Journal of Modern Physics, 2019, 10, 1505-1521
https://www.scirp.org/journal/jmp
ISSN Online: 2153-120X
ISSN Print: 2153-1196

Design of Hanman Entropy Network from Radial Basis Function Network

M. Hanmandlu[1], Rao Shivansh[2], Shantaram Vasikarla[3]

[1]CSE Department MVSR Engg. College, Nadergul, Hyderabad, India
[2]Pennsylvania State University State College, PA, USA
[3]CSE Department California State University Northridge, CA, USA
Email: mhmandlu@ee.iitd.ernet.in

How to cite this paper: Hanmandlu, M., Shivansh, R. and Vasikarla, S. (2019) Design of Hanman Entropy Network from Radial Basis Function Network. *Journal of Modern Physics*, **10**, 1505-1521.
https://doi.org/10.4236/jmp.2019.1013100

Received: August 25, 2019
Accepted: November 10, 2019
Published: November 13, 2019

Abstract

Different learning algorithms have been developed in the literature for training the radial basis function network (RBFN). In this paper, a new neural network named as Hanman Entropy Network (HEN) is developed from RBFN based on the Information set theory that deals with the representation of possibilistic uncertainty in the attribute/property values termed as information source values. The parameters of both HEN and RBFN are learned using a new learning algorithm called JAYA that solves the constrained and unconstrained optimization problems and is bereft of algorithm-specific parameters. The performance of HEN is shown to be superior to that of RBFN on four datasets. The advantage of HEN is that it can use both information source values and their membership values in several ways whereas RBFN uses only the membership function values.

Keywords

RBFN, HEN, Gradient Descent (GD), Pseudo-Inverse, JAYA

1. Introduction

The artificial neural networks (ANNs) that include back propogation (BP) networks [1], radial basis function networks (RBFNs) [2], counter propagation networks [3] to mention a few show their power in data classification, pattern recognition and function approximation. In this paper, we are mainly concerned with incorporating a new learning agorithm, called JAYA into the architecture of RBFN to mitigate the drawbacks of its gradient descent learning.

A radial basis function network (RBFN) [4] [5] is a three-layer feed-forward neural network. Each hidden layer neuron evaluates its kernel function on the incoming input. The network output is simply a weighted sum of the values of the kernel functions in the hidden layer neurons. The value of a kernel function is highest when the input falls on its center and decreases monotonically as it moves away from the center. A Gaussian function is normally used as the kernel function. The training of an RBFN is done by finding the centers and the widths of the kernel functions and the weights connecting the hidden layer neurons to the output layer neurons.

Next, we will foray into the learning domain. Finding the global optimum of a function is the main task of many of the scientific applications. Gradient descent approach is widely used but it suffers from local minima. Another limitation is that it cannot be used in optimization problems that have non-differentiable objective functions. Many modern population based heuristic algorithms focus on finding a near optimum solution to overcome this requirement of differentiability associated with gradient descent learning.

A brief survey of the population based heuristic algorithms will enlighten the readers how much work has been done in the domain of learning. These algorithms can be clubbed into two important groups: evolutionary algorithms (EA) and swarm intelligence (SI) based algorithms. Some of the recognized evolutionary algorithms are: Genetic Algorithm (GA), Evolution Strategy (ES), Evolution Programming (EP), Differential Evolution (DE), Bacterial Foraging Optimization (BFO), Artificial Immune Algorithm (AIA), etc. Some of the well known swarm intelligence based algorithms are: Particle Swarm Optimization (PSO), Shuffled Frog Leaping (SFL), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Fire Fly (FF) algorithm, etc. Besides the evolutionary and swarm intelligence based algorithms, there are some other algorithms that work on the principles of different natural phenomena. Some of them are: Harmony Search (HS) algorithm, Gravitational Search Algorithm (GSA), Biogeography-Based Optimization (BBO), Grenade Explosion Method (GEM), etc. All the evolutionary and swarm intelligence based algorithms are probabilistic algorithms that require common controlling parameters like population size, number of generations, elite size, etc. Besides the common control parameters, different algorithms require their own algorithm-specific control parameters. A recent meta-heuristic learning method called Human Effort for Achieving Goals (HEFAG) by Jyotsana and Hanmandlu contains the comparison of several learning methods in [6]. A new learning algorithm called JAYA is developed in [7] to overcome the need for the algorithm-specific parameters but the need for the common control parameters still exists. This algorithm helps the initial solutions move towards the best solution by avoiding the worst solution.

2. Design of RBFN

For the detailed study on artificial neural networks (ANN) and fuzzy systems and their applications readers may refer to Jang *et al.* [8]. As Multilayer Perceptron (MLP) is a major leap in ANNs and RBFN has arisen out of simplying computational burden involved in MLP; hence it is widely used [9] for the traditional classification problems. A comparison between the traditional neural networks and RBFN is presented in [10].

RBFN deals with attrubute/feature values that are clustered. The attribue values in a cluster are fitted with the radial basis function which is another name for Gaussian function. RBFN fuzzifies the attribute values in a cluster into the membership function values. Each RBFN neuron stores a cluster centre or centroid, which is initially taken to be one of the samples from the training set. When we want to classify a new input, each neuron computes the Euclidean distance between the input and its centroid and computes the membership function using the standard deviation or width of the Gaussian function. The output of the RBFN is a weighted sum of the membership function values as shown in **Figure 1**. In this μ_i denotes the ith membership function (MF) of a neuron. The MF vector is of size k and each value of this vector is multiplied with the output weight and then summed up to get the computed output.

