a input

layer, a single hidden layer and the output layer are having $J_1 - J_2 - J_3$ number of neurons respectively as shown in Fig.6.1 An activation function called $\phi(r)$ is used by each hidden layer neuron, which is non-linear. The Gaussian function applied to all neurons in the hidden layer is often the same. It has the following definition $\phi_i(\vec{x}) = \phi(\vec{x} - \vec{c}_i); i = 1, 2, ..., J_2$ where, $\vec{c}_i$ is center, $\vec{x}$ is input vector and $\phi(x)$ is any basis function. A matrix form of N training pairs, $(\vec{x}_i, \vec{y}_i)|i = 1, 2, ..., n$, can be represented as $Y = W^T\phi$; where $W = [\vec{w}_1, \vec{w}_2, ..., \vec{w}_{J_3}]$ is $J_2 \times J_3$ matrix with $\vec{w}_i = (w_{1i}, w_{2i}, ..., w_{J_2 i})^T$, $\Phi = [\vec{\phi}_1, \vec{\phi}_2, ..., \vec{\phi}_n]$ is $J_2 \times n$ matrix with $\vec{\phi}_i = (\phi_{i1}, \phi_{i2}, ..., \phi_{i_{J_2}})^T$ is output of the hidden layer for the i^{th} sample and $Y = [\vec{y}_1, \vec{y}_2, ..., \vec{y}_n]$ is $J_3 \times n$ matrix with $\vec{y}_i = (y_{i1}, y_{i2}, ...y_{iJ_3})^T$. The learning process revolves around modifying the network's parameters in order to replicate a predetermined set of input-output patterns.

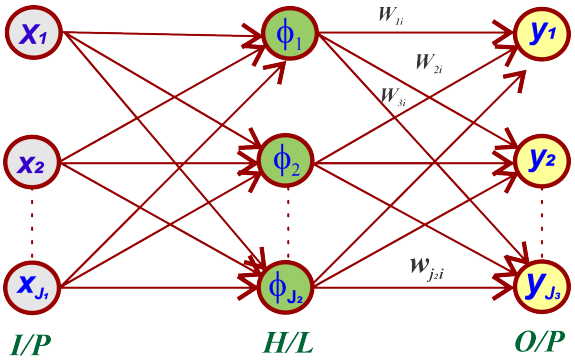


Figure 6.1: Architecture of Radial Basis Function Network (RBFN)

- Input layer: Data is simply passed from the input layer to the hidden layer. No weights are connected through input to hidden layer. The input neurons are fully connected to the hidden neurons and send the information they receive to the hidden neurons.

- Hidden layer: The input, in which the pattern may or may not be linearly separable, is transformed by the hidden layer into a new space that is more linearly separable. Due to the necessity to translate non-linearly separable patterns into higher-dimensional space in order to make them linearly separable, the hidden layer has a higher dimensionality than the input layer. Cover's theorem on pattern separability states that if a pattern is transformed into a higher-dimensional space via nonlinear transformation, then it is more likely to be linearly separable, and so the number of neurons in the hidden layer should be larger than the number of neurons in the input layer [31]. Hidden layer is also known as feature vector. Each node in the hidden layer consist

of individual Basis Function. There are different basis function like Gaussian, Quadratic, Multi-Quadratic, etc. have been used in the experiments.

In the hidden layer, prototype vectors i.e. receptors or centers are calculated, which are vectors from the training set. The distance between receptor and input vector is calculated using norm. Here ϕ is Gaussian activation or Basis function consist of centers C and spread σ. The computation that takes place in the hidden layer can be expressed mathematically as follows in eq. 6.1:

$$\phi_i = \sum_{i=1}^{J_2} e^{\frac{-\|\vec{x_i} - \vec{C}\|^2}{2\sigma_i^2}} \tag{6.1}$$

where, $\vec{x_i}$ is input vector
$\vec{c_i}$ center or prototype vector
σ_i is smoothness parameter or variance-spread of RBF function
ϕ_i is non-linear RBF activation function or output of the i^{th} hidden neuron

Here, norm is taken to be the Euclidean norm. For each node in the hidden layer, center is calculate using k-means clustering algorithm.

- Output layer: The output layer computation is done in the same way as a traditional artificial neural network, which is a linear combination of the input vector (output of hidden layer will become input for output layer) and the weight vector. The Weight vectors between hidden and output layer are learned and updated using optimization algorithm like Gradient Descent, Adam, Particle Swarm optimization, etc. The computation in the output layer can be stated mathematically as follows in eq. 6.2:

$$y_i(\vec{x}) = \sum_{k=1}^{J_2} w_{ki}\phi(\|\vec{x} - \vec{c_k}\|); i = 1, 2, ..., J_3 \tag{6.2}$$

Where, w_i is the weight connection between each hidden node to output node, ϕ_i is the i^{th} neuron's output from the hidden layer and y is network's predicted output.

k-means clustering

k-Means Clustering is an unsupervised learning approach used to address clustering issues in machine learning and data science. This method clusters the unlabeled

data set into distinct categories. Here, k determines how many distinct clusters will be formed; for example, if $k = 2$, only two clusters will be formed, whereas if $k = 3$, there would be three clusters, and so on. It is an iterative technique that splits the unlabeled data set into k distinct clusters, with each data set belonging to only one group with identical or common attributes. There is a centroid assigned to each cluster in this technique. This algorithm's primary goal is to find the smallest possible sum of distances between data points and their respective clusters. As input, the algorithm takes an unlabeled data set, which it then divides into k clusters and iteratively searches until it fails to discover the optimal clusters. When using this procedure, k should have a known value beforehand. k-means clustering is used for primarily two purposes: i) Uses iteration to find the optimal location for k centres, and ii) It gives each data point the k-center that is closest to it. A cluster is formed by the data points that are located in close proximity to a specific k-center.

Algorithm for k-means clustering

1. Decide number of cluster k
2. Initialize k random centroids for each center of each cluster to a different randomly selected training pattern.
3. Assign each data points (training pattern) to the nearest cluster. Then calculated euclidean distance between the training patterns and the cluster centers.
4. After assigning all of the training patterns, then compute the average position for each cluster centre. They then become the new centres of their respective clusters.
5. It is necessary to repeat steps 3 and 4 until the cluster centres remain constant during the succeeding rounds.

Spread (σ)

Once the RBF centres are determined, the k-nearest neighbours approach can be used to determine the spread of each RBF unit. The k closest centres to each of the centres are determined. The Root-Mean Squared distance is used to find the spread value, which is calculated between teh current cluster and its k nearest neighbours. Therefore, if the current cluster centre is c_j, the following expression will describe the σ value:

$$\sigma_i = \sqrt{\frac{\sum_{i=1}^{J_2}(c_j - c_i)^2}{k}}$$

RBF network learning i.e. Weight

RBF network learning can be expressed as the Mean Square Error(MSE) function's minimization which is defined as follows:

$$E = \frac{1}{n}\sum_{i=1}^{n} ||\vec{y_i} - w^T\vec{\phi_i}||^2 = \frac{1}{n}||Y - W^T\Phi||_{F'}^2$$

Where, $Y = [\vec{y_1}, \vec{y_2}, ..., \vec{y_n}]$; $\vec{y_i}$ is the target output for the i^{th} sample in the training set and $||\cdot||_F^2$ is the Frobenious norm defined as $||A||_F^2 = tr(A^TA)$.

