

Chapter 2

Classification Using Fuzzy Rough Granular Neural Networks

2.1 Introduction

In Chap. 1, a comprehensive description of granular computing, rough fuzzy computing and different classification, clustering and feature selection methods is provided. This chapter deals with pattern classification in the framework of granular neural networks. As mentioned in Chap. 1, the problem of classification is basically one of partitioning the feature space into regions, one region for each category of input. A finite and usually a small number of samples, called a *training set*, provides partial information for optimal design of classifying system, in terms of the values of the parameters, for different tasks of pattern recognition. Different types of classifiers are briefly described in Sect. 1.5.3 of Chap. 1.

As mentioned there, integration of various components of soft computing like fuzzy sets, rough sets, fuzzy rough sets and artificial neural networks provides a new way in which the fuzzy neural networks and rough fuzzy neural networks for classification are developed in pattern recognition framework. These methods can handle uncertainties arising from overlapping classes. In doing so, fuzzy sets are associated with multilayer perceptron (MLP) to design neuro-fuzzy inference model [6]. A fuzzy MLP for classification is designed in [15]. Here fuzzy sets are used to define input vectors of MLP in terms of 3-dimensional (3D) linguistic terms corresponding to *low*, *medium* and *high*.

In this chapter we first describe the salient features of neuro-fuzzy inference model and fuzzy MLP. The networks subsequently developed based on these models are then discussed. These are followed by a rough fuzzy neural network, by integrating rough set theory with fuzzy MLP, along with its characteristic features. Incorporation of rough reducts, as obtained from the training data, as initial network parameters is seen to enhance the classification performance while reducing training time significantly. Finally we describe in detail some of the recent granular neural networks, namely, FRGNN1 [4] and FRGNN2 [5] with experimental results.

2.2 Adaptive-Network-Based Fuzzy Inference System

The adaptive-network-based fuzzy inference system (ANFIS), as described in [6], has two inputs x and y and one output z . It is assumed that the network consists of two fuzzy if-then rules of Takagi and Sugenos type, namely,

Rule 1: If x is A_1 and y is B_1 , then $f_1 = p_1x + q_1y + r_1$,

Rule 2: If x is A_2 and y is B_2 , then $f_2 = p_2x + q_2y + r_2$.

The architecture of ANFIS is available in [6]. The node functions in the layers are described as follows:

Layer 1: Every node i in this layer is a square node with a node function

$$O_i^1 = \mu_{A_i}(x) \quad (2.1)$$

where x is the input to node i , and A is the linguistic label like, small and large, associated with this node function. O_i^1 is the output of the membership function of A_i and it specifies the degree to which the given x satisfies the quantifier A_i . Here bell shaped membership function with maximum 1 and minimum 0 is used.

Layer 2: Every node in this layer is a circle node labeled with $\prod$ which multiplies the incoming signals (input values) and sends the product as output signal. For instance,

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1 \text{ and } 2. \quad (2.2)$$

Each node output represents the firing strength of a rule. A T-norm operator, that performs generalized AND, can be used as the node function in this layer.

Layer 3: Every node in this layer is a circle node labeled with N . The i th node calculates the ratio of the i th rules firing strength to the sum of all rules firing strengths.

$$\bar{w}_i = \frac{w_i}{w_1 + w_2}, i = 1 \text{ and } 2. \quad (2.3)$$

Outputs of this layer are called normalized firing strengths.

Layer 4: Every node i in this layer is a square node with a node function

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i), \quad (2.4)$$

where $\bar{w}_i$ is the output of layer 3, and $\{p_i, q_i, r_i\}$ is a parameter set.

Layer 5: The single node in this layer is a circle node labeled with $\sum$, that computes the overall output, summation of all incoming signals,

$$O_1^5 = \text{overall output} = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (2.5)$$

A learning algorithm of ANFIS involves forward pass and backward pass. In the forward pass, functional signals go forward till layer 4 and the parameters are identified by least square estimate as in [6]. In the backward pass, the error rates propagate backward and the parameters are updated using the gradient descent method. The details of the learning algorithm are elaborated in [6].

