International Journal of Operations Research and Information Systems
Volume 11 • Issue 4 • October-December 2020

Evaluation of Parameter Settings for Training Neural Networks Using Backpropagation Algorithms:
A Study With Clinical Datasets

Leema N., Anna University, Chennai, India
Khanna H. Nehemiah, Anna University, Chennai, India
Elgin Christo V. R., Anna University, Chennai, India
Kannan A., Anna University, Chennai, India

ABSTRACT

Artificial neural networks (ANN) are widely used for classification, and the training algorithm commonly used is the backpropagation (BP) algorithm. The major bottleneck faced in the backpropagation neural network training is in fixing the appropriate values for network parameters. The network parameters are initial weights, biases, activation function, number of hidden layers and the number of neurons per hidden layer, number of training epochs, learning rate, minimum error, and momentum term for the classification task. The objective of this work is to investigate the performance of 12 different BP algorithms with the impact of variations in network parameter values for the neural network training. The algorithms were evaluated with different training and testing samples taken from the three benchmark clinical datasets, namely, Pima Indian Diabetes (PID), Hepatitis, and Wisconsin Breast Cancer (WBC) dataset obtained from the University of California Irvine (UCI) machine learning repository.

KEYWORDS

Artificial Neural Network, Backpropagation Algorithm, Classification, Convergence Rate, Network Parameters

INTRODUCTION

Researchers aim to build a computing system that will operate intelligently like a human brain. The Artificial Neural Network (ANN) facilitates the information processing in an intelligent manner (Akinyokun, 2002; Bezdek, 1993), and is inspired by the biological neural system. A biological nervous system is a large interconnection of neurons located within the brain. The functional equivalent of an artificial neuron is known as computational neuron or a node (Eluyode, Akomolafe & MNCS, 2013). These neurons are structured hierarchically by layers and interconnected between them like the biological nervous systems. Artificial neural network determines the rate of adjustment required for

DOI: 10.4018/IJORIS.2020100104

internal network parameters. This adjustment is known as learning or training the network. The neuron functions are described by the activation function. Activation functions are used in the hidden and the output layer. Hidden layer implements the non-linear activation function, whereas the output layer implements the linear activation function. Linear activation function used in neural network training is purelin and the non-linear activation functions are hardlim, sigmoid and logistic (Sharma, 2014).

Parameters of both biological and Artificial Neural Networks are structures, layers, number of neurons, the functional capabilities of neurons, their learning capabilities, processing elements, connections, strength, processing speed, style of computation, information storage, signal transduction, information transmission communication media selection and fault tolerance. Major difficulty faced in correlating artificial neural networks with biological neural networks are adjusting weights and synaptic strengths. Weights are altered mathematically in an ANN, based on differences in error values. Synaptic strengths are modified in response to synaptic activity. A simple feed-forward system behaves similar to biological neurons and they are used for pattern recognition. Once input values are given to the input layer, neuron computes the output, layer by layer. The dependence of output values and input values require adjusting every weight, and threshold, which can be complex and time consuming. After training is complete, the network is able to give reasonable outputs for any type of input, even if the test data does not match with the training data. This is referred to as the generalization capability of the network. In that case, the ANN attempts to determine the best output depending on its training method.

Based on the structure, ANN is divided into two types, namely, single layer neural network and multilayer neural network. Single layer neural network is used for linearly separable problems, whereas the multilayer neural network is used for linearly non separable problems. One major drawback of a single layer network is that it can predict the output, which is similar to the input pattern. For many practical problems, very similar input patterns may have very different output requirements (FFNN, 2010). To overcome the above limitation multilayer neural network has been developed with one or more hidden layers, called multi-layer perceptron (MLP) networks. Hidden layer in the MLP is used to deal with nonlinear relationships between input features and the output layer is used to obtain the predicted output. This MLP can be used to solve many real world problems like predicting the future trends based on the historical data (Kosko, 1994.). ANN have been implemented in many science and engineering fields such as, decision making and control, biological modeling, health care and medicine, marketing, engineering and manufacturing for classification task (Krasnopolsky & De ´ricChevallier, 2003; Coppin, 2004; Basheer & Hajmeer, 20007; He, Wu & Gong, 1992).

In machine learning, Backpropagation (BP) is a supervised learning algorithm for training the Artificial Neural Network (ANN). Most of the researchers used different BP algorithms to train ANN without knowing the performance of different BP algorithms and network parameter adjustments. Backpropagation training algorithms receive the inputs, adjust the weights and produce the required output. The commonly used supervised training algorithm is gradient descent backpropagation in which the weights are altered based on the quadratic error function (Rumelhart, Hinton & Wiliams, 1986b). The BP algorithm is a universal approximator, which can approximate any smooth function to an arbitrary degree of accuracy, when the network parameters are optimized. Hence the training process needs the appropriate combination of network parameters to obtain higher accuracy in classification. ANN is usually designed for specific applications such as data classification, pattern recognition, optimization, time series prediction, curve fitting, sensitivity analysis, dynamic modeling and the control of systems over time (Rajasekaran & Pai, 2003).

This work provides the comparative study and discusses the impact of various network parameter combinations using twelve different backpropagation training algorithms. The different combination of network parameters was tested. For each combination, training was carried out ten times, and minimum mean squared error was found out. The performance of each BP trained network is evaluated based on the accuracy, convergence rate and MSE. From the evaluation results, the most appropriate combination of network parameters and the best type of BP algorithm are identified.

The rest of the paper is organized as follows: Section 2 presents an overview of related research. Section 3 provides necessary background materials and methods used in the comparative study carried out. The implementation and comparative analysis of the results are discussed in section 4. Section 5 deals with conclusions and scope for future work.

RELATED WORK

Related works carried out by researchers using backpropagation algorithm for training the neural network is discussed below.

Rosenblatt (1958) developed a hypothetical nervous system called perceptron, that answered the questions of "how the information about physical world is sensed", "in what form, the information is remembered" and "how does the information retained in memory influence recognition". This work provides the relationship between psychology and biophysics and predicts the learning curves from neurological variables and vice versa. Rumelhart, Hinton & Wiliams (1986a) introduced a multilayer feed forward network with an error backpropagation method to train the network. The algorithm backpropagates the error repeatedly by adjusting the weights in the network. The weight adjustments to the hidden nodes are independent of input and output features.

Verma & Mulawka (1994) developed a modified Backpropagation (BP) algorithm, which was based on solving the weight matrix in the output layer using the theory of equations and least square techniques. This overcomes the drawback of long training process in the BP algorithm. Drago, Morando & Ridella(1995) developed an adaptive momentum BP algorithm for achieving fast minimum search. In this work, the network weight updation rule is chosen to accelerate the faster convergence in the training process using adaptive momentum term to reduce the error function. This achieves high convergence speed and generalization of the network.

Bossan, Seixas, Caloba,Penha & Nadal(1995) developed a modified BP algorithm for neural classifiers. This method reduced the time to achieve the low Mean Squared Error, by ignoring patterns in the sparse regions in very populated regions of the pattern space until a large number of training epochs occurred. Yu & Chen (1997) considered efficient BP learning using dynamically optimal learning rate and momentum factor. This approach uses the products with respect to the momentum factor and learning rate. The learning rate and momentum term were adjusted using conjugate gradient method at each iteration to reduce the training time. This provides faster convergence to achieve the low mean squared error.

Fukuoka, Matsuki,Minamitani & Ishida (1998) modified the BP algorithm to avoid local minima. The modification is done by multiplying a factor within the range of (0,1) at a constant interval of each connected weight in a network during the training process. This method uses sigmoid activation function when the error rate is high. This overcomes the problem of local minima. Ng, Leung & Luk(1999) developed the generalized BP algorithm with constant learning rate. The network weights in the generalized BP algorithm are approximated by using OrdinaryDifferential Equation (ODE). When the learning rate increases from zero, the generalized backpropagation algorithm provides faster convergence and overcomes the local minima problem.

Zweiri, Whidborne & Seneviratne (2003) introduced a third term in BP algorithm for ANN training. The existing BP algorithm uses two terms for training namely, learning rate and momentum factor. The author introduced the third term named as proportional factor, which speedup the weight updating process. This overcomes the slow convergence and local minima problem of the existing BP algorithm. Zhang & Suganthan (2016) did a survey on training neural networks using randomized algorithms. The drawback of the existing neural network training algorithms is that it tunes the parameters iteratively. This suffers from local minima and slow convergence. This training approach uses randomization either to change the data distribution or to change network configuration.

Christopher, Nehemiah & Kannan (2015) developed a rule based clinical decision support system to diagnose the presence or absence of allergic rhinitis. The developed clinical decision support system is based on the results of an intradermal skin test. This framework compares the efficiency of five traditional classification approaches namely, k-nearest neighbourclassifier (KNN), Decision tree classifier (C4.5), Multi-layer perceptron classifier Support Vector Machine and Naïve Bayes classifier (NB). The rule based approach presents the knowledge model in an IF-THEN rule format. As rule based Clinical Decision Support System (CDSS) provides better comprehensibility than other classification approaches, physicians prefer them.The clinical decision support system achieved an accuracy of 88.31% for diagnosis of allergic rhinitis. Furthermore, the clinical decision support system can be used as an aid for decision support for junior clinicians in the absence of allergy specialist.