2.1. The Derivation of the Model of RBFN

We will derive an input-output relation underlying the architecture of RBFN in **Figure 1** in which prototype refers to the cluster centre. In this architecture there are two phases. The first phase is fuzzification and second phase is regression. For the fuzzification let us assume a cluster consisting of feature vectors of dimension k. Let ith feature X_{ui} in this vector X_u be fuzzified using the ith membership function μ_i and u stands for uth input vector-output pair. Thus we have k fuzzy sets. Here we have as many neurons as the number of the input feature values. We don't require any equation for this phase. In the regression phase we employ Takagi-Sugeno-Kang fuzzy rule [8] on k-input fuzzy sets and one output as:

If X_{u1} is A_1 and X_{u2} is A_2 and ⋯ X_{uk} is A_k then

$$Y_u = b_0 + b_1 X_{u1} + b_2 X_{u2} + \cdots + b_k X_{uk} \tag{1}$$

where the fuzzy set $A_i = \{X_{ui}, P_{ui}\} \mid u = 1,\cdots,m; i = 1,2,\cdots,k$. Now substituting the fuzzified inputs, *i.e.* $P_{ui} = \mu_i(X_{ui})$, for the inputs we get

$$Y_u = b_0 + b_1 P_{u1} + b_2 P_{u2} + \cdots + b_{uk} \tag{2}$$

This equation is valid if there is one class. We now extend this equation to the multi-class case. We feed the input vector of size k denoted by X_u and the neurons compute the membership function values P_{uj}. Let the number of classses be c. The regression equation that computes the outputs Y_l in multi-class is framed as:

$$Y_l = w_{0l} + w_{1l} P_{u1} + w_{2l} P_{u2} + \cdots + w_{kl} P_{uk};\ l = 1,\cdots,c \tag{3}$$

where we have replaced the weight vector $\{b_i\}$ by the weight matrix $\{w_{il}\}$ to account for multi-class. This is the governing equation for the architecture in **Figure 1**. The calculation of the output weights is deferred to Section III. The case when a class is represented by more than one cluster is now explained.

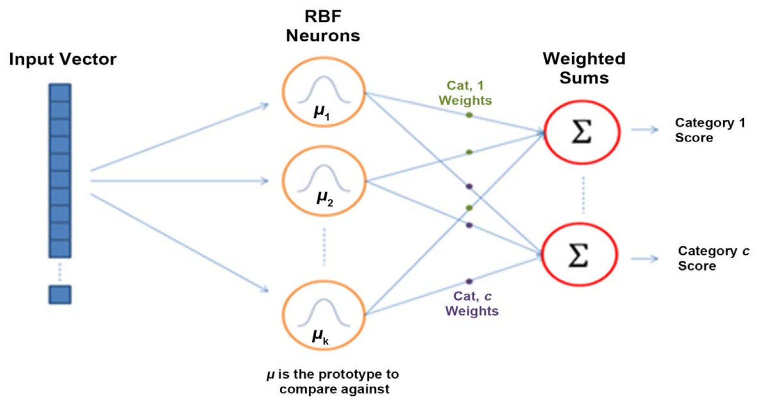

Figure 1. Architecture of RBFN.

2.2. Procedure for Learning of Weights

Consider first the problem of Iris flower recognition. In this we have 4 features. That is each feature vector is 4 dimensional. Assume that these are clustered into say, 3. It means that each cluster contains some number of feature vectors. According to the fuzzy set theory, we can form 4 fuzzy sets in each cluster corresponding to four features. Now each fuzzy set is defined by its attribute values and their membership function values. As we are using clustering, we can obtain mean values as well as scaling factors that are functions of variances involved in MFs of the fuzzy sets resulting from clustering. Our attempt is to focus on learning of weights.

Let us assume that we are feeding each feature vector of a cluster. Then we will have four neurons that convert four feature values of the feature vector into four membership function values. Then these membership values will be summed up. As we have assumed three clusters for one class (each flower type of Iris), this procedure is repeated on all feature vectors of the remaining two clusters. By this, we get three sums which will be multiplied with three weights (*i.e.* forming one weight vector) and the weighted sum is the computed output that represents one class.

The above procedure is repeated for the other three classes and the three weight vectors so obtained correspond to the remaining three flower types. There will also be three weighted sums called the computed outputs. In this paper, we are concerned with one cluster per class for simplicity.

3. Training of RBFN

The training process for RBFN consists of finding the three sets of parameters: the centrods of clusters, scaling parameters for each of the neurons of RBFN, and a set of the output weight vectors between the neurons and the output nodes of RBFN. The approaches for finding the centriods and their variances are discussed next.

3.1. Cluster Centrods

The possible approaches for the selection of clustercentrods are: Random selection of centroids, Clustering based approach and Orthogonal Least Squares (OLS). We have selected the K-means clustering for the computation of centroids of clusters or cluster centres from the training setr. Note that this clustering method partitionsn observations into K number of clusters such that each observation having its value closest to the cluster centre belongs to that cluster.

3.2. Scaling Parameters

Camped with the centoid of each cluster, the variance is computed as the average distance between all points in the cluster and the centrod.

$$\sigma_i^2 = \frac{1}{m}\sum_{u=1}^{m}\left(X_{ui} - C_i\right)^2 \tag{4}$$

Here, C_i is the centroid of ith cluster, m is the number of training samples belonging to this cluster, X_{ui} is the uth training sample in the ith cluster. Next we use σ_i^2 to compute the scaling parameters denoted by $\beta_i = \frac{1}{\sigma_i^2}$.

3.3. Output Weights

In the literature, there are two popular methods for the determination of the output weights: one learning method called gradient descent [11] and another computational method called pseudo inverse [12] [13]. As gradient descent learning has problems of slow convergence to local minima, we embark on a new learning algorithm called JAYA. Prior to using JAYA for learning the parameters of RBFN, we will discuss how the weights can be determined by Pseudo-inverse (PINV) method.

Consider an input vector which is generally a feature vector of some dimension n. When all the feature vectors are clustered, we will have C number of clusters (Note that c denotes the number of classes). In some datasets such as Iris dataset, we can easily separate out all the feature vectors belonging to each class of one flower type. Thus the feature vectors belonging to a class form a cluster. Out of these feature vectors some are selected for training and the rest for testing.