Usually, RBF network's learning performed on the two stages, namely find the centers and spread respectively and then update the weights(W) of the network. After calculation of spread and centers, network's error is minimized and weight is calculated using $W = (\Phi^\dagger)^\dagger Y^T = (\Phi\Phi^T)^{-1}\Phi Y^T$. Where, $[\cdot]^\dagger$ is the pseudo-inverse of the matrix within.

6.3 Lasso-Ridge Regularization

The term "Regularization" is used to describe the methods by which machine learning models are updated to prevent over-fitting, under-fitting and to minimize the loss function. Regularization is an essential concept that is utilized to prevent the data from being over-fit, particularly in situations in which the trained data and the test data differ greatly from one another. Over-fitting can sometimes occur when modelling a high-dimensional data set with an excessive number of features (model captures both real and random effects). An overly complex model with a large number of features, especially if those elements are connected, can be difficult to understand. To reduce the variance of the tested data, regularization adds a penalty term to the best fit from the training data and also compresses the coefficients of the predictor variables that have an effect on the output variable. In the process of regularization, the standard practise is to maintain the same number of features while simultaneously decreasing the absolute value of the coefficients. By employing a variety of different sorts of regression approaches, one of which is regularization, we will be able to lessen the magnitude of the coefficients and thereby solve this

problem. Lasso and Ridge Regularization is two techniques to deal with over-fitting problem.

6.3.1 Lasso Regularization

Lasso stands for Least Absolute Shrinkage and Selection Operator and also known as L1 Regularization. It resolves the problem by adding a penalty that is comparable to the total of the absolute values of the coefficients to the cost (loss or error) function , which changes the over-fitted or under-fitted models. Lasso Regularization is another method that can be used to do coefficient reduction. However, unlike other methods, it does not square the magnitudes of the coefficients; rather, it uses the coefficients' actual values. Since negative coefficients are allowed, the sum of the coefficients can be 0. When using L1 regularisation, a penalty is applied that is proportional to the absolute value of the magnitude of the coefficient. Because of the nature of this regularisation, the resulting model may be sparse and contain few coefficients. It is possible that some of the coefficients will become 0 and then can be removed from the model. Mathematically L1 regularization is defined as follows:
Residual Sum of Squares (i.e. Cost or loss or error function) + λ_1 * (Sum of the absolute value of the magnitude of coefficients)

$$L1 = \text{Loss function} + \lambda_1 \sum_{i=1}^{J_3} \|\vec{w_i}\|_1 = \text{Loss function} + \lambda_1 \sum_{i=1}^{J_3} |\vec{w_i}|.$$

Where, λ_1 is Penalty parameter or shrinkage for the errors.
$\vec{w_i}$ is slope of the curve.

When the penalties are higher, the resulting coefficient values are nearer to zero (ideal for producing simpler models). However, sparse models and coefficients are preserved with L2 regularisation. Therefore, as compared to the Ridge, the Lasso Regression model is simpler to understand.

6.3.2 Ridge Regularization

Ridge regularization is also known as L2 Regularization. As a penalty term, L2 adds "squared magnitude" of coefficient as penalty term to the cost function. Mathemat-

ically, it is represented by as follows:

$$L2 = \text{Loss function} + \lambda_2 \sum_{i=1}^{J_3} \|\vec{w_i}\|_2^2 = \text{Loss function} + \lambda_2 \sum_{i=1}^{J_3} \vec{w_i}^2.$$

Adjusting the penalty function's parameters allows us to manipulate the penalty value. The magnitude of the coefficients is decreased as the penalty is increased to a greater level and hence it shrinks the parameters. Because of this, it is utilised to prevent multi-collinearity, and it also minimises the complexity of the model by means of coefficient shrinking. If λ_2 is 0 at this point, then L2 becomes Ordinary Least Square (OLS). On the other hand, if λ_2 is excessively large, then it will add an excessive amount of weight, which will result turn to under-fitting problem. However, the method of selecting λ_2 is crucial. Over-fitting problems can be effectively avoided with this method.

6.4 Proposed Lasso-Ridge Radial Basis Function Network

Over-fitting and under-fitting problems have been addressed in this model by using an RBF network which integrates the $L1$ penalty and $L2$ penalty. While $L1$ regularization seeks to estimate the data's median, $L2$ regularization seeks to do so by estimating the data's mean in an effort to minimize over-fitting.

Weights w_i parameters are added as penalties to the cost function in $L1$ penalty and the squared value of weight w_i is added to the cost function in $L2$ penalty, respectively. The proposed approach uses mean square error function (loss function) to assess the classification model's performance, as illustrated in 6.3.

$$E = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{J_3} (e_{j,i})^2. \tag{6.3}$$

Approximate error at i^{th} output node for the j^{th} pattern is given by 6.4.

$$e_{j,i} = y_{j,i} - \vec{w_i}^T \vec{\phi}_j. \tag{6.4}$$

A regularization term has been incorporated to the loss function as shown in 6.3, of the novel RBFN to enhance the model's efficiency which is evince in the following

6.5.

$$E = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{J_3}(e_{j,i})^2 + \lambda_1 \sum_{i=1}^{J_3} \|\vec{w}_i\|_1 + \lambda_2 \sum_{i=1}^{J_3} \|\vec{w}_i\|_2^2. \tag{6.5}$$

Where, $j = 1, 2, ..., n$ is number of pattern, $i = 1, 2, ..., J_3$ which is number of output neuron and $m = 1, 2, ..., J_2$ which is number of neurons at the hidden layer and λ_1 and λ_2 are tuning parameter of $L1$ and $L2$ respectively which controls bias-variance trade off.

Here, $L1$ penalty term is non-differentiable function. Gradient may not exist or undefined. $L1$ penalty function is non-differentiable when $w_i = 0$. Hence, sub-gradient of $L1$ term is given as follows:

$$\frac{\partial}{\partial w_i}|w_i| = \begin{cases} -1 & ; w_i < 0 \\ [-1, 1] & ; w_i = 0 \\ 1 & ; w_i > 0 \end{cases}$$

By taking the first derivative of E with respect to $w_{m,i}$, we have following equation.

$$\frac{\partial E}{\partial w_{m,i}} = -2\rho_j + 2w_{m,i}z_j + 2\lambda_2 w_i + \begin{cases} -\lambda_1 & ; w_i < 0 \\ [-\lambda_1, \lambda_1] & ; w_i = 0 \\ \lambda_1 & ; w_i > 0 \end{cases}$$

Where, $\rho_j = \frac{1}{n}\sum_{j=1}^{n} \phi(\|\vec{x}_j - \vec{c}_m\|)\Big\{y_{j,i} - \sum_{k\neq m} w_{k,i}\phi(\|\vec{x}_j - \vec{c}_m\|)\Big\}$ and $z_j = \frac{1}{n}\sum_{j=1}^{n} \phi^2(\|\vec{x}_j - \vec{c}_m\|)$.