Nahato, Nehemiah & Kannan (2016) developed a classifier which combines fuzzy logic and Extreme Learning Machine (ELM). The classifier was tested with UCI machine learning repository datasets namely, Cleveland Heart disease (CHD), Pima Indian Diabetes (PID) and Statlog Heart Disease (SHD). The datasets were preprocessed by using the nearest neighbor method based on Euclidian distance. Fuzzification of selected features was done using Trapezoidal Membership Function. The ELM uses single hidden layer feed forward neural network for classification. The classifier achieved 73.77%, 93.55%, 94.44% and 92.54% accuracies for CHD with five class labels, CHD with two class labels SHD and PID datasets respectively.

Nahato, Nehemiah & Kannan (2015) used rough set with backpropagation neural network for classification of clinical datasets. The clinical datasets were obtained from the University of California Irvine (UCI) machine learning repository. Missing values from the clinical dataset was handled by either imputing or rejecting based on the percentage of the missing values. Handling missing values and selection of attributes are performed using indiscernibility relation. The selected attributes are used to train back propagation neural network. The network is a single hidden layer feed forward neural network having Tangent sigmoid activation function applied to the hidden neuron and linear activation function applied to the output neuron. The accuracy obtained from their proposed method is 97.3%, 98.6%, and 90.4% for hepatitis, Wisconsin Breast Cancer, and Cleveland Heart Disease datasets respectively.

Leema, Nehemiah & Kannan (2016) developed a optimization technique for training neural network using differential evolution with global information (DEGI) with BP algorithm for clinical datasets. The DEGI was developed by drawing the relative advantages of particle swarm optimization's (PSO) global search ability and differential evolution's modified mutation operation is used to improve the search exploration of PSO. The DEGI algorithm is used for global search and the BP algorithm is used for local search. This optimization technique overcomes drawback of the local minima problem of BP and premature convergence due to stagnation problem of PSO. The classifier performance was tested using three datasets namely, Pima Indian Diabetes, Cleveland Heart Disease and Wisconsin Breast Cancer obtained from the UCI machine learning repository. The developed classifier provides 98.52%, 85.71% and 86.66% of accuracies for the above datasets.

Elgin, Nehemiah, Minu & Kannan (2019)developed a framework for clinical decision support system (CDSS) which uses Correlation based ensemble feature selection and gradient descent back propagation neural network for classification. The Hot deck imputation has been used for handling missing values and min max-normalization was used to transformation. Correlation based ensemble feature selector is performed to get the optimal feature set by the rejection of high similarity features from majority voting on the output of Differential evolution, LION optimization and glow-worm swarm optimization. The CDSS was experimented on Hepatitis dataset and Wisconsin Diagnostic Breast Cancer (WDBC) dataset from University of California Irvine (UCI) Machine Learning repository and was observed that it was obtained an accuracy of 95.51% for Hepatitis and 98.47% WDBC.

Wu, Zhao, Zhang, Sang, Dong, & Jiang (2020, January) proposed a framework for diagnosing early diabetic retinopathy detection using back-propagation artificial neural network (BP-ANN) improved by a priori knowledge. The retinal blood vessels are segmented using a fuzzy clustering algorithm based on texture features. Based on the a priori knowledge obtained from the experienced ophthalmologists, geometric features of blood vessels namely, width and tortuosity are extracted.

A total of 72 retinal vessel features were extracted and are classified using an Improved BP Neural Network Classifier. A total of 240 fundus images were used for experimentation which were obtained from 120 early-stage DR and 120 normal participants. The average accuracy of 10 randomization tests with different hidden neurons based on BP-ANN were obtained. The maximum accuracy obtained for BP-ANN using a priori knowledge was 98.46%.

Geetha, Aprameya & Hinduja (2020) proposed a diagnostic system for diagnosing dental caries from digital radiographs. The framework comprised of preprocessing steps such as Laplacian filtering, window based adaptive threshold, morphological operations, statistical feature extraction. Back propagation neural network used to classify a tooth surface as normal or having dental caries. The dataset consists of 49 caries and 56 sound dental X-ray images obtained from SJM Dental College Chitradurga, India using intra oral Gendex X-ray machine with RVG sensor was experimented. The system achieved an accuracy of 97.1%.

Sudha (2017) developed a CDSS that uses Genetic algorithm based rough set method for selecting relevant features and Back Propagation Neural Network for classification. The CDSS named as Application Specific Intelligent Computing (ASIC) decision support system. The CDSS has been tested with breast cancer, fertility diagnosis and heart disease data set from UCI machine learning repository. Sigmoid function is used as the activation function and MSE is used the fitness function for BPNN. The CDSS provides 93%, 97.61% and 92.3% for the above datasets respectively.

Helwan, Idoko & Abiyev (2017) proposed an automated Breast tissue classifier using two different neural networks namely feed forward neural network using the Backpropagation learning algorithm (BPNN) and Radial Basis Function Network (RBFN) . Breast Tissue dataset from the UCI machine learning repository is used for experimentation. Feature selection is performed through a filter approach by computing the information gain. Seventy percentage of the samples were used for training and thirty percentage for testing. The performance of BPNN and RBFN are compared in terms of accuracy. The classification accuracies obtained from the BPNN and RBFN are 93.39% and 94.33% respectively. It can be inferred that the Radial basis function network outperformed the back propagation network for classifying six different breast tissues. The performance of the classifier is measured in terms of accuracy, minimum error, maximum epochs and training time.

Desai, Giraddi, Narayankar, Pudakalakatti & Sulegaon (2019) performed a comparison of two classifiers namely BPNN and Logistic Regression (LR). The performance of the classifier is tested with Cleveland Hear Disease (CHD) dataset obtained from UCI machine learning repository. The accuracies of the classifier is 85.074% and 92.58% for BPNN (nonparametric) and LR (parametric) models, respectively. The BPNN used MSE as the optimization function with 10-fold cross validation.

Ravindra, Sriraam & Geetha (2018, February) in their work have developed a BPNN classifier for the diagnosis of Chronic Kidney Disease. Details of 230 patients collected from a local hospital in Karnataka, India is used for experimentation. The objective of the work is to discriminate between chronic kidney disease and non-chronic kidney disease. The BPNN is used MSE as the optimization function and Tan-sigmoid as the activation function. Levenberg-marquardt back propagation algorithm is used for the training of neural network. The developed method provides an accuracy of 95.3% for the diagnosis of Chronic Kidney Disease.

Tarle & Jena (2017, August) have proposed a BPNN for the diagnosis of heart disease. Experiments were conducted on CHD dataset from UCI machine learning repository. The BPNN used sigmoid function as the activation function and MSE as the optimization criterion. Min-max Normalization was used to normalize the dataset. The performance of the classifier is evaluated using Five-fold cross validation. The accuracy obtained from their proposed method is 83% for CHD dataset.

Paing, Hamamoto, Tungjitkusolmun & Pintavirooj (2019) in their work, have proposed a Computer Aided Diagnosis (CAD) system for detecting and staging lung cancer from computedtomography (CT) images. The proposed CAD system has two stages of classification; the first stage discriminate the true tumour nodules from other false lesions. The second stage intents to classify the associated stages of the truly predicted tumours. The proposed CAD was developed and tested using 1560 CT exams from four popular public datasets: LIDC-IDRI, NSCLC-Radiomics-Genomics, NSCLC-Radiomics and NSCLC Radio genomics. The BPNN is used in both the stages of classification. The developed CAD achieved a detection accuracy of 92.8%, a sensitivity of 93.6%, a specificity of 91.8%, a precision of 91.8%, a F-score of 92.8%, and an AUC of 96.8%. For the staging, the proposed CAD achieved an accuracy of 90.6%, a sensitivity of 77.4%, a specificity of 97.4%, a precision of 93.8%, an F1-score of 79.6% and an AUC of 84.6% respectively.

Agharezaei, Agharezaei, Nemati, Bahaadinbeigy, Keynia, Baneshi & Agharezaei (2016) proposed a CDSS to assist diagnosis and prediction of the risk level of pulmonary embolism in patients, by means of artificial neural network. Two types of artificial neural networks, namely Feed-Forward Back Propagation and Elman Back Propagation were used for the research. 294 Patients admitted in educational hospitals affiliated with Kerman University of Medical Sciences, located in the south eastern Iran were used for experimentation. 80 percent of the dataset is randomly assigned as training data and 20 percent is assigned as testing data. Both Feed-Forward Back Propagation and Elman Back Propagation ended up with the same performance (93.23% accuracy). The advantage of Feed-Forward Back Propagation Network has a higher convergence speed and requires fewer neurons in its hidden layer.

This work provides a comparative study on variations in various network parameters using different BP algorithms. The training algorithms are tested using three clinical datasets obtained from the UCI machine learning repository. The experimental results show, which combination of network parameter values and the training algorithms provides faster convergence and minimum error in the classification process.

MATERIALS AND METHODS

In this section, the materials and methods that have been used to investigate the performance of twelve different BP algorithms with the impact of variations in network parameter values for neural network training are discussed.