Let $\{X_u, Z_u\}; u = 1,\cdots,m$ be the set of feature vectors with each feature vector having the size of n, *i.e.* $X_u \in R^k$ with target, $Z_{ul} \in R^c$, and $P_{uj} = \mu_i\left(X_{uj}\right)$ be the membership function of the jth basis radial function μ_j with the uth feature vector. X_{uj} is the jth component of the feature vector X_u and Z_{ul} is the lth target output. Note that this formulation is meant for one cluster per one class. After the fuzzification of X_{uj} into P_{uj}, we can form a matrix P by taking $u = 1,\cdots,m$; $j = 1,2,\cdots,k$. The matrix Q is written as

P_{11}	P_{12}	···	P_{1k}
P_{21}	P_{22}	···	P_{2k}
	..	···	..
P_{m1}	P_{m2}	···	P_{mk}

As we have $W_l = [w_{1l}, w_{2l}, \cdots, w_{kl}]$; $Z_l; l = 1, 2, \cdots, c$; $Z = [Z_1, Z_2, \cdots, Z_c]$. Let us denote $Q = [P_1, P_2, \cdots, P_m]$ with $P_u = [P_{u1}, P_{u2}, \cdots, P_{uk}]$ and $W = [W_1, W_2, \cdots, W_k]$. The objective function to be minimized is given by:

$$f(W, Q, Z) = \|W * Q - Z\|^2 \tag{5}$$

where $Y = W * Q$ with $Y = [Y_1, Y_2, \cdots, Y_c]$ as per Equation (3). The solution to the above equation lies in the assumption that be $Y = W * Q = Z$ which leads to $W = Q^+ Z$ where Q^+ denotes the pseudo inverse matrix of Q, defined as follows:

$$Q^+ = \lim_{\alpha \to 0} \left(Q^T Q + \gamma I_n\right)^{-1} Q^T \tag{6}$$

where I_k is the k-dimensional unity matrix and γ is a small positive constant. The pseudo inverse $Q^+ = \left(Q^T Q\right)^{-1} Q^T$ exists if $(Q^T Q)$ is nonsingular. After calculating the weights at the output layer, all the parameters of RBFN with its 3-layered architecture in Figure 1 can be determined.

4. Learning of the Output Weights by JAYA

We will now discuss JAYA algorithm to be used for learning the parameters of RBFN.

Description of the JAYA Algorithm

It is a simple and powerful learning method for solving the constrained and unconstrained optimization problems. As mentioned above JAYA algorithm is the offshoot of Teacher-Learner Based Optimization (TLBO) algorithm proposed in [14] [15]. This needs only the common controlling parameters like population size and number of iterations. The guidelines for fixing these parameters can be seen in [15]. Here we have fixed the population size as 10 and the number of iterations as 3000.

Let $f(W, Q, Z)$ be the objective function to be minimized. Let the best candidate be the one associated with the least value of the function (*i.e.* $f_{best}(W, Q, Z)$) and the worst candidate is the one with the highest value of the function (*i.e.* $f_{worst}(W, Q, Z)$) in all the candidate solutions. We choose B to stand for the weights W when the cluster centres and scale parameters are found separately. In case we use to learn all the parameters, B includes the cluster centres, scaling parameters and the output weights, *i.e.* $B = [C, \beta, W]$.

At any run of the algorithm, assume that there are '*j*' design variables and '*k*' candidate solutions and '*i*' iterations. So to fit B into the JAYA algorithm, it is denoted by $B_{j,k,i}$ which is the value of the *j*th variable of the *k*th candidate during the *i*th iteration. $B_{j,k,i}$ is updated to $B'_{j,k,i}$ during the iteration as,

$$B'_{j,k,i} = B_{j,k,i} + r_{1,j,i}\left(B_{j,best,i} - \left|B_{j,k,i}\right|\right) - r_{2,j,i}\left(B_{j,worst,i} - \left|B_{j,k,i}\right|\right) \tag{7}$$

where $B_{j,best,i}$ is the value of the *j*th variable for the best candidate, $B_{j,worst,i}$ is the value of the *j*th variable for the worst candidate at *i*th iteration and $r_{1,j,i}$ and $r_{2,j,i}$ are the two random numbers in the range 0 to 1 for the *j*th variable at the *i*th iteration. The term $r_{1,j,i}\left(B_{j,best,i} - \left|B_{j,k,i}\right|\right)$ indicates the tendency to move closer to the best solution whereas the term $r_{2,j,i}\left(B_{j,worst,i}-\left|B_{j,k,i}\right|\right)$ indicates the tendency to avoid the worst solution. $B'_{j,k,i}$ is accepted if its function value is better than that of $B_{j,k,i}$. All the accepted function values at the end of iteration are retained and these values become the input to the next iteration. The flowchart of JAYA algorithm is shown in Figure 2. Unlike TLBO algorithm that has two phases (*i.e.* teacher and learner), JAYA algorithm has only one phase and it is comparatively simpler to apply. Rao *et al.* [16] have used TLBO algorithm in the machining processes. A tea-category identification (TCI) system is developed in [17] and it uses a combination of JAYA algorithm and fractional Fourier entropy on three images captured by a CCD camera. In two studies involving heat transfer and pressure drop, *i.e.* thermal resistance and pumping power, two objective functions are used to ascertain the performance of the micro-channel heat sink. Multi-objective optimization aspects of plasma arc machining (PAM), electro-discharge machining (EDM), and micro electro-discharge machining (μ-EDM) processes are investigated in [18]. These processes are optimized while solving the multi-objective optimization problems of machining processes using MO-JAYA algorithm.

There are three learning parameters, viz., the cluster centers C_i, the scaling parameters (β_i) and the output weights W between the hidden and output layers. The learning of these parameters is depicted in Figure 3. The first parameter is found using K-means clustering algorithm.

We make use of JAYA algorithm for learning the second parameter, β_i. The weights are learned by optimizing the objective function using JAYA algorithm. The RBFN model so obtained can then be used for both classification and function approximation.

