DYNAMIC GROWTH OF HIDDEN-LAYER NEURONS USING THE NON-EXTENSIVE ENTROPY

MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF

Master of Technology

In

Information Systems

Submitted By:

MAYANK DWIVEDI

(2K12/ISY/17)

Under the Guidance of

Dr. Seba Susan

(Assistant professor, Department of IT)

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(2012-2014)

CERTIFICATE

This is to certify that **Mayank Dwivedi (2K12/ISY/17)** has carried out the major project titled **"Dynamic growth of hidden-layer neurons using the non-extensive entropy"** in partial fulfilment of the requirements for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2012-2014**. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Dr. Seba Susan
Assistant professor
Department of Information Technology
Delhi Technological University
Delhi-110042

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project mentor **Dr. Seba Susan**, Assistant professor, Department of Information Technology, Delhi Technological University, Delhi for providing valuable guidance and constant encouragement throughout the project. It is my pleasure to record my sincere thanks to her for her constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Mayank Dwivedi
Roll No. 2K12/ISY/17
M. Tech. (Information Systems)
E-mail: mayankd.tct@gmail.com

ABSTRACT

The dynamic growth of hidden units and the pruning strategy has been sufficiently investigated in case of Radial Basis Function (RBF) neural network but relatively less in the case of feedforward multilayer perceptron (MLP) due to similarity in the hidden units in the MLP. So in this study I present a dynamic neural network that dynamically grows the number of hidden layer neurons based on an increase in the entropy of the weights during training. Before computing the entropy value weights are normalized to probability values. The entropy which is used being referred is the non-extensive entropy proposed recently by Susan and Hanmandlu for the representation of structured data. Along with the description of dynamic growth of hidden layer neurons using the Susan and Hanmandlu non-extensive entropy, the results are also compared with the Shannon, Pal and Pal, the Tsallis entropies and various static neural network configurations, in terms of execution time of the set of training samples, growth of hidden layer neurons and the testing accuracy. The experiment is performed on basically three standard machine learning datasets and on synthetic dataset. Incrementally growing the hidden layer as per requirement leads to better tuning of network weights and high classification performance as proved by the empirical results.

TABLE OF INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

In computer science and some more related fields Artificial Neural Network (ANN) is a computational model mainly inspired by animal central nervous system which is capable of machine learning and pattern recognition. So in simple way it can be well explained as an Artificial Neural Network is a network of well organised, interconnected neurons which takes inputs to train it-self and perform pattern recognition as well as classification.

Artificial Neural Network architecture mainly consists of three layers. First layer termed as input layer, from where an input is provided to the network for training, Second layer is known as hidden layer which is sandwiched between input layer and output layer and third layer is called as output layer where the result is collected for classification as well as for pattern recognition. Neural network, with their remarkable ability, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

1.1 Dynamic neural Network

A neural network which has fixed number of hidden layer neurons is known as static neural network. Due to non-adaptive nature of static neural network the concept of dynamic neural network comes in to the picture. The advent of dynamic neural network marks a significant advancement in the field of machine learning due to manifolds benefits it offers over the conventional neural network with a fixed architecture. The word dynamic in dynamic neural network refers to either dynamically changing the neural network architecture by adding or deleting some neurons [21] from hidden layer or incorporating dynamic units [22]. In Dynamic Neural Network mainly the network architecture is adaptive. And due to which it has several advantage over conventional or static neural network. Dynamic nature of neural network is mostly decided by the weights emanating from the training process. In [15] the neurons are added and weights are updated incrementally till the error is minimized.

1.2 Problem statement

It has been observed that when input data is little bit complex or multiple pattern has been observed among features of training data samples, the conventional neural network is not able to perform better due to its non-adaptive nature towards input pattern due to which it leads to more number of iteration than normal while training and which is responsible for increment in execution time as well as decrement in rate of convergence. So to overcome from all these problems it requires bringing the concept of dynamic neural network. In dynamic neural

network if input pattern is too complex, network dynamically insert a neuron in the hidden layer on one by one basis as per requirement or it depends upon the complexity of input pattern. Due to increment in number of hidden layer neurons in network, respond distribution of computation among multiple neurons due to which number of iteration will get decrease and rate of convergence will increase. In the field of machine learning it is necessary to have high convergence rate within a certain limit of time so up to certain extent it is better to use dynamic neural network instead conventional multi-layer perceptron or static neural network configurations.

1.3 Proposed solution

To decrease the execution time of training of input samples and make an enhancement in rate of convergence, some basic changes that have been done in conventional machine learning algorithm to make it dynamic. In my research work I incorporated the fusion of machine learning with statistics. Entropy part is taken from statistics and neural network is from machine learning, to bring the change in conventional multilayer perceptron. The use of statistical probability theory for improvisation in the performance of neural networks has been investigated before [23-25]. The scope of using statistics in neural network is due to the possibility of visualizing the neural network as a statistical model of non-linear regression and discriminant analysis. Since the entropy function is a measure of the chaos or uncertainty regarding an event, we use it to measure the state of chaos among the weights associated with the hidden layer in neural network. We grow the hidden layer dynamically by incrementing the hidden layer neurons by one if the chaos among the weights increases after a training session. Since we want that as the training progresses a declination in the calculated entropy must be there, so we use the Susan and Hanmandlu non-extensive entropy to measure uncertainty among the tuned weights, since it is proved successful for the representation of structured information in regular textures [31]. In short if present entropy is higher than the previous one then this states that there is chaos among the tuned weights and require increment in hidden layer neurons by one.

1.4 Justification and need of the algorithm

1. It decreases the number of iteration while training of input samples.
2. Rate of convergence become faster as compared to conventional neural network.
3. Network becomes dynamic or adaptive to decrease the execution time of algorithm.
4. It improves the efficiency and accuracy of dynamic neural networks.

CHAPTER 2
LITERATURE REVIEW

2.1 RBF Neural Network

2.1.1 Radial functions

Radial functions are a special class of function. Their characteristic feature is that their response decreases or increases monotonically with distance from a central point. The center, the distance scale and the precise shape of the radial function are parameters of the model, all fixed if it is linear.

A typical radial function is Gaussian which in the case of scalar input is,

$$h(\mathrm{x}) = \exp\left(-\frac{(\mathrm{x}-\mathrm{c})^2}{r^2}\right) \qquad (1)$$

Its parameters are its center c and its radius r.

2.1.2 Radial Basis Neural Network

Radial functions are simply a class of functions [1]. In principle, they could be employed in any sort of model (linear or nonlinear) and any sort of network (single layer or multi-layer). However, since Broomhead and Lowe's 1988 seminal paper [2], radial basis function networks (RBF networks) have traditionally been associated with radial functions in a single-layer network such as shown in figure.

A RBF network is nonlinear if the basis functions can move or change size or if there is more than one hidden layer.

2.1.3 Research on RBF

The dynamic growth of hidden units and the pruning strategy has been sufficiently investigated in the case of the Radial Basis Function (RBF) neural network.

Minimal Resource Allocation Network (M-RAN) has been proposed in [3]. M-RAN is a sequential learning radial basis function neural network which combines the growth criterion of the resource allocating network (RAN) of Platt with a pruning strategy based on the relative contribution of each hidden unit to the overall network output. The resulting network leads toward a minimal topology for the RAN. It presents a comprehensive comparison of the performance of M-RAN with MFN's on established benchmark problems in the function approximation and pattern classification areas. Platt's motivation for RAN stemmed from the

fact that learning with a fixed-size network is a NP-complete problem and by allocating new resources, learning could be achieved in polynomial time.

In [4], Supervisory Control of PV-Battery Systems is proposed using RBF neural network. It deals with a neural network based supervisor control system for a Photovoltaic (PV) plant. The aim of the work is to feed the power line with the 24 hours ahead forecast of the PV production. An on-line self-learning prediction algorithm is used to forecast the power production of the PV plant. The learning algorithm is based on a Radial Basis Function (RBF) network and combines the growing criterion and the pruning strategy of the minimal resource allocating network technique. The power feeding the electric line is scheduled by a Fuzzy Logic Supervisor (FLS) which controls the charge and discharge of a battery used as an energy buffer.

Growing and pruning (GAP)-RBF proposed in [5] uses the concept of "Significance" of a neuron and links it to the learning accuracy. "Significance" of a neuron is defined as its contribution to the network output averaged over all the input data received so far. Using a piecewise-linear approximation for the Gaussian function, a simple and efficient way of computing this significance has been derived for uniformly distributed input data. In the GAP-RBF algorithm, the growing and pruning are based on the significance of the "nearest" neuron. The growing and pruning criteria allows adding and pruning of neurons only when it is significant to the overall performance of the network. This results in a smooth growth of neurons and a compact network. Further, only the nearest neuron needs to be checked for growing and pruning.

In [6], architecture of dynamic fuzzy neural networks (D-FNN) implementing Takagi–Sugeno–Kang (TSK) fuzzy systems based on extended radial basis function (RBF) neural networks is proposed. A novel learning algorithm based on D-FNN is also presented. The salient characteristics of the algorithm are: 1) hierarchical on-line self-organizing learning is used; 2) neurons can be recruited or deleted dynamically according to their significance to the system's performance; and 3) fast learning speed can be achieved. Comparing with standard RBF neural networks, the term "extended RBF neural networks" implies that: 1) there are more than three layers; 2) no bias is considered; and 3) the weights may be a function instead of a real constant. Each node in layer 1 represents an input linguistic variable. Each node in layer 2 represents a membership function (MF) which is in the form of Gaussian functions. Each node in layer 3 represents a possible IF-part for fuzzy rules. Layer 4 represents

normalized nodes. Each node in layer 5 represents an output variable as the summation of incoming signals. Neuron generation depends upon system errors and accommodation boundary. Hierarchical learning is used where the accommodation boundary of each RBF unit is not fixed but adjusted dynamically in the following way: initially, the accommodation boundaries are set large for achieving rough but global learning. Then, they are gradually reduced for fine learning. The key idea of the hierarchical learning is to first find and cover the most troublesome positions, which have large errors between the desired and the actual outputs but are not properly covered by existing RBF units. This is called coarse learning. Thus in this based on the D-FNN, a hierarchical on-line self-organizing learning algorithm, whereby both the structure and parameter identification can be achieved quickly and simultaneously without iterative learning and initialization of the structure and parameters.

2.2 Multi-Layer Perceptron

A **multi-layer perceptron** (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called back-propagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

2.2.1 Activation function

A multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then it is easily proved with linear algebra that any number of layers can be reduced to the standard two-layer input-output model. What makes a multilayer perceptron different is that each neuron uses a nonlinear activation function which was developed to model the frequency of action potentials, or firing, of biological neurons in the brain. This function is modeled in several ways.

The two main activation functions used in current applications are both sigmoid, and are described by

$$y(\mathrm{v}_i) = \tanh(\mathrm{v}_i) \qquad (2)$$

$$y(v_i) = (1 + e^{-v_i})^{-1} \qquad (3)$$

The former function is a hyperbolic tangent which ranges from -1 to 1, and the latter, the logistic function, is similar in shape but ranges from 0 to 1. Here y_i is the output of the i^{th} node (neuron) and v_i is the weighted sum of the input synapses. Alternative activation functions have been proposed, including the rectifier and soft-plus functions.

2.2.2 Layers

The multilayer perceptron consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes and is thus considered a deep neural network. Each node in one layer connects with a certain weight w_{ij} to every node in the following layer.

2.2.3 Applications of MLP

Object extraction has been done using MLP and fuzziness measures [7]. A self-organizing multilayer neural network architecture suitable for image processing is proposed. The proposed architecture is also a feedforward one with back-propagation of errors; but like MLP it does not require any supervised learning. Each neuron is connected to the corresponding neuron in the previous layer and the set of neighbours of that neuron. The output status of neurons in the output layer is described as a fuzzy set. A fuzziness measure of this fuzzy set is used as a measure of error in the system (instability of the network). Neural networks are designated by the network topology, connection strength between pairs of neurons, node characteristics, and the status updating rules. The neurons operate in parallel, thereby providing output in real time. Since there are interactions among all the neurons, the collective computational property inherently reduces the computational task.

Image segmentation and edge detection has been done using MLP [37]. The system consists of a multilayer perceptron (MLP)-like network that performs image segmentation by adaptive thresholding of the input image using labels automatically pre-selected by a fuzzy clustering technique. The proposed architecture is feed forward, but unlike the conventional MLP the learning is unsupervised. The output status of the network is described as a fuzzy set. Fuzzy entropy is used as a measure of the error of the segmentation system as well as a criterion for determining potential edge pixels.

Image Restoration is another application of MLP [38]. The problem of restoring a blurred and noisy image having many gray levels, without any knowledge of the blurring function and the statistics of the additive noise, is considered. A multilevel sigmoidal function is used as the node nonlinearity. The same number of nodes as in the case of a binary image is sufficient for an image with multiple gray levels. Restoration is achieved by exploiting the generalization capabilities of the multilayer perceptron network.