Mean Square Error

The MSE computes the difference between the neural network output and the target value. The main objective of ANN training is to minimize the MSE and improve the ability of classification. The MSE $\left(E_k\right)$ is computed using Equation (1):

$$E_k = \sum_{i=1}^{O} \left(z_i^k - C_i^k\right)^2 \tag{1}$$

where z_i^k is the neural network output from the network, C_i^k is the target output and E_k is the mean squared error of the network. The error is backpropagated until minimum mean square error is achieved. Although several algorithms are used to train the ANN, most commonly used supervised algorithm is backpropagation algorithm.

Backpropagation (BP) Algorithms

BP algorithm was introduced by Rumelhart, Hinton and Wiliams (1986a). BP algorithm is a supervised learning algorithm for training the ANN. The BP training includes the following steps: initialization of weights, MSE computation, optimization of weights and backpropagation of errors. The BP algorithm is classified into two types namely, faster and slower training algorithms (Bezdek, 1993.). The faster training algorithms falls into two main categories, namely, heuristic and standard numerical optimization. Heuristic techniques are developed from the performance analysis of the standard steepest descent algorithm.

The heuristic techniques are further divided into two types, namely, Gradient Descent with variable learning rate BP and Resilient BP. The standard numerical optimization algorithms are divided into three types such as, Conjugate Gradient algorithms, Quasi Newton and Levenberg-Marquardt. The Conjugate Gradient algorithms are further divided into four types namely, Conjugate Gradient backpropagation with Fletcher-Reeves updates, Conjugate Gradient backpropagation with Polak-Ribiere updates and Conjugate Gradient backpropagation with Powell-Beale Restarts and Scaled Conjugate Gradient. The Quasi Newton algorithm is further divided into two types such as BFGS algorithm and One Step Secant algorithm.

The slower training algorithms are gradient descent algorithms and are not used for solving practical problems, because the training process is very slow. The gradient descent algorithms has two modes, they are, incremental mode and batch mode. The input is applied to the network and then, the weights are updated and gradient is computed in the incremental mode of training. In the batch mode of training, all inputs are applied before the network weights are updated. The BP training strongly depends on the network parameters and training algorithms (Eluyode, Akomolafe & MNCS, 2013).

ANN Parameters

The ANN parameters used to train BPNN are namely, initial weights and biases, number of hidden layers and the number of neurons per hidden layer, activation function, number of training epochs, learning rate, minimum error and momentum term. Selection of optimal set of initial weights and biases for training the neural network reduces the training time and initial error. Hidden layer is used to solve complex non-linearly separable problem.

Multiple hidden layers and the number of neurons in the hidden layer are used, when accuracy is the main criteria and there is no limitation for the complexity of the network and training time. During training, the learning rate controls the rate of change in weights and biases of the network. Momentum parameter prevents the system from converging to local minima. The activation function determines the complexity and performance of the network. It plays a major role in the convergence of the learning algorithms. If the network parameters are not chosen properly, then the network slows down the training process.

Initial Weight Selection

The training algorithms are very sensitive to the random selection of initial weight values. Selection of an optimal set of initial weight values for neural network training reduces the training time and initial error. Training iterations are reduced, if the initial weights chosen are closer to the true minimum. An inappropriate choice of initial weights leads to getting stuck in local minima.

Researchers introduced several weight initialization techniques. Yam & Chow (2000) implemented a weight initialization technique for feed forward networks based on a linear algebraic method and Cauchy‘s inequality. This method ensures the hidden neuron’s output is in the active region, which means the activation function’s derivative has a larger value. When the optimal initial weights are chosen, this method reduces the initial error and increases the convergence rate. Hence the number of iterations to reach the error criterion is significantly reduced.

Masters (1993) used least square method for weight initialization. The researcher had used neural networks with one hidden layer and the weights between input and hidden layer are initialized by using simulated annealing and genetic algorithm. The output layer weights are calculated using singular value decomposition. Nguyen &Widrow (1990) developed an algorithm for weight initialization where all the hidden nodes are scattered uniformly in the input space, which results in substantial improvement of the learning speed of the network. The initial weights and biases in the region of interest are distributed by dividing into smaller intervals. As the initial weights are divided into smaller intervals, the input pattern is learned quickly by the network.

Number of Hidden Layers and Number of Neurons Per Hidden Layer

The hidden layer is an intermediate layer between the input and the output layer. Hidden layer is a collection of neurons and an activation function is applied to it. If the problem is linearly separable, then there is no need for the hidden layer. Hence an activation function is used in the input layer to solve linearly separable problems. In case of non-linearly separable problems, there is a need for one or more hidden layers with activation function applied to it. If the number of neurons in the hidden layer is too low, then underfitting occurs. If the number of neurons in the hidden layer is too high, then overfitting or memorization of the training dataset will occur (Alsmadi, Omar & Noah, 2009; Karsoliya, 2012). Different approaches have been followed by other researchers in the computation of number of hidden layers and number of neurons in each hidden layer.

Boger & Guterman (1997) suggested that the hidden layer size is 2/3rd of the input layer. Berry & Linoff (1997) suggested that the hidden layer size should be less than twice the size of the input layer. Blum (1992) stated that the hidden layer size is in between the input layer and the output layer. Morshed & Kaluarachi (1998) demonstrated that the size of the hidden layer is (2n+1), where n is the number of neurons in the input layer. The size of the hidden layer can be determined by the activation function used in the neurons, the training algorithm, the neural network architecture and training samples in the dataset. Multiple hidden layers are used in applications, where there are no limitations on training time and accuracy is the main criteria. The limitation of using multiple hidden layers in the neural network leads to the problem of local minima (Liu, Starzyk & Zhu 2007).

Activation Function

The activation function strongly influences the complexity and performance of neural networks and plays a major role in the convergence of the learning algorithms (Chandra & Singh, 2004; Duch & Jankowski1999; 2001; Saduf, 2013; Singh & Chandra, 2003.). Several activation functions are used in the ANN training such as Linear, Sigmoid, Sigmoid Stepwise, Sigmoid Symmetric, Sigmoid Symmetric Stepwise, Gaussian, Gaussian Symmetric, Elliot, Elliot Symmetric, Linear Piecewise and Linear Piece Symmetric (Sibi, Jones & Siddarth, 2013). To increase the speed of the training process, several modifications have been done by researchers in the activation functions. Masters (1993) determined that the local minima problem arises when the error saturation in the hidden layer occurs. To overcome the local minima problem, the author introduced a new algorithm by adjusting the sigmoid activation functions in the hidden layer of each neuron, which provides slight modification of weights between the hidden and the output layer. The sigmoid function is adjusted by varying the gain parameter of the hidden neurons. The gain parameter modification is done according to the degree of approximation of the desired output in the output layer.

Nguyen & Widrow (1990) introduced an improvement to the basic BP by adjusting the slope of the activation function of the output layer nodes and using different learning rates for the hidden and output layer nodes. Wang, Tang, Tamura, Ishii & Sun, (2004) implemented an algorithm for modifying the gradient based search direction by adaptively varying the slope parameter (gain) of the sigmoid function.

Learning Rate (η)

The learning rate is a tunable factor that controls the speed of the training process. The sigmoid activation function used in the BP algorithm will slow down the training process, when the output is near to 0 or 1, asthe learning rate is very small and results only in a slight change in the weight adjustments. Lee, Chen & Huang (2001) found the error saturation condition is the main cause for the premature convergence. To overcome the error saturation problem, the author introduced an error saturation prevention function (ESP) which is a parabolic function to the learning rate in the nodes of the output layer. The ESP function scales up the learning rate within the range of [0, 1]. However the learning rate for the hidden layer is very small when the actual output of a unit reaches the saturation area of 0 or 1. The learning rate with constant value of 0.01 is used to overcome the above drawback (Wani, 2014). This constant value was chosen after analyzing many experimental results whereas to improve the learning speed and faster convergence.

Momentum

In BP learning, the impact of learning rate reveals that a smaller learning rate gives smaller changes of weights in the network from one iteration to the next. If the learning rate is too large, it results in larger change in weights and makes the network unstable. To overcome this problem, a momentum term (*m*) is added to the weight updation rule. The momentum affects the weight to be tuned in up slope instead of down slope. To overcome the above limitations in learning, researchers have introduced dynamically varying momentum term. Swanston, Bishop & Mitchell, (1994) introduced an adaptive momentum approach by considering the current negative gradient and the last weight changes. If the current weight change vector is similar in direction to the previous weight change, then the momentum term is increased. If the current weight change is in opposite direction to previous weight change, then the momentum is reduced to zero.

Limitations of BP Algorithms

'Over-fitting' or memorization of data and slow learning problems will occur, when the number of hidden layers and neurons in the hidden layer are too high. When the number of hidden layers and neurons in the hidden layer are too low, 'under-fitting' or unable to learn the data will occur. If the learning rate is too small, the learning process is done slowly and can easily get stuck in the local minima. If the learning rate is too big, instability or poor performance will occur in the network. The parameters are fixed in the network using trial and error method. Hence successful application of BPNN requires time and experience.

Data Set Description

The BP training methods has been experimented with three clinical datasets namely, Pima Indian Diabetes (PID), Hepatitis and Wisconsin Breast Cancer (WBC) dataset obtained from the UCI machine learning repository (Lichman,2013). The description of clinical dataset is presentedin Table 1 to Table 3.