2.3 Fuzzy Multi-layer Perceptron

This is a pioneering model where the integration of fuzzy sets and multilayer perceptron (MLP) resulted in the fuzzy MLP (FMLP), described in [15], for classification. The model involves fuzzy input vector and fuzzy output vector. The fuzzy input vector consists of 3-dimensional feature values whereas fuzzy output vector has membership values corresponding to different classes. Each feature is represented as a 3-dimensional membership vector using Π -membership functions, corresponding to the linguistic terms *low*, *medium* and *high*. The values of parameters for the Π -membership function, corresponding to the three linguistic terms, are chosen as in [15]. Network parameters are updated using the back propagation algorithm depending on the membership values at output nodes. The model provides a judicious integration of the capabilities of MLP in generating non linear boundaries and of fuzzy set in handling uncertainties arising from overlapping regions. Unlike, MLP, FMLP can accept both numerical and linguistic input. The superiority of FMLP over MLP in terms of classification performance and learning time has been adequately established. In Sect. 2.6, FMLP is used to design the granular models FRGNN1 [4] and FRGNN2 [5] with random weights within 0–1.

2.4 Knowledge Based Fuzzy Multi-layer Perceptron

A knowledge based fuzzy MLP for classification and rule generation is developed in [10]. Here, the number of hidden nodes and the connection weights of the network are determined by considering the values of the 3-dimensional features within an interval and outside the interval (compliment of the interval) for a particular class as in [14]. The membership values for the features are defined and these are encoded into the network as its connection weights. The hidden nodes of the network correspond to the intervals.

An interval $[F_{j1}, F_{j2}]$ is denoted as the range of feature F_j , covered by class C_k . The membership value of the interval is represented as $\mu([F_{j1}, F_{j2}]) = \mu(\text{between } F_{j1} \text{ and } F_{j2})$. It is computed as

$$\mu(\text{between } F_{j1} \text{ and } F_{j2}) = \{\mu(\text{greater than } F_{j1}) * \mu(\text{less than } F_{j1})\}^{1/2}, \quad (2.6)$$

where

$$\mu(\text{greater than } F_{j1}) = \begin{cases} \{\mu(F_{j1})\}^{1/2}, & \text{if } F_{j1} \leq c_{prop}, \\ \{\mu(F_{j1})\}^2, & \text{otherwise,} \end{cases} \quad (2.7)$$

and

$$\mu(\text{less than } F_{j2}) = \begin{cases} \{\mu(F_{j2})\}^{1/2}, & \text{if } F_{j2} \geq c_{prop}, \\ \{\mu(F_{j2})\}^2, & \text{otherwise,} \end{cases} \quad (2.8)$$

Here c_{prop} denotes c_{jl} , c_{jm} and c_{jh} for each of the corresponding three overlapping fuzzy sets *low*, *medium*, and *high*. These are defined as in [15]. The output membership for the corresponding class C_k is obtained as in [10].

The interval which does not include the class is considered as complement of the interval. The complement of the interval $[F_{j1}, F_{j2}]$ of the feature F_j is the region where the class C_k does not lie and is defined as $[F_{j1}, F_{j2}]^c$. The linguistic membership values for $[F_{j1}, F_{j2}]^c$ are defined by $\mu([F_{j1}, F_{j2}]^c) = \mu(\text{not between } F_{j1} \text{ and } F_{j2})$. It is calculated as

$$\mu(\text{not between } F_{j1} \text{ and } F_{j2}) = \max\{\mu(\text{less than } F_{j1}), \mu(\text{greater than } F_{j2})\} \quad (2.9)$$

since, not between F_{j1} and $F_{j2} \equiv$ less than F_{j1} OR greater than F_{j2} . The linguistic membership values for class C_k in the interval $[F_{j1}, F_{j2}]$ are denoted by $\{\mu_L([F_{j1}, F_{j2}]), \mu_M([F_{j1}, F_{j2}]), \mu_H([F_{j1}, F_{j2}])\}$. Similarly, for the complement of the interval, using Eq. 2.9, we have $\{\mu_L([F_{j1}, F_{j2}]^c), \mu_M([F_{j1}, F_{j2}]^c), \mu_H([F_{j1}, F_{j2}]^c)\}$.