2.3 The Shannon Entropy

In information theory, entropy is a measure of the uncertainty in a random variable. In this context, the term usually refers to the Shannon entropy, which quantifies the expected values of the information contained in a message. Shannon entropy is basically the average unpredictability in a random variable, which is equivalent to its information content. Shannon entropy provides an absolute limit on the best possible lossless encoding or compression of any communication, assuming that the communication may be represented as a sequence of independent and identically distributed random variables.

Shannon entropy is may be used globally, for the whole data, or locally, to evaluate entropy of probability density distributions around some points. This notion of entropy can be generalized to provide additional information about the importance of specific events.

Shannon [1]-[2] defined the entropy of an n-state system as

$$H = -\sum_{i=1}^{n} p_i \log(p_i) \qquad (4)$$

Where p_i is the probability of occurrence of the event i and

$$\sum_{i=1}^{n} p_i = 1 \qquad 0 \le p_i \le 1. \qquad (5)$$

It is concluded that the gain in the information from an event is inversely related to its probability of occurrence. Shannon used as the measure for such a gain.

$$\Delta I = \log(1/p_i) = -\log(p_i) \qquad (6)$$

. The expected value of the gain function is taken as the entropy, i.e.

$$\mathrm{H} = \mathrm{E}(\Delta I) = -\sum_{i=1}^{n} p_i \log(p_i) \qquad (7)$$

2.4 The Pal and Pal Entropy

Unlike the logarithmic behaviour of Shannon's entropy, the gain function considered here is of exponential nature. Based on the new concept, three definitions (e.g., global, local and conditional) of entropy of an image are then introduced and applied to formulate four algorithms for image segmentation. One of the algorithms assumes a passion distribution to describe the gray level variation within the object and background.

New definition of entropy:

Before giving any explanation about new entropy the following important points should be noted.

1. It is to be noted from the logarithmic entropic measure that as $p_i \to 0$, $\Delta I(p_i) \to \infty$ and $\Delta I(p_i) = -\log(p_i)$ is not defined for $p_i = 0$. Om the other hand, as $p_i \to 1$, $\Delta I(p_i) \to 0$ and $\Delta I(p_i = 1) = 0$. In practice, the gain in information from an event, whether highly probable or highly unlikely, is expected to lie between two finite limits.
2. In Shannon's theory the measure of ignorance or the gain in information is taken as $\log(1/p_i)$. But statically ignorance can be better represented by $(1- p_i)$ than $\log(1/p_i)$. The gain in information corresponding to the occurrence of i^{th} event can be defined as follows:

$$\Delta I(p_i) = \log(1- p_i) \text{ or } -\log(1- p_i) \qquad (8)$$

Then $\Delta I < 0$ or increases with p_i, which is intuitively unappealing.

The above problem can be circumvented by considering an exponential function of $(1- p_i)$.

Definition: here are some of the basic properties of new entropic function.

P1: $\Delta I(p_i)$ is defined at all points in [0, 1].

P2: Lim $p_i \to 0$ $\Delta I(p_i) = \Delta I(p_i = 0) = k_1, k_1 > 0$ and finite.

P3: Lim $p_i \to 1$ $\Delta I(p_i) = \Delta I(p_i = 1) = k_{2,} k2 > 0$ and finite.

P4: $k_2 < k_1$

P5: with increase in p_i, $\Delta I(p_i)$ decrease exponentially.

P6: $\Delta I(p_i)$ and H, the entropy, are continuous for $0 \le p \le 1$.

P7: H is max when all $p_{i,'s}$ equal.

In other words H ($p_1,...,p_n$) ≤ H ($1/n,...,1/n$)

The gain in information from an event with probability p_i as

$$\Delta I\,(p_i) = e^{(1-p_i)} \qquad (9)$$

And the entropy

$$H = E\,(\Delta I) = \sum_{i=1}^{n} p_i e^{(1-p_i)} \qquad (10)$$

The normalized entropy H_{nor} can be defined as

$$H_{nor} = k\,(H-1) \qquad (11)$$

Where k = $1/(e^{1-1/n}-1)$.

2.5 The Tsallis entropy

The **Tsallis entropy** [39**]** is a generalization of the standard Boltzmann–Gibbs entropy in physics. It was introduced in 1988 by Constantino Tsallis as a basis for generalizing the standard statistical mechanics. In the scientific literature, the physical relevance of the Tsallis entropy was occasionally debated. However, from the years 2000 on, an increasingly wide spectrum of natural, artificial and social complex systems have been identified which confirm the predictions and consequences that are derived from this non-additive entropy, such as non-extensive statistical mechanics, which generalizes the Boltzmann–Gibbs theory. Tsallis introduced a new definition for entropy which successfully describes the statistical feature of complex systems:

$$S_q = -k\frac{1-\sum_{i=1}^{N} p_i^{\,q}}{1-q} \qquad (12)$$

Where k is a positive constant, p_i stands for probability for occupation of i^{th} state of the system, N counts the known microstates of the systems and q is appositive real parameter. In my experiment q=1.5. Tsallis entropy is non-extensive entropy, which means that if two identical systems combine the entropy, the entropy of combined system is not equal to summation of entropy of its subsystems. Non-extensive statistical mechanics which is established by optimization of Tsallis entropy in presence of appropriate constraints, can interpret properties of many physical systems [40–43].

The more generalized equation of Tsallis entropy can be defined as

$$H(P) = \sum_{i=1}^{n} \frac{p_i - {p_i}^q}{q-1} \qquad (13)$$

Where q = 1.5.

2.6 Non-extensive entropy function having Gaussian information measure

This entropy is proposed in [21].In a quest for increasing the nonlinearity of the exponential family of entropies, the authors proposed non-extensive entropy function having exponential information gain with a quadratic exponent. Consider a random variable $X = \{x_1, x_2, ..., x_n\}$ with associated probabilities $P = \{p_1, p_2, ..., p_n\}$. Assume that the probability distribution is complete, i.e., $p_i \in [0,1]$ and $\sum_{i=1}^{n} p_i = 1$ for i=1,2,….n where, n is the number of probabilistic experiments.

Let the information gain on the i^{th} event of X with an associated probability p_i be defined by the Gaussian function:

$$I(p_i) = e^{-p_i^2} \qquad (14)$$

The entropy of X is defined as

$$H(P) = E(I(p_i)) = \sum_{i=1}^{n} p_i I(p_i) = \sum_{i=1}^{n} p_i e^{-{p_i}^2} \qquad (15)$$

The normalized entropy H_N is of the form:

$$H_N = \frac{(H - H_{\min})}{(H_{\max} - H_{\min})} \qquad (16)$$

i.e.

$$H_N = \frac{(H - e^{-1})}{e^{-\frac{1}{n^2}} - e^{-1}} \qquad (17)$$

The entropy in (17) is non-additive or non-extensive since for two random variables X and Y, the sum of individual entropies is always greater than the joint entropy of X and Y, i.e.,

H(X) + H(Y) > H(X, Y)

As proved in [44]. Unlike Shannon entropy, the equality condition is not satisfied for the statistically independent case. The non-extensive entropy has certain inherent advantages as compared with the other forms of entropy for the detection of anomalies or deviants from normal behaviour. The main reason behind its success is the nonlinearity of its Gaussian information gain function which ensures that low probability events fall inside the bell of the Gaussian and high probability events are rejected as being non-relevant or carrying less information. This leads to a better detection of anomalies or deviants from the normal behaviour. In the presence of highly correlated fields, the non-additive Gaussian information is more apt than the logarithmic information measure.

2.6.1 Weighted sum of non-extensive entropies

The entropy associated with each state x is computed using the set of conditional probabilities $p(\mathrm{y}|x)$. The entropy of the source S is defined as the weighted average of the entropies $H_N(p(\mathrm{y}|x))$ associated with the set of states in X, where the entropy of each state $\mathrm{x} \in \mathrm{X}$ is weighted by its probability p(x).

The entropy of the source S is defined by,

$$H(S) = \sum_{x \in X} p(x) H(p(y/x)) \qquad (18)$$

Where the sum of probabilities of the various states x is one, i.e.

$$\sum_{x \in X} p(\mathrm{x}) = 1$$

The entropy $H_N(p\ (y\ |x))$ in (18) is the normalized non extensive entropy associated with state x and is computed using (17) as,

$$H(p(y/x)) = \sum_{y \in Y} p(y/x) e^{-p(y/x)^2} \qquad (19)$$

Where, the sum of probabilities of all y leading from state x is one, i.e.

$$S_q = -k\frac{1-\sum_{i=1}^{N} p_i^{\,q}}{1-q} \qquad (20)$$

Here, both p(x) ∈ [0, 1] and p(y |x) ∈ [0, 1].

2.7 Extreme Learning Machine

The single hidden layer feedforward neural network (SLFN) that has several practical applications [15-16] and uses a random number of hidden nodes [28] was improvised by Huang *et al* in [29] to devise a three step learning methods called the Extreme Learning Machine (ELM) that does not use back-propagation (BP) or other iterative techniques. ELM is a unified SLFN with randomly generated hidden nodes independent of the training data [8], [9]. The output of an SLFN with hidden nodes (which may not be neuron alike [9]) can be represented by

$$f_L(\mathrm{x}) = \sum_{i=1}^{L} \beta_i G(\mathrm{a}_i, \mathrm{b}_i, \mathrm{x}), x \in R^n, a_i \in R^n \qquad (21)$$

where a_i and b_i are the learning parameters of hidden nodes and β_i the weight connecting the i^{th} hidden node to the output node. Here $G(\mathrm{a}_i, \mathrm{b}_i, \mathrm{x})$ is the output of the i^{th} hidden node with respect to the input. SLFNs with a wide type of random computational hidden nodes have the universal approximation capability as long as SLFNs with this type of adjustable hidden nodes (tuned by some learning algorithms) can be universal approximators. Such computational hidden nodes include additive/radial basis function (RBF) hidden nodes, multiplicative nodes, fuzzy rules, etc. It was established that ELM is an extremely fast batch learning algorithm and can provide good generalization performance. The key advantages of ELM as compared to other classical methods[10] (such as back-propagation algorithm (BP) and support vector machine (SVM) [11]) are that ELM needs no iteration when determining the hidden node parameters, which dramatically reduces the computational time for training the model, and it is very simple and easy to implement ELM. However, the preliminary ELM [12, 13] does not provide an effective solution for architectural design of the network. For ELM, in most cases, the suitable number of hidden nodes is pre-determined by a trial and error method, which may be tedious in some applications. Hence, some researchers have advocated that the network structure should not be determined by a trial and error method but

should instead be computed by the learning algorithm [14]. There are normally two heuristic approaches to modify the structure of SLFNs: constructive methods (or growing methods) and destructive methods (or pruning methods).

The ELM algorithm which only consists of three steps can be summarized as:

ELM algorithm: Given a training set, activation function g(x), and hidden node Number L

(1) Assign random hidden nodes by randomly generating learning parameters (a_i, b_i), i=1…L.

(2) Calculate the hidden layer output matrix H .

(3) Calculate the output weight $\beta: \beta = H^{\dagger}T$, where $H^{\dagger}$

T, where $H^{\dagger}$ is the Moore–Penrose generalized inverse of the hidden layer output matrix H. The singular value decomposition (SVD) method is used to calculate the Moore–Penrose generalized inverse $H^{\dagger}$ in ELM and most of the implementations of ELM.

Error Minimized Extreme Learning Machine with growth of hidden nodes is proposed in [15]. In the paper, a simple and efficient method to automatically determine the number of hidden nodes in generalized SLFNs is proposed. Computational hidden nodes adopted in such SLFNs include additive/RBF hidden nodes, multiplicative nodes, fuzzy rules [16], fully complex nodes [17], [18], hinging functions etc. Unlike the other sequential/incremental/growing learning algorithms such as I-ELM, RAN, and MRAN, which add hidden nodes on by one, our new approach allows the random hidden nodes to be added one by one or group by group (with varying group size). The output weights can be incrementally updated efficiently during the growth of the networks. The simulation results on sigmoid type of hidden nodes show that the new approach can significantly reduce the computational complexity of ELM.

Constructive hidden nodes selection of extreme learning machine for regression has been proposed in [19].The main idea of CS-ELM is to identify the significance of each hidden node and select the optimal subset of hidden nodes when the stopping criterion reaches its minimum. The hidden node is selected one-by-one from the candidate reservoir. The proposed algorithm achieves a compact network structure with competitive generalization performance when compared to the preliminary ELM. In addition, CS-ELM selects the

hidden nodes based on the correlation between hidden nodes and the current residual, while EM-ELM adds the randomly generated hidden nodes without any selection.