5. Design of Hanman Entropy Network

As the RBFN is not geared up to take care of the uncertainty in the input which may be an attribute or property value, we will make use of the Information set

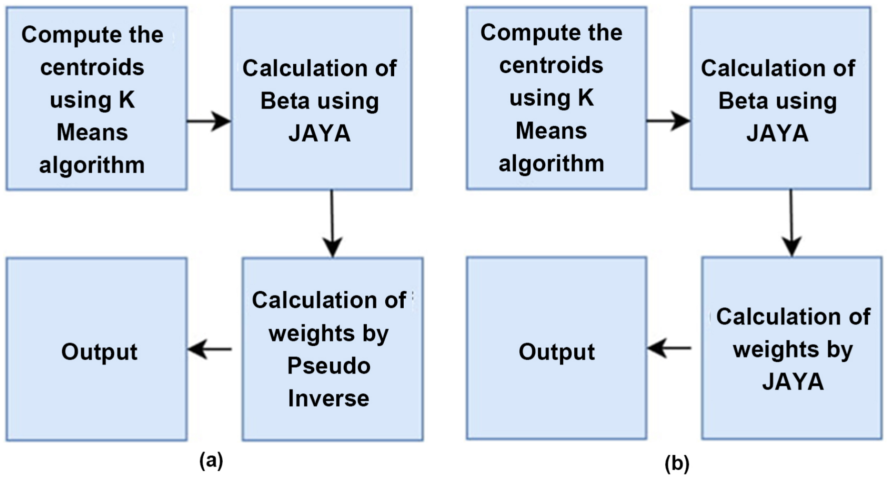
Compute the centroids using K Means algorithm
Calculation of Beta using JAYA
Calculation of weights by Pseudo Inverse
Output
(a)
Compute the centroids using K Means algorithm
Calculation of Beta using JAYA
Calculation of weights by JAYA
Output
(b)

Figure 2. The learning processes in RBFN. (a) Weights using pseudo inverse; (b) Weights using JAYA Algorithm.

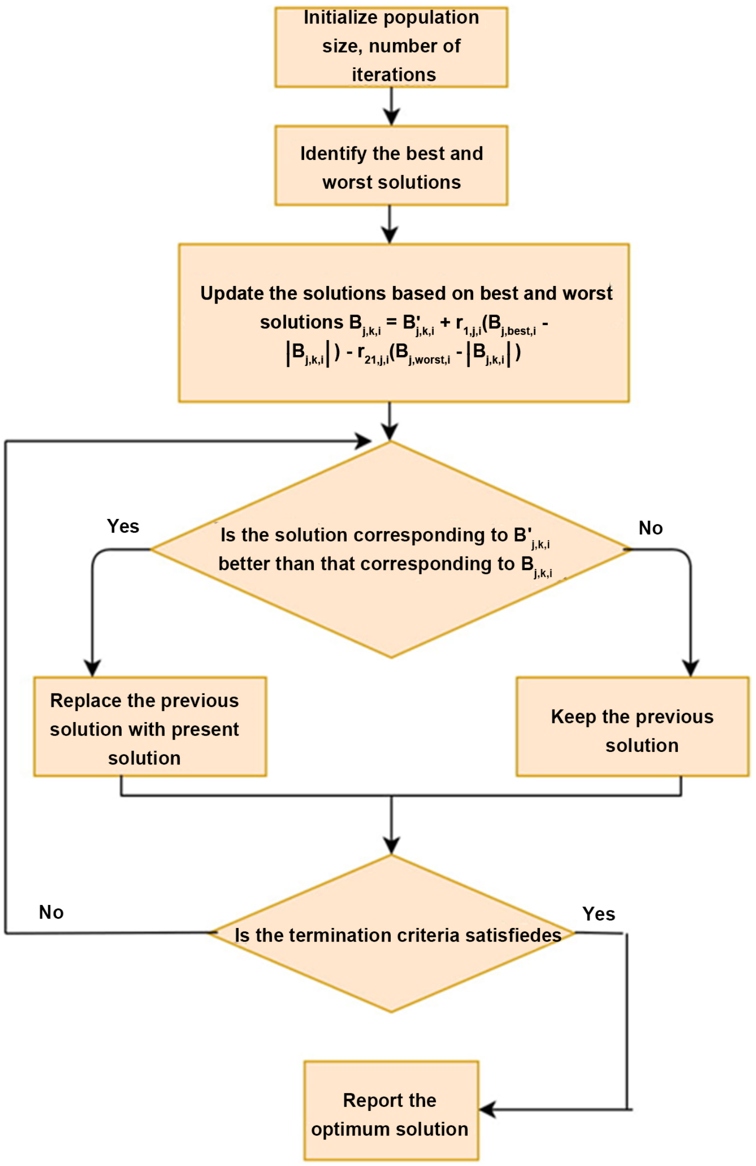
Initialize population size, number of iterations
Identify the best and worst solutions
Update the solutions based on best and worst solutions $B_{j,k,i} = B'_{j,k,i} + r_{1,j,i}(B_{j,best,i} - |B_{j,k,i}|) - r_{21,j,i}(B_{j,worst,i} - |B_{j,k,i}|)$
Is the solution corresponding to $B'_{j,k,i}$ better than that corresponding to $B_{j,k,i}$
Yes
No
Replace the previous solution with present solution
Keep the previous solution
Is the termination criteria satisfiedes
No
Yes
Report the optimum solution

Figure 3. Flowchart of JAYA algorithm.

theory expounded by M. Hanmandlu and his co-workers [19] [20] [21]. In this theory each input variable is termed as the information source value. It centres around the concept of information set that expands the scope of a fuzzy set in which each element is a pair comprising a property (Information source) value and its degree of association with the set/concept called the Membership function value. In most of the applications involving fuzzy theory only the membership function is at the centres tage of operations. This limitation is sought to be removed by proposing the concept of information set. In real life contexts, we operate on information values. The information sources received by our senses are perceived by the mind as information values. That is the reason why we fail to perceive sound even when it strikes our ears. Like fuzzy variables, information values are also natural variables.