A fuzzy multi-layer perceptron (FMLP) [15] with one hidden layer is considered, taking two hidden nodes corresponding to $[F_{j1}, F_{j2}]$ and its complement, respectively. Links are defined between the input nodes and the two nodes in the hidden layer iff $\mu_A([F_{j1}, F_{j2}])$ or $\mu_A([F_{j1}, F_{j2}]^c) \geq 0.5, \forall j$, where $A \in \{L, M, H\}$. The weight $w_{k_{\alpha_p}}^{(0)} j_m$ between the k_{α_p} node of the hidden layer (the hidden node corresponding to the interval $[F_{j1}, F_{j2}]$ for class C_k) and j_m ($m \in \{\text{first}(L), \text{second}(M), \text{third}(H)\}$)th node of the input layer corresponding to feature F_j is set by

$$w_{k_{\alpha_p}}^{(0)} j_m = p_k + \epsilon, \quad (2.10)$$

p_k is the a priori probability of class C_k and ϵ is a small random number. This hidden node is designated as positive node. A second hidden node k_{α_n} is considered for the compliment case and is termed as a negative node. Its connection weights are initialized as

$$w_{k_{\alpha_n}}^{(0)} j_m = (1 - p_k) + \epsilon. \quad (2.11)$$

It is to be mentioned that the method described above can suitably handle convex pattern classes only. In case of concave classes one needs to consider multiple intervals for a feature F_j corresponding to the various convex partitions that may be generated to approximate the given concave decision region. This also holds for the complement of the region in F_j in which a particular class C_k is not included.

Hence, in such cases, hidden nodes, positive and negative, are introduced for each of the intervals with connections being established by Eqs. 2.10 and 2.11 for the cases of a class belonging and not belonging to a region, respectively. This would result in multiple hidden nodes for each of the two cases. In this connection, one may note that a concave class may also be subdivided into several convex regions as in [9].

Let there be $(k_{pos} + k_{neg})$ hidden nodes, where $k_{pos} = \sum_{\alpha_p} k_{\alpha_p}$ and $k_{neg} = \sum_{\alpha_n} k_{\alpha_n}$ generated for class C_k such that $k_{pos} \geq 1$ and $k_{neg} \geq 1$. Now connections are established between k th output node (for class C_k) and only the corresponding $(k_{pos} + k_{neg})$ hidden nodes.

The connection weight $w_{kk_\alpha}^{(1)}$ between the k th output node and the k_α th hidden node is calculated from a series of equations generated as follows. For an interval as input for class C_k , the expression for output $y_k^{(2)}$ of the k th output node is given by

$$y_k^{(2)} = f(y_{k_\alpha}^{(1)} w_{kk_\alpha}^{(1)} + \sum_{r \neq \alpha} 0.5 w_{kk_r}^{(1)}) \quad (2.12)$$

where $f(\cdot)$ is the sigmoid function and the hidden nodes k_r correspond to the intervals not represented by the convex partition α . Thus for a particular class C_k we have as many equations as the number of intervals (including *not*) used for approximating any concave and/or convex decision region C_k . Thereby, we can uniquely compute each of the connection weights $w_{kk_\alpha}^{(1)} \forall \alpha$ (corresponding to each hidden node k_α and class C_k pair).

The network architecture, so encoded, is then refined by training it on the pattern set supplied as input. In case of “all connections” between input and hidden layers, all the link weights are trained. In case of “selected connections” only the selected link weights are trained, while the other connections are kept clamped at zero. If the network achieves satisfactory performance, the classifier design is complete. Otherwise, node growing or link pruning is applied to the network as in [10].

2.5 Rough Fuzzy Multi-layer Perceptron

Rough fuzzy MLP (rfMLP) [1] is designed by associating rough sets with fuzzy MLP (FMLP). Rough sets are used to extract domain knowledge about data. The process of extracting the knowledge about data and encoding it into FMLP is described as follows:

Figure 2.1 shows a block diagram for knowledge encoding procedure for rough fuzzy MLP in which the data is first transformed into a 3-dimensional linguistic space. A threshold Th ($0.5 \leq Th < 1$) is imposed on the resultant linguistic data in order to make the data into binary (0 or 1). The binary valued data is represented in a decision table $S = \langle U, A \rangle$ with C , a set of condition attributes, and $D = \{d_1, \dots, d_n\}$, a set of decision attributes.