Growing neural network has been proposed in [20].In the growing NN with hidden neurons, the growth is formulated as an extension of BP learning. At first, the output and the error are calculated, and the error signal is propagated backward. If the output neuron does not have enough connections, the neuron diffuses the signal as a chemical substance. Then, the concentration gradient is formed by the diffusion around the output neuron. The growing neuron extends its axon according to the concentration gradient. The hidden neurons that do not have enough connections diffuse another substance constantly not depending on the error. By the diffusion of the hidden neurons, the input neurons extend its axons and make connections to the hidden neurons at first. After that, the weight is increased gradually until the hidden neuron makes a connection to an output neuron. The hidden neuron that has formed the connections with the input neurons extends its axon to an output neuron referring to the input signals. From just after making the connection (synapse), the learning according to the regular BP algorithm begins.

CHAPTER 3
PROPOSED METHODOLOGY

The dynamic growth of hidden units and the pruning strategy has been sufficiently investigated in the case of Radial Basis Function (RBF) neural network but relatively less in the case of feed forward multilayer perceptron (MLP). So in my research work I used MLP instead of RBF in dynamic neural network.

3.1 Architecture

In dynamic neural network the architecture we used is basically 3 layer architecture. First layer is called as input layer from where input is given to the network for training of input samples. Second layer is called as hidden layer where multiple layers sandwiched between input layer and output layer, mainly used to process information from input to output layer. And third and last layer is called as output layer where output is collected for further classification. The number of nodes or neurons present in the input layer is depends upon the number of features in the input training samples. So architecture may differ according to training samples. Number of neurons in the hidden layer has been taken three for initiation, but as training progresses it will increases automatically as per complexity of the training samples or as per requirement. Output layer contains only one neuron for classification which is nothing but our aim.

The main computing takes place in these three layers only. Associated with each hidden neuron is a set of weights that are scales of connection between the hidden layer neurons and the neurons in the input and output layers. There is no connection between neurons of same layer in the network. These weights are significant since they are responsible for the fraction of the input that is fed in to the **activation function** of the hidden neurons and thereby influence its output. Activation function is mainly calculated on every other node of hidden layers as well as output layer, which is responsible to processes information forward. The hidden layer units are significant due to their ability to interpret data in a meaningful way.

3.2 Flow of information in the neural networks

Initially first input sample has been provided to the input layer in start of the training process. Individual input values then multiplied with corresponding weights and we get sum of the product of input value with weights associated with each hidden layer neurons. This sum of product values goes through activation function attached with each hidden layer neurons. Output of activation function will act as an input for further information processes. This whole process will continue till information will reach up to output layer. This process will

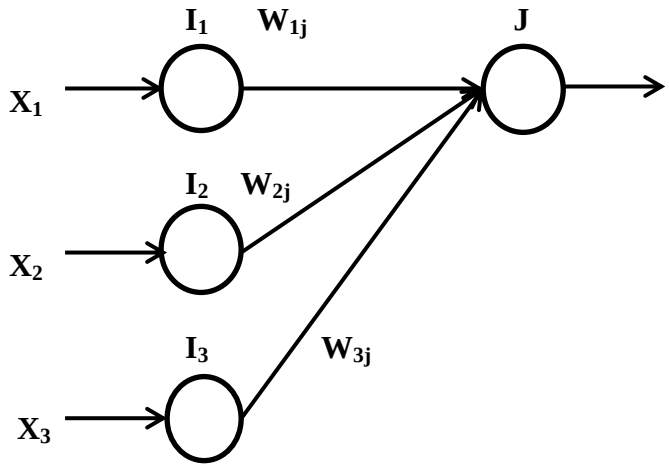
work according to the **feed-forward algorithm** to process information to the output layer. At output layer it is necessary to calculate error difference between resultant output and desired output. If error difference is greater than the given threshold value then it require to back-propagate up to the input layer for clear classification. **Back-propagation** is an algorithm which is used to process information backwards up to input layer and updates the weights associated with neurons. These feed forwards and back-propagation algorithms simultaneously used to propagate information from input layer to output layer and vice-versa and weights get tuned for classification.

3.3 Feed-forward algorithm

In feed-forward neural network information is moved only in forward direction and weights are not updated during this step. Features of input training samples are passed in to the input layer. These features value then multiplied with corresponding weights which is associated with hidden layer neurons to input layer neurons called as input layer weights. At the hidden layer neurons the sum of product of features value to the corresponding weight, is passed in to the activation function. Though if there are multiple layers in a hidden layer then the process of forwarding of information between two layers will be done as per feed-forward neural network algorithm. That's why the output of activation function is acting as an input in next information sharing process.

Let x_1, x_2, x_3……..x_n are features of input training samples.

Fig. 3.1 Shows the connection of input along with its weights.

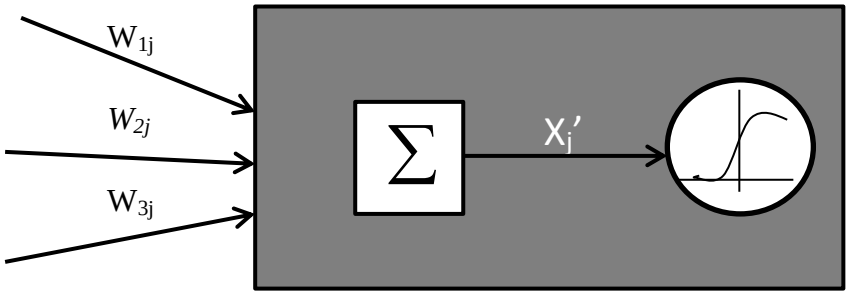


As shown above I_1, I_2, I_3……I_n are input layer neurons which is equal to the number of features in input training samples. W_{1j}, W_{2j}, W_{3j}……W_{nj} are the weights associated with input layer neurons to hidden layer neurons. Neuron *'J'* is a hidden layer neuron which has an activation function along with it. Now sum of product of input values with corresponding weights can be calculated as follows:

$$x_j' = \sum_i w_{ij} x_i + w_j \qquad (21)$$

Here *Wj* is well known for biased weight which is a fixed value throughout the training process.

fig. 3.2 Description of activation function inside a hidden layer neuron or output layer



Node J

In the above diagram x_j' represents the sum of product of input with corresponding weights. Now x_j, which is the output of activation function in the hidden layer neuron, can be calculated as follows:

$$x_j = f(x_j') = \tanh(x_j') \qquad (22)$$

Here $\tanh(x_j')$ is hyperbolic tangent activation function, used in every hidden layer nodes. So this process of forward feeding of information will continue up to last layer of neural network.

3.4 Back-propagation algorithm

The **backward error propagation**, also called as the **back-propagation (BP)** or the **generalized delta rule (GDR)** is mainly used to feed information back up to the input layer and on the basis of error gradient which is calculated at output layer, update the weights

associated with neurons to minimizing the error. But first, a squared error measure for the p^{th} input-output pair is defined as

$$E_p = \sum_k (\mathrm{d}_k - \mathrm{x}_k)^2 \qquad (23)$$

Where d_k is the desired output for node k, and x_k is the actual output for node k when the input part of the pth data pair is presented. To find the gradient vector, an error term ε_i' for node i is defined as

$$\varepsilon_i' = \frac{\partial^+ E_p}{\partial x_i'} \qquad (24)$$

According to the chain rule the recursive formula for above equation can be written as follows.

$$\varepsilon_i' = \begin{cases} -2(\mathrm{d}_i - \mathrm{x}_i)\dfrac{\partial x_i}{\partial x_i'} = -2(\mathrm{d}_i - \mathrm{x}_i)\,\mathrm{x}_i(1-\mathrm{x} & \text{If node } i \text{ is a output node} \\ \dfrac{\partial x_i}{\partial x_i'} = \sum_{j,i<j} \dfrac{\partial^+ E_p}{\partial x_j'}\dfrac{\partial x_j'}{\partial x_i} = x_i(1-\mathrm{x}_i)\sum_{j,i<j} \varepsilon_j' w_{ij} & \text{else} \end{cases} \qquad (25)$$

The above two equations are used for calculating the error gardient which is further used in updation of weights. Weight updation in back propagation algorithm can be done as follows

$$\Delta w_{ki} = -\eta \frac{\partial^+ E_p}{\partial w_{ki}} = -\eta \frac{\partial^+ E_p}{\partial x_i'}\frac{\partial x_i'}{\partial w_{ki}} = -\eta \varepsilon_i' x_k \qquad (26)$$

$$w_{ki} = w_{ki} + \Delta w_{ki} \qquad (27)$$

Where η is a learning rate that affect the convergence speed and stability of the weight during learning. The update formula for the bias of each node can derived similarly. w_{ki} is a connection weight between *k*th hidden layer node to *i*th output layer node if node *i* is output node. Here Δw_{ki} is change in weight which is calculated using error gradient in back-propagation algorithm.

Feed-forward and back-propagation are two basic algorithm which are used to tarin any neural network. But in my proposed work I introduced a concept of Susan and Hanmandlu non-extensive entropy which is mainly used to make decision about weather number of neurons in the hidden layer should increased or to keep stable.

3.5 Dynamic growth of hidden units

The growth or pruning of the hidden units is mostly decided by the weights emanating from the training process and in Susan and Hanmandlu non-extensive entropy, normalized weights are used as a probabilistic values. As explained above the susan and hanmandlu non-extensive etropy was propaosed as a feature for texture identification and classification[31]. Its speciality lies in identifying regular textures containg repetative patterns that translate to high co-occurrence probabilities. But in my work the weighted sum of non-extensive entropies is used to identify the irregular patterns among features of input tarining samples. The weighted sum of non-extensive entropies over two consecutive frame pairs was used in [35] for the detection of video anomalies. The significant of this weighted sum of non-extensive entropy is that any significant departur from normal will result in a positive spike indicating an anomaly. The positive spike in the weighted sum of non-extensive entropy is due to more information gain. Because as we know that if there are similar patterns in a input training samples the new information gathering is less due to similarity in features and resulting less information gain but if there is no repetitive pattern among features then information gathering is high resulting higher information gain and result in a positive spike.

So in my work I computed over time the weighted sum of non-extensive entropies associated with the trained weights of a feedforward network, tuned by back-propagation algorithm. So by using weighted sum of non-extensive entropy, as a decision factor to increment the neurons, if a positive spike or increase in te weighted sum indicate a kind of chaos or uncertainty among tarining input features and increments the number of hidden layer neurons by one. The weighted sum of non-extensive entropies, also called as the entropy of the source *H(S)* is defined as

$$H(S) = \sum_{x \in X} p(x) H(p(y/x)) \qquad (28)$$

Solving for the entropy *H(p(y/x))* using [above equation number], we have

$$H(p(y/x)) = \sum_{y \in Y} p(y/x) e^{-p(y/x)^2} \qquad (29)$$

Here *p(x)* denotes the probability associated with the weight between the output neurons and the x^{th} hidden unit. Before applying non-extensive entropy to the network weights must be normalized in to the range of [0, 1], then it is converted in to probabilistic values. The set *{p(y/x)}* indicates the probabilities associated with the input connection weights of the x^{th} hidden unit, and *H (p(y/x))* is non-extensive entropy associated with *{p(y/x)}* computed using (29). To calculate weighted sum of non-extensive entropy it is necessary normalized the weights for converting them in to probabilistic values then non-extensive entropy should be calculated on the basis of weights associated with input layer neurons to hidden layer neurons. Now sum of the product of the probabilities associated with the weights between hidden layer neurons to output layer neuron and non-extensive entropy calculated on the basis of probabilities associated with the weights between input layer neurons to hidden layer neurons. Resulting weighted sum of non-extensive entropies, also called as entropy of the source.