Pima Indian Diabetes

The PID dataset is the result of a research survey carried out in the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), United States. PID dataset samples are taken from Pima Indian Heritage, which contains the details of female patients having gestational diabetes of age group greater than 21 years. The dataset samples were collected during the first trimester of pregnancy.The PID data set has 768 samples with eight features and one class label is associated with each sample to indicate whether the individual is affected with Gestational diabetes or not. Among 768 samples collected, 268 samples (34.9%) had been diagnosed with Gestational diabetes and 500 samples (65.1%) without Gestational diabetes. The description of the attributes in PID dataset is shown in Table 1.

Hepatitis

Hepatitis dataset was donated by the Jozef Stefan Institute. Hepatitis dataset consist of 155 instances with 19 features including class label.The class label Histology describes whether the patient with hepatitis will live or not. The dataset has 123 instances with 'Live' class and 32 instances with 'Die' class. Six features have numerical data type and the remaining thirteen attributes have binary data type. The description of attributes in the Hepatitis dataset is shown in Table 2.

Table 1. Description of attributes in Pima Indian Diabetes Dataset

S.No.	Attribute	Description
1	Preg	Number of pregnancies
2	Plas	Plasma glucose concentration in an oral glucose tolerance test
3	Pres	Diastolic blood pressure
4	Skin	Triceps skin fold thickness
5	Insu	2-Hour serum insulin
6	Mass	Body mass index
7	Pedi	Diabetes pedigree function
8	Age	Age of an individual
	Class	Tested positive / negative

Table 2. Description of attributes in Hepatitis Dataset

S.No.	Attribute	Description
1	Age	Age of an individual
2	Sex	Gender
3	Steroid	Use of anabolic steroids
4	Antivirals	Use of Anti-virals
5	Fatigue	Extreme tiredness
6	Malaise	A vague feeling of bodily discomfort
7	Anorexia	Lack or loss of appetite for food
8	Liver big	Enlargement of Liver
9	Liver firm	Firmness of the liver
10	Spleen palpable	Enlargement of spleen
11	Spiders	Blood vessels near the skin's surface due to the increased estrogen levels
12	Ascites	Accumulation of fluid in the peritoneal cavity
13	Varices	Bleeding from varices
14	Bilirubin	Amount of bilirubin in a blood sample
15	Alk phosphate	Level of alkaline phosphatase
16	SGOT	Amount of serum glutamic oxaloacetic transaminase in blood
17	Albumin	Amount of serum albumin protein in the clear liquid portion of the blood.
18	Protime	Time taken for blood plasma to clot
	Histology	Die or Live

Wisconsin Breast Cancer

Wisconsin Breast Cancer (WBC) dataset has been created by Dr. William H. Wolberg from the University of Wisconsin hospitals. The WBC dataset has 699 instances with nine features from breast fine needle aspiration. The class label for an instance can be Malignant (Cancerous) and Benign (non cancerous). The dataset has 241 instances with malignant class and 458 instances with benign class. All features have numerical data type ranging from 1 to 10. The description of attributes in the WBC dataset is shown in Table 3.

Table 3. Description of attributes in Wisconsin Breast Cancer Dataset

S.No.	Attribute	Description
1	Clump Thickness	Assesses if cells are mono or multi-layered.
2	Uniformity of Cell Size	Evaluates the consistency in size of the cells in the sample.
3	Uniformity of Cell Shape	Estimates the equality of cell shapes and identifies marginal variances.
4	Marginal Adhesion	Quantifies how much cells on the outside of the epithelium tend to stick together.
5	Single Epithelial Cell Size	Relates to cell uniformity, determines if epithelial cells are significantly enlarged.
6	Bare Nuclei	Presence and size of nuclei
7	Bland Chromatin	Rates the uniform “texture” of the nucleus in a range from fine to coarse.
8	Normal Nucleoli	Determines whether the nucleoli are small and barely visible or larger, more visible, and more plentiful.
9	Mitoses	Describes the level of mitotic (cell reproduction) activity.
	Class	benign, malignant

DESIGN AND IMPLEMENTATION

This research work investigates the parameter settings for twelve different BPNN training is carried out. The system framework consists of preprocessing subsystem, training subsystem and classification subsystem.

Preprocessing Subsystem

The preprocessing subsystem handles noisy, missing and irregular values. Table 4 presents the number of features, classes and instances of the three clinical datasets obtained from the UCI machine learning repository.

Table 4. Number of features, classes and instances

S.No.	Dataset	Number of Features	Number of Classes	Number of Instances
1	WBC	9	2	683
2	PID	8	2	768
3	Hepatitis	19	2	155

Handling Missing Values

Hepatitis and Wisconsin Breast Cancer datasets have missing values, whereas Pima Indian Diabetes dataset has no missing values. In this research work, missing values are handled by rejecting or imputing using the steps given below.

Input: Clinical Dataset
Process:
Step 1: Count missing values for each feature.
Step 2: Eliminate features where missing values are greater than 25%, else retain the feature.
Step 3: Count missing values in each instance.
Step 4: Eliminate instances with greater than 25% missing values, otherwise retain them.
Step 5: Impute the missing values of the remaining instances with most frequent feature values corresponding to that class.
Output: Clinical dataset without missing values.

The Hepatitis dataset has 167 missing values. The features alkphosphate and protime have the missing values greater than 25%, hence they are removed from the dataset. The WBC dataset has 16 missing values. The feature Bare Nuclei have 16 missing values. These missing values are handled by imputing the most frequent values of the corresponding class. The PID dataset has no missing values, but there are noisy values in the dataset. In hepatitis dataset six samples have missing values greater than 25%, hence these samples are rejected from the dataset. The dataset is reduced to 149 samples from 155 samples. Table 5 shows the number of missing values in each dataset.

Table 5. Details about the number of missing values in each dataset

S.No.	Data Set	Presence of Missingness	Number of Missing Values	Number of Noisy Values
1	WBC	YES	16	-
2	PID	NO	-	432
3	Hepatitis	YES	167	-

Smooth Noisy Data

In Pima Indian Diabetes dataset, the value zero corresponds to each feature is considered as noisy value. In this research work, noisy values are handled by eliminating or imputing attribute values using the steps below.

Input: Clinical dataset without missing values.
Process:
Step 1: Count noisy values for each feature.
Step 2: Eliminate features where noisy values are greater than 25%, else retain the feature.
Step 3: Count noisy values in each instance.
Step 4: Eliminate instances with greater than 25% noisy values, otherwise retain them.
Step 5: Impute the noisy values of the remaining instances with most frequent feature values corresponding to that class.

In Pima Indian Diabetes dataset, among the 768 samples in the dataset 432 samples have one or more zero values associated with it. The number of instances greater than 25% features having value 0, is 256. The number of instances is reduced from 768 to 512. Among 512, 176 instances have feature value zero. The value zero is replaced by frequently occurring values of an attribute belonging to that class. From the 512 instances, 343 instances indicate the absence of diabetes (class 0) and 169 instances indicate the presence of diabetes (class 1). There is no noisy value in the WBC and hepatitis datasets.

Output: Clinical dataset without noisy values.

Data Normalization

Input: Clinical dataset without noisy values.

Process:

Normalize irregular feature values in the clinical dataset into a specified range using Min-Max normalization (Kamber, Han & Pei, 2012) which is presented in Equation (2):

$$Normalized\left(X\right)=\frac{E-E_{min}}{E_{max}-E_{min}}\left(E_{new\_max}-E_{new\_min}\right)+E_{new\_min} \tag{2}$$

where E is the feature value to be normalized, X is the normalized feature value, E_{min} is the minimum value of the feature, E_{max} is the maximum value of the feature, E_{new_max} and E_{new_min} represents the normalized value within the range [0,1].

Output: Normalized Clinical Dataset

Training Subsystem

The training subsystem uses multi layer feed forward neural network with one hidden layer for training the neural network classifier. The input layer corresponds to each feature in the dataset. The hidden layer neurons is structured using Equation (6) and the output layer with output nodes corresponding to the class label. The activation function used is sigmoid. The following Equations (2) and (3) are used to compute the sigmoidal activation functions (Mhaskar & Micchelli, 1994):

$$logsig\left(n\right)=1\ /\left(1+exp\left(-n\right)\right) \tag{3}$$

$$tansig\left(n\right)=2\ /\left(1+exp\left(-2*n\right)\right)-1 \tag{4}$$

where n is the number of input features in the network.

Backpropagation Training Algorithm

Notations Used

Input: Training Dataset
Process:

Step 1: Initialize the input layer with n input nodes, the hidden layer with H hidden nodes and the output layer with O output nodes.