5.1. Definition of Information Set and Generation of Features

Consider a fuzzy set constructed from the feature values $\{X_{uj}\}$ termed as the Information source values and their membership function values which we take as the Gaussian function values $\{P_{uj}\}$. If the information source values do not fit the Gaussian function, we can choose any other mathematical function to describe their distribution. Thus each pair (X_{uj}, P_{uj}) consisting of information source value and its membership value is an element of a fuzzy set. P_{uj} gives the degree of association of X_{uj} and the sum of P_{uj} values doesn't provide the uncertainty associated with the fuzzy set. In the fuzzy domain, only P_{uj} is used in all applications of fuzzy logic thus ignoring X_{uj} altogether. This limitation is eliminated by applying the information theoretic entropy function called the Hanman-Anirban function [22] to the fuzzy set. This function combines the pair of values X_{uj} and P_{uj} into a product termed as the information value given by

$$H_{uj} = X_{uj} P_{uj} \tag{8}$$

The above relation owes its derivation to the non-normalized Hanman-Anirban entropy function expressed as

$$H = \sum_{j=1}^{n} X_{uj} \mathrm{e}^{-\left(aX_{uj}^3 + bX_{uj}^2 + cX_{uj} + d\right)} \tag{9}$$

where a, b, c and d are the real-valued parameters. In this equation normalization by n is not needed as the number of attributes is very small (less than 10 in the databases used) but needed if the value of H exceeds more than 1.

With the choice of parameters: $a = 0$, $b = \frac{1}{2\sigma^2}$, $c = -\frac{2\bar{X}}{2\sigma^2}$ and $d = \frac{\bar{X}^2}{2\sigma^2}$ where $\bar{X}$ is the mean value and σ^2 is the variance of the information source values X_{uj}, the exponential gain function is converted into the Gaussian function P_u. As a result, Equation (9) is modified to

$$H = \sum_{j=1}^{n} X_{uj} P_{uj} \tag{10}$$

The set of information values constitutes the information set denoted by $\mathcal{H} = \{H_{uj}\} = \{X_{uj} P_{uj}\}$ whereas the corresponding fuzzy set is simply $\{X_{uj}, P_{uj}\}$. Consider another entropy function called Mamta-Hanman entropy function [20] which is a generalized form of Hanman-Anirban entropy function, expressed as:

$$H_{MH} = \sum X_{uj}^{\alpha} \mathrm{e}^{-\left(cX_{uj}^{\gamma} + d\right)^{\rho}} \tag{11}$$

Substituting $c = \frac{1}{\sigma}$ and $d = -\frac{\bar{X}}{\sigma}$ in (11) modifies H_{MH} to the following:

$$H_{MH} = \sum_{j=1}^{n} X_{uj}^{\alpha} G_{uj} \tag{12}$$

where G_{uj} is the generalized Gaussian function given by

$$G_{uj} = \mathrm{e}^{-\left\{\frac{X_{uj}^{\gamma} - \bar{X}}{\sigma}\right\}^{\rho}}$$

This function takes different shapes as we vary the value of ρ from 1 to 5.

Assuming each information value as a unit of information, we can derive several modified information sets. For instance application of sigmoid function on $X_{uj}^{\alpha}G_{uj}$ leads to

$$S = \sum_{j=1}^{n} \frac{1}{1+\mathrm{e}^{-X_{uj}^{\alpha}G_{uj}}} \tag{13}$$

In Equations (9)-(13) the information values are the ones inside the summation sign. Thus a family of information forms can be deduced from both Hanman-Anirban and Mamta-Hanman entropy functions for dealing with different problems. For the derivation of different forms of H and H_{MH} the readers may refer to [19] and [21] respectively.

5.2. The Hanman Transform and its Link to Intuitionistic Set

This is a higher form of information set. To derive this transform, we have to consider the adaptive form of Hanman-Anirban entropy function in which the parameters in the exponential gain function are taken to be varaibles. Assuming $a=b=d=0$ and $c=P_{uj}$ in (9) we obtain the Hanman Transform [21]:

$$H_T = \sum_{j=1}^{n} X_{uj}\mathrm{e}^{-X_{uj}P_{uj}} = \sum_{j=1}^{n} X_{uj}\mathrm{e}^{-H_{uj}} \tag{14}$$

Note that the exponentional gain function in (14) is a function of the information value. This transform acts as an evaulator of information values based on the information values obtained on them. The higher form of information set $\left\{X_{uj}\mathrm{e}^{-H_{uj}}\right\}$ is recursive because r.h.s of (14) can be rewritten as

$$H_{uj}(new) = X_{uj}\mathrm{e}^{-H_{uj}(old)} \tag{15}$$

An interesting result termed as Shannon transform emerges from Hanman transform by changing the substitution such that $d=-1$ instead of 0 in (9) and then simplifying the resulting exponential function as follows:

$$H_{Sh} = \sum_{j=1}^{n} X_{uj}\mathrm{e}^{1-H_{uj}} = \sum_{j=1}^{n} X_{uj}\log H_{uj} \tag{16}$$

Let us consider the adaptive Hanman-Anirban entropy function involving the membership functions alone. Then we have

$$H = \sum_{j=1}^{n} P_{uj}\mathrm{e}^{-\left(a(.)P_{uj}^{3}+b(.)P_{uj}^{2}+c(.)P_{uj}+d(.)\right)} \tag{17}$$