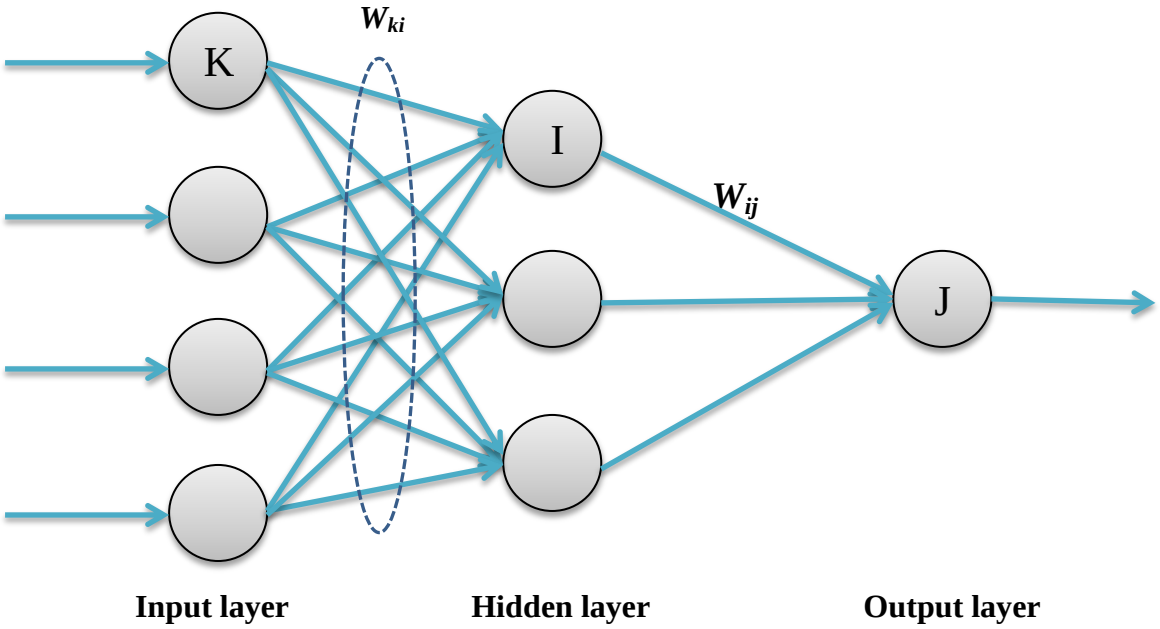
The multilayer perceptron architecture 4-3-1 with the actual weights associated with each layer is shown in fig. 3, is a basic architecture used in dynamic neural network. The multilayer perceptron is a neural network having multiple layers in a hidden layer, but in my architecture there is only one layer in the hidden layer with randomly initialized 3 neurons. Let W_{ki}^{t} denote the weight associated with the connection between the k^{th} input dimension and the i^{th} hidden layer neuron after the training of the i^{th} training data sample. The computation of weighted sum of non-extensive entropy in [28] from the network weights $\{W_{ki}^{t}\}$ and $\{W_{ij}^{t}\}$ is explained below. The normalized weights $\{W_{kiN}^{t}\}$ are obtained from $\{W_{ki}^{t}\}$ by transforming the value of the weights $\{W_{ki}^{t}\}$ in the range of 0 to 1, and dividing by the total sum of weights as shown in (30) to (31).

$$w_{kip}^{t} = \frac{w_{ki}^{t} - \min_{\forall k}(w_{ki}^{t})}{\max_{\forall k}(w_{ki}^{t}) - \min_{\forall k}(w_{ki}^{t})} \quad , \text{k=1, 2… M} \qquad (30)$$

Normalized weights in the form of probabilities are given below as

$$w_{kiN}^{t} = \frac{w_{kip}^{t}}{\sum_{k=1}^{M} w_{kip}^{t}}, k = 1, 2, 3, ..., M \qquad (31)$$

Fig. 3.3 The multi-layer perceptron (MLP) neural and connection weights during training



Here *M* is the number of connection weights between the input layer and the hidden neuron *i*. The normalized weights $\{W_{1iN}^{t}, W_{2iN}^{t} \ldots W_{MiN}^{t}\}$ are equal to probability values $\{p(y/x)\}$ for a given x, with values in the range of 0 to 1 and $\sum_{k=1}^{M} W_{kiN}^{t} = 1$. This probability distribution can be used for the computation of the entropy $H\ (p(y/x))$ using [equation]. The probabilities $\{p(x)\}$ associated with the *H* hidden neurons and the single output neuron (j=1) are the normalized weights $\{W_{ijN}^{t}\}$ which are computed from $\{W_{ij}^{t}\}$ in a similar manner. The weighted sum of non-extensive entropies given by (28) is generalized for *O* significant outputs as

$$\mathrm{H}(S) = \sum_{j=1}^{O} \frac{1}{O} \sum_{i=1}^{H} w_{ijN^{t}} \sum_{k=1}^{M} w_{kiN^{t}} e^{-w_{kiN^{t}}^{2}} \qquad (32)$$

In our example *O*=1. The entropy value computed using (32) for the t^{th} training sample is compared with the entropy value for the previous training data samples *t-1*. If incoming entropy value is higher than the previous input training samples entropy then it is assumed

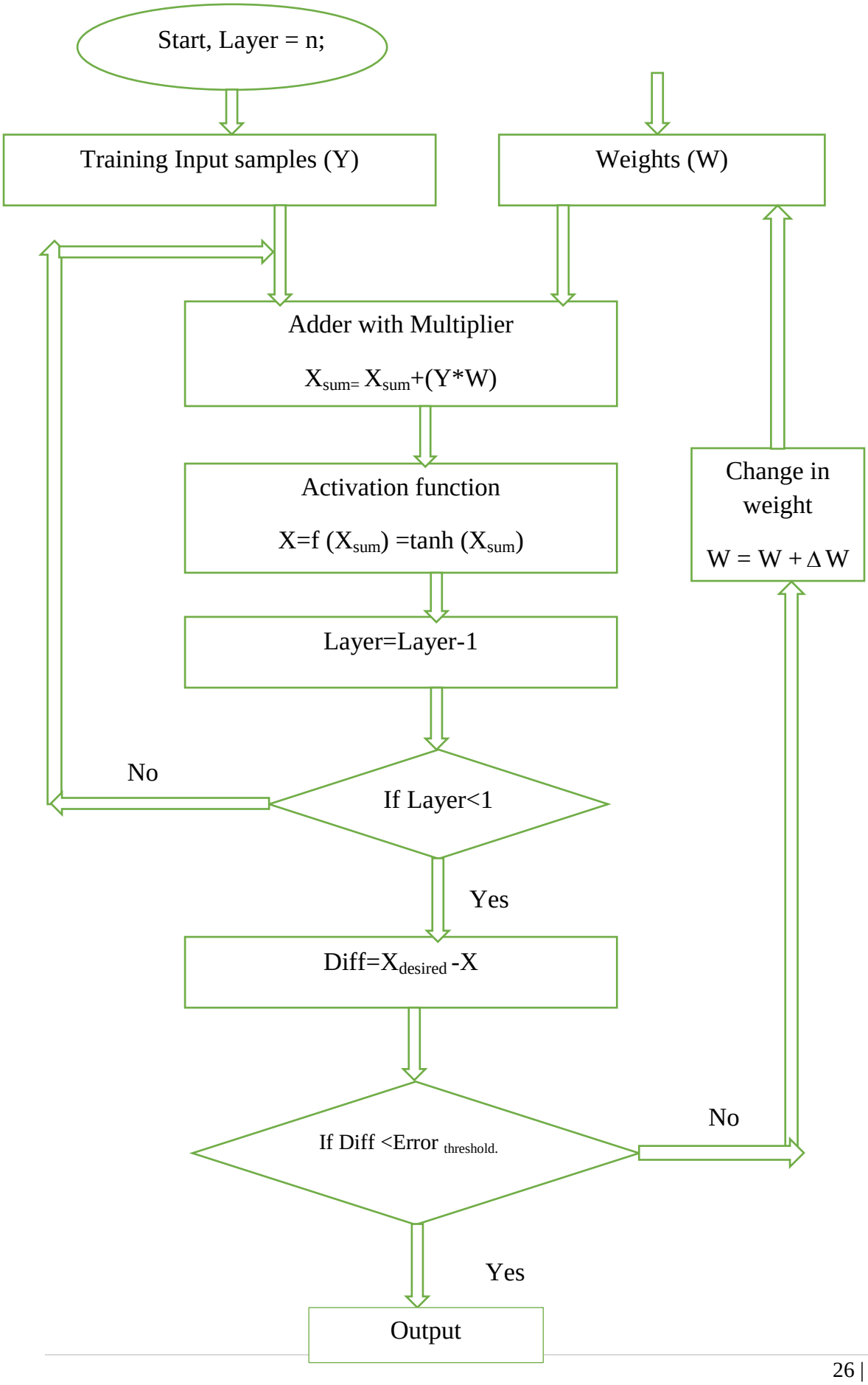
that there is a positive spike in entropy *H(S)*, the number of hidden layer neuron is increased by one and the training proceeds for sample *t+1* . A single memory storage for the previous entropy value is only required for comparison purpose.

The steps of proposed methods are summarized below.

1. Initialize the neural network architecture with initially 3 hidden neurons. For input layer number of neurons should be equal to the number of features in input training samples. Number of neuron at output layer is always remains one. Hidden layer weights initialized to 0.11 and output layer weights initialized to 0.13.
2. Tuning of the weights for a training inputs will be done by feed-forward and back-propagation algorithm until error goes below to the given threshold value.
3. To calculate the probabilistic value it is necessary to normalize the weights in the range of 0 to 1 and then divide the individual weight by the sum of all the weights. So to get probability of the tuned weights of each layer using (30) and (31).
4. After tuning of all weights through back-propagation compute the weighted sum of non-extensive entropies using (32). We called it weighted sum of non-extensive entropy because individual weights multiplied with corresponding neuron's entropy.
5. Though we are storing the entropy of previously executed input training sample, so if current calculated weighted sum of non-extensive entropy value is higher than the previous training sample's entropy then increment the number of hidden layer neurons by one, and weight corresponding to newly added neuron being initialized to 0.11.
6. Repeat the training process for the next sample using the tuned weights of the existing neurons and the newly added neuron.
7. Repeat above steps 2-6 until the training is complete.

The highlight of our method is thus that a new computation unit is added whenever an unusual pattern occurs in the training. This trend was also observed before in the case of the resource allocating network in [36] which however requires memorizing of specific input patterns that perform poorly, in order to achieve desired accuracy.

Fig. 3.4 flow chart of working of single layer feedforward multilayer perceptron



CHAPTER 4
EXPERIMENTAL RESULTS

4.1 Experiment performed on synthetic data set

4.1.1 Description about synthetic data set:

The synthetic data set I used in my experiment is basically having four features and total numbers of data samples are 100 in numbers. Basically it is not inspired with any other data set but simply classified itself in two different classes as 0 and 1. Features of class 1 are in the range of 0 to 15 and of class 0 are in the range of 100 to 200. Among data set of 100 samples 40 samples are used for training and 60 samples are used for testing purpose. Total number of samples belongs to class 1 are 60 and number of samples in class 0 are 40. None of the features in any sample are missing in synthetic data set.

4.1.2 Experimental results of synthetic data set

The single hidden layer feed forward multilayer perceptron 4-3-1 shown in fig. 3**,** is used as the initial configuration for experiment on synthetic data set, with a learning rate η of 0.02 and an error threshold 'ε' of 0.01. The error threshold is basically different for different data sets. The number of hidden layer units is initialized to 3 at the beginning of training procedure and is grown dynamically as explained in section 3.5. The result of dynamic growth of hidden layer neurons as training progresses is shown in table no.1. When consecutive training pattern follows a similar pattern irrespective of the actual range of values, the non-extensive entropy of the weights decreases (or remain stable) indicating a decline in in chaos in the weights as some weights are assigned higher values than other due to higher significance their connections. As explained above in Susan and Hanmandlu non-extensive entropy that its specialty lies in identifying regular texture containing repetitive patterns that translate to high co-occurrence probabilities. So whenever an unnatural training pattern is said to be encountered when it is different from its predecessors in its relationship between the features. In any training sample if there is similar pattern like equal difference between features or if numbers are in the range of 0 and 1 only then how many non-zero and zero values are there. So this is what a pattern called on the basis of number of neurons will increases. The concept of information gain works here in the way that if there are multiple sample following similar pattern in among its features it means pattern is well known by algorithm, so if a next input training sample has same pattern as previous one it mean information gain is less because nothing is new in that, but pattern has changed or it differs from previous sample's pattern it means some is new encountered by the algorithm and

information gain is high. Due to high information gain the weighted sum of non-extensive entropy will also become high and there will be a positive spike which indicates a chaos among the weights or weighted sum of non-extensive entropy is higher than the previous one and number of hidden layer neurons increment by one. For example feature number 2 and 3 of training samples 9 and 34 are closer than the usual. The distinct training inputs are detected correctly by the non-extensive entropy in synthetic data set (4, 9, 34, and 36) while the Shannon entropy is highly sensitive and includes lots of false alarms (2, 5, 7, 18, 33, 35, 39), for instance sample 2 is normally occurring pattern in the dataset. The same kinds of fact were also observed for video anomaly example in [35] where Shannon entropy gives rise to false alarm due to random crowd movement.

The results are summarized in Table 2 for proposed dynamic neural network and the static neural network with fixed number of hidden layer neurons. We observed a slow and steady rise of the number of hidden neurons for the non-extensive entropy (hidden neurons=7), while Shannon entropy results in growth of 11 hidden neurons, Pal and Pal entropy results in growth of 8 hidden neurons and Tsallis entropy results in growth of 7 hidden layer neurons. Though Pal and Pal and Tsallis entropies are results in approx. similar number of hidden layer neurons but they took more execution time required to train a given set of input samples than non-extensive entropy. Apart from a reduced MLP architecture with only 7 hidden layer nodes, the overall training time required by the proposed method is much less than the equivalent fixed neurons architecture, while its classification performance is (98.33%) is higher than any possible static neural network configuration. As it is clearly derived from the Table 2 that static growth of neurons up to 22 can only give you approximate efficiency equal to non-extensive entropy, but this can be easily achieved by dynamic neural network with non-extensive entropy having only 7 hidden layer neurons.