Hence, weights can be determined using following (6.6):

$$w_{m,i} = \begin{cases} \frac{-\lambda_1 + 2\rho_j}{2(z_j + \lambda_2)}; & \rho_j < \frac{-\lambda_1}{2} \\ 0 & \frac{-\lambda_1}{2} < \rho_j < \frac{-\lambda_2}{2} \\ \frac{\lambda_1 + 2\rho_j}{2(z_j + \lambda_2)}; & \rho_j < \frac{-\lambda_2}{2} \end{cases} \tag{6.6}$$

Weight updation is given by (6.7).

$$w_{m,i}^{new} = w_{m,i}^{old} - \eta_1 \frac{\partial E}{\partial w_{m,i}}. \tag{6.7}$$

Again taking first derivative of loss function E with respect to $\vec{c}_m$, we have

$$\frac{\partial E}{\partial \vec{c}_m} = -\frac{2}{n} \sum_{j=1}^{n} \sum_{i=1}^{J_3} e_{j,i} \left[\sum_{m=1}^{J_2} w_{m,i} \big(\sigma^2 + \|\vec{x}_j - \vec{c}_m\|^2\big)^{-\frac{3}{2}} \big(\vec{x}_j - \vec{c}_m\big) \left(\frac{-\big(\vec{x}_j - \vec{c}_m\big)}{\|\vec{x}_j - \vec{c}_m\|} \right) \right].$$

Hence, update center $\vec{c}_m$ is given by 6.8.

$$c_m^{new} = c_m^{old} - \eta_2 \frac{\partial E}{\partial c_m}. \tag{6.8}$$

Different types of basis functions can be used in RBFN ([128], [79]; [69]; [94]). In this experiments, we have used Gaussian, Multi-quadratic and Inverse Multi-quadratic basis functions which are defined in the experiment section.

6.5 Ensemble Learning

Ensemble methods are used in machine learning and statistics to improve predicted performance over that of individual learning algorithms [101] [96]. Unlike a statistical ensemble in statistical mechanics, which is normally unlimited, a machine learning ensemble consists of only a specific finite number of different models having considerably more flexible structure. The term "Ensemble Learning" refers to the method of generating and combining several models such as classifiers or experts to address a specific computational intelligence challenge. The primary goal of ensemble learning is to enhance the performance of the classification, function approximation, etc of the model. Ensemble learning can also be used for data fusion, incremental learning, non-stationary learning, error correction and assigning confidence to the model's choice.

Ensemble methods are classified into two types: sequential ensemble and parallel ensemble approaches. For example, Adaptive Boosting is an ensemble method that sequentially generates base learners (AdaBoost). Dependence between basis learners is encouraged by the sequential creation of the base learners. When this is done,

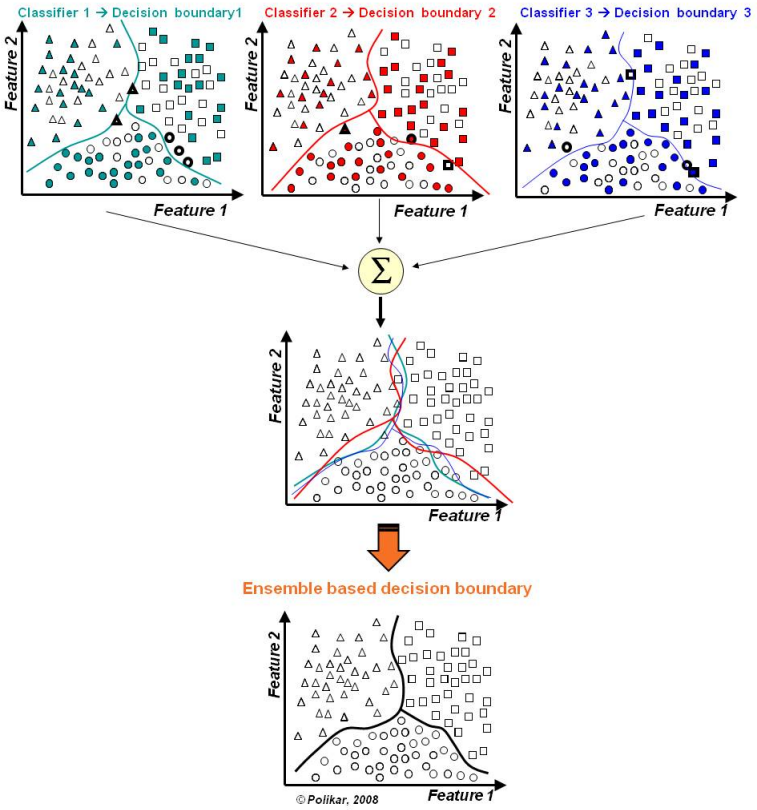


Figure 6.2: Combining an ensemble of classifiers for reducing classification error and/or model selection [96]

the model's performance can be enhanced by giving more weight to the types of students that were initially under-represented. The basis learners for parallel ensemble methods like random forest are created simultaneously. By generating new basis learners in parallel, parallel techniques enhance independence in the base learners. As a result of their independence, the base learners considerably reduce the average mistake. The vast majority of ensemble approaches use just one algorithm for base learning, which leads to all of the base learners having the same characteristics. Base learners of the same type and with similar characteristics are said to be homogeneous. Alternatively, other approaches use a variety of different types of foundation learners to create a variety of different types of ensembles. Learners of various types make up a heterogeneous base.

Hyun Chul Kim et. al. introduced the SVM ensemble with bagging and boosting as an alternative to SVM ensemble [60]. The suggested SVM ensemble with bagging or boosting outperforms a single SVM in terms of classification accuracy in simulations, including those for IRIS data classification, hand-written digit classification and fraud detection. With the help of bagging and boosting, Jork Stork et. al. created SVM ensembles [112]. They came up with the novel idea of examining

SVM ensembles composed of several kernel types like linear, polynomial and RBF. For big data set, their goal is to train a single strong SVM ensemble classifier with fewer resources than a single SVM on all the data. In the context of food quality assessment, Fady Mohareb et. al. proposed an ensemble technique for classification and regression based on SVM classifiers [80]. Min Wei Huang et. al. Used different kernel functions such as linear, polynomial, RBF and compared SVM with SVM Ensemble for prediction of breast cancer in order to determine which was superior [49]. Tina Elizabeth Mathew proposed an ensemble decision tree classifier for the diagnosis of breast cancer using Wisconsin diagnostic data [75]. Josef Kittler et. al. used different decision tree classifier including the C4.5 and J48 classifier as well as the Standard CART, REPTree, Hoeffding Tree, BFTree, Logostic model tree etc. Nevertheless, it has been well known that combining multiple classifier or classifier ensembles, which is another major study field in pattern classification ([63]). Vincent F. Adegoke et. al. proved that they are known to perform better then single classifiers. The author suggested that EKF-RBFN-Adaboost was able to speed up training times while also improving the efficiency of RBFN's training setting ([5]).

Proposed Ensemble RBF network

In order to make the most practical and precise prediction for breast cancer diagnosis, we develop machine learning models. It is possible that a single model, due to issues like unpredictability and bias, doesn't always make the most accurate forecasts. Errors can be reduced and prediction quality can be improved by combining many models into a single model. The purpose of ensemble learning is to achieve a specific task by combining the results of numerous base models. As a result of their simplicity, they have earned the title "weak learners," and this is why ensemble learning takes use of them. A strong learner is produced when a weak learner joins forces with other weak learners. Those with stronger learning abilities tend to do better than those with weaker ones. In order to enhance the performance of the machine learning process, we create and analyse the Ensemble RBF learning model.

RBF networks have various advantages over other types of networks. These advantages are due to the fact that RBF networks use RBF as their activation function and have only one hidden layer. These functions are highly powerful when it comes to approximation. RBF networks have the ability to learn online. They are very resistant to input noise and have good generalization ability.

