3.1 INTRODUCTION

One of the most popular training algorithms in the domain of neural networks used for weather forecasting is the Back Propagation Neural Network (BPNN) algorithm. BPNN algorithm is used to develop accuracy of weather prediction. BPNN is a systematic method of training multilayer artificial neural networks. The back propagation is a gradient descent method in which gradient of the error is calculated with respect to the weights for a given input by propagating the error backwards. The combination of weights which minimizes the error function is considered to be a solution of the problem. Although Back propagation algorithm is an efficient technique applied to classification problems, system modeling, adaptive robotics control, it suffers from local minima problem, scaling problem, long training time etc.

One typical method for training a network is to first partition the data series into three disjoint sets. These sets are: the training set, the validation set, and the test set. The network is trained (e.g., with back propagation) directly on the training set, its generalization ability is monitored on the validation set, and its ability to forecast is measured on the test set. A network’s generalization ability indirectly measures how well the network can deal with unforeseen inputs, in other words, inputs on which it was not trained [FDH01]. A network that produces high forecasting error on unforeseen inputs, but low error on training inputs is said to have over-fit the training data. Over-fitting occurs when the network is blindly trained to a minimum in the total squared error based on the training set.

BPNN is one of the most common supervised training methods [KM05]. Training is usually carried out by iterative updating of weights based on minimizing the Mean Square Error. In the output layer, the error signal is the difference between the desired and the output values. Then the error signal is fed back through the steepest descent algorithm to the lower layers to update the weights of the network. The weights of the network are adjusted by the algorithm such that the error is decreased along a descent direction. The Back propagation algorithm requires the activation function to be continuous and differentiable [AP07, HA05].
Back propagation is the most broadly used learning method for feed forward neural networks [SZ1, IM09, AP07]. The feed forward neural network [SZ11, Su09, RG07] is the simplest ANN architecture in terms of information flow direction. Many of the neural network architectures are variations of the feed forward neural network [GA09]. There are two practical ways to implement the Back propagation algorithm: batch updating approach and online updating approach. Corresponding to the standard gradient method, the batch updating approach accumulates the weight correction over all the training samples before actually performing the update. On the other hand, the online updating approach updates the network weights immediately after each training sample is fed [WW11].

In science and engineering problems, there are many literatures that have examined the effectiveness of each category of algorithms on the performance of the Back propagation neural network.

Esugasini et al., analysed the problem of breast cancer diagnosis and compared the classification accuracy of the standard steepest descent against the gradient descent with momentum, adaptive learning and levenberg marquardt algorithm. The simulations show that the neural network using the levenberg marquardt algorithm achieved the best classification performance [EM05].

Brian A. Smith et al., aims at creating a ANN models with lesser average prediction error by means of enhancing the number of distinct observations utilized in training, adding together extra input expressions that explain the date of an observation, raising the duration of prior weather data considered in all observation, and reexamining the number of hidden nodes utilized in the network. Models were generated to predict air temperature at hourly intervals from one to 12 hours before it happens. The entire ANN model, containing a network architecture and set of associated parameters, was calculated by instantiating and training 30 networks and computing the mean absolute error of the resulting networks for few set of input patterns [BS07].
Arvind Sharma et al., briefly provided the way of the various connectionist models could be created with the help of various learning techniques and then examines whether they can afford the necessary level of performance, that are adequately good and robust so as to afford a reliable prediction model for stock market indices [AM06].

A feed forward neural network model was developed and implemented to predict the rainfall on yearly and monthly basis. A feed forward neural network with back propagation algorithm was implemented and tested for the purpose of weather forecasting.

The basic back propagation algorithm consists of three steps.

- The input pattern is presented to the input layer of the network. These inputs are propagated through the network until they reach the output units. This forward pass produces the actual or predicted output pattern.

- Back propagation is a supervised learning algorithm, the desired outputs are given as part of the training vector. The actual network outputs are subtracted from the desired outputs and an error signal is produced.

- The error signal is then the basis for the back propagation step, whereby the errors are passed back through the neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the neural network has just “learned” from an experience.

The general structure of weather forecasting system using back propagation neural network is shown in figure 3.1.
3.1.1. Phases in Back Propagation Technique

The back propagation [RR06] learning algorithm can be divided into two phases: propagation and weight update.

Phase 1: Propagation

Each propagation involves the following steps:

1. Forward propagation of a training pattern's input is given through the neural network in order to generate the propagation's output activations.