Phase 1: Feed Forward Phase

Step 2: Compute the output of the j^{th} hidden node using Equation (5):

$$f\left(y_j\right) = 1 \Big/ \left(1 + \exp\left(-\left(\sum\nolimits_{i=1}^{n} w_{ji} x_i - \theta_j\right)\right)\right), \text{j= 1, 2..., H} \tag{5}$$

Step 3: Compute the output of the output layer using Equation (6):

$$z_k = \sum_{j=1}^{H} w_{kj} f\left(y_j\right) \; k = 1, 2, \ldots\ldots, O \tag{6}$$

Step 4: The number of hidden nodes in the hidden layer is represented in Equation (7):

$$H = (2n + 1) \tag{7}$$

Step 5: Compute Mean Squared Error (MSE) E_k using Equation (8):

$$E_k = \sum_{i=1}^{O} \left(z_i^k - C_i^k\right)^2 \tag{8}$$

Phase 1: Backpropagation of Error

Step 6: Compute the error in the network with respect to the weights and biases increment, which is represented in Equation (9):

$$\Delta w_k = \eta\left(z_i - c_i\right) x_i \tag{9}$$

Phase 1: Weight and Bias Update

Step 6.1: The error in the network with respect to weights and biases are computed and is updated using **Gradient Descent BP** algorithm. The updation is a negative gradient of the performance function and is represented in Equation (10):

$$\Delta w_k = -\eta . g_k \tag{10}$$

Step 6.2: Compute the error in the network weights and biases. They are updated using **Gradient Descent with Momentum** algorithm. The momentum term is added by a fractional change of the new weight, which is represented in Equation (11):

$$\Delta w_k = -\eta_k . g_k + p \Delta w_{k-1} \tag{11}$$

Step 6.3: Compute the error in the network with respect to the weights and biases increment using **Gradient Descent Variable Learning Rate BP** which is represented in Equation (12):

$$\Delta w_k = \eta . \frac{\Delta E_k}{\Delta w_k} \tag{12}$$

Step 6.4: Compute the error in the network with respect to the weights and biases. The increments are calculated using **Gradient Descent with Momentum and Variable Learning Rate BP** algorithm, in which the variable learning rate and the momentum term are added using Equation (13):

$$\Delta w_k = p\Delta w_{k-1} + \eta . p . \frac{\Delta E_k}{\Delta w_k} \tag{13}$$

Step 6.5: Compute the error in the network with respect to the weights and biases. This computation is performed during **Resilient BP** algorithm. the temporal behavior of sign function in this algorithmis used to determine the direction of the weight update.Weightupdation is calculated using Equation (14):

$$\Delta w_k = -sign\left(\frac{\Delta E_k}{\Delta w_k}\right) . \Delta w_{k-1} \tag{14}$$

Step 6.6: Compute the error in the network with respect to the weights and biases. The increments are computed using **Conjugate Gradient Backpropagation with Fletcher-Reeves Updates** algorithm. In this algorithm, the line search starts in the steepest descent direction. The line search method determines the optimal current search direction α. The next search direction β is determined by conjugate to previous search directions. This produces faster convergence than steepest descent directions and is represented in Equation (15) to (18):

$$p_0 = -g_0 \tag{15}$$

$$\Delta w_k = \alpha_k . p_k \tag{16}$$

$$p_k = -g_k + \beta_k p_{k-1} \tag{17}$$

$$\beta_k = \frac{g_k^{'}}{g_{k-1}^{'}} \frac{g_k}{g_{k-1}} \tag{18}$$

Step 6.7: Compute the error in the network with respect to the weights and biases increment using **Conjugate Gradient Backpropagation with Polak-Ribiere Updates** algorithm. In this algorithm, new search direction is computed as the product of the previous change in the gradient with the current gradient divided by the square of the previous gradient and is represented in Equation (18).

Step 6.8: Compute the error in the network with respect to the weights and biases increment using **Conjugate Gradient Backpropagation with Powell-Beale Restarts** algorithm. In this algorithm, the direction of search periodically resets to the negative of the gradient and is represented in Equation (19):

$$\left|g_{k-1}^{'} g_k\right| \geq 0.2 g_k^{\,2} \tag{19}$$

Step 6.9: Compute the error in the network with respect to the weights and biases increment using **Scaled Conjugate Gradient (SCG)** algorithm. The algorithm is based on conjugate directions calculated using Equations (18) and (19).

Step 6.10: Compute the error in the network with respect to the weights and biases increment using **Quasi-Newton BFGS** algorithm. In this algorithm the second derivative of the Hessian matrix is computed using Equation (20):

$$\Delta w_k = -H_k^{'} g_k \tag{20}$$

Step 6.11: Compute the error in the network with respect to the weights and biases increment using **One Step Secant** algorithm. The search starts in the negative direction of the performance gradient. The next search direction is computed from the new gradient and the previous gradient according to the Equation (21):

$$\Delta w_k = -g_k + z_{k-1} * \Delta w_{k-1} + \beta_k * \Delta g_{k-1} \tag{21}$$

Step 6.12: Compute the error in the network with respect to the weights and biases increment using **Levenberg-Marquardt** algorithm which is represented in Equation (20), (22) and (23):

$$H' = J'J \tag{22}$$

$$g = J'E_k \tag{23}$$

Output: Backpropagation Trained Neural Network Classifier.

Classification Subsystem

The classification subsystem uses the BP trained ANN classifier for disease diagnosis. The efficiency of the developed BP trained ANN classifier has been tested with three clinical datasets namely, Wisconsin Breast Cancer, Pima Indian Diabetes and Hepatitis obtained from the UCI machine learning repository.

Input: Testing Samples
Process:
Step 1: Preprocessthe input features using the steps given in section 4.1.
Step 2: Preprocessed samples are given as the inputs to the backpropagation trained ANN classifier to obtain diagnostic results.
Output: Diagnostic Results.

EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the results of the experiments along with analysis of BPNN classifier. The expriments were conducted on personal compute with Intel (R) core (TM) 2 Duo CPU E7500 processor with speed of 2.93 GHz and 2.00 GB ram the BPNN classifier is implemented using the neural network toolbox MATLAB R2013a computing platform. The steps of this comparative BPNN classifier are demonstrated by showing the outputs considering the input clinical datasets. The developed BPNN classifier has been experimented on three different clinical datasets obtained from the UCI machine learning repository namely, Pima Indian Diabetes, Hepatitis, and Wisconsin Breast Cancer.

From Table 6, it can be inferred that, the neural network experimented with (2n+1) hidden neurons and the activation function used in the hidden and the output layer are logsig-logsig and tansig-logsig respectively. For PID dataset, the LM algorithm converges to 48 epochs in 0.01 seconds to get the MSE less than 0.01. The gradient descent with adaptive learning rate BP algorithm does not converge at maximum epochs (500) with the MSE 0.265, but it achieves maximum accuracy 83.4% for PID dataset. For WBC dataset, the Polak-Ribiere and One Step Secant BP algorithm exceeds maximum of 500 epochs with MSE of 0.006, this yields highest accuracy of 98.6% for WBC dataset. For Hepatitis dataset, the classification accuracy is low for all twelve BP algorithms, as the number of samples in the dataset is low. BFGS algorithm provides an accuracy of 45.8% for hepatitis dataset. The parameters that work well for Wisconsin Breast Cancer and Pima Indian Diabetes dataset has given an average result for Hepatitis dataset. However, changing these parameters may yield a better result for Hepatitis dataset.

Performance Analysis

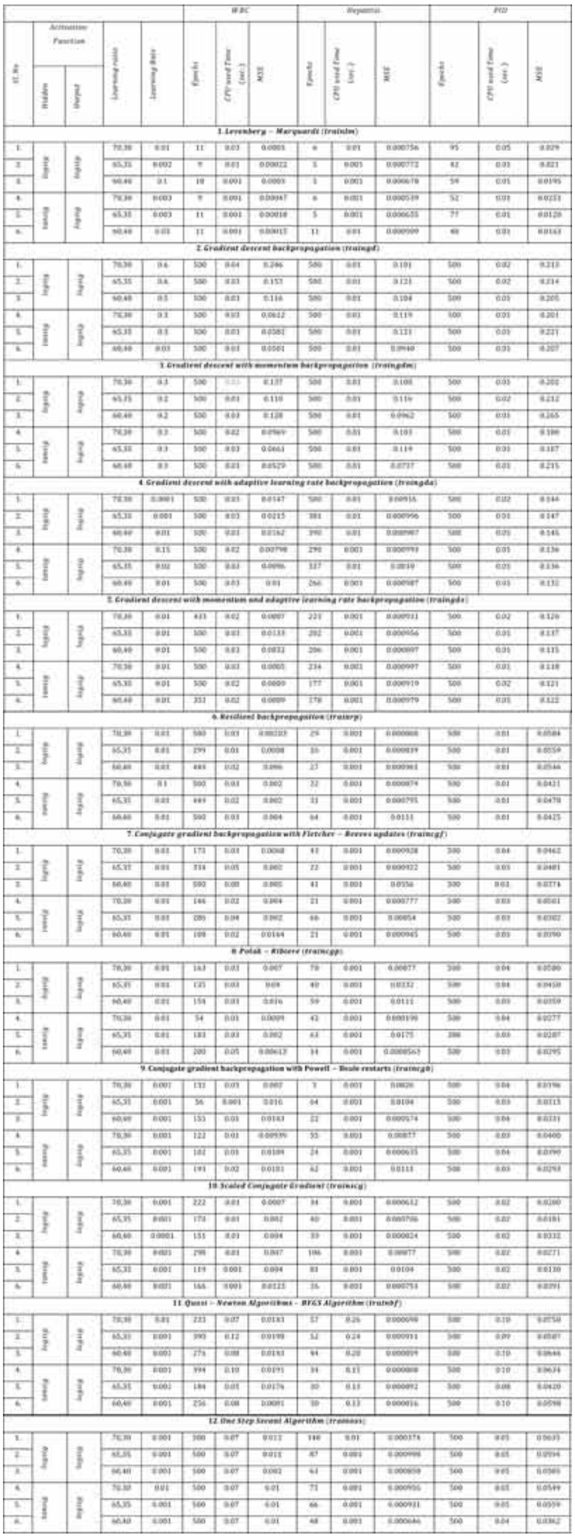
The classifier results are compared against the statistical measures namely, accuracy, sensitivity and specificity. The statistical measures are evaluated using metrics derived from confusion matrices such as True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). The elements are determined as follows:

TP – Diseased sample correctly identified as Diseased.
TN – Normal sample correctly identified as Normal.
FP – Diseased sample incorrectly identified as Normal.
FN – Normal sample incorrectly identified as Diseased.