It is probable that a Standard RBFN will not learn the exact parameters of the

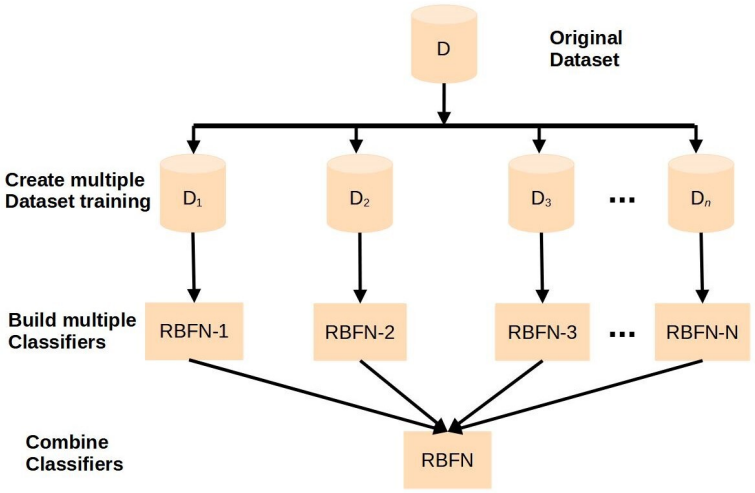


Figure 6.3: A general architecture of the Ensemble RBFN

global optimum because of the approximation techniques that are employed in the implementation of a Standard RBFN. The RBFN that was obtained via the process of learning might not be adequate for classifying all of the unknown test samples. Because of the possibility that a Standard RBFN would not always produce the global ideal classification performance across all test samples, we suggest employing a group of RBFNs as a solution. A general layout for the planned Ensemble RBFN can be seen in the Fig.6.3

There have been many different ways of constructing an ensemble classifier. A few different methods, including bagging, boosting, blending and stacking are utilized to choose the training samples. Particularly, we concentrated on advanced ensemble bagging as a model for the technique to follow. After the training has been completed, it is important to integrate a number of the RBFNs that were previously trained into a single model. The bagging procedure is determined by the outcome of the voting with the most votes winning.

6.6 A Novel Modified Gaussian Kernel and Particle Swarm Optimization

6.6.1 A Novel Modified Gaussain Kernel

It is known that Euclidean distance is not only the measure that can be used to find whether the feature vectors are spaced out uniformly or not. In this case Euclidean distance will have no measurable impact. To address this problem, we proposed a

generalized RBF kernel along with cosine based RBF kernel which can be formulated as follows:

$$\phi(x_i, c_i) = \phi_1(x_i, c_i)\phi_2(x_i, c_i)$$

Where, $\phi_1(x_i, c_i) = \frac{x_i \cdot c_i}{||x_i||||c_i||+\gamma}; \gamma > 0$ is cosine similarity and $\phi_2(x_i, c_i) = \frac{||x_i - c_i||^2}{2\sigma^2}$ is Gaussian kernel. Where, c and σ has their usual meaning. Hence, it is defined as follows:

$$\phi(x_i, c_i) = \left(\frac{x_i \cdot c_i}{||x_i||||c_i|| + \gamma} \right) e^{\frac{||x_i - c_i||^2}{2\sigma^2}}$$

The cosine similarity metric assesses similar two vectors in an inner product space. It evaluates whether or not two vectors are pointing in nearly the same direction by measuring the cosine vector and comparing the results. The similarity attains the value between -1 and +1 . If the cosine similariy is +1 then it implies that two vectors are aligned or have the same direction, if it attians 0 values then it means x and y are orthogonal to each other and if it attains -1 then it indicates that both vectors are aligned but in opposite direction.

6.6.2 Particle Swarm Optimization

The bio-inspired Particle Swarm Optimization (PSO) technique is a straightforward approach to find the optimum solution. This optimization method is distinct from others because it does not rely on the gradient or any differential form of the objective function. Moreover, it uses few hyperparameters. An initial population of random solutions is used to begin off the search for the best possible solution, which is then refined through successive generational updates. In 1995, Kennedy and Eberhart proposed Particle Swarm Optimization technique. Bird flocking behaviour is the inspiration behind Particle Swarm Optimization. When birds flock together, they forage in an area at random. Unfortunately, not every bird knows where the food is hidden. However, they know the food's distance each time. The PSO algorithm is fast optimization method that can be implemented quickly since it only requires a small number of parameters to achieve optimal performance, in addition to its fast convergence speed, great robustness, strong global search capability and its independence from gradient information.

PSO begins with a population called a swarm, which is initialised with a set of

random values called particles (solutions), and then looks for the optimal position of particles by updating its population at each iteration. Each particle is updated in each iteration by taking the average of its two most recent fitness values and applying the appropriate fitness function to that problem. The first number is the most optimal starting point for each particle, and it is refereed as personal best (pbest). Another value is the best swarm position that has been attained thus far by any swarm; this best value is known as a global best (gbest). Every particle has a velocity that determines its direction and moves it to a new location. It is possible for the magnitude velocities to grow significantly when the velocities are calculated. Therefore, it is best to limit the maximum velocity (Vmax) that the user can select in order to effect PSO performance. The basic algorithm of PSO is mention as follows:

1. Initialize each particle i of the swarm, with random values for position (x_i) and velocity (v_i) in the search space according to the dimensionality of problem.

2. Evaluate fitness value of particle by using fitness function.

3. Compare the value obtained from the fitness function from particle i, with the value of Pbest.

4. If the value of the fitness function is better than the Pbest value, then update the particle position to takes the place of Pbest.

5. If the value of Pbest form any particle is better than gbest, then update gbest = Pbest.

6. Modify the position X_i and velocity V_i of the particles using following equation. Calculate particle position by:

$$x_i^{t+1} = x_i^t + v_i^t t$$

Calculate velocity by:

$$v_i^{k+1} = wv_i^k + c_1 r_1 (xBest_i^t - x_i^t) + c_2 r_2 (gBest_i^t - x_i^t)$$

Where, xBest is best particle position, gBest is best group position, r_1 and r_2 are two random parameter within [0,1], $i = 1, 2, ..., M; d = 1, 2, ..., n$, t+1 is the current iteration number, t is the previous iteration number, w is the inertia weight, c_1 and c_2 are the acceleration constant which are usually between [0, 2].

7. If the maximum number of iterations or the ending criteria is not achieved so far, then return to step 2.

6.7 Method Evaluation

In order to illustrate the utility of the proposed method as a medical diagnostic tool, the LR-RBFN and Ensemble LR-RBFN are applied to the Wisconsin Breast Cancer data set (WBCD) ([34]). By employing Multivariate Imputation by Chained (MICE) Equation, 16 missing values are assigned in the data set. Appendix A provides a brief overview of the data set, while this link: http://archive.ics.uci.edu/ml provides further information.

A combination of bagging and majority voting is employed in the assembly of proposed Ensemble LR-RBFN for testing and approval of the results. We also investigate the efficacy of the proposed Ensemble LR-RBFN in comparison to the Standard RBFN. Moreover, Standard RBFN is also compared to the proposed hybrid of Lasso and Ridge RBFN (LR-RBFN) to test the performance of the model.

The WBCD data set is pre-processed using normalization and scaled into [0,1] using the following (6.9), once it is loaded into the RBF network to ensure that no attributes are missing and that the data is consistent.

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}. \tag{6.9}$$

In this experiment, the Gaussian, Multi-quadratic and Inverse Multi-quadratic functions have been employed as the basis function in RBFN. Which are defined as follows in eqs. 6.10, 6.11 and 6.12

$$\phi\big(x, c\big) = \exp\left(\frac{-\|x - c\|^2}{2\sigma_m^2}\right); c > 0, c \in \mathbb{R} \tag{6.10}$$

$$\phi\big(x, c\big) = \big(x^2 + c^2\big)^{\frac{1}{2}}; c > 0, c \in \mathbb{R}. \tag{6.11}$$

$$\phi\big(x, c\big) = \frac{1}{\big(x^2 + c^2\big)^{\frac{1}{2}}}; c > 0, c \in \mathbb{R}. \tag{6.12}$$

Centers and weights must be determined for use in the RBF networks learning

process. The smoothness parameter σ must also be calculated for a RBFN with a Gaussian RBF. There are two stages to the RBF network learning process. Phase one begins with the identification of suitable centers and their associated standard deviations. The weights are modified during the second phase using (6.7). RBF network centers are found using k-means clustering method.