2. Back propagation of the output activation’s propagation through the neural network using the training pattern's target in order to generate the deltas of all output and hidden neurons.

Phase 2: Weight Update

For each weight-synapse:

1. Multiplying its input activation and output delta to get the gradient of the weight.
2. Bringing the weight in the direction of the gradient by adding a ratio of it from the weight.

This ratio impacts on the speed and quality of learning is called the learning rate. The sign of the gradient of a weight designates where the error is increasing, this is why the weight must be updated in the opposite direction.

The phase 1 and 2 is repeated until the performance of the network is satisfactory.

The steps of the Back Propagation algorithm are [HTTP4]

Step 1. Initialize the weights in the network (often randomly)
Step 2. Do
Step 3. For each e in the training set
   a. O = neural-net-output (network, e); forward pass
   b. T = teacher output for e
Step 4. Calculate error (T - O) at the output units
Step 5. Compute $\Delta w_h$ for all weights from hidden layer to output layer;
   Backward pass
Step 6. Compute $\Delta w_i$ for all weights from input layer to hidden layer;
   Backward pass continued
Step 7. Update the weights in the network
Step 8. Until all e’s classified correctly or stopping criterion satisfied
Step 9. Return the network

3.1.2 LEARNING WITH THE BACK PROPAGATION ALGORITHM

The back propagation algorithm is an involved mathematical tool; however, execution of the training equations is based on iterative processes [AB01]. Before starting the back propagation learning process, need the following:
The set of training patterns, input, and target

A value for the learning rate

A criterion that terminates the algorithm

A methodology for updating weights

The nonlinearity function (usually the sigmoid)

Initial weight values (typically small random values)

Uses of back propagation neural network are

- A large amount of input/output data is available, but not sure how to relate it to the output.

- It is easy to create a number of examples of the correct behavior.

- The solution to the problem may change over time, within the bounds of the given input and output parameters.

- Outputs can be "fuzzy", or non-numeric.

3.2 BACK PROPAGATION ARCHITECTURE

The feed-forward back-propagation neural network architecture shown in figure 3.2 is fully connected, which means that a neuron in any layer is connected to all neurons in the previous layer. Signal flow through the network progresses in a forward direction, from left to right and on a layer-by-layer basis.
In the figure 3.2,

1. The output of a neuron in a layer moves to all neurons in the following layer.
2. Each neuron has its own input weights.
3. The weights for the input layer are assumed (fixed) to be 1 for each input. In other words, input values are not changed.
4. The output of the neural network is obtained by applying input values to the input layer, passing the output of each neuron to the following layer as input.
5. The Back Propagation neural network should have at least an input layer and an output layer. It could have zero or more hidden layers.

3.2.1 Two passes of computation

In the application of the back-propagation algorithm, two distinct passes of computation may be distinguished. The first pass is referred to as the forward pass, and the second one as the backward pass. In the forward pass, the synaptic weights remain unaltered and the function signals of the network are computed on a neuron-by-neuron
basis according to figure 3.2. Thus the forward phase of computation begins at the first hidden layer by presenting it with the input vector, and terminates at the output layer by computing the error signal for each neuron of this layer. The backward pass, on the other hand, starts at the output layer by passing the error signals leftward through the sensitivity network, layer by layer, and recursively computing the local gradient for each neuron. This recursive process permits the synaptic weights of the network to undergo changes in accordance with the delta rules. For a neuron located in the output layer, the gradient is simply equal to the error signal of that neuron multiplied by the first derivative of its nonlinearity. The recursive computation is continued, layer by layer by propagating the changes to all synaptic weights made.

### 3.2.2 Stopping criteria

The back-propagation algorithm cannot, in general, be shown to converge, nor are there well defined criteria for stopping its operation. Rather, there are some reasonable criteria, each with its own practical merit, which may be used to terminate the weight adjustments. To formulate such a criterion, the logical thing to do is to think in terms of the unique properties of a local or global minimum of the error surface.

1. The back-propagation algorithm is considered to have converged when the Euclidean norm of the gradient vector reaches a sufficiently small gradient threshold. The drawback of this convergence criterion is that, for successful trials, learning times may be long.

2. The back-propagation algorithm is considered to have converged, when the absolute rate of change in the average squared error per epoch is sufficiently small. Typically, the rate of change in the average squared error is considered to be small enough if it lies in the range of 0.1 to 1 percent per epoch.

3. The back-propagation algorithm is terminated when the weight updates are sufficiently small.
4. After each learning iteration, the network is tested for its generalization performance.