The performance measures are computed using Equations (24) through (26) and the results are presented in Table 7:

$$\text{Accuracy} = \frac{TP + TN}{\left(TP + FP + FN + TN\right)} \tag{24}$$

Table 6. Performance evaluation of twelve BP algorithms for three clinical datasets



					WBC			Hepatitis			PID		
Sl. No	Activation Function		Learning ratio	Learning Rate	Epochs	CPU used Time (sec.)	MSE	Epochs	CPU used Time (sec.)	MSE	Epochs	CPU used Time (sec.)	MSE
	Hidden	Output											
1. Levenberg – Marquardt (trainlm)													
1.	logsig	logsig	70,30	0.01	11	0.03	0.0003	6	0.01	0.000756	95	0.05	0.029
2.			65,35	0.002	9	0.01	0.00022	5	0.005	0.000772	82	0.01	0.021
3.			60,40	0.1	10	0.001	0.0003	5	0.001	0.000678	59	0.01	0.0195
4.	tansig	logsig	70,30	0.003	9	0.001	0.00047	6	0.001	0.000539	52	0.01	0.0233
5.			65,35	0.003	11	0.001	0.00018	5	0.001	0.000655	77	0.01	0.0120
6.			60,40	0.03	11	0.001	0.00015	11	0.01	0.000909	60	0.01	0.0163
2. Gradient descent backpropagation (traingd)													
1.	logsig	logsig	70,30	0.6	500	0.04	0.246	500	0.01	0.101	500	0.02	0.213
2.			65,35	0.6	500	0.03	0.153	500	0.01	0.121	500	0.02	0.214
3.			60,40	0.3	500	0.03	0.116	500	0.01	0.104	500	0.03	0.205
4.	tansig	logsig	70,30	0.3	500	0.03	0.0612	500	0.01	0.119	500	0.01	0.201
5.			65,35	0.3	500	0.03	0.0582	500	0.01	0.121	500	0.01	0.221
6.			60,40	0.03	500	0.03	0.0501	500	0.01	0.0940	500	0.01	0.207
3. Gradient descent with momentum backpropagation (traingdm)													
1.	logsig	logsig	70,30	0.3	500	[illegible]	0.137	500	0.01	0.108	500	0.01	0.202
2.			65,35	0.2	500	0.01	0.110	500	0.01	0.116	500	0.02	0.212
3.			60,40	0.2	500	0.03	0.128	500	0.01	0.0962	500	0.01	0.265
4.	tansig	logsig	70,30	0.3	500	0.02	0.0969	500	0.01	0.103	500	0.01	0.180
5.			65,35	0.3	500	0.03	0.0661	500	0.01	0.119	500	0.01	0.187
6.			60,40	0.3	500	0.03	0.0529	500	0.01	0.0737	500	0.01	0.215
4. Gradient descent with adaptive learning rate backpropagation (traingda)													
1.	logsig	logsig	70,30	0.0001	500	0.03	0.0147	500	0.01	0.00916	500	0.02	0.144
2.			65,35	0.001	500	0.03	0.0215	381	0.01	0.000996	500	0.01	0.147
3.			60,40	0.01	500	0.03	0.0162	390	0.01	0.000987	500	0.01	0.145
4.	tansig	logsig	70,30	0.15	500	0.02	0.00798	290	0.001	0.000993	500	0.01	0.136
5.			65,35	0.02	500	0.03	0.0096	327	0.01	0.0810	500	0.01	0.136
6.			60,40	0.01	500	0.03	0.01	266	0.001	0.000987	500	0.01	0.131
5. Gradient descent with momentum and adaptive learning rate backpropagation (traingdx)													
1.	logsig	logsig	70,30	0.01	413	0.02	0.0007	221	0.001	0.000911	500	0.02	0.126
2.			65,35	0.01	500	0.03	0.0133	202	0.001	0.000956	500	0.01	0.137
3.			60,40	0.01	500	0.03	0.0032	206	0.001	0.000997	500	0.01	0.115
4.	tansig	logsig	70,30	0.01	500	0.03	0.0005	234	0.001	0.000997	500	0.01	0.118
5.			65,35	0.01	500	0.02	0.0009	177	0.001	0.000919	500	0.02	0.121
6.			60,40	0.01	331	0.02	0.0009	178	0.001	0.000979	500	0.01	0.122
6. Resilient backpropagation (trainrp)													
1.	logsig	logsig	70,30	0.03	500	0.03	0.00223	29	0.001	0.000868	500	0.01	0.0584
2.			65,35	0.03	299	0.01	0.0008	16	0.001	0.000839	500	0.01	0.0559
3.			60,40	0.03	489	0.02	0.096	27	0.001	0.000963	500	0.01	0.0546
4.	tansig	logsig	70,30	0.1	500	0.03	0.002	22	0.001	0.000874	500	0.01	0.0421
5.			65,35	0.03	489	0.02	0.002	33	0.001	0.000795	500	0.01	0.0478
6.			60,40	0.03	500	0.03	0.004	64	0.001	0.0111	500	0.01	0.0425
7. Conjugate gradient backpropagation with Fletcher – Reeves updates (traincgf)													
1.	logsig	logsig	70,30	0.03	173	0.03	0.0068	47	0.001	0.000928	500	0.04	0.0462
2.			65,35	0.03	314	0.05	0.002	22	0.001	0.000922	500	0.03	0.0481
3.			60,40	0.03	500	0.08	0.005	41	0.001	0.0356	500	0.03	0.0374
4.	tansig	logsig	70,30	0.03	146	0.02	0.004	21	0.001	0.000777	500	0.03	0.0501
5.			65,35	0.03	285	0.04	0.002	66	0.001	0.00054	500	0.03	0.0302
6.			60,40	0.03	108	0.02	0.0144	21	0.001	0.000945	500	0.03	0.0390
8. Polak – Ribiere (traincgp)													
1.	logsig	logsig	70,30	0.03	163	0.03	0.007	79	0.001	0.00077	500	0.04	0.0580
2.			65,35	0.03	135	0.03	0.08	40	0.001	0.0232	500	0.04	0.0458
3.			60,40	0.03	158	0.03	0.016	59	0.001	0.0111	500	0.03	0.0359
4.	tansig	logsig	70,30	0.03	54	0.01	0.0009	42	0.001	0.000199	500	0.04	0.0277
5.			65,35	0.03	183	0.03	0.002	63	0.001	0.0175	288	0.03	0.0287
6.			60,40	0.03	200	0.05	0.00613	14	0.001	0.0000563	500	0.03	0.0295
9. Conjugate gradient backpropagation with Powell – Beale restarts (traincgb)													
1.	logsig	logsig	70,30	0.001	131	0.03	0.007	7	0.001	0.0820	500	0.04	0.0196
2.			65,35	0.001	56	0.001	0.016	64	0.001	0.0104	500	0.03	0.0113
3.			60,40	0.001	153	0.03	0.0143	22	0.001	0.000574	500	0.04	0.0331
4.	tansig	logsig	70,30	0.001	122	0.01	0.00939	33	0.001	0.00077	500	0.03	0.0400
5.			65,35	0.001	102	0.03	0.0109	24	0.001	0.000635	500	0.04	0.0390
6.			60,40	0.001	193	0.02	0.0101	62	0.001	0.0111	500	0.03	0.0293
10. Scaled Conjugate Gradient (trainscg)													
1.	logsig	logsig	70,30	0.001	222	0.03	0.0007	34	0.001	0.000612	500	0.02	0.0200
2.			65,35	0.001	173	0.03	0.002	40	0.001	0.000786	500	0.02	0.0181
3.			60,40	0.0001	155	0.01	0.004	33	0.001	0.000024	500	0.02	0.0332
4.	tansig	logsig	70,30	0.001	298	0.03	0.007	106	0.001	0.00077	500	0.02	0.0271
5.			65,35	0.001	119	0.001	0.004	83	0.001	0.0104	500	0.02	0.0130
6.			60,40	0.001	166	0.001	0.0123	35	0.001	0.000753	500	0.02	0.0391
11. Quasi – Newton Algorithms – BFGS Algorithm (trainbf)													
1.	logsig	logsig	70,30	0.01	223	0.07	0.0183	57	0.26	0.000699	500	0.10	0.0750
2.			65,35	0.001	390	0.12	0.0199	52	0.24	0.000931	500	0.09	0.0587
3.			60,40	0.001	273	0.08	0.0183	44	0.20	0.000059	500	0.10	0.0644
4.	tansig	logsig	70,30	0.001	394	0.10	0.0191	34	0.11	0.000088	500	0.10	0.0634
5.			65,35	0.001	184	0.05	0.0176	30	0.13	0.000092	500	0.08	0.0420
6.			60,40	0.001	256	0.08	0.0091	30	0.13	0.000016	500	0.10	0.0598
12. One Step Secant Algorithm (trainoss)													
1.	logsig	logsig	70,30	0.001	300	0.07	0.012	140	0.01	0.000374	500	0.05	0.0633
2.			65,35	0.001	500	0.07	0.011	87	0.001	0.000998	500	0.05	0.0594
3.			60,40	0.001	500	0.07	0.002	63	0.001	0.000859	500	0.05	0.0585
4.	tansig	logsig	70,30	0.01	500	0.07	0.01	73	0.001	0.000956	500	0.05	0.0549
5.			65,35	0.001	500	0.07	0.01	66	0.001	0.000931	500	0.05	0.0559
6.			60,40	0.001	500	0.07	0.01	48	0.001	0.000646	500	0.04	0.0362