WBC data set consists of 699 records each having 9 features with integral values ranging from 1 to 10. A single incidence is classified as either benign or malignant. We simulated the network with diverse number of RBFs. Performance and error reduction are two of the many advantages of employing an ensemble rather than a single model. An evaluation of the RBFN was carried out utilising Gaussian, Multi-quadratic and Inverse Multi-quadratic basis function to train and test the network. When calculating the network output and the desired output, we employed the Mean Square Error.

The proposed model's efficacy in classifying breast cancer as benign or malignant is evaluated using test data acquired after RBFN has been trained. Once the model has been developed, it can be used to predict the class labels of previously unknown data on a test set. Measurement of the model's performance based on test data is typically useful since it offers an unbiased estimate of generating errors.

For WBC data set, the model is built with nine input neurons, one hidden layer and one output neuron. In order to reach the optimum rate of precision, it is required to alter the number of neurons in the hidden layer. To emphasize the convergence of the models, they are trained using learning rates of 0.01, 0.001, and 0.0001. In addition, the proposed models are validated for several train-test pairs, such as 90-10%, 80-20% and 70-30%.

6.8 Simulation Results and Conclusion

6.8.1 Experiments for LR-RBFN

A new Lasso (L) and Ridge (R) penalty term is added in the loss function. Loss function can be strengthened by including a regularization terms as shown in (6.5). It is essential to use the regularization terms in order to reduce error by fitting a function appropriately on the available training set while avoiding the over fitting. As a result of penalizing for complexity, this regularization strategy aids in the reduction of variance in the proposed model. Due to fusion of Lasso and Ridge

regularization to the model, we effectively yield better performance for the training data as well as better generalization ability for the test data. Lasso and Ridge regularization is introduced into the loss error function in order to penalise heavy weights.

The embedded approach of $L1$ and $L2$ penalty reduces the coefficients of 'unimportant' features to zero or close to zero. Having a hybrid of the two penalties, $L1$ and $L2$, is advantageous since it allows for a higher performance on some issues than a model with only one penalty applied. λ_1 and λ_2 are another hyperparameters that regulates the weighting of the total of both penalties to the loss function. For λ_1 and λ_2 values ranging from 0 to 1, our proposed model attained an accuracy of 98.57% for $\lambda_1 = 0.03$ and $\lambda_2 = 0.02$ and learning rates: $\eta_1 = 0.75$ and $\eta_2 = 0.75$. Here, λ_1 and λ_2 also controls bias-variance trade-off. Accuracy, precision, recall, the F-score, Kappa statics (k), and the Matthews Correlation Coefficient (MCC) are all employed to improve the overall performance of the proposed LR-RBFN model.

The pseudo-code of the proposed LR-RBFN model is outlined in Algorithm 4:

Algorithm 4 Proposed LR-RBFN model

Step 1. Pass training pattern $\left\{(\vec{x}_p, \vec{y}_p)\middle| p = 1, 2, ..., N\right\}$.

Step 2. Set learning rate η for different range.
Set stopping criteria $\epsilon = 0.001$.
Set initial weights randomly.

Step 3. Calculate centers $\vec{c}_m$ using k-means clustering.

Step 4. Calculate spread using σ_m using $\sigma_m = \sqrt{\frac{1}{p}\sum_{k=1}^{p}(t_i - t_k)^2}$.

Step 5. Find the value of each RBF's i.e. $\phi_i(x)'s$ for each input vector using Gaussian basis function.

Step 6. Take linear combination at hidden layer to output layer using following equation: $y_i(\vec{x}) = \sum_{k=1}^{J_2} w_{ki}\phi(\|\vec{x} - \vec{c}_k\|); i = 1, 2, ..., J_3$.

Step 7. Compute network's error E from 6.3 and compare with desired output.

Step 8. Update weights and centers using 6.7 and 6.8.

Step 9. While stopping criteria is true, stop else repeat the procedure until convergence.

For Multi-quadratic and Inverse Multi-quadratic radial basis function same procedure is performed. The flow of proposed LR-RBFN algorithm is depicted in Fig.6.4.

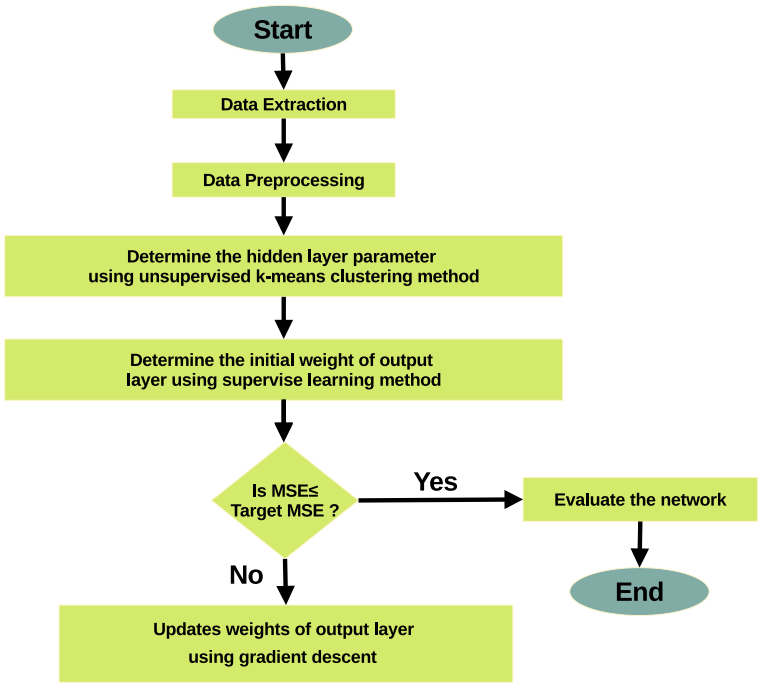


Figure 6.4: Flow chart of proposed LR-RBFN algorithm

Results and discussion

For several sets of centres, learning rates, and train-test sets (such as 90-10%, 80-20% and 70-30%), the Standard RBFN is simulated using various radial basis functions like Gaussian, Multi-quadratic and Inverse Multi-quadratic. The following Table 6.3, Table 6.4 and Table 6.5 illustrate the best performance of the model. The efficacy of the model is assessed using kappa statistic (k) and MCC and it is indicated in tables.

Because of the embedded method, both penalties i.e. $L1$ and $L2$ can be balanced out, which can lead to better results in some cases than a model with just one or the other penalty. In addition to the "lambda (λ)" hyperparameter, the loss function can be tweaked to change the weighting of the total of both penalties. Adding penalties to the loss function during training to encourage standard models with smaller coefficient values. Better results are attained when compared to a standard model.

The results of Table 6.1, Table 6.2 and Table 6.3 states that Gaussian basis function attains 97.14% of classification accuracy in 6.42 seconds.