Table 4.1: **Dynamic growth of hidden layer neurons using the Susan and Hanmandlu non-extensive, Shannon, Pal and Pal and Tsallis entropies based on synthetic dataset.**

Input training Sample number	Training feature 1	Training feature 2	Training feature 3	Training feature 4	Class	Growth of neurons using non-extensive entropy	Growth of neurons using Shannon Entropy	Growth of neurons using Pal and Pal Entropy	Growth of neurons using Tsallis entropy
1	0.01	0.02	0.03	0.04	1	3	3	3	3
2	0.4	0.5	0.6	0.7	1	3	4	3	3
3	0.10	0.11	0.12	0.13	1	3	4	3	3
4	0.12	0.1	0.16	0.18	1	4	4	4	4
5	0.35	0.25	0.45	0.55	1	4	5	4	4
6	0.55	0.40	0.56	0.65	1	4	5	5	4
7	0.05	0.06	0.07	0.08	1	4	6	5	5
8	0.61	0.63	0.67	0.69	1	4	6	5	5
9	0.71	0.73	0.74	0.77	1	5	6	5	5
10	0.94	0.91	0.93	0.95	1	5	7	5	5
11	2.3	2.5	2.57	2.9	1	5	7	5	5
12	3.1	3.25	3.7	3.9	1	5	7	5	5
13	4.3	4.5	4.7	4.9	1	5	7	5	5
14	5.1	5.4	5.6	5.9	1	5	7	5	5
15	6.7	6.9	6.4	6.45	1	5	7	5	5
16	8.0	8.5	8.3	8.4	1	5	7	5	5
17	7.5	7.7	7.9	7.2	1	5	7	5	5
18	1.2	1.3	1.5	1.7	1	5	8	5	5
19	9.2	9.3	9.5	9.7	1	5	8	5	5
20	3.9	24	5.4	5.9	1	5	8	5	5
21	10.0	11.1	12.2	13.3	1	5	8	5	5

22	5.2	5.5	5.7	5.9	1	5	8	5	5
23	5.1	5.3	5.4	5.7	1	5	8	5	5
24	3.2	3.4	3.6	3.8	1	5	8	5	5
25	3.9	3.1	3.5	3.7	1	5	8	5	5
26	7.9	7.4	7.2	7.5	1	5	8	5	5
27	3.2	7.7	7.5	9.2	1	5	8	5	5
28	2.3	2.5	2.7	2.9	1	5	8	5	5
29	7.3	7.9	7.7	7.5	1	5	8	5	5
30	4.7	4.55	4.3	4.1	1	5	8	5	5
31	1.2	1.9	1.59	1.7	1	5	8	5	5
32	104.5	109.6	108.7	107.8	0	5	8	5	5
33	100	101	102	103	0	5	9	5	5
34	110	113	114	117	0	6	9	5	5
35	121	123	125	127	0	6	10	6	5
36	131	121	141	151	0	7	10	6	6
37	141	143	147	149	0	7	10	7	6
38	80.1	85.11	87.2	84.9	0	7	10	7	7
39	70.0	70.2	70.3	70.7	0	7	11	8	7
40	67.2	65.5	75.5	85.0	0	7	11	8	7

Table 4.2: Performance analysis of the proposed dynamic neural network, Shannon, Pal and Pal, Tsallis entropies and various static neural network configurations, in terms of training time in seconds (s), number of hidden nodes and the testing accuracy in (%), based on Synthetic data set.

Methods	Growth of hidden layer neurons using Susan and Hanmandlu Non-extensive Entropy.	Growth of hidden layer neurons using Shannon Entropy.	Growth of hidden layer neurons using Pal and Pal Entropy.	Growth of hidden layer neurons using Tsallis Entropy.	Static 3 Hidden neurons	Static 7 Hidden neurons	Static 11 Hidden neurons	Static 22 Hidden neurons	Static 34 Hidden neurons	Static 40 Hidden neurons
Total training time (for 40 samples)	0.39 sec	0.78 sec	0.589sec	0.602sec	0.25sec	0.51sec	0.58sec	0.6sec	0.62sec	0.62sec
Number of hidden nodes	7	11	8	7	3	7	11	22	34	40
Testing accuracy (for 60 samples)	98.33%	98.33%	98.33%	98.33%	83.43%	83.4%	93.4%	93.4%	83.4%	83.4%

4.2 Experiment performed on dermatology dataset

4.2.1 Description about dermatology data set.

This database contains mainly 34 attributes 33 of which are linear valued and one of them is nominal. The differential diagnosis of erythemato-squamous diseases is a real problem in dermatology. They all share the clinical features of erythema and scaling, with very little differences. The deceases in this group are psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris. Usually a biopsy is necessary for the diagnosis but unfortunately these diseases share many histopathological features as well. Another difficulty for the differential diagnosis is that a disease may show the features of another disease at the beginning stage and may have the characteristic features at the following stages. Patients were first evaluated clinically with 12 features. Afterwards, skin samples were taken for the evaluation of 22 histopathological features. The values of the histopathological features are determined by an analysis of the samples under a microscope.

In the dataset constructed for this domain, the family history feature has the value 1 if any of these diseases has been observed in the family and 0 otherwise. The age feature simply represents the age of the patient. Every other feature (clinical and histopathological) was given a degree in the range of 0 to 3. Here, 0 indicates that the feature was not present, 3 indicates the largest amount possible, and 1, 2 indicate the relative intermediate values.

The name and id numbers of the patients were recently removed from the database.

No. of instances: 366

No. of attributes or features: 34

Attribute Information:

Clinical Attributes: (take values 0, 1, 2, 3, unless otherwise indicated)

1: Erythema
2: Scaling
3: Definite borders
4: Itching
5: Koebner phenomenon
6: Polygonal papules

7: Follicular papules

8: Oral mucosal involvement

9: Knee and elbow involvement

10: Scalp involvement

11: Family history, (0 or 1)

34: Age (linear)

Histopathological Attributes: (take values 0, 1, 2, 3)

12: Melanin incontinence

13: Eosinophils in the infiltrate

14: PNL infiltrate

15: Fibrosis of the papillary dermis

16: Exocytosis

17: Acanthosis

18: Hyperkeratosis

19: Parakeratosis

20: Clubbing of the rete ridges

21: Elongation of the rete ridges

22: Thinning of the suprapapillary epidermis

23: Spongiform pustule

24: Munro microabcess

25: Focal hypergranulosis

26: Disappearance of the granular layer

27: Vacuolisation and damage of basal layer

28: Spongiosis

29: Saw-tooth appearance of retes

30: Follicular horn plug

31: Perifollicular parakeratosis

32: Inflammatory monoluclear inflitrate

33: Band-like infiltrate

Class distribution:

Class code:	Class:	Number of instances:
1	Psoriasis	112
2	Seboreic dermatitis	61

3	Lichen planus	72
4	Pityriasis rosea	49
5	Cronic dermatitis	52
6	Pityriasis rubra pilaris	20

4.2.2 Experimental results of dermatology data set:

The experiments are conducted on data set containing 366 data samples distributed among 6 different classes as described above. But in single layer feed forward multilayer perceptron has only one single output neurons which can only able to classify the datasets having two classes, so to match the compatibility criteria the classes 1,2 and 3 are merged in to class 1 and classes 4, 5 and 6 are merged in class 0. The total numbers of data samples in class 1 are 245 and in class 0 are 121. Among the dataset of 366 data samples alternate samples (ex. 1, 3, 5**…**365) are used for training purpose and remaining (2, 4,…366) for testing purpose, this way total 183 samples has been taken for training as well for testing also. The experiment were performed on dynamic neural network using non-extensive, Shannon, Pal and pal, Tsallis and various static neural network configurations, in terms of training time in seconds (s), number of hidden nodes and the testing accuracy in (%). As it is depicted from Table 3 that growth of hidden layer neurons in the case of non-extensive entropy and Shannon entropy are same but time required to train 183 input training data samples is less (0.192 sec) in the case of non-extensive entropy as compared to Shannon (0.242 sec), Pal and Pal (0.196 sec), and Tsallis (0.198 sec) entropies. In terms of accurate classification, the Susan and Hanmandlu non-extensive entropy classify the testing input data samples more accurately (85.80%) as compare to Shannon (80.0%), Pal and Pal (84.16%) and Tsallis (83.61%) entropies. If we talk about various static neural network configurations in that case also the Susan and Hanmandlu non-extensive entropy works better as compared to static neural network configuration having hidden layer neurons 3, 7, etc. This whole discussion concludes that the dynamic growth of hidden layer neurons using the Susan and Hanmandlu non-extensive entropy as proposed in my work not only gives the optimal number of hidden layer neurons for a data set, at the same time it provides a better tuning of weights and faster training than any other non-extensive, extensive and static neural network configurations. Though the pattern of growth of hidden layer neuron using the Susan and Hanmandlu non-extensive entropy is quite different with other entropies it means definitely there must be some false alarms given by other entropies because other entropies have more number of hidden layer neurons as compared to the Susan and Hanmandlu non-extensive entropy.

Table 4.3: Dynamic growth of hidden layer neurons using the Susan and Hanmandlu non-extensive, Shannon, Pal and Pal and Tsallis entropies based on Dermatology dataset.

Serial number	Sample number	Class	Growth of hidden neurons using the non-extensive Susan and Hanmandlu entropy	Growth of hidden neurons using the Shannon entropy	Growth of hidden neurons using the Pal and Pal	Growth of hidden neurons using the Tsallis entropy
1	1	1	3	3	3	3
2	3	1	3	3	3	3
3	5	1	4	4	4	4
4	7	1	4	4	4	4
5	9	1	5	4	4	5
6	11	1	5	5	5	5
7	13	1	6	5	5	6
8	15	1	6	5	5	6
9	17	1	6	6	6	6
10	19	1	7	6	6	7
11	21	1	7	6	6	7
12	23	1	8	7	7	8
13	25	1	8	7	7	8
14	27	1	9	7	7	9
15	29	1	9	8	8	9
16	31	1	10	8	8	10
17	33	1	10	8	8	10
18	35	1	11	9	9	11
19	37	1	11	9	9	11
20	39	1	12	9	10	12
21	41	1	12	10	10	12

22	43	1	12	10	11	12
23	45	1	13	11	11	13
24	47	1	13	11	12	13
25	49	1	13	12	12	13
26	51	1	13	12	12	13
27	53	1	14	13	13	14
28	55	1	14	13	13	14
29	57	1	14	13	13	14
30	59	1	14	14	14	14
31	61	1	15	14	14	15
32	63	1	15	14	15	15
33	65	1	15	14	15	15
34	67	1	16	15	16	16
35	69	1	16	15	16	16
36	71	1	17	15	17	17
37	73	1	17	15	17	17
38	75	1	18	16	17	18
39	77	1	18	16	18	18
40	79	1	19	16	18	18
41	81	1	19	17	18	18
42	83	1	20	17	19	19
43	85	1	20	18	19	19
44	87	1	20	18	19	19
45	89	1	20	18	19	19
46	91	1	21	19	19	19
47	93	1	21	19	20	19
48	95	1	22	19	20	19

49	97	1	22	20	20	20
50	99	1	22	20	20	20
51	101	1	22	20	21	20
52	103	1	22	20	21	21
53	105	1	22	21	21	21
54	107	1	22	21	22	21
55	109	1	22	21	22	21
56	111	1	22	21	22	21
57	113	1	22	21	22	21
58	115	1	23	22	22	21
59	117	1	23	22	23	21
60	119	1	23	22	23	22
61	121	1	23	22	23	22
62	123	1	24	23	24	22
63	125	1	24	23	24	22
64	127	1	25	23	24	23
65	129	1	25	23	25	23
66	131	1	25	24	25	23
67	133	1	25	24	25	23
68	135	1	25	25	25	23
69	137	1	25	25	25	23
70	139	1	25	26	25	24
71	141	1	25	26	25	24
72	143	1	25	27	25	24
73	145	1	25	27	25	25
74	147	1	26	27	25	25
75	149	1	26	28	25	25

76	151	1	26	28	25	25
77	153	1	27	28	25	25
78	155	1	27	28	26	25
79	157	1	27	28	26	25
80	159	1	27	29	26	25
81	161	1	27	29	27	26
82	163	1	27	29	27	26
83	165	1	27	30	27	26
84	167	1	28	30	28	26
85	169	1	28	30	28	26
86	171	1	29	30	28	26
87	173	1	29	30	28	26
88	175	1	29	30	28	26
89	177	1	29	30	28	26
90	179	1	29	30	28	26
91	181	1	29	30	28	26
92	183	1	29	30	28	27
93	185	1	30	30	28	27
94	187	1	30	30	28	27
95	189	1	30	30	28	28
96	191	1	30	30	28	28
97	193	1	30	30	28	29
98	195	1	30	30	28	29
99	197	1	30	30	28	29
100	199	1	30	30	28	29
101	201	1	30	30	28	29
102	203	1	30	30	28	29