5. The simplest way to stop the training is to limit the number of iterations to a predetermined value. This stopping criterion is frequently used mainly when a new problem is solved and nothing is known about the shape and properties of the error surface.

3.2.3 Pattern and batch modes of training

In a practical application of the back-propagation algorithm, learning results from the many presentations of training set to the network. One complete presentation of the entire training set during the learning process is called an epoch. The learning process is maintained on an epoch-by-epoch basis until the synaptic weights stabilize and the average squared error over the entire training set converges to some minimum value. It is good practice to randomize the order of presentation of training examples from one epoch to the next. This randomization tends to make the search in weight space stochastic over the learning cycles, thus avoiding the possibility of limit cycles in the evolution of the synaptic weight vectors. For a given training set, back-propagation learning may thus proceed in one of two basic ways [NA04]:

1. In the pattern mode of back-propagation learning, weight updating is performed immediately after the presentation of each training example. This is the very mode of operation for which the derivation of the back-propagation algorithm presented here applies.

2. In the batch mode of back-propagation learning, weight updating is performed after the presentation of all the training examples that constitute an epoch. The pattern mode of training is generally preferred over the batch mode, because it requires less local storage for each synaptic connection. Moreover, given that the patterns are presented to the network in a random manner, the use of
sample-by-sample updating of weights makes the search in weight space stochastic in nature, which, in turn, makes it less likely for the back-propagation algorithm to be trapped in a local minimum. On the other hand, the use of batch mode of training provides a more accurate estimate of the gradient vector. The relative effectiveness of the two training modes depends on the problem at hand.

### 3.2.4 Weight initialization

The first step in back-propagation learning is to initialize the network. A good choice for the initial values of the free parameters (i.e. adjustable synaptic weights) of the network can be of tremendous help in a successful network design. In cases where prior information is available, it may be better to use the prior information to guess the initial values of the free parameters. The initial values should not be too small because gradients are multiplied by weights and small weight values will cause the error gradients to be very small. The wrong choice of initial weights (e.g. too large weights) can lead to a phenomenon known as premature saturation. This phenomenon refers to a situation where the instantaneous sum of squared errors remains almost constant for some period of time during the learning process. Such a phenomenon cannot be considered as a local minimum, because the squared error continues to decrease after this period is finished.

When a training pattern is applied to the input layer of a multilayer perceptron, the output values of the network are calculated through a sequence of forward computations that involves inner products and sigmoidal transformations. This is followed by a sequence of backward computations that involves the calculation of error signals and pertinent slope of the sigmoid activation function, and culminates in synaptic weight adjustments. Output value for the neuron will be close to -1 or +1. Such situation is called saturation. The neurons that are in saturation do not learn any more.

### 3.2.5 Training, testing, and validation sets
The essence of back-propagation learning is to encode an input-output relation, represented by a set of \( \{x, d\} \), with a back-propagation network well trained in the sense that it learns enough about the past to generalize to the future. This can be trained in the same network several times and get different synaptic connections for each training run. A standard tool in statistics, known as cross-validation, provides a guiding principle. First, the available data set is randomly partitioned into a training set and testing set. The training set is further partitioned into two subsets.
1. A subset used for training the network.

2. A subset used for evaluation of the performance of the network (i.e. validation). Splitting the data into subsets is shown in the figure 3.3. The validation subset is typically 10 to 20 percent of the training set.

![Figure 3.3 Splitting the data into subsets](image)

**Figure 3.3 Splitting the data into subsets**

The motivation here is to validate the model on a data set different from the one used for parameter estimation. In this way, back propagation algorithm uses the testing set to assess the performance of various candidate model structures, and thereby chooses the best one. The particular model with the best-performing parameter values is then trained on the full training set, and the generalization performance of the resulting network is measured on the test set.

Back propagation is an iterative process that starts with the last layer and moves backwards through the layers until the first layer is reached. It may be assumed that for each layer, the error in the output of the layer is known. If the error of the output is known, then it is not hard to calculate changes for the weights, so as to reduce that error. The problem is that the error in the output of the very last layer only can be observed [MN05].