Table 7. Classification performance comparison of twelve BP algorithms for three clinical datasets

MSE	Accuracy (%)	Sensitivity (%)	Specificity (%)	MSE	Accuracy (%)	Sensitivity (%)	Specificity (%)	MSE	Accuracy (%)	Sensitivity (%)	Specificity (%)
WBC				*Hepatitis*				*PID*			
1. Levenberg – Marquardt (trainlm)											
0.00015	97.9	98.1	97.0	0.0005	33.9	100	32.8	0.01	74.1	93.7	29
2. Gradient descent Backpropagation (traingd)											
0.0501	83.9	93.0	54.5	0.0940	15.3	100	13.8	0.201	68.3	97.2	1.6
3. Gradient Descent with Momentum Backpropagation (traingdm)											
0.0529	95.7	98.6	86.4	0.0737	20.3	100	19	0.180	71.7	90.0	27.4
4. Gradient Descent with adaptive Learning Rate Backpropagation (traingda)											
0.00798	97.1	98.1	93.9	0.00098	33.9	100	32.8	0.132	79.5	85.3	66.1
5. Gradient Descent with Momentum and adaptive Learning Rate Backpropagation											
0.0005	97.5	98.1	95.5	0.00089	40.7	100	39.7	0.115	83.4	88.1	72.6
6. Resilient Backpropagation (trainrp)											
0.0008	96.8	97.2	95.5	0.00079	25.4	100	24.1	0.0421	76.1	84.6	56.5
7. Conjugate gradient Backpropagation with Fletcher – Reeves Updates (traincgf)											
0.002	98.2	98.6	97	0.00077	40.7	100	39.7	0.0302	76.6	83.2	61.3
8. Polak – Ribiere (traincgp)											
0.0009	98.6	98.6	98.5	0.00019	35.6	100	34.5	0.0277	76.1	81.1	64.5
9. Conjugate gradient Backpropagation with Powell – Beale Restarts (traincgb)											
0.002	98.2	98.1	98.5	0.00057	33.9	100	32.8	0.0196	76.1	81.1	64.5
10. Scaled Conjugate Gradient (trainscg)											
0.0007	97.5	98.6	93.9	0.00061	40.7	0	41.4	0.0130	76.1	81.8	62.9
11. Quasi – Newton Algorithms – BFGS Algorithm (trainbf)											
0.0081	97.9	98.1	97	0.00069	45.8	100	44.8	0.0420	78	81.8	69.4
12. One Step Secant Algorithm (trainoss)											
0.002	98.6	98.6	98.5	0.00037	32.2	100	31	0.0362	76.6	81.8	64.5

$$\text{Sensitivity} = \frac{TP}{\left(TP + FN\right)} \tag{25}$$

$$\text{Specificity} = \frac{TN}{\left(FP + TN\right)} \tag{26}$$

CONCLUSION

In this work, an analytic study has been performed to evaluate the network parameter values for backpropagation neural network training. The investigation is performed on the network parameters namely, number of neurons in the hidden layer, activation function used in the hidden and the output layer, learning rate, number of training epochs and mean square error while training the twelve different Backpropagation Neural Network (BPNN). The performance of the twelve BPNN classifier has been evaluated using three clinical datasets namely Pima Indian diabetes, Wisconsin Breast cancer and

Hepatitis obtained from the UCI machine learning repository. From the comparative study it can be inferred that the accuracy of the classifier depends upon the type of backpropagation algorithm chosen, the combination of network parameter values chosen and the dataset chosen. This work can serve as a guideline to design network parameters for any neural network based CDSS.

REFERENCES

Agharezaei, L., Agharezaei, Z., Nemati, A., Bahaadinbeigy, K., Keynia, F., Baneshi, M. R., & Agharezaei, M. (2016). The prediction of the risk level of pulmonary embolism and deep vein thrombosis through artificial neural network. *Acta Informatica Medica*, *24*(5), 354. doi:10.5455/aim.2016.24.354.359 PMID:28077893

Akinyokun, O. C.(2002). Neuro-Fuzzy Expert System for Human Resource Performance Evaluation. *First Bank of Nigeria Plc Endowment Fund Public Lecture, Series 1.*

Alsmadi, M. K. S., Omar, K. B., & Noah, S. A. (2009). Back propagation algorithm: The best algorithm among the multi-layer perceptron algorithm. *International Journal of Computer Science and Network Security*, *9*(4), 378–383.

Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*, *43*(1), 3–31. doi:10.1016/S0167-7012(00)00201-3 PMID:11084225

Berry, M. J., & Linoff, G. (1997). *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc.

Bezdek, J. C. (1993). A review of probabilistic, fuzzy, and neural models for pattern recognition. *Journal of Intelligent & Fuzzy Systems*, *1*(1), 1–25. doi:10.3233/IFS-1993-1103

Blum, A. (1992). *Neural networks in C*. Wiley.

Boger, Z., & Guterman, H. (1997). Knowledge extraction from artificial neural network models. In Systems, Man, and Cybernetics. *IEEE International Conference on Computational Cybernetics and Simulation*, *4,* 3030-3035. doi:10.1109/ICSMC.1997.633051

Bossan, M. C., Seixas, J. M., Caloba, L. P., Penha, R. S., & Nadal, J. (1995). A modified backpropagation algorithm for neural classifiers. *Proceedings of the 38th Midwest SymposiumIn Circuits and Systems*, *1,* 562-565.

Chandra, P., & Singh, Y. (2004). An activation function adapting training algorithm for sigmoidal feedforward networks. *Neurocomputing*, *61*, 429–437. doi:10.1016/j.neucom.2004.04.001

Christopher, J. J., Nehemiah, H. K., & Kannan, A. (2015). A clinical decision support system for diagnosis of Allergic Rhinitis based on intradermal skin tests. *Computers in Biology and Medicine*, *65*, 76–84. doi:10.1016/j.compbiomed.2015.07.019 PMID:26298488

Coppin, B. (2004). *Artificial intelligence illuminated.* Jones & Bartlett Learning.

Desai, S. D., Giraddi, S., Narayankar, P., Pudakalakatti, N. R., & Sulegaon, S. (2019). Back-propagation neural network versus logistic regression in heart disease classification. Advanced Computing and Communication Technologies, 133-144. doi:10.1007/978-981-13-0680-8_13

Drago, G. P., Morando, M., & Ridella, S. (1995). An adaptive momentum back propagation (AMBP). *Neural Computing & Applications*, *3*(4), 213–221. doi:10.1007/BF01414646

Duch, W., & Jankowski, N. (1999). Survey of neural transfer functions. *Neural Computing Surveys*, *2*(1), 163–212.

Duch, W., & Jankowski, N. (2001). Transfer functions: hidden possibilities for better neural networks. ESANN, 81-94.

Elgin Christo, V. R., Khanna Nehemiah, H., Minu, B., & Kannan, A. (2019). Correlation-Based Ensemble Feature Selection Using Bioinspired Algorithms and Classification Using Backpropagation Neural Network. *Computational and Mathematical Methods in Medicine*, *2019*, 1–17. doi:10.1155/2019/7398307 PMID:31662787

Eluyode, O. S., & Akomolafe, D. T., & MNCS, M. (2013). Comparative study of biological and artificial neural networks. *European Journal of Applied Engineering and Scientific Research*, *2*(1).

FFNN. Feed-forward Neural Networks. (2010). *Short Notes*. http://www.cs.nott.ac.uk/~pszqiu/Teaching/G53MLE/ffnets-note.pdf

Fukuoka, Y., Matsuki, H., Minamitani, H., & Ishida, A. (1998). A modified back-propagation method to avoid false local minima. *Neural Networks*, *11*(6), 1059–1072. doi:10.1016/S0893-6080(98)00087-2 PMID:12662775

Geetha, V., Aprameya, K. S., & Hinduja, D. M. (2020). Dental caries diagnosis in digital radiographs using back-propagation neural network. *Health Information Science and Systems*, *8*(1), 1–14. doi:10.1007/s13755-019-0096-y PMID:31949895

Hamamoto, Tungjitkusolmun, & Pintavirooj. (2019). Automatic Detection and Staging of Lung Tumors using Locational Features and Double-Staged Classifications. *Applied Sciences*, *9*(11), 2329.