103	205	1	30	30	28	29
104	207	1	30	30	28	29
105	209	1	30	30	28	29
106	211	1	30	30	28	29
107	213	1	30	30	28	29
108	215	1	30	30	28	29
109	217	1	30	30	28	29
110	219	1	30	30	28	29
111	221	1	30	30	28	29
112	223	1	30	30	28	29
113	225	1	30	30	28	29
114	227	1	30	30	28	29
115	229	1	30	30	29	29
116	231	1	30	30	29	29
117	233	1	30	30	29	29
118	235	1	30	30	29	29
119	237	1	30	30	29	29
120	239	1	30	30	29	29
121	241	1	30	30	29	29
122	243	1	30	30	29	29
123	245	1	30	30	29	29
124	247	0	30	31	29	30
125	249	0	30	31	30	30
126	251	0	30	32	30	31
127	253	0	30	32	30	31
128	255	0	30	32	30	31
129	257	0	31	33	30	32

130	259	0	31	33	31	32
131	261	0	32	33	31	33
132	263	0	32	34	32	33
133	265	0	32	34	32	33
134	267	0	33	34	32	34
135	269	0	33	35	32	34
136	271	0	34	35	33	34
137	273	0	34	35	33	35
138	275	0	35	36	33	35
139	277	0	35	36	34	35
140	279	0	36	37	34	36
141	281	0	36	37	35	36
142	283	0	36	37	35	36
143	285	0	36	37	36	36
145	287	0	37	38	36	37
146	289	0	37	38	36	37
147	291	0	37	38	37	38
148	293	0	38	38	37	38
149	295	0	38	39	38	38
150	297	0	38	39	38	39
151	299	0	38	40	39	39
152	301	0	39	40	39	40
153	303	0	39	41	40	40
154	305	0	39	41	40	40
155	307	0	39	41	40	41
156	309	0	39	41	41	41
157	311	0	40	42	41	42

158	313	0	40	42	41	42
159	315	0	41	43	42	42
160	317	0	41	43	42	43
161	319	0	42	44	43	43
162	321	0	42	44	43	43
163	323	0	42	44	44	43
164	325	0	43	44	44	44
165	327	0	43	44	44	44
166	329	0	44	45	45	44
167	331	0	44	45	45	45
168	333	0	45	45	45	45
169	335	0	45	46	46	45
170	337	0	46	46	46	46
171	339	0	46	47	47	46
172	341	0	47	47	47	47
173	343	0	47	47	47	47
174	345	0	47	47	48	48
175	347	0	48	48	48	48
176	349	0	48	48	49	48
177	351	0	48	48	49	49
178	353	0	48	48	50	49
179	355	0	49	48	50	50
180	357	0	49	49	50	50
181	359	0	50	49	51	51
182	361	0	50	50	51	51
183	363	0	51	50	52	52
184	365	0	51	51	52	52

Table 4.4: Performance analysis of the proposed dynamic neural network, Shannon entropy, Pal and Pal entropy, Tsallis entropy and various static neural network configurations, in terms of training time in seconds (s), number of hidden nodes and the testing accuracy in (%), based on Dermatology dataset.

Methods	Growth of hidden layer neurons using Susan and Hanmandlu Non-extensive Entropy.	Growth of hidden layer neurons using Shannon Entropy.	Growth of hidden layer neurons using pal n pal entropy.	Growth of hidden layer neurons using Tsallis Entropy.	Static 3 Hidden neurons	Static 7 Hidden neurons	Static 11 Hidden neurons	Static 22 Hidden neurons	Static 34 Hidden neurons	Static 40 Hidden neurons	Static 50 Hidden neurons
Total training time (for 183 samples)	0.192 sec	0.242sec	0.196sec	0.198sec	0.198sec	0.198sec	0.156sec	0.184sec	0.236sec	0.472sec	2.793sec
Number of hidden nodes	51	51	52	52	3	7	11	22	34	40	50
Testing accuracy (for 183 samples)	85.80%	80.00%	84.16%	83.61%	67.22%	67.22%	67.59%	82.52%	83.07%	84.16%	84.70%

4.3 Experiment performed on Iris dataset

4.3.1 Description about Iris dataset:

This dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2.

Attribute predicted: Class of Iris plant.

This data differs from the data presented in Fishers article (identified by Steve Chadwick, spchadwick@espeedaz.net). The 35th sample should be: 4.9, 3.1, 1.5, 0.2, "Iris-setosa" where the error is in the fourth feature and 38th sample: 4.9, 3.6, 1.4, and 0.1, "Iris-setosa" where the errors are in the second and third features.

Number of attributes: 4 numeric, predictive attributes and the class.

Attribute Information:

1. Sepal length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm

Class:

-- Iris Setosa

-- Iris Versicolour

-- Iris Virginica

4.3.2 Experimental results of Iris data set:

The experiment is conducted on linearly separable Iris data set containing 150 data samples distributed among 3 different classes, 50 samples each. The single layer feedforward multilayer perceptron as shown in fig. 3 is used as the initial configuration for the experiments, with a leaning rate η of 0.02 and an error threshold 'ε' of 0.01. The error threshold is basically different for different data sets. The number of hidden layer units is initialized to 3 at the beginning of training procedure and is grown dynamically as explained in [section number]. The result of dynamic growth of hidden layer neurons as training progresses is shown in Table 5. Though in single layer feedforward multilayer perceptron has only one neuron in the output layer which can able to classify only two different classes, so to match the compatibility criteria class 1 and 2 are merged in to class 0 and class 3

renamed as class 1. This way 100 input data samples belongs to class 0 and remaining 50 data samples belongs to class 1. The alternate input data samples of Iris data set are basically used for training and testing purpose. Among 150 input data set 75 data samples of dataset are used as training and remaining 75 input data samples are used as testing purpose. In testing data samples number 50 input training data samples are belongs to class 0 and 25 input training data samples are belongs to class 1. In Table 5, the dynamic growth of hidden layer neurons using the Susan and Hanmandlu non-extensive entropy is less as compared to Shannon, Pal and Pal and Tsallis entropies. The steady and slow rise of hidden layer neurons in the Susan and Hanmandlu non-extensive entropy shows that the MLP architecture will remain small as compared to other extensive and non-extensive entropies which will take less time to train a set of input training data samples, enhance the efficiency of neural network with good classification accuracy. The same thing what I explained above can easily concluded from Table 6, where the growth of hidden layer neurons in case of the Susan and Hanmandlu non-extensive entropy, are 18 and in case of Shannon, Pal and Pal and Tsallis entropy, are 20, 20, 21 simultaneously. In terms of execution time of training samples the Susan and Hanmandlu non-extensive entropy took less time (0.051 sec) as compared to the Shannon (0.136 sec), Pal and Pal (0.152 sec), Tsallis (0.168 sec) and various static neural network configurations. Though in terms of training time and growth of hidden nodes the Susan and Hanmandlu non-extensive entropy performed better as compared to all other entropies as well as various static neural network configurations but in terms of testing accuracy all entropies classify the testing data samples with accuracy of 93.44%. To make sure the correctness of the algorithm the cross validation also been done on Iris dataset by taking testing data samples as a training and training data samples as testing.

Table 4.5: Dynamic growths of hidden layer neurons using the Susan and Hanmandlu non-extensive, Shannon, Pal and Pal and Tsallis entropy based on Iris dataset.

Serial Number	Training feature 1	Training feature 2	Training feature 3	Training feature 4	Class	Growth of hidden neurons using the S&H non-Ext. Entropy	Growth of hidden neurons using the Shannon Entropy	Growth of hidden neurons using the pal & Pal Entropy	Growth of hidden neurons using the Tsallis entropy
1	5.1	3.5	1.4	0.2	0	3	3	3	3
2	4.7	3.2	1.3	0.2	0	3	3	3	3
3	5	3.6	1.4	0.2	0	3	3	3	3
4	4.6	3.4	1.4	0.3	0	3	3	3	3
5	4.4	2.9	1.4	0.2	0	4	4	4	4
6	5.4	3.7	1.5	0.2	0	4	4	4	4
7	4.8	3	1.4	0.1	0	5	5	4	5
8	5.8	4	1.2	0.2	0	5	5	4	5
9	5.4	3.9	1.3	0.4	0	6	6	5	6
10	5.7	3.8	1.7	0.3	0	6	6	5	6
11	5.4	3.4	1.7	0.2	0	7	7	6	6
12	4.6	3.6	1	0.2	0	7	7	6	7
13	4.8	3.4	1.9	0.2	0	7	7	6	7
14	5	3.4	1.6	0.4	0	7	8	7	7
15	5.2	3.4	1.4	0.2	0	7	8	7	7
16	4.8	3.1	1.6	0.2	0	7	9	7	7
17	5.2	4.1	1.5	0.1	0	7	9	8	8
18	4.9	3.1	1.5	0.2	0	7	9	8	8
19	5.5	3.5	1.3	0.2	0	8	10	8	9

20	4.4	3	1.3	0.2	0	8	10	8	9
21	5	3.5	1.3	0.3	0	8	10	9	9
22	4.4	3.2	1.3	0.2	0	8	11	9	9
23	5.1	3.8	1.9	0.4	0	9	11	9	10
24	5.1	3.8	1.6	0.2	0	9	11	10	10
25	5.3	3.7	1.5	0.2	0	9	11	10	10
26	7	3.2	4.7	1.4	0	9	11	10	10
27	6.9	3.1	4.9	1.5	0	10	12	10	10
28	6.5	2.8	4.6	1.5	0	10	12	11	11
29	6.3	3.3	4.7	1.6	0	10	12	11	11
30	6.6	2.9	4.6	1.3	0	10	13	11	12
31	5	2	3.5	1	0	10	13	11	12
32	6	2.2	4	1	0	10	13	12	12
33	5.6	2.9	3.6	1.3	0	11	13	12	13
34	5.6	3	4.5	1.5	0	11	14	12	13
35	6.2	2.2	4.5	1.5	0	11	14	13	13
36	5.9	3.2	4.8	1.8	0	11	14	13	13
37	6.3	2.5	4.9	1.5	0	12	15	13	14
38	6.4	2.9	4.3	1.3	0	12	15	14	14
39	6.8	2.8	4.8	1.4	0	12	15	14	14
40	6	2.9	4.5	1.5	0	12	16	14	14
41	5.5	2.4	3.8	1.1	0	12	16	14	15
42	5.8	2.7	3.9	1.2	0	13	16	14	15
43	5.4	3	4.5	1.5	0	13	16	14	15
44	6.7	3.1	4.7	1.5	0	14	16	14	15
45	5.6	3	4.1	1.3	0	14	17	14	15
46	5.5	2.6	4.4	1.2	0	14	17	14	15

47	5.8	2.6	4	1.2	0	14	18	14	16
48	5.6	2.7	4.2	1.3	0	15	18	15	16
49	5.7	2.9	4.2	1.3	0	15	19	15	17
50	5.1	2.5	3	1.1	0	16	19	16	17
51	6.3	3.3	6	2.5	1	16	20	16	18
52	7.1	3	5.9	2.1	1	16	20	16	18
53	6.5	3	5.8	2.2	1	16	20	16	18
54	4.9	2.5	4.5	1.7	1	16	20	16	18
55	6.7	2.5	5.8	1.8	1	16	20	16	18
56	6.5	3.2	5.1	2	1	16	20	16	18
57	6.8	3	5.5	2.1	1	16	20	16	18
58	5.8	2.8	5.1	2.4	1	16	20	16	18
59	6.5	3	5.5	1.8	1	16	20	16	18
60	7.7	2.6	6.9	2.3	1	16	20	16	18
61	6.9	3.2	5.7	2.3	1	16	20	16	18
62	7.7	2.8	6.7	2	1	16	20	16	18
63	6.7	3.3	5.7	2.1	1	16	20	16	18
64	6.2	2.8	4.8	1.8	1	16	20	16	18
65	6.4	2.8	5.6	2.1	1	16	20	16	18
66	7.4	2.8	6.1	1.9	1	16	20	16	18
67	6.4	2.8	5.6	2.2	1	16	20	16	18
68	6.1	2.6	5.6	1.4	1	17	20	16	18
69	6.3	3.4	5.6	2.4	1	17	20	17	18
70	6	3	4.8	1.8	1	17	20	17	19
71	6.7	3.1	5.6	2.4	1	17	20	18	19
72	5.8	2.7	5.1	1.9	1	17	20	18	20
73	6.7	3.3	5.7	2.5	1	18	20	19	20

74	6.3	2.5	5	1.9	1	18	20	19	21
75	6.2	3.4	5.4	2.3	1	18	20	20	21

Table 4.6: Performance analysis of the proposed dynamic neural network, Shannon entropy, Pal and Pal entropy, Tsallis entropy and various static neural network configurations, in terms of training time in sec (s), number of hidden nodes and the testing accuracy in (%), based on Iris dataset.