Figure 6.5, 6.6 and 6.7 shows the iteration process at each epoch for the best results

Table 6.1: Performance analysis for 90-10% train-test set - Standard RBFN

BF	NC	CI	II	Acc(%)	Pre (%)	Re (%)	F(%)	T (sec)	k	MCC
GA	50	68	2	97.14	1.0	92.0	96.0	6.42	0.94	0.94
MQ	20	66	4	94.29	87.0	95.0	91.0	3.84	0.87	0.87
IMQ	15	67	3	95.71	1.0	87.0	93.0	2.15	0.90	0.90

Table 6.2: Performance analysis for 80-20% train-test set - Standard RBFN

BF	NC	CI	II	Acc(%)	Pre (%)	Re (%)	F(%)	T (sec)	k	MCC
GA	35	135	5	96.43	97.0	90.0	94.0	7.52	0.91	0.91
MQ	20	133	7	95.0	87.0	98.0	92.0	3.98	0.88	0.89
IMQ	20	133	7	95.0	98.0	89.0	93.0	3.71	0.89	0.90

(a) (b) (c)

Figure 6.5: Epoch vs error for Gaussian RBF for different train-test set - Standard RBFN: (a) 90-10% train-test set, (b) 80-20% train-test set, (c) 70-30% train-test set

of the Standard RBFN for various learning rates, various centres with Gaussian, Multi-quadratic and Inverse Multi-quadratic RBFs for various train-test pairs.

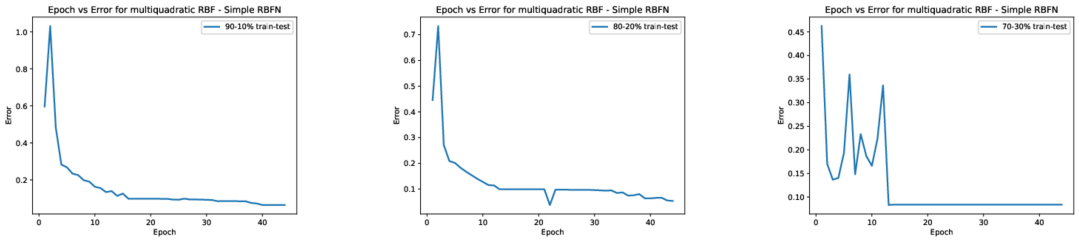
Table 6.4 presents the confusion matrix and accuracy for the best centres and best learning rate of the Standard RBFN with Gaussian, Multi-quadratic, and Inverse Multi-quadratic RBFs in various train-test sets. It can be seen in Table 6.4 and Table 6.5 that of these three RBF's, Gaussian RBF outperforms them all. The Gaussian RBF achieves the best results with the 50 centres and with $\eta_1 = 0.75$ and $\eta_2 = 0.75$, with the highest values of 98.14% accuracy, 100% precision, 92.0% recall and 96.0% F-score achieved in just 6.42 seconds.

Table 6.5 shows the best performance of the proposed LR-RBFN with multiple centres and different tuning parameter pairs, i.e. λ_1 and λ_2. Performance matrices such as the confusion matrix, accuracy, precision, recall, F-score, k, and MCC are

Table 6.3: Performance analysis for 70-30% train-test set - Standard RBFN

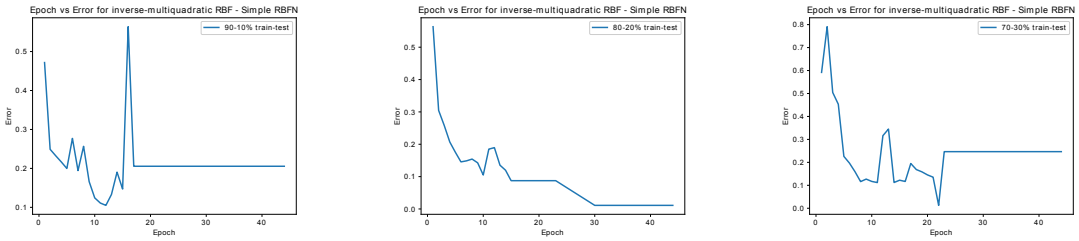
BF	NC	CI	II	Acc(%)	Pre (%)	Re (%)	F(%)	T (sec)	k	MCC
GA	50	201	9	95.71	94.0	95.0	94.0	9.35	0.91	0.91
MQ	40	201	9	95.71	97.0	90.0	93.0	3.48	0.92	0.90
IMQ	15	194	16	92.38	91.0	83.0	87.0	1.75	0.82	0.82

*Terms: BF - Basis Function, NC - No. of Centroids, CI - Correct Instances, II - Incorrect Instances, Acc - Accuracy, Pre - Precision, Re - Recall, F - F-score, T (sec) - Time, GA - Gaussian, MQ - Multi-Quadratic, IMQ - Inverse Multi-Quadratic



(a) (b) (c)

Figure 6.6: Epoch vs error for Multi-quadratic RBF for different train-test set - Standard RBFN: (a) 90-10% train-test set, (b) 80-20% train-test set, (c) 70-30% train-test set



(a) (b) (c)

Figure 6.7: Epoch vs error for Inverse Multi-Quadratic RBF for different train-test set - Standard RBFN: (a) 90-10% train-test set, (b) 80-20% train-test set, (c) 70-30% train-test set

used to evaluate each model. For 20 centres and $\eta_1 = 0.75$ "and" $\eta_2 = 0.75$ with $\lambda_1 = 0.03$ and $\lambda_2 = 0.02$, the model attains a classification accuracy of 98.57%.

Fig.6.8 depicts the results of a performance assay i.e. accuracy, precsion, recall and F-score of several λ_1 and λ_2 combinations. According to the results, when compared

Table 6.4: Confusion matrix and accuracy with different RBF's for different train-test set - Standard RBFN

RBF's	Train-Test set	Center	Accuracy (%)	Confusion Matrix
GA	90-10%	50	97.14	45 0 2 23
	80-20%	35	96.43	97 1 4 38
	70-30%	50	95.71	128 5 4 73
MQ	90-10%	20	94.29	46 3 1 20
	80-20%	15	94.28	87 1 7 45
	70-30%	40	95.71	137 2 7 64
IMQ	90-10%	15	95.71	47 0 3 20
	80-20%	20	95.0	84 1 6 49
	70-30%	15	92.38	141 5 14 59

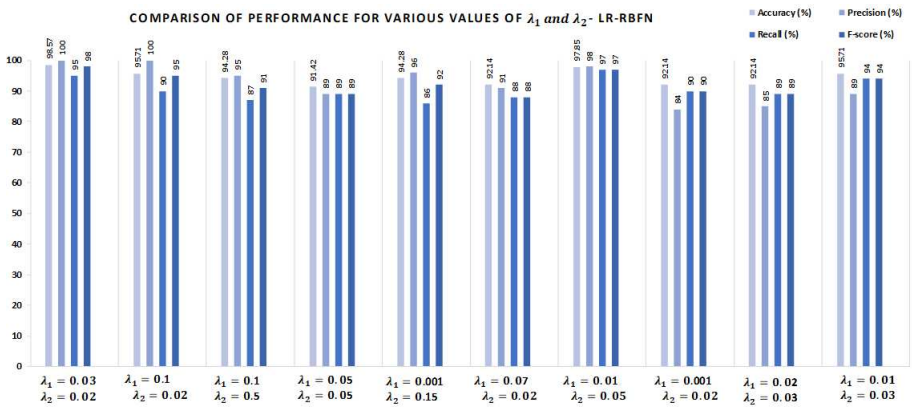


Figure 6.8: Comparison of performance for various values of λ_1 and λ_2 with different centers - LR-RBFN

to the standard RBFN, LR-RBFN attained the best classification accuracy 98.57%, and it did it in just 12.56 seconds as depicts in Fig.6.9.

Fig. 6.10 represents the training error at each epoch for different values of centers, λ_1 and λ_2 for proposed LR-RBFN.