He, Z., Wu, M., & Gong, B. (1992). Neural network and its application on machinery fault diagnosis. *IEEE International Conference on Systems Engineering*, 576-579.

Helwan, A., Idoko, J. B., & Abiyev, R. H. (2017). Machine learning techniques for classification of breast tissue. *Procedia Computer Science*, *120*, 402–410. doi:10.1016/j.procs.2017.11.256

Kamber, M., Han, J., & Pei, J. (2012). *Data mining: Concepts and techniques*. Elsevier.

Karsoliya, S. (2012). Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *International Journal of Engineering Trends and Technology*, *3*(6), 714–717.

Kosko, B. (1994). *Neural Network and Fuzzy Systems* (1st ed.). Prentice Hall of India.

Krasnopolsky, V. M., & Chevallier, F. (2003). Some neural network applications in environmental sciences. Part II: Advancing computational efficiency of environmental numerical models. *Neural Networks*, *16*(3), 335–348. doi:10.1016/S0893-6080(03)00026-1 PMID:12672429

Lee, H. M., Chen, C. M., & Huang, T. C. (2001). Learning efficiency improvement of back-propagation algorithm by error saturation prevention method. *Neurocomputing*, *41*(1), 125–143. doi:10.1016/S0925-2312(00)00352-0

Leema, N., Nehemiah, H. K., & Kannan, A. (2016). Neural network classifier optimization using Differential Evolution with Global Information and Back Propagation algorithm for clinical datasets. *Applied Soft Computing*, *49*, 834–844. doi:10.1016/j.asoc.2016.08.001

Lichman, M. (2013). *UCI Machine Learning Repository*. University of California, School of Information and Computer Science.

Liu, Y., Starzyk, J. A., & Zhu, Z. (2007). Optimizing number of hidden neurons in neural networks. *Artificial Intelligence and Applications (Commerce, Calif.)*, *2*, 138–143.

Masters, T. (1993). *Practical neural network recipes in C*. Morgan Kaufmann.

Mhaskar, H. N., & Micchelli, C. A. (1994). How to choose an activation function. *Advances in Neural Information Processing Systems*, 319–319.

Morshed, J., & Kaluarachchi, J. J. (1998). Parameter estimation using artificial neural network and genetic algorithm for free-product migration and recovery. *Water Resources Research*, *34*(5), 1101–1113. doi:10.1029/98WR00006

Nahato, K. B., Harichandran, K. N., & Arputharaj, K. (2015). Knowledge mining from clinical datasets using rough sets and backpropagation neural network. *Computational and Mathematical Methods in Medicine*, *2015*, 1–13. doi:10.1155/2015/460189 PMID:25821508

Nahato, K. B., Nehemiah, K. H., & Kannan, A. (2016). Hybrid approach using fuzzy sets and extreme learning machine for classifying clinical datasets. *Informatics in Medicine Unlocked*, *2*, 1–11. doi:10.1016/j.imu.2016.01.001

Ng, S. C., Leung, S. H., & Luk, A. (1999). Fast convergent generalized back-propagation algorithm with constant learning rate. *Neural Processing Letters*, *9*(1), 13–23. doi:10.1023/A:1018611626332

Nguyen, D., & Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *International Joint Conference In Neural Networks*, 21-26.

Rajasekaran, S., & Pai, G. V. (2003). *Neural networks, fuzzy logic and genetic algorithm: synthesis and applications (with cd)*. PHI Learning Pvt. Ltd.

Ravindra, B. V., Sriraam, N., & Geetha, M. (2018, February). Chronic kidney disease detection using back propagation neural network classifier. *2018 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, 65-68. doi:10.1109/IC3IoT.2018.8668110

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. doi:10.1037/h0042519 PMID:13602029

Rumelhart, D. E., Hinton, G. E., & Wiliams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi:10.1038/323533a0

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. *Parallel Distributed Processing*, *1*, 318–362.

Saduf, M. A. W. (2013). Comparative study of back propagation learning algorithms for neural networks. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(2), 1151–1156.

Sharma, B. (2014). Comparison of neural network training functions for hematoma classification in brain CT images. *IOSR Journals*, *1*(16), 31–35. doi:10.9790/0661-16123135

Sibi, P., Jones, S. A., & Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, *47*(3), 1264–1268.

Singh, Y., & Chandra, P. (2003). A class+ 1 sigmoidal activation functions for FFANNs. *Journal of Economic Dynamics & Control*, *28*(1), 183–187. doi:10.1016/S0165-1889(02)00157-4

Sudha, M. (2017). Evolutionary and neural computing based decision support system for disease diagnosis from clinical data sets in medical practice. *Journal of Medical Systems*, *41*(11), 178. doi:10.1007/s10916-017-0823-3 PMID:28956226

Swanston, D. J., Bishop, J. M., & Mitchell, R. J. (1994). Simple adaptive momentum: New algorithm for training multilayer perceptrons. *Electronics Letters*, *30*(18), 1498–1500. doi:10.1049/el:19941014

Tarle, B., & Jena, S. (2017, August). An artificial neural network based pattern classification algorithm for diagnosis of heart disease. *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 1-4. doi:10.1109/ICCUBEA.2017.8463729

Verma, B. K., & Mulawka, J. J. (1994). A modified backpropagation algorithm. *IEEE World Congress on Computational Intelligence*, *2,* 840-844.

Wang, X., Tang, Z., Tamura, H., Ishii, M., & Sun, W. D. (2004). An improved backpropagation algorithm to avoid the local minima problem. *Neurocomputing*, *56*, 455–460. doi:10.1016/j.neucom.2003.08.006

Wani, M. A. (2014). Comparative Study of High Speed Back-Propagation Learning Algorithms. *International Journal of Modern Education and Computer Science*, *6*(12), 34–40. doi:10.5815/ijmecs.2014.12.05

Wu, H., Zhao, S., Zhang, X., Sang, A., Dong, J., & Jiang, K. (2020, January). Back-propagation Artificial Neural Network for Early Diabetic Retinopathy Detection Based On A Priori Knowledge. *Journal of Physics: Conference Series*, *1437*(1), 012019. doi:10.1088/1742-6596/1437/1/012019

Yam, J. Y., & Chow, T. W. (2000). A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing*, *30*(1), 219–232. doi:10.1016/S0925-2312(99)00127-7

Yu, X. H., & Chen, G. A. (1997). Efficient backpropagation learning using optimal learning rate and momentum. *Neural Networks*, *10*(3), 517–527. doi:10.1016/S0893-6080(96)00102-5

Zhang, L., & Suganthan, P. N. (2016). A survey of randomized algorithms for training neural networks. *Information Sciences*, *364*, 146–155. doi:10.1016/j.ins.2016.01.039

Zweiri, Y. H., Whidborne, J. F., & Seneviratne, L. D. (2003). A three-term backpropagation algorithm. *Neurocomputing*, *50*, 305–318. doi:10.1016/S0925-2312(02)00569-6

Leema N. completed B.E. (Electronics and Communication Engineering) (2008), M.E. (Computer Science and Engineering) (2010) and Ph.D. (Computer Science and Engineering) (2018) from Anna University, Chennai, Tamil Nadu, India Her areas of interest include Data Mining, Machine Learning and Bio-inspired Computing.

Khanna Nehemiah H. completed B.E. (Computer Science and Engineering) (1997), M.E. (Computer Science and Engineering) (1998) from University of Madras, Chennai, Tamil Nadu, India and Ph.D. (Computer Science and Engineering) (2007) from Anna University, Chennai, Tamil Nadu, India. He is currently working as a Professor in Ramanujan Computing Centre, Anna University, Chennai, Tamil Nadu, India. His areas of interest include Software Engineering, Database Systems, Data Mining, Medical Image Processing, Artificial Intelligence, Soft computing and Bio-inspired Computing.

V.R. Elgin Christo completed B.Tech. (Information Technology) (2013) and M.Tech. (Information Technology) (2015) from Anna University, Chennai, Tamil Nadu, India. Currently he is pursuing Ph.D. (Computer Science and Engineering), in Anna University, Chennai, Tamil Nadu, India. He is a Research Scholar under "Visvesvaraya Ph.D. Scheme for Electronics and Information Technology", Deity, Ministry of Communication and Information Technology, Government of India. His areas of interest include Bio-inspired Computing, Medical Image Processing and Data Mining.

Kannan Arputharaj completed M.E (Computer Science and Engineering) (1991) and Ph.D. (Computer Science and Engineering) (2000) from Anna University, Chennai, Tamil Nadu, India. He retired as a Professor from the Department of Information Science and Technology, Anna University, Chennai, Tamil Nadu, India. He is currently working as a Senior Professor in Vellore Institute of Technology, Vellore. His areas of interest include Database Management Systems, Artificial Intelligence, Bio-inspired Computing and Information Security.