Methods	Growth of hidden layer neurons using Susan and Hanmandlu Non-extensive Entropy.	Growth of hidden layer neurons using Shannon Entropy.	Growth of hidden layer neurons using Pal and Pal entropy.	Growth of hidden layer neurons using Tsallis Entropy.	Static 3 Hidden neurons	Static 7 Hidden neurons	Static 11 Hidden neurons	Static 22 Hidden neurons	Static 34 Hidden neurons	Static 40 Hidden neurons
Total training time (for 75 samples)	0.051 sec	0.136sec	0.152sec	0.168sec	0.090sec	0.082sec	0.173sec	0.129sec	0.107sec	0.116sec
Number of hidden nodes	18	20	20	21	3	7	11	22	34	40
Testing accuracy (for 75 samples)	93.44%	93.44%	93.44%	93.44%	89.44%	89.44%	90.77%	93.44%	93.44%	90.77%

4.3.3 Cross validation of Iris data set:

In the cross validation of Iris data set, as explained above the training input data samples of Iris dataset will become testing data samples and testing data samples become training input data samples, to check the correctness of algorithm whether it is accurately classifying the dataset or not. The result of dynamic growth of hidden layer neurons as well explained in Table 7. And in terms of execution time of training samples along with the testing accuracy the results are shown in Table 8. In both the tables the dynamic growth of hidden layer neurons using the non-extensive entropy performed better as compared to other extensive,

non-extensive entropies which have been taken under experimental consideration and various static neural network configurations.

Table 4.7: Dynamic growths of hidden layer neurons using the Susan and Hanmandlu non-extensive, Shannon, Pal and Pal and Tsallis entropy based on Iris dataset.

Serial Number	Training feature 1	Training feature 2	Training feature 3	Training feature 4	Class	Growth of hidden neurons using the S&H non-Ext. Entropy	Growth of hidden neurons using the Shannon Entropy	Growth of hidden neurons using the pal & Pal Entropy
1	4.9	3	1.4	0.2	3	3	3	3
2	4.6	3.1	1.5	0.2	3	3	3	3
3	5.4	3.9	1.7	0.4	3	3	3	4
4	5	3.4	1.5	0.2	3	3	3	4
5	4.9	3.1	1.5	0.1	3	4	4	4
6	4.8	3.4	1.6	0.2	4	4	4	4
7	4.3	3	1.1	0.1	4	4	5	5
8	5.7	4.4	1.5	0.4	5	4	5	5
9	5.1	3.5	1.4	0.3	5	5	5	5
10	5.1	3.8	1.5	0.3	5	5	5	5
11	5.1	3.7	1.5	0.4	5	5	6	5
12	5.1	3.3	1.7	0.5	6	6	6	5
13	5	3	1.6	0.2	6	6	6	5
14	5.2	3.5	1.5	0.2	6	6	7	6
15	4.7	3.2	1.6	0.2	7	7	7	6
16	5.4	3.4	1.5	0.4	7	7	7	7
17	5.5	4.2	1.4	0.2	7	7	7	7
18	5	3.2	1.2	0.2	7	8	8	7

19	4.9	3.6	1.4	0.1	7	8	8	8
20	5.1	3.4	1.5	0.2	7	9	9	8
21	4.5	2.3	1.3	0.3	7	9	9	9
22	5	3.5	1.6	0.6	8	10	9	9
23	4.8	3	1.4	0.3	8	10	9	9
24	4.6	3.2	1.4	0.2	8	11	9	9
25	5	3.3	1.4	0.2	8	11	9	9
26	6.4	3.2	4.5	1.5	9	12	10	9
27	5.5	2.3	4	1.3	9	12	10	9
28	5.7	2.8	4.5	1.3	9	13	10	10
29	4.9	2.4	3.3	1	10	13	10	10
30	5.2	2.7	3.9	1.4	10	13	11	11
31	5.9	3	4.2	1.5	10	14	11	11
32	6.1	2.9	4.7	1.4	11	14	12	12
33	6.7	3.1	4.4	1.4	11	14	12	12
34	5.8	2.7	4.1	1	11	14	13	12
35	5.6	2.5	3.9	1.1	12	14	13	12
36	6.1	2.8	4	1.3	12	15	13	13
37	6.1	2.8	4.7	1.2	12	15	14	13
38	6.6	3	4.4	1.4	12	15	14	14
39	6.7	3	5	1.7	13	16	14	14
40	5.7	2.6	3.5	1	13	16	15	14
41	5.5	2.4	3.7	1	13	16	15	15
42	6	2.7	5.1	1.6	14	16	15	15
43	6	3.4	4.5	1.6	14	16	16	16
44	6.3	2.3	4.4	1.3	14	16	16	16
45	5.5	2.5	4	1.3	14	16	17	17

46	6.1	3	4.6	1.4	15	17	17	17
47	5	2.3	3.3	1	15	17	17	17
48	5.7	3	4.2	1.2	16	18	18	18
49	6.2	2.9	4.3	1.3	16	18	18	18
50	5.7	2.8	4.1	1.3	17	18	18	19
51	5.8	2.7	5.1	1.9	17	19	19	19
52	6.3	2.9	5.6	1.8	17	19	19	19
53	7.6	3	6.6	2.1	17	19	19	19
54	7.3	2.9	6.3	1.8	17	19	19	19
55	7.2	3.6	6.1	2.5	17	19	19	19
56	6.4	2.7	5.3	1.9	17	19	19	19
57	5.7	2.5	5	2	17	19	19	19
58	6.4	3.2	5.3	2.3	17	19	19	19
59	7.7	3.8	6.7	2.2	17	19	19	19
60	6	2.2	5	1.5	17	19	19	19
61	5.6	2.8	4.9	2	17	19	19	19
62	6.3	2.7	4.9	1.8	17	19	19	19
63	7.2	3.2	6	1.8	17	19	19	19
64	6.1	3	4.9	1.8	17	19	19	19
65	7.2	3	5.8	1.6	17	19	19	19
66	7.9	3.8	6.4	2	17	19	19	19
67	6.3	2.8	5.1	1.5	17	19	19	19
68	7.7	3	6.1	2.3	17	19	19	19
69	6.4	3.1	5.5	1.8	17	19	19	19
70	6.9	3.1	5.4	2.1	17	19	19	19
71	6.9	3.1	5.1	2.3	18	20	20	20
72	6.8	3.2	5.9	2.3	18	20	20	20

73	6.7	3	5.2	2.3	18	21	21	20
74	6.5	3	5.2	2	19	21	21	20
75	5.9	3	5.1	1.8	19	21	21	20

Table 4.8: Performance analysis of the proposed dynamic neural network, Shannon entropy, Pal and Pal entropy, Tsallis entropy and various static neural network configurations, in terms of training time in sec (s), number of hidden nodes and the testing accuracy in (%), based on Iris dataset.

Methods	Growth of hidden layer neurons using Susan and Hanmandlu Non-extensive Entropy.	Growth of hidden layer neurons using Shannon Entropy.	Growth of hidden layer neurons using Pal and Pal entropy.	Growth of hidden layer neurons using Tsallis Entropy.	Static 3 Hidden neurons	Static 7 Hidden neurons	Static 11 Hidden neurons	Static 22 Hidden neurons	Static 34 Hidden neurons	Static 40 Hidden neurons
Total training time (for 75 samples)	0115sec	0.163sec	0.186sec	0.170sec	0.1122sec	0.194sec	0.199sec	0.190sec	0.141sec	0.199sec
Number of hidden nodes	19	21	21	20	3	7	11	22	34	40
Testing accuracy (for 75 samples)	90.77%	89.44%	89.44%	90.77%	88%	89.44%	89.44%	90.77%	90.77%	90.77%

4.4 Experiment performed on SPECT Heart dataset

4.4.1 Description of SPECT Heart dataset:

The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature patterns were created for each patient. The pattern was further processed to obtain 22 binary feature patterns. The CLIP3 algorithm was used to generate classification rules from these patterns. The CLIP3 algorithm generated rules that were 84.0% accurate (as compared with cardiologist diagnoses).

Number of Instances: 267

Number of Attributes: 23 (22 binary + 1 binary class)

Attribute Information:

1. OVERALL_DIAGNOSIS: 0, 1 (class attribute, binary)
2. F1: 0, 1 (the partial diagnosis 1, binary)
3. F2: 0, 1 (the partial diagnosis 2, binary)
4. F3: 0, 1 (the partial diagnosis 3, binary)
5. F4: 0, 1 (the partial diagnosis 4, binary)
6. F5: 0, 1 (the partial diagnosis 5, binary)
7. F6: 0, 1 (the partial diagnosis 6, binary)
8. F7: 0, 1 (the partial diagnosis 7, binary)
9. F8: 0, 1 (the partial diagnosis 8, binary)
10. F9: 0, 1 (the partial diagnosis 9, binary)
11. F10: 0, 1 (the partial diagnosis 10, binary)
12. F11: 0, 1 (the partial diagnosis 11, binary)
13. F12: 0, 1 (the partial diagnosis 12, binary)
14. F13: 0, 1 (the partial diagnosis 13, binary)
15. F14: 0, 1 (the partial diagnosis 14, binary)
16. F15: 0, 1 (the partial diagnosis 15, binary)
17. F16: 0, 1 (the partial diagnosis 16, binary)
18. F17: 0, 1 (the partial diagnosis 17, binary)
19. F18: 0, 1 (the partial diagnosis 18, binary)
20. F19: 0, 1 (the partial diagnosis 19, binary)

21. F20: 0, 1 (the partial diagnosis 20, binary)

22. F21: 0, 1 (the partial diagnosis 21, binary)

23. F22: 0, 1 (the partial diagnosis 22, binary)

-- Dataset is divided into:

-- Training data (80 instances)

-- Testing data (187 instances)

8. Missing Attribute Values: None

9. Class Distribution:

-- Entire data

Class	# examples
0	55
1	212

-- Training dataset

Class	# examples
0	40
1	40

-- Testing dataset

Class	# examples
0	15
1	172

4.4.2 Experimental results of SPECT Heart dataset:

The experiment is conducted on linearly separable SPECT Heart data set containing 267 data samples distributed among 2 different classes. Among 267 samples of SPECT Heart dataset 55 input data samples are belongs to class 0 and remaining 212 input data samples belongs to class 1. The single layer feedforward multilayer perceptron as shown in fig. 3, is used as the initial configuration for the experiments, with a leaning rate η of 0.02 and an error threshold 'ε' of 0.11. The single layer feed forward multilayer perceptron is used as an initial configuration for this experiment with initial 3 hidden layer neurons and is grown dynamically as explained above in section [section]. The SPECT Heart dataset is well structured data set, that's why the Susan and Hanmandlu non-extensive entropy works better as compare to other extensive entropies. The number of hidden layer neurons in the case of the Susan and Hanmandlu non-extensive entropy, are 38 which is quite lesser as compared to

the Shannon (43), Pal and Pal (42) and Tsallis (49) entropies. Less number of hidden layer neurons shows that the neural network architecture will also become small and execution time required for training the given set of input data samples will also decrease. The results of execution time of training data along with testing accuracy in Table 10, is also shows that the Susan and Hanmandlu non extensive entropy took less execution time (0.136 sec) while training of 80 input training data samples as compared to the Shannon (0.287 sec), Pal and Pal (0.187 sec), Tsallis (0.186 sec) and various other static neural network configurations. In terms of testing accuracy the Susan and Hanmandlu non-extensive entropy and Tsallis entropy classify the given input testing data samples up to 93.05%, which is higher than the any non-extensive, extensive and any possible static neural network configuration. This suggest that the dynamic growth of hidden layer as proposed in my work not only gives the optimal number of hidden neurons for a dataset, at the same time it provides a better tuning of weights and faster training than any static neural network configuration.

Table: 4.9 Dynamic growths of hidden layer neurons using the Susan and Hanmandlu non-extensive, Shannon, Pal and Pal and Tsallis entropy based on SPECT Heart dataset.