This gives the uncertainty in the membership function values. This is useful when a mathematical function describing the information source values is not appropriate thus leading to error in the fuzzy modeling. Now with a particular substitution of values for $a(.)=b(.)=d(.)=0$ and $c(.)=X_{uj}$. Equation (14) takes the form

$$H_T = \sum_{j=1}^{n} P_{uj}\mathrm{e}^{-H_{uj}} \tag{18}$$

This equation paves the way for the recursive membership function value. The r.h.s. of (18) without summation can be written as:

$$P_{uj}(new) = P_{uj}(old)\mathrm{e}^{-H_{uj}(old)} \tag{19}$$

On the lines of derivation of (19), we can have another derivation from (11) as follows:

$$P_{uj}(new) = P_{uj}^{\alpha}(old)\mathrm{e}^{-H_{uj}(old)} \tag{20}$$

This is a useful relation becuase it can be used to make RBFN adaptive by changing the membership function. At this juncture we can make an interesting connection between the modified membership function in Equation (19) and hesitancy function in the Intuitionistic fuzzy set [23]. The hesitancey function is defined as follows:

$$h_{uj} = 1 - P_{uj} - P_{ucj} \tag{21}$$

where $P_{ucj} = 1 - P_{uj}$, the complemetary of P_{uj}. The hesitancy function reflects the uncertainty in the modeling of P_{uj} and P_{ucj}. As Equation (19) bestows the way to evaluate P_{uj}, we can use the new values of P_{uj} and P_{ucj} in determining the updated value of h_{uj} as follows:

$$h_{uj}(new) = 1 - P_{uj}(new) - P_{ucj}(new) \tag{22}$$

where $P_{ucj}(new) = P_{ucj}(old)\mathrm{e}^{-H_{ucj}(old)}$ and $H_{ucj} = X_{uj}P_{ucj}$. We can use this hesitancy function for the design of a new network in future.

5.3. Properties of Information Set

We will now present a few useful properties of Information set.

1) In the information set, the product of the complementary membership function value and the information source value gives the complementary information value.

2) Information values are natural variables just as signals received by biological neuron from visual cortex after modification by synapse.

3) The information values can be modified by applying various functions to provide effective features.

4) Higher form of information values like Hanman Transform provides a better representation of the information source values.

5) The fuzzy rules can be easily aggregated using the information set concept.

5.4. The Architecture and Model of Hanman Entropy Network

We will now discuss the architecture of HEN in **Figure 4**.

The architecture of HEN is the same as that of RBFN but for the function φ_i, which assumes the specified form of an entropy function of the input. In HEN each n-input vector needs to be categorized into any one of "*c*" classes. The *i*th function denoted by φ_i converts all the values of the input vector into the entropy function values. This will be clear if we consider the Takagi-Sugeno-Kang fuzzy rule for multi-class case:

If X_{u1} is A_1 and X_{u2} is A_2 and $\cdots$ X_{un} is A_n then

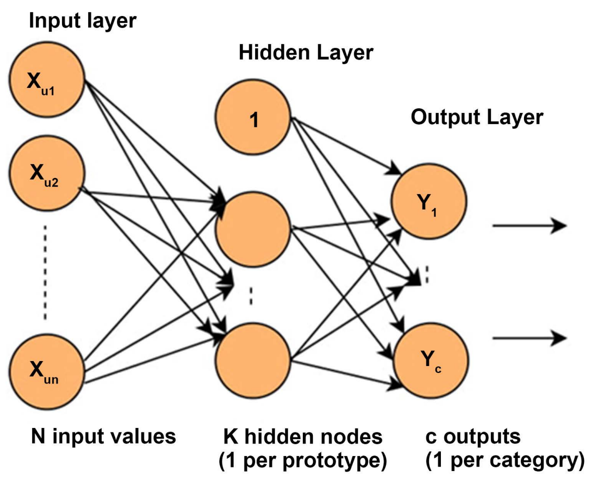

Figure 4. The Architecture of HEN.

$$Y_v = b_{v0} + b_{v1}X_{u1} + b_{v2}X_{u2} + \cdots + b_{vn}X_{un} \tag{23}$$

As mentioned above that at the fuzzification phase we replace b_{vj} with P_{vj} and set P_{v0} to 0 in (23) to get the neuron output as:

$$\varphi_v = P_{v1}X_{u1} + P_{v2}X_{u2} + \cdots + P_{vn}X_{un}; v = 1, \cdots \tag{24}$$

On substituting H_{uj} for the information values in (24) we get:

$$\varphi_v = H_{v1} + H_{v2} + \cdots + H_{vn}; v = 1, 2, \cdots, k \tag{25}$$

Thus the fuzzification phase in HEN is different from that of RBFN. In HEN the input feature vectors of size n are clustered into k clusters but in RBFN there is a single cluster for each feature of a feature vector. Each neuron in RBFN of Figure 1 has only one radial basis function whereas each neuron in HEN has k radial basis functions in Figure 4. So in Equation (25) k sums φ_1 to φ_k will be multiplied with the corresponding weights w_{11} to w_{kl} to yield the lth output Y_l in the regression phase as follows:

$$Y_l = w_{0l} + w_{1l}\cdot\varphi_1 + w_{2l}\cdot\varphi_2 + \cdots + w_{kl}\cdot\varphi_k; l = 1, \cdots, c \tag{26}$$

Here $\varphi_0 = 1$ and $\varphi = [1, \varphi_1, \varphi_2, \cdots, \varphi_k]$; $W_l = [w_{0l}, w_{1l}, \cdots, w_{kl}]$. This is the governing equation of Hanman Entropy Network in Figure 4. The objective function to be optimized by the JAYA + HEN combination is different. In view of this, the objective function becomes:

$$f(\varphi, W_l, Z_l) = \left\| W_l * \varphi - Z_l \right\|^2 \tag{27}$$

Instead of μ_j in RBFN, we will now use φ_j in HEN. In the general case φ_j can be taken as any relation linking the information source values to their membership function values. Thus, we can assume the following relations for this function:

$$\varphi_j = \sum_u \left\{ X_{uj}^{\alpha} G_{uj} \right\}_{\text{MH}} \text{ or } \sum_u \left\{ \frac{1}{1 + \mathrm{e}^{-X_{uj}^{\alpha} G_{uj}}} \right\}_{\text{S}} \text{ or } \sum_u \left\{ X_{uj} \mathrm{e}^{-H_{uj}} \right\}_{\text{T}} \text{ or } \sum_u \left\{ X_{uj} \log H_{uj} \right\}_{\text{Sh}} \tag{28}$$

where subscripts MH, S, T and Sh indicate Mamta-Hanman, Sigmoid, Hanman transform and Shannon transform respectively. Note that RBFN network simply responds to the pattern in the input vector but the Hanman entropy network responds not only to the pattern but also to the uncertainty associated with it.