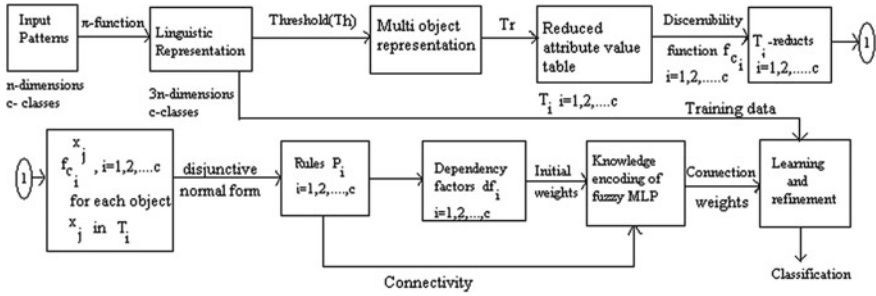


Fig. 2.1 Block diagram of the knowledge encoding procedure in rough fuzzy MLP [1]

(1) *Rule generation*: The decision table $S = \langle U, A \rangle$ is divided into n tables $S_i = \langle U_i, A_i \rangle$, $i = 1, 2, \dots, n$ corresponding to n decision attributes $d_1, \dots, d_n$, where $U = U_1 \cup U_2, \dots, \cup U_n$ and $A_i = C \cup \{d_i\}$. The size of each S_i , $i = 1, 2, \dots, n$, is first reduced with the help of a threshold on the number of occurrences of the same pattern of attribute values. The reduced decision table is denoted by τ_i , and $\{x_{i_1}, \dots, x_{i_p}\}$ is a set of objects of U_i that occur in τ_i , $i = 1, 2, \dots, n$. The discernibility matrix and discernibility function $f_{d_i}^{x_j}$ for every object $x_j \in \{x_{i_1}, \dots, x_{i_p}\}$ are defined as in [1]. Then, $f_{d_i}^{x_j}$ is brought to its CNF a dependency rule r_i , viz., $P_i \rightarrow d_i$, where P_i is the disjunctive normal form (DNF) of $f_{d_i}^{x_j}$, $j \in \{i_1, \dots, i_p\}$.

(2) *Knowledge encoding*: A $n_k \times 3n$ -dimensional attribute value table for each class C_k is generated, where n_k indicates the number of objects in C_k . There are m sets $O_1, O_2, \dots, O_m$ of objects in the table having identical attribute values and $\text{card}(O_i) = n_k$, $i = 1, \dots, m$, such that $n_{k_1} \geq \dots \geq n_{k_m}$ and $\sum_{i=1}^m n_{k_i} = n_k$. The attribute value table can now be represented as $m \times 3n$ array. Let $n'_{k_1}, \dots, n'_{k_m}$ denote the distinct elements among $n_{k_1}, \dots, n_{k_m}$ such that $n'_{k_1} \geq \dots \geq n'_{k_m}$. A heuristic threshold function is defined as

$$Tr = \left\lceil \frac{\sum_{i=1}^m \frac{1}{n'_{k_i} - n'_{k_{(i+1)}}}}{TH} \right\rceil. \quad (2.13)$$

All entries having frequency less than Tr are eliminated from the table, resulting in the reduced attribute-value table. The main motive of introducing this threshold function lies in reducing the size of the resulting network thereby; eliminating noisy pattern representatives (having lower values of n_{k_i}) from the reduced attribute-value table. Based on the reduced attribute-value table, rough rules are generated as explained above. These are used to determine the dependency factors df_i , $i = 1, 2, \dots, c$ which are then considered as the initial connection weights of the fuzzy MLP.