Table 6.5: Performance analysis - LR-RBFN

λ_1	λ_1	C	CM	Acc(%)	Pre (%)	Re (%)	F	T (sec)	k	MCC
0.03	0.02	20	47 1 0 22	98.57	100.0	95.0	98.0	12.56	0.97	0.97
0.1	0.02	15	41 0 3 26	95.71	100.0	90.0	95.0	12.40	0.91	0.91
0.1	0.5	30	91 2 6 41	94.28	95.0	87.0	91.0	15.05	0.87	0.87
0.05	0.05	22	77 6 6 51	91.42	89.0	89.0	89.0	15.87	0.82	0.82
0.01	0.15	17	86 2 6 46	94.28	96.0	86.0	92.0	12.37	0.88	0.88
0.07	0.02	23	90 4 7 39	92.14	91.0	88.0	88.0	25.78	0.82	0.82
0.01	0.05	15	94 1 2 43	97.85	98.0	97.0	97.0	14.46	0.95	0.95
0.001	0.02	27	80 9 2 49	92.14	84.0	90.0	90.0	29.82	0.84	0.84
0.02	0.03	23	84 8 3 45	92.14	85.0	89.0	89.0	17.63	0.83	0.83
0.01	0.03	35	86 6 0 48	95.71	89.0	94.0	94.0	15.71	0.91	0.91

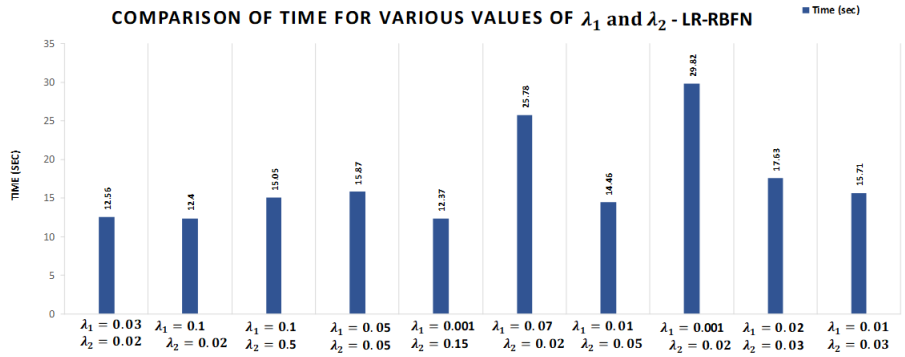


Figure 6.9: Comparison of time for various values of λ_1 and λ_2 with different centers - LR-RBFN

6.8.2 Experiments for Ensemble LR-RBFN

The ensemble learning technique known as "bagging" or "bootstrap aggregation", reduces variance in a noisy data set. To build the Ensemble LR-RBFN, we employ a bagging method. Bagging is a process in which several RBFN are trained independently using the bootstrap approach and then they are aggregated using the majority voting method.

There is a training and a testing subset of the experimental data. Bootstrap sam-

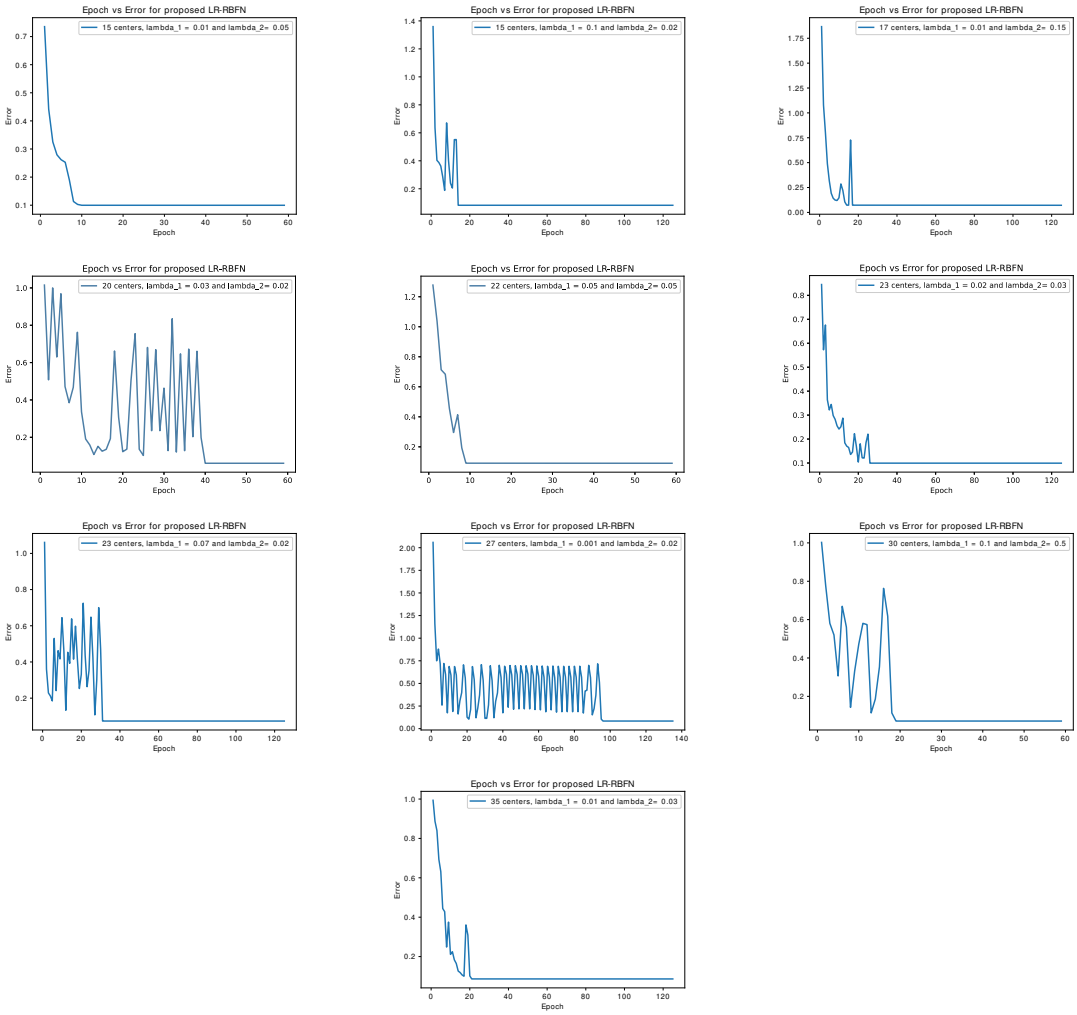


Figure 6.10: Epoch vs error Gaussian RBF for different values of centers and λ_1 and λ_2 - LR-RBFN

pling is used to reproduce overlapping random samples of the training subset for each RBFN classifier, depending on the size of training subset and classification algorithm. Once the ensemble has been used to evaluate the unseen test data, all RBFN classifiers are fused together using a maximum voting technique. We anticipate the class label $y_i(\vec{x})$ by applying the following formula to obtain a majority (plurality) vote from each classifier RBFN-N : $y_i(\vec{x}) = max_{i=1} \sum_{i=1}^{N} y_i$.