6. Results of Case Studies

The experimentation is conducted on four datasets: IRIS, Wine, and Waveform from UCI repository and Signature dataset [24] in two phases. In the first phase, we have entirely dealt with the performance analysis of RBFN and in the second phase only with that of Hanman Entropy Network (HEN). We have split up our computations into two cases. In Case-1 which is applicable only to RBFN, the learning/computation of the output weights is delinked from the computation of centriods and scaling parameters. Next, we have employed two learning methods such as Genetic algorithm (GA) and Gradient Descent (GD) and one computational method called Pseudo inverse (PINV) for the weights and K-means clustering for the centriods and scaling parameters.

There is another combination, JAYA + PINV + RBFN wherein JAYA is used for learning scaling parameters and PINV is used for computing the output weights. Of course, the centroids are found by K-means clustering.

The notations GA + RBFN, GD + RBFN, PINV + RBFN and JAYA +P INV refer to the learning of weights of RBFN by Genetic Algorithm, Gradient descent, Pseudo-Inverse, JAYA + PINV combination respectively. The results of classification accuracy with these four methods GA, GD, PINV and JAYA + PINV along with RBFN are given in **Table 1**. The last combination gives the best results. **Table 2** gives the comparison of JAYA + PINV + RBFN with JAYA + RBFN where the latter shows the best results.

A brief exposition on how to form fuzzy sets from which information sets are formed is the need of the hour. Assuming n feature types of an object, say, signature we form n fuzzy sets by collecting all the feature values of each feature type and fitting the radial basis function with the help of centroid (mean value of feature values) and scaling parameter (variance). Then conversion of this fuzzy set to information set is a simple matter.

The dataset-wise discussion of results follows the next.

IRIS dataset: This dataset consists of three classes with 50 samples for each class. There are four attributes for each sample. These are: sepal length, sepal

Table 1. A comparison of the Classification accuracy of RBFN using several learning methods on different datasets.

Dataset	GA + RBFN	GD + RBFN	PINV+RBFN	JAYA + PINV + RBFN
IRIS	96.3%	96.3%	96.3%	**96.7%**
WINE	86.88%	84.8%	82.6%	**92.13%**
WAVE FORM	87%	87.4%	87.5%	**85.8%**

Table 2. A comparison of the Classification accuracy of RBFN using JAYA for the learning of all parameters.

Dataset	JAYA + PINV + RBFN	JAYA + RBFN
IRIS	96.7%	**98%**
WINE	92.13%	**92.6%**
WAVEFORM	85.8%	**87.5%**

width, petal length and petal width all in cms. The classes are: Setosa, Versicolour and Virginica. It may be noted that GA, GD and PINV learning methods yield the classification accuracy of 96.3% when used for learning the weights of RBFN by classifying 144 instances out of 150. However, the combination symbolized by JAYA + PINV + RBFN gives the accuracy of 96.7% which uses 1) JAYA to learn the scaling parameters and 2) PINV to learn the weights of RBFN. This is the best result among all the results obtained on this dataset by the methods compared.

WINE dataset: The dataset consists of three classes with class 1, class 2, class 3 having 59, 71, 48 samples repectively. Each sample has 13 attribultes that include Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline. The classification accuracies of 84.8% and 82.6% are achieved with GD and PINV respectively when these are used for the determination of the output weights of RBFN whereas GA + RBFN combination gives an accuracy of 86.88%. However, the best accuracy of 92.13% is achieved with JAYA-RBFN combination.

WAVEFORM dataset: The dataset consists of 5000 samples with each sample comprising 22 attributes. Each class is generated from a combination of 2 out of 3 "base" waves and each instance is generated by adding noise (mean 0, variance 1) to every attribute. RBFN classifies 4350 instances correctly out of 5000 instances with an accuracy of 87% with GA. The weights of RBFN computed using GD and PINV yield the best accuracies of 87.4% and 87.5% respectively but with JAYA + RBFN combimation the accuracy comes down to 85.8% in Table 2.

As can be seen from the results, the efficiency of the classification task increases when we use JAYA algorithm even for learning the weights of the network in comparison to learning the scaling parameters of the membership function. When the concept of information set is incorporated into our approach, the output is computed as Weights*Information values, *i.e.* $W_l * \varphi$. If the parameter vector, B also includes the centrods and the scaling parameters in addition to the weights then these parameters modify P_u indirectly. Then we will write $\varphi' = \varphi(C, \beta)$. Accordingly the objective function is modified as:

$$f(\varphi', W_l, Z_l) = \left\| W_l * \varphi' - Z_l \right\|^2 \tag{29}$$

For, RBFN the above with $P_i' = P_i(C, \beta)$ is written as

Table 3. A comparison of Verification accuracy on Signature dataset.

Dataset	JAYA + PINV + RBFN	JAYA + RBFN	HEN
SVC2004	90.7%	99.6%	99.8%

Table 4. Comparison of Classification accuracy.

Dataset	JAYA + RBFN	JAYA + HEN	MLP
IRIS	98%	**98.7%**	**95.3%**
WINE	92.6%	**93.2 %**	**81.4%**
WAVE FORM	87.5%	**88.6%**	**87.1%**

$$f\left(P_i', W_l, Z_l\right) = \left\| W_l * P_i' - Z_l \right\|^2 \tag{30}$$

Applying JAYA on (29) and (30) learns B.

SIGNATURE dataset: This dataset (SVC2004) in [24] has been used for a competition and it consists of 20 skilled forgeries and 20 genuine signatures of 40 users. Each signature in the dataset is represented as a sequence of points containing X and Y co-ordinates, time stamp and pen status (pen up or down) along with the additional information like azimuth, altitude and pressure. The text file contains a sequence of 7-dimensional measurements (feature types) for each signature. Our previous work on signature verification using Information set features on this dataset shows the effectiveness of these features [25]. We have used JAYA for learning both scaling parameters and the output weights of Hanman Entropy network (HEN) just as in JAYA + RBFN combination. The results of classification accuracy with JAYA + HEN are slightly better than those of JAYA + RBFN. But with JAYA + PINV + RBFN combination the results are very poor as shown in **Table 3**. The power of Hanman Entropy network can only be realized when the dataset is very large.