2.6 Architecture of Fuzzy Rough Granular Neural Networks

The fuzzy rough granular neural networks, FRGNN1 [4] and FRGNN2 [5] for classification are formulated by integrating fuzzy sets, fuzzy rough sets and multilayer perceptron. The advantages of the granular neural network include tractability, robustness, and close resemblance with human-like (natural) decision making for pattern recognition in ambiguous situations. It may be noted that while multilayer perceptron is a capable of classifying all types of real life data (both convex and non convex shapes), rough set handles uncertainty in class information, and fuzzy sets are used to resolve the overlapping regions among the classes. The information granules derived from the lower approximation of rough set are employed in the process of designing the network models. Each of the networks contains the input layer, the hidden layer and the output layer. A gradient decent method is used for training the networks. The number of nodes in input layer is determined by $3n$ dimensional granular input vector where n is total number of features in data. Every feature is transformed into 3-dimensional features, that constitute a granular input vector, along a feature axis of the three linguistic terms *low*, *medium* or *high*. The number of nodes in the hidden layer of FRGNN1 is set equal to the number of decision tables (c) where every decision table contains $3n$ dimensional linguistic training data belonging to the c classes. In FRGNN2, the number of nodes in the hidden layer is set is equal to c where c is number of classes, and only one decision table that consists the entire training data is used. The initial connection weights of FRGNN1 and FRGNN2, based on the decision tables, are defined using the concepts of fuzzy rough sets.

Let us now describe in detail the architecture of fuzzy rough granular neural networks (FRGNN1 [4] and FRGNN2 [5]) for classification. The networks use multi-layer perceptron with back propagation algorithm based on the gradient decent method for training purpose. The initial architecture of FRGNN1 and FRGNN2 is shown in Fig. 2.2. The FRGNN1 and FRGNN2 contain the input, hidden and output layers. The number of nodes/neurons in the input layer of FRGNN1 and FRGNN2 is set equal to 3-dimensional (3D) granular vector. While the number of nodes in the hidden layer of FRGNN1 is set equal to the number decision tables, this number is equal to the number of classes for FRGNN2. The number of nodes in the output layer for both FRGNN1 and FRGNN2 is the number of classes present in the data. Each neuron in the hidden layer is fully connected to the neurons in the previous layer and the next layer. The nodes in the input layer, non-computational nodes, receives an input and distributes to all the neurons in the next layer via its connection weights, during training. The input vector presented at the nodes in the input layer is in the granular form. The outputs of the input neurons are considered as the activation values to the nodes in the hidden layer. Similarly, the nodes in the output layer are received the outputs of the hidden layer nodes as their activation values. The node in the output layer corresponding to a class is assigned with a membership value for a pattern, while the other nodes, representing the other classes, are assigned with zeros. The fuzzy rough rules are initialized as initial connection weights of the links connecting

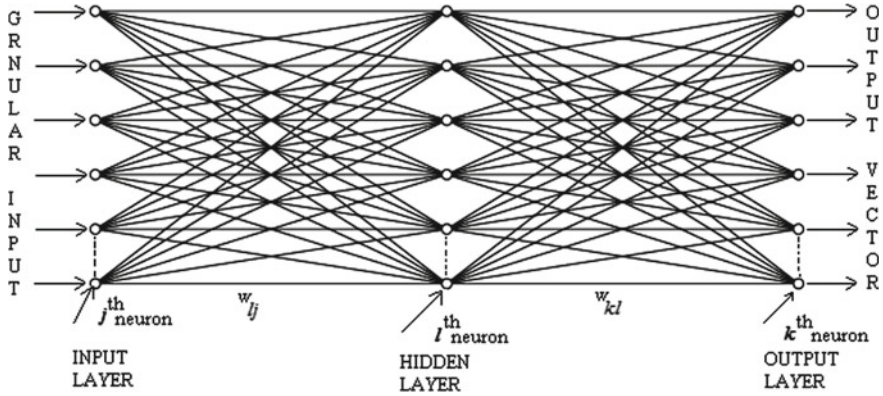


Fig. 2.2 Initial architecture of FRGNN1 and FRGNN2 with granular input layer, hidden layer and output layer with the same number of nodes

the nodes between input & hidden and hidden & output layers. The FRGNN1 and FRGNN2 are trained using back propagation algorithm with gradient-decent method which is described as follows:

Input

D: Data set with training patterns in the granular form and their associated target vectors in terms of membership values and zeros (see Sect. 2.7.3 for details)

α : Learning rate

η : Momentum term

$b_l = b$, Bias term which is kept constant at the node (l) in hidden and output layers