The Fig.6.11 depicts the overall execution of the ensemble based LR-RBFN with bagging strategy. First and Foremost, the original data set D is divided into two subsets: a training subset (TR) and a testing subset (TS). The training subset TR is subdivided into training tr1 and testing ts1. Subset is created using re-sampling technique. After that, ts is bootstrapped into a 300 samples. This procedure is repeated several times until the entire ensemble is generated. The ensemble is used in order to label the unseen testing subset D samples. In order to enhance the overall

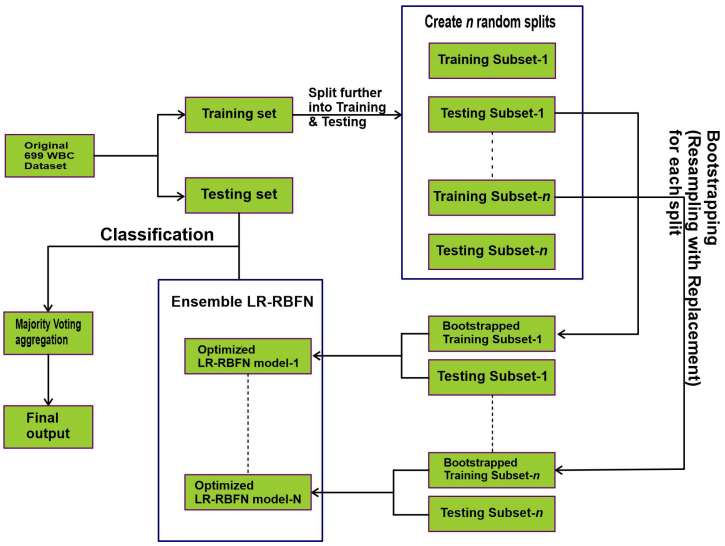


Figure 6.11: Flowchart of development process for Ensemble LR-RBFN.

performance of the proposed ensemble model, accuracy, precision, recall, F-score, kappa statics (k) and Matthews Correlation Coefficient (MCC) are used.

Table 6.6: Comparison of Standard RBFN vs Proposed LR-RBFN vs Ensemble LR-RBFN

Classifiers	Evaluation criteria			
	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)
Standard RBFN	97.14	100.0	92.0	96.0
LR-RBFN	98.57	100.0	95.0	98.0
Ensemble LR-RBFN	100.0	100.0	100.0	100.0
Novel RBFN	97.14	100.0	92.0	65.0

Results and discussion

The bagging approach is used to construct an ensemble-based system for predicting breast cancer, and the results are promising. The data set was divided into two subsets: train and test. In order to complete the optimization procedure depicted in Fig. 6.11, the training subset for each of the classifiers in the ensemble is bootstrapped to generate a subset of 90%, 80%, and 70% samples consisting of 699 data, which was then partitioned into train-test subsets. We train number of LR-RBFNs independently over the replicated training data set. The accuracy of the ensemble prediction was calculated using the testing subset, and the final output was com-

Table 6.7: Performance comparison of various author's work

Author	Year	Algorithm	Accuracy (%)
T. Kiyan and T. Yildrim	2004	RBFN	96.18
A. M. Thandar and M. K. Khaing	2012	Consistency based RBFN	85.19
S. Vijayalakshmi and J. Priyadarshini	2017	RBFN	98.26
S. Mojrian et. al.	2020	ELM-RBF	99.23
Our Study	2022	Standard RBFN	97.14
		LR-RBFN	98.57
		Ensemble LR-RBFN	100.0

puted using the majority voting aggregation approach. The use of ensemble methods helped to alleviate (reduce) prediction variance. Analysing the performance of multiple models and averaging them together helped to keep the spread of performance under control. The variability in performance is minimized by averaging the results of the multiple model at the same time. The bagging process contributes to the reduction of variation. As a result, model's capacity to forecast the prediction is enhanced.

A total of 2, 3, 5, 7, and 10 LR-RBFN classifiers are incorporated into each ensemble system. This quantity is found to be adequate for stabilising the accuracy of the ensemble prediction. Various LR-RBFN were added at a time to each ensemble in an accumulative fashion to test for stability, which is illustrated in Fig. 6.12 and Fig. 6.13 by a comparison of accuracy and time for several models with different train-test combinations for Ensemble LR-RBFN.

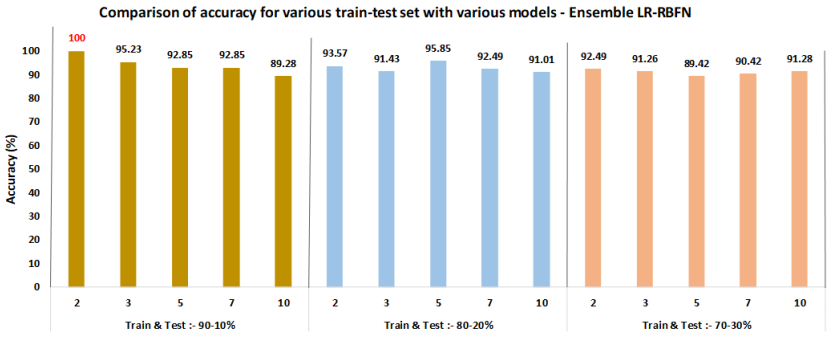


Figure 6.12: Comparison of accuracy for various models with different train-test - Ensemble LR-RBFN.

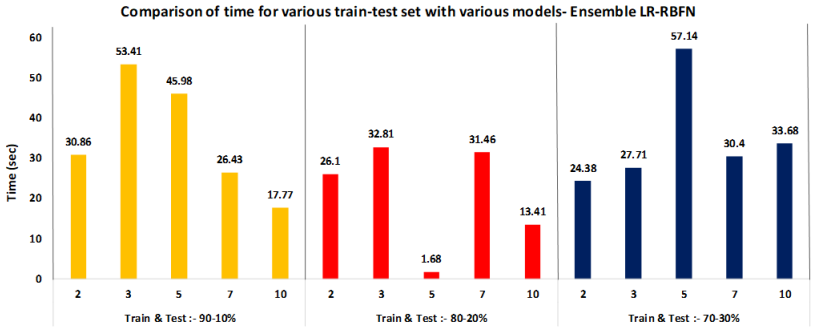


Figure 6.13: Comparison of time for various models with different train-test - Ensemble LR-RBFN.

From the above Fig.6.12 and Fig.6.13, we see that the highest classification accuracy of 100% is achieved with only two LR-RBFN in the Ensemble model in just 30.86 seconds, for 90-10% of train-test set. The above accuracy is obtained using the following parameter values: $\eta_1 = 0.75$, $\eta_2 = 0.75$, $\lambda_1 = 0.08$ and $\lambda_2 = 0.02$ with 30 centres.

We also see that, in comparison to the Standard RBFN performance, the bagging strategy and majority voting approach used in the Ensemble model has increased overall prediction accuracy by 3% approximately, as shown in Table 6.6.

Table 6.7 represents the comparative analysis of the performance of the accuracy obtained by the various researchers in different models with the proposed models in this study. A proposed novel RBF kernel with PSO optimization technique gives 96.23% classification accuracy for WBC data set and 97.14% classification accuracy for WDBC data set as shown in Taable 6.6.

The detailed information of the both data set are given in the Appendix. Also, above all computation is carried out using Python programming and it is given in the Appendix.

6.8.3 Conclusion

In this study, standard RBFN, LR-RBFN and Ensemble LR-RBFN models are developed and implemented on Wisconsin Breast Cancer data set. In the LR-RBFN model, Lasso and Ridge regularization technique is employed along with L1 and L2 penalty into the loss function and obtained better accuracy of 98.57% compare to the accuracy of 97.14% in Standard RBFN. The Ensemble based LR-RBFN, using bagging strategy with majority voting technique is proposed and its accuracy

is compared with LR-RBFN. It is found that, Ensemble based LR-RBFN gives the highest 100.0% accuracy. Also for each model performance analysis in terms of accuracy, precision, recall, F-score is carried out and found that the proposed Lasso-Ridge RBFN (LR-RBFN) and Ensemble LR-RBFN models outperforms the standard RBFN model.