Serial number	Sample number	Class	Growth of hidden neurons using the non-extensive Susan and Hanmandlu entropy	Growth of hidden neurons using the Shannon entropy	Growth of hidden neurons using the Pal and Pal	Growth of hidden neurons using the Tsallis entropy
1	S. N. 1	1	3	3	3	3
2	S. N. 2	1	4	4	4	4
3	S. N. 3	1	4	4	4	4
4	S. N. 4	1	4	5	4	5
5	S. N. 5	1	5	5	5	5
6	S. N. 6	1	5	5	5	6
7	S. N. 7	1	6	6	6	6
8	S. N. 8	1	6	6	6	7
9	S. N. 9	1	6	6	6	7

10	S. N. 10	1	7	6	7	7
11	S. N. 11	1	7	7	7	8
12	S. N. 12	1	8	7	8	8
13	S. N. 13	1	8	7	8	9
14	S. N. 14	1	8	8	8	9
15	S. N. 15	1	8	8	8	10
16	S. N. 16	1	9	8	9	10
17	S. N. 17	1	9	9	9	10
18	S. N. 18	1	9	9	9	11
19	S. N. 19	1	10	10	10	11
20	S. N. 20	1	10	10	10	11
21	S. N. 21	1	10	11	10	12
22	S. N. 22	1	10	11	10	12
23	S. N. 23	1	10	11	10	12
24	S. N. 24	1	11	11	11	12
25	S. N. 25	1	11	11	11	12
26	S. N. 26	1	11	12	11	12
27	S. N. 27	1	11	12	11	12
28	S. N. 28	1	11	12	11	12
29	S. N. 29	1	11	13	12	13
30	S. N. 30	1	11	13	12	13
31	S. N. 31	1	12	13	13	13
32	S. N. 32	1	12	13	13	14
33	S. N. 33	1	13	13	14	14
34	S. N. 34	1	13	13	14	15
35	S. N. 35	1	13	14	14	15
36	S. N. 36	1	14	14	15	16

37	S. N. 37	1	14	14	15	16
38	S. N. 38	1	14	15	16	16
39	S. N. 39	1	14	15	16	16
40	S. N. 40	1	15	16	17	16
41	S. N. 41	0	15	16	17	16
42	S. N. 42	0	15	16	17	17
43	S. N. 43	0	15	16	17	17
44	S. N. 44	0	15	16	17	18
45	S. N. 45	0	16	17	17	18
46	S. N. 46	0	16	17	18	18
47	S. N. 47	0	16	17	18	19
48	S. N. 48	0	17	17	18	19
49	S. N. 49	0	17	17	18	19
50	S. N. 50	0	17	17	19	20
51	S. N. 51	0	17	17	19	20
52	S. N. 52	0	18	18	20	21
53	S. N. 53	0	18	18	20	22
54	S. N. 54	0	19	19	21	23
55	S. N. 55	0	19	20	22	24
56	S. N. 56	0	20	21	23	25
57	S. N. 57	0	20	22	24	26
58	S. N. 58	0	21	23	25	27
59	S. N. 59	0	21	23	25	28
60	S. N. 60	0	22	24	26	29
61	S. N. 61	0	23	25	27	30
62	S. N. 62	0	23	26	27	31
63	S. N. 63	0	24	27	28	32

64	S. N. 64	0	25	28	29	33
65	S. N. 65	0	26	29	30	34
66	S. N. 66	0	27	30	31	35
67	S. N. 67	0	28	31	32	36
68	S. N. 68	0	28	32	32	37
69	S. N. 69	0	29	33	33	38
70	S. N. 70	0	29	34	33	39
71	S. N. 71	0	30	35	34	40
72	S. N. 72	0	31	36	35	41
73	S. N. 73	0	32	37	36	42
74	S. N. 74	0	33	38	37	43
75	S. N. 75	0	34	38	37	44
76	S. N. 76	0	34	39	38	45
77	S. N. 77	0	35	40	39	46
78	S. N. 78	0	36	41	40	47
79	S. N. 79	0	37	42	41	48
80	S. N. 80	0	38	43	42	49

Table 4.10: Performance analysis of the proposed dynamic neural network, Shannon entropy, Pal and Pal entropy, Tsallis entropy and various static neural network configurations, in terms of training time in sec (s), number of hidden nodes and the testing accuracy in (%), based on SPECT Heart dataset.

Methods	Growth of hidden layer neurons using Susan and Hanmandlu Non-extensive Entropy.	Growth of hidden layer neurons using Shannon Entropy.	Growth of hidden layer neurons using Pal and Pal Entropy.	Growth of hidden layer neurons using Tsallis Entropy.	Static 3 Hidden neurons	Static 7 Hidden neurons	Static 11 Hidden neurons	Static 22 Hidden neurons	Static 34 Hidden neurons	Static 40 Hidden neurons
Total training time (for 80 samples)	0.136 sec	0.287sec	0.187sec	0.186sec	0.127sec	0.231sec	0.167sec	0.142sec	0.142sec	0.207sec
Number of hidden nodes	38	43	42	49	3	7	11	22	34	40
Testing accuracy (for 187 samples)	93.05%	92.52%	92.52%	93.05%	83.43%	89.84%	90.38%	90.91%	91.45%	91.45%

CHAPTER 5
CONCLUSIONS AND FUTURE SCOPE

In this study the dynamic growth of hidden units is proposed that is based on the non-extensive entropy of the network weights that are normalized to probability values. A decrease in entropy of eights is observed as the training progresses and if this criterion is not met the number of hidden units is incrementally increased. As shown in the experimental results section, the dynamic growth of hidden layer neurons using non-extensive entropy is less as compared with other extensive, non-extensive entropies which are considered for experiment. In terms of testing accuracy and execution time required for set of training samples, the Susan and Hanmandlu non-extensive entropy performed better in comparison of other entropies and various static neural network configurations. It is observed that a slow and steady rise of the number of hidden layer neurons for the non-extensive entropy, while Shannon, Pal an Pal and Tsallis entropies result in growth of more hidden layer neurons as in compare with Susan and Hanmandlu non-extensive, taking more training time than the non-extensive entropy. Apart from a reduced MLP architecture with less number of hidden layer neurons, the overall training time required by the proposed method is much less than the equivalent fixed neurons architecture, while its classification performance is much higher than any possible static neural network configuration. This suggest that the dynamic growth of hidden layer as proposed in this study not only gives the optimal number of hidden neurons for a dataset, at the same time it provides a better tuning of associated weights and faster training than any static neural network configuration. The highlight of our approach is that it can grow the hidden layer dynamically while training and the weights are tune better than any configuration of the static neural network leading to high classification performance.

REFERENCES

[1] Mark J. L. Orr, “Introduction to Radial Basis Function Networks”, april 1996.

[2] D. S. Broomhead and D. Lowe, “Multivariate functional interpolation and adaptive networks”, Complex Systems (1988): 321-355.

[3] Ciabattoni, Lucio, et al. "Supervisory control of PV-battery systems by online tuned neural networks." Mechatronics (ICM), 2013 IEEE International Conference on IEEE, 2013.

[4] Yingwei, Lu, Narashiman Sundararajan, and Paramasivan Saratchandran. "Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm." Neural Networks, IEEE Transactions on 9.2 (1998): 308-318.

[5] Huang, Guang-Bin, Paramasivan Saratchandran, and Narasimhan Sundararajan. "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks." Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34.6 (2004): 2284-2292.

[6] Wu, Shiqian, and Meng Joo Er. "Dynamic fuzzy neural networks-a novel approach to function approximation." Systems, Man, and Cybernetics,Part B: Cybernetics, IEEE Transactions on30.2 (2000): 358-364.

[7] Ashish Ghosh, Nikhil R. Pal, “Self-Organization for Object Extraction Using a Multilayer Neural Network and Fuzziness Measures”, IEEE transactions on fuzzy systems, vol. 1, no. 1, (1993): 54-68.

[8] G.-B. Huang, L. Chen, and C.-K. Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes,”IEEE Trans. Neural Netw., vol. 17, no. 4(2006):879–892.

[9] G.-B. Huang and L. Chen, “Convex incremental extreme learning machine,”Neurocomputing, vol. 70(2007):3056–3062.

[10] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, New Jersey, 1999.

[11] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.

[12] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[13] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of International Joint Conference on Neural Networks, IJCNN'04, vol. 2, Budapest, Hungary, 25–29 July 2004, pp. 985–990.

[14] E. Alpaydin, Gal: networks that grow when they learn and shrink when they forget, International Computer Science Institute, TR 91-032, USA, 1991.

[15] Feng, Guorui, et al. "Error minimized extreme learning machine with growth of hidden nodes and incremental learning." Neural Networks, IEEE Transactions on20.8 (2009): 1352-1357.

[16] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine," inProc. IASTED Int. Conf. Comput. Intell., Calgary, AB, Canada, Jul. 4–6(2005): 232–237.

[17] T. Kim and T. Adali, "Approximation by fully complex multilayer perceptrons,"Neural Comput., vol. 15(2003):1641–1666, 2003.

[18] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine,"Neurocomputing, vol. 68(2005):306–314.

[19] Lan, Yuan, Yeng Chai Soh, and Guang-Bin Huang. "Constructive hidden nodes selection of extreme learning machine for regression." Neurocomputing73.16 (2010): 3191-3199.

[20] Kurino, Ryusuke, Masanori Sugisaka, and Katsunari Shibata. "Growing neural network with hidden neurons." Proc. 9th Int. Symp. on Artificial Lifeand Robotics (AROB'04). Vol. 1. 2004.

[21] Yang, Hong, and Jun Ni. "Dynamic neural network modelling for nonlinear, nonstationary machine tool thermally induced error." *International Journal of Machine Tools and Manufacture* 45.4 (2005): 455-465.

[22] Gupta, Madan M., Liang Jin, and Noriyasu Homma. Static and dynamic neural networks: from fundamentals to advanced theory. *Wiley-IEEE Press, 2004.*

[23] Erdogmus, Deniz, and Jose C. Principe. "An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems." *Signal Processing, IEEE Transactions on* 50.7 (2002): 1780-1786.

[24] Ben-David, Arie. "Monotonicity maintenance in information theoretic machine learning algorithms." *Machine Learning* 19.1 (1995): 29-43.

[25] Morejon, Rodney A., and Jose C. Principe. "Advanced search algorithms for information-theoretic learning with kernel-based estimators." *Neural Networks, IEEE Transactions on* 15.4 (2004): 874-884.

[26] Robert J.Schalkoff, Artificial Neural Networks, McGraw-Hill, 2011.

[27] Singh, Rampal, and S. Balasundaram. "Application of extreme learning machine method for time series analysis." International Journal of Intelligent Technology 2.4 (2007): 256-262.

[28] Huynh, Hieu Trung, Jung-Ja Kim, and Yonggwan Won. "Performance comparison of SLFN training algorithms for DNA microarray classification." Software Tools and Algorithms for Biological Systems. Springer New York, 2011. 135-143.

[29] Huang, Guang-Bin, Lei Chen, and Chee-Kheong Siew. "Universal approximation using incremental constructive feedforward networks with random hidden nodes." Neural Networks, IEEE Transactions on 17.4 (2006): 879-892.

[30] Huang, Guang-Bin, Lei Chen, and Chee-Kheong Siew. "Universal approximation using incremental constructive Feed-forward networks with random hidden nodes." Neural Networks, IEEE Transactions on 17.4 (2006): 879-892.

[31] Seba Susan, Madasu Hanmandlu, A Non-Extensive entropy feature and its application to texture classification", Neurocomputing, Volume 120, November 2013, pp. 214–225.

[32] Tsallis, Constantino. "Possible generalization of Boltzmann- Gibbs statistics." Journal of statistical physics 52.1-2 (1988): 479-487.

[33] C.E.Shannon, “A Mathematical theory of communication”, The Bell system Technical Journal, Vol.27, pp.379-423 July,1948.

[34] Seba Susan, Madasu Hanmandlu,” A novel Fuzzy Entropy based on the Non-Extensive Entropy and its application for feature selection”, FUZZ-IEEE 2013.

[35] Susan, Seba, and Madasu Hanmandlu. "Unsupervised detection of nonlinearity in motion using weighted average of non-extensive entropies." Signal, Image and Video Processing (2013): 1-15.

[36] Platt, John. "A resource-allocating network for function interpolation." Neural computation 3.2 (1991): 213-225.
[36] Platt, John. "A resource-allocating network for function interpolation." Neural computation 3.2 (1991): 213-225

[37] Victor Boskovitz and Hugo Guterman. “An Adaptive Neuro-Fuzzy System for Automatic Image Segmentation and Edge Detection” IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 10, NO. 2, APRIL (2002): 247-262

[38] K. Sivakumar and U. B**.** Desai. “Image Restoration Using a Multilayer Perceptron with a Multilevel Sigmoidal Function” IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 41**,** NO. *5***,** MAY (1993): 2018-2022

[39] Amir Hossein Darooneh, Ghassem Naeimi, Ali Mehri and Parvin Sadeghi. “Tsallis Entropy, Escort Probability and the Incomplete Information Theory” Entropy 2010, 12, 2497-2503; doi: 10.3390/e12122497.

[40]. Tsallis, C. “Non-Extensive Statistical Mechanics and Its Applications”; Abe, S., Okamoto Y., Eds.; Springer: Berlin, Germany, 2001.

[41]. Tsallis, C. “Non-extensive Entropy-Interdisciplinary Applications”; Gell-Mann, M., Tsallis, C., Eds.; Oxford University Press: New York, NY, USA, 2004.

[42]. Tsallis, C.; Gell-Mann, M.; Sato, Y. “Special issue on the non-extensive statistical mechanics”. Europhys. News 2005, 36, 186–189.

[43]. Tsallis, C. “Introduction to Non-extensive Statistical Mechanics”; Springer: Berlin, Germany, 2009.

[44] Susan, S., Hanmandlu, M.: “A non-extensive entropy feature and its application to texture classification”. Neurocomput. Elsevier Special Issue Image Feature Detect. Descr. (Article in press) (2012). doi: 10.1016/j.neucom.2012.08.059