Network: Granular feed-forward network

Method

1. Assign initial weights, in terms of dependency factors, between the nodes (units) in all the layers of the network, where the weights are determined by fuzzy rough sets based on a fuzzy similarity relation or fuzzy reflexive relation;
2. While the network is trained for all training patterns for a particular number of iterations {

Forward Propagation:

3. For each unit j of the input layer, the output, say x_j , of an input unit, say I_j is

$$\{$$

$$x_j = I_j;$$

$$\}$$

4. For each node l of the hidden or output layers, compute the output o_l of each unit l with respect to the previous layer, j , as

$$\{$$

$$I_l = \sum_j w_{lj}x_j + b;$$

$$\}$$
5. Apply logistic activation function to compute the output of each node l

$$\{$$

$$\phi(x_l) = \frac{1}{1+e^{-I_l}};$$

$$\}$$
 /* Note that, the logistic activation function is set at each node l in the hidden and the output layers*/
Back propagation
6. For each node in the output layer say k , compute the error using

$$\{$$

$$Error_k = \phi(x_k)(1 - \phi(x_k))(TG_k - \phi(x_k));$$

$$\}$$
 where TG_k denotes the target value for each node k in the output layer.
7. Compute the error for l th node in the hidden layer by using the error obtained at k th node in the output layer (with respect to the next layer)

$$\{$$

$$\Upsilon_l = \phi(x_l)(1 - \phi(x_l)) \sum_k Error_k w_{kl};$$

$$\}$$
 where $Error_k$ is an error value at the node k in the output layer (next layer).
8. Update the weight w_{lj} in the network using

$$\{$$

$$\Delta w_{lj} = (\alpha)x_j \Upsilon_l;$$

$$\Delta w_{lj}(e) = (\alpha)x_j \Upsilon_l + (\eta)\Delta w_{lj}(e - 1);$$

$$\}$$
 where Υ_l is the error value at the node l of the hidden or output layers.
9. For each constant value of bias b at node l of the hidden or output layers in the network

$$\{$$

$$\Delta b = (\alpha)\Upsilon_l;$$

$$b(e) += \Delta b(e - 1);$$

$$\}$$
 /* end while loop*/

Output: Classified patterns

Here, the momentum term η is used to escape the local minima in the weight space. The value of the bias b is kept constant at each node of the hidden and output layers of the network, and e denotes the number of iterations. For every iteration, the training data is presented to the network for updating the weight parameters w_{lj} and bias b . The resulting trained network is used for classifying the test patterns.

2.7 Input Vector Representation

A formal definition of fuzzy granule is defined by the generalized constraint form “X is_r R” where ‘R’ is a constrained relation, ‘r’ is a random set constraint which is a combination of probabilistic and possibilistic constraints, and ‘X’ is a fuzzy set which contains the values of the patterns in terms of linguistic terms *low*, *medium* and *high*. By using fuzzy-set theoretic techniques [12], a pattern x is assigned a membership value, based on a π function, to fuzzy set A and is defined as

$$A = \{(\mu_A(x), x)\}, \quad \mu_A(x) \in [0, 1] \quad (2.14)$$

where $\mu_A(x)$ represents the membership value of the pattern x . The π membership function, with range $[0, 1]$ and $x \in \mathbf{R}^n$, is defined as

$$\pi(x, C, \lambda) = \begin{cases} 2(1 - \frac{\|x-C\|_2}{\lambda})^2, & \text{for } \frac{\lambda}{2} \leq \|x - C\|_2 \leq \lambda, \\ 1 - 2(\frac{\|x-C\|_2}{\lambda})^2, & \text{for } 0 \leq \|x - C\|_2 \leq \frac{\lambda}{2}, \\ 0, & \text{otherwise} \end{cases} \quad (2.15)$$

where $\lambda > 0$ is a scaling factor (radius) of the π function with C as a central point, and $\|\cdot\|_2$ denotes the Euclidean norm.