On conducting tests on three datasets as shown in **Table 4**, we find that the performance of JAYA + RBFN combination is somewhat inferior to that of JAYA + HEN combination on three datasets (Iris, Wine and Waveform) but the performance of Multi-layer perceptron (MLP) network is the worst. The use of high level information set features may help improve the performance of JAYA + HEN.

7. Conclusions

In this paper not only the performance of Radial Basis Function Network (RBFN) is improved by learning its parameters with a new evolutionary method called JAYA but also the design of Hanman Entropy network is given based on the Hanman-Anirban entropy function. Of all the combinations of RBFN with GA, GD, PNV, MLP and JAYA, JAYA + RBFN gives the best results. The proposed Hanman Entropy network (HEN) along with JAYA outperforms this combination on all the datasets considered in this paper.

As HEN is based on information set theory that caters to uncertainty representation; there is so much flexibility in the choice of information forms. This advantage is missing in RBFN where only the membership function values rule the roost. The only silver lining with RBFN is that we can use Type-2 fuzzy sets where the membership function values can be varied by changing the variance parameter of Gaussian membership function.

The present work opens up different directions to change the information at the hidden neurons of HEN.

Acknowledgements

The authors acknowledge the comments of anonymous reviewers, which helped improve the paper.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) *Nature*, **323**, 533-536. https://doi.org/10.1038/323533a0

[2] Moody, J. and Darken, C.J. (1989) *Neural Computation*, **1**, 281-294. https://doi.org/10.1162/neco.1989.1.2.281

[3] Hecht-Nielsen, R. (1987) *Applied Optics*, **26**, 4979-4984. https://doi.org/10.1364/AO.26.004979

[4] Bullinaria, J.A. (2015) Radial Basis Function Networks: Algorithms, Neural Computation: Lecture 14. http://www.cs.bham.ac.uk/~jxb/INC/l14.pdf

[5] Poggio, T. and Girosi, F. (1990) *Proceedings of IEEE*, **78**, 1481-1497. https://doi.org/10.1109/5.58326

[6] Grover, J. and Hanmandlu, M. (2018) *Applied Intelligence*, **48**, 3394-3410. https://doi.org/10.1007/s10489-018-1154-x

[7] Rao, R.V. (2016) *International Journal of Industrial Engineering Computations*, **7**, 19-34. https://doi.org/10.5267/j.ijiec.2015.8.004

[8] Jang, J.S.R., Sun, C.T. and Mizutani, E. (2009) Neuro-Fuzzy and Soft-Computing: A Computational Approach to Learning and Machine Intelligence. PHI Learning Private Limited, New Delhi.

[9] Orr, M.J.L. (1995) *Neural Computation*, **7**, 606-623. https://doi.org/10.1162/neco.1995.7.3.606

[10] Xie, T.T., Yu, H. and Wilamowski, B. (2011) Comparison between Traditional Neural Networks and Radial Basis Function Networks. Industrial Electronics (ISIE). *IEEE International Symposium*, Gdansk, 27-30 June 2011, 1194-1199.

[11] Parappa, S.N. and Pratap Singh, M.P. (2013) *International Journal of Advancements in Research & Technology*, **2**, 112-125.

[12] Pierre, C. (2008) *Neural Information Processing—Letters and Reviews*, **8**, 25-29.

[13] Schwenker, F., Kestler, H.A. and Palm, G. (2001) *Neural Networks*, **14**, 439-458.

https://doi.org/10.1016/S0893-6080(01)00027-2

[14] Rao, R.V. and Patel, V. (2013) *Scientia Iranica*, **20**, 710-720. https://doi.org/10.1016/j.scient.2012.12.005

[15] Rao, R.V., Savsani, V.J. and Vakharia, D.P. (2011) *Computer-Aided Design*, **43**, 303-315. https://doi.org/10.1016/j.cad.2010.12.015

[16] Rao, R.V. and Kalyankar, V.D. (2012) *Journal of Materials and Manufacturing Processes*, **27**, 978-985. https://doi.org/10.1080/10426914.2011.602792

[17] Zhang, Y., Yang, X., Cattani, C., Rao, R.V. and Wang, S. (2016) *Entropy*, **18**, 77. https://doi.org/10.3390/e18030077

[18] Rao, R.V., Rai, D.P., Ramkumar, J. and Balic, J. (2016) *Advances in Production Engineering & Management*, **11**, 271-286. https://doi.org/10.14743/apem2016.4.226

[19] Hanmandlu, M. (2013) *Expert Systems with Applications*, **40**, 6478-6490. https://doi.org/10.1016/j.eswa.2013.05.020

[20] Hanmandlu, M. (2014) *Engineering Applications of Artificial Intelligence*, **36**, 269-286. https://doi.org/10.1016/j.engappai.2014.06.028

[21] Sayeed, F. and Hanmandlu, M. (2017) *Knowledge and Information Systems*, **52**, 485-507.

[22] Hanmandlu, M. and Das, A. (2011) *Defence Science Journal*, **61**, 415-430. https://doi.org/10.14429/dsj.61.1177

[23] Xu, Z. (2007) *IEEE Transactions on Fuzzy Systems*, **15**, 1179-1187. https://doi.org/10.1109/TFUZZ.2006.890678

[24] Chang, W. and Shin, J. (2008) DPW Approach for Random Forgery Problem in Online Handwritten Signature Verification. 4*th International Conference on Networked Computing and Advanced Information Management*, Gyeongju, 2-4 September 2008, 347-352. https://doi.org/10.1109/NCM.2008.118

[25] Hanmandlu, M., Sayeed, F. and Shantaram, V. (2015) Online Signature Verification Using the Entropy Function. 2015 *IEEE AIPR Workshop*, Washington DC, 13-15 October 2015, 1-7. https://doi.org/10.1109/AIPR.2015.7444522