Back propagation gives a way to determine the error in the output of a prior layer by giving the output of a current layer as feedback. The process is therefore iterative: starting at the last layer and calculating the changes in the weight of the last layer. Then calculate the error in the output layer and the process can be repeated.
3.3 RESULTS AND DISCUSSION

Back propagation neural network is used to predict the weather forecasting based on the training set provided to the neural network. Through the implementation of this system, it is illustrated, how an intelligent system can be efficiently integrated with a neural network prediction model to predict the rain and no rain classification. This algorithm improves convergence. This method proves to be a simplified conjugate gradient method. Back propagation neural network approach for weather forecasting is capable of yielding good results and can be considered as an alternative to traditional meteorological approaches. This approach is able to determine the non-linear relationship that exists between the historical data supplied to the system during the training phase and on that basis, make a prediction of what the prediction would be in future.

To experiment the proposed system, a sample dataset was taken (Detailed explanation given in Chapter 1). To test and validate the proposed method, totally 2000 data sets are selected from Meteorological department Kanyakumari District, Tamil Nadu. Out of 2000 data sets, 1000 pertain to rain and 1000 to no rain. To train and test the classifier, out of 1000 rain datasets, 750 datasets pertaining to rain features are randomly assigned for training and the remaining 250 datasets are assigned for testing. Out of 1000 no rain datasets 750 data sets are assigned for training and the remaining 250 data sets are assigned for testing. Test and validations are done with confusion matrix.

A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. The accuracy is the proportion of the total number of predictions that were correct. The recall or true positive rate (TP) is the proportion of positive cases that were correctly identified. The true negative rate (TN) is defined as the proportion of negative cases that were classified correctly. The false negative rate (FN) is the proportion of positive cases that were incorrectly classified as negative. Finally, precision (P) is the proportion of the predicted positive cases that were
All experiments are validated with the help of four traditional performance metrics, namely sensitivity, accuracy, specificity, and precision. Table 3.1 defines events that assign true positive (TP), false negative (FN), true negative (TN), and false positive (FP) used in the performance metrics equations which are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$ (3.1)

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$ (3.2)

$$\text{Specificity} = \frac{TN}{FP + TN}$$ (3.3)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$ (3.4)

Table 3.1 Events that assign TP, FN, TN and FP

<table>
<thead>
<tr>
<th>Weather prediction</th>
<th>Rain</th>
<th>No rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified as Rain</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Classified as No rain</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

A confusion matrix in table 3.2 can be used to calculate classification statistics by using weather classification correctly by the back propagation network for the given type of weather pattern. The performance analysis of weather forecasting is shown in figure 3.4.

Table 3.2 Confusion Matrix-Results of BPNN in Weather prediction

<table>
<thead>
<tr>
<th>BPNN Weather prediction</th>
<th>Rain</th>
<th>No Rain</th>
<th>Total</th>
</tr>
</thead>
</table>
Figure 3.4 Performance Analysis of BPNN

Receiver operating Characteristic (ROC) graphs is an alternate method besides confusion matrices to examine the performance of classifiers. A ROC graph is a plot with the false positive rate on the X axis and the true positive rate on the Y axis. The point (0,1) is the perfect classifier: it classifies all the positive cases and the negative cases correctly. It is (0,1) because the false positive rate is 0 (none), and the true positive rate is 1 (all). The point (0,0) represents a classifier that predicts all cases to be negative, while the point (1,1) corresponds to a classifier that predicts every case to be positive. Point (1,0) is the classifier that is incorrect for all classifications. In many cases, a classifier has a parameter that can be adjusted to increase TP at the cost of an increased FP or decreased FP at the cost of a decrease in TP. Each parameter setting provides a (FP, TP) pair and a series of such pairs can be used to plot an ROC curve. A
non-parametric classifier is represented by a single ROC point, corresponding to its (FP,TP) pair.

Features of ROC Graphs

- An ROC curve or point is independent of class distribution or error costs.
- An ROC graph encapsulates all information contained in the confusion matrix, since FN is the complement of TP and TN is the complement of FP.
- ROC curves provide a visual tool for examining the tradeoff between the ability of a classifier to correctly identify positive cases and the number of negative cases that are incorrectly classified.

All available rain and no rain features are taken for classification. The validation parameters; viz. Precision, Sensitivity, Specificity and Accuracy are 83.2%, 81.9%, 82.9%, and 82.4% respectively. Here the 42 misclassified datasets pertain to rain features and 46 misclassified dataset to no rain features. Time taken for training the network is 18.46 seconds. Receiver Operating Characteristic (ROC) curves and best validation performance 0.001 at 141 epochs for this analysis is shown in figure 3.5 and 3.6.
Figure 3.5 ROC curve for BPNN

Figure 3.6 Validation Performance for BPNN