2.7.1 Incorporation of Granular Concept

An n -dimensional i th pattern is represented by a $3n$ -dimensional linguistic vector [15] as

$$\vec{F}_i = [\mu_{low(F_{i1})}(\vec{F}_i), \mu_{medium(F_{i1})}(\vec{F}_i), \mu_{high(F_{i1})}(\vec{F}_i), \dots, \mu_{high(F_{in})}(\vec{F}_i)] \quad (2.16)$$

where $\vec{F}_i$ is a feature vector for i th pattern, μ indicates the value of π - function corresponding to linguistic terms (fuzzy granule) *low*, *medium* or *high* along each feature axis. For example, the fuzzy granule for the feature F_{i1} of i th pattern (in terms of *low*, *medium* and *high*) is quantified as

$$F_{i1} \equiv \left\{ \frac{0.689}{L}, \frac{0.980}{M}, \frac{0.998}{H} \right\}.$$

When the input is numerical, we use the π -membership function (Eq.2.15) with appropriate values of center C and scaling factor λ . Their selection is explained in the following section.

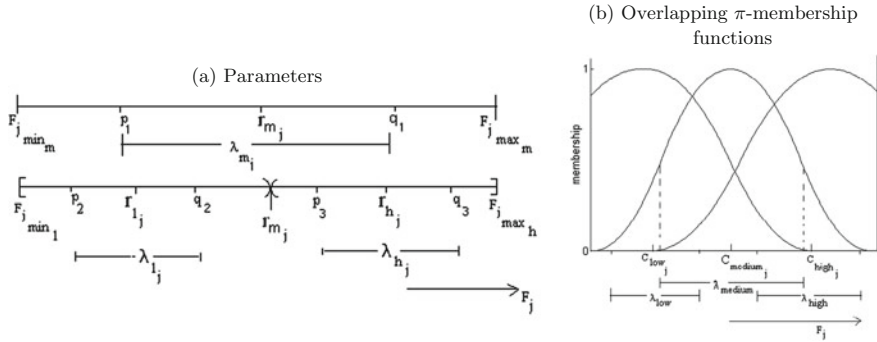


Fig. 2.3 Parameters for overlapping linguistic π -membership functions corresponding to linguistic terms, *low*, *medium* and *high* along the j th feature F_j

2.7.2 Choice of Parameters of π Membership Functions

Let $\{F_{ij}\}$ for $i = 1, 2, \dots, s$, $j = 1, 2, \dots, n$, represent a set of s patterns with n features, and F_{jmin_m} and F_{jmax_m} denote the minimum and maximum values along j th feature considering all the s patterns. Initially, the average of feature values of all the s patterns along the j th feature F_j is considered as the center of the linguistic term *medium* along that feature and denoted by r_{m_j} , as shown in Fig. 2.3a. Then, the average values (along the j th feature F_j) of the patterns having label values in the ranges $[F_{jmin_m}, r_{m_j})$ and $(r_{m_j}, F_{jmax_m}]$ are defined as the means of linguistic terms *low* and *high*, denoted by r_{l_j} and r_{h_j} , respectively. Similarly, considering the patterns with the values in the ranges $[F_{jmin_m}, r_{m_j})$ and $(r_{m_j}, F_{jmax_m}]$ along j th axis, we define $F_{jmin_l} = F_{jmin_m}$, $F_{jmax_l} = r_{m_j}$, $F_{jmin_h} = r_{m_j}$, and $F_{jmax_h} = F_{jmax_m}$. Then the center C and corresponding scaling factor λ for linguistic terms *low*, *medium* and *high* along the j th feature F_j are as follows:

$$\begin{aligned}
 C_{medium_j} &= r_{m_j}, \\
 p_1 &= C_{medium_j} - \frac{F_{jmax_m} - F_{jmin_m}}{2}, \\
 q_1 &= C_{medium_j} + \frac{F_{jmax_m} - F_{jmin_m}}{2}, \\
 \lambda_{m_j} &= q_1 - p_1, \\
 \lambda_{medium} &= \frac{\sum_{j=1}^n \lambda_{m_j}}{n}.
 \end{aligned} \tag{2.17}$$

$$\begin{aligned}
 C_{low_j} &= r_{l_j}, \\
 p_2 &= C_{low_j} - \frac{F_{jmax_l} - F_{jmin_l}}{2}, \\
 q_2 &= C_{low_j} + \frac{F_{jmax_l} - F_{jmin_l}}{2}, \\
 \lambda_{l_j} &= q_2 - p_2, \\
 \lambda_{low} &= \frac{\sum_{j=1}^n \lambda_{l_j}}{n}.
 \end{aligned} \tag{2.18}$$

$$\begin{aligned}
C_{high_j} &= r_{h_j}, \\
p_3 &= C_{high_j} - \frac{F_{jmax_h} - F_{jmin_h}}{2}, \\
q_3 &= C_{high_j} + \frac{F_{jmax_h} - F_{jmin_h}}{2}, \\
\lambda_{h_j} &= q_3 - p_3, \\
\lambda_{high} &= \frac{\sum_{j=1}^n \lambda_{h_j}}{n}.
\end{aligned} \tag{2.19}$$

The value of scaling factor for linguistic term *low* along the j th feature axis (λ_{l_j}) is defined as in Eq. 2.17, where the minimum and maximum values of λ_{l_j} are represented by p_1 and q_1 . Moreover, the values of scaling factors for linguistic terms *medium* and *high* along the j th feature axis are defined as in Eqs. 2.18 and 2.19, where the corresponding minimum & maximum values are denoted by (p_2 & q_2) and (p_3 & q_3). The λ -value for a linguistic term is defined by taking the average of values of the linguistic term along all feature axes. Clearly, the λ -value for a particular linguistic term along all the axes would remain the same. Equations 2.17–2.19 automatically ensure that each input feature value along j th axis for a pattern $\vec{F}_i$ is assigned three membership values corresponding to the three-dimensional (3D) granular space of Eq. 2.16 in such a way that at least one of the $\mu_{low(F_{ij})}(\vec{F}_i)$, $\mu_{medium(F_{ij})}(\vec{F}_i)$ and $\mu_{high(F_{ij})}(\vec{F}_i)$ is greater than 0.5. In other words, this allows a pattern $\vec{F}_i$ to have a strong membership to at least one of the linguistic properties *low*, *medium* or *high*. This representation is shown diagrammatically in Fig. 2.3b.

2.7.3 Defining Class Membership at Output Node

The membership of i th pattern to k th class is defined as [12]

$$\mu_k(\vec{F}_i) = \frac{1}{1 + (Z_{ik}/f_d)^{f_e}} \tag{2.20}$$

where Z_{ik} is a weighted distance, and f_d and f_e are the denominational and exponential fuzzy generators controlling the amount of fuzziness in the class membership. Obviously, the class membership lies in $[0, 1]$. Here, the weighted distance Z_{ik} is defined in [15] as

$$Z_{ik} = \sqrt{\sum_{j=1}^n \left[\sum_{p=1}^3 \frac{1}{3} (\mu_p(F_{ij}) - \mu_p(o_{kj}))^2 \right]}, \quad \text{for } k = 1, 2, \dots, c \tag{2.21}$$

where o_{kj} is the center of the j th feature vector from the k th class, μ_p is a membership value of the p th linguistic term (granule), and c is the number of classes. In the fuzziest case, we may use fuzzy modifier, namely, contrast intensification (INT) operator, to

enhance the contrast in membership values of each pattern within that class in order to decrease the ambiguity in taking a decision.

2.7.4 Applying the Membership Concept to the Target Vector

The target vector at the output layer is defined in Eq. 2.22 by a membership value and zeros. For example, if a training pattern belongs to k th class, its desired output vector would have only one non-zero membership value corresponding to the k th node representing that class and zero value for the remaining nodes in the output layer. Therefore, for the i th training pattern $\vec{F}_i$ from the k th class, we define the desired output of the k th output node TG_k considering the fuzziest case, as

$$TG_k = \begin{cases} \mu_{INT}(\vec{F}_i), & \text{if the } i\text{th pattern is from } k\text{th class denoting} \\ & \text{the } k\text{th output node,} \\ 0, & \text{otherwise.} \end{cases} \quad (2.22)$$

The network then back-propagates the errors with respect to the desired membership values at the output stage.

2.8 Fuzzy Rough Sets: Granulations and Approximations

In the context of fuzzy rough set theory, fuzzy set theory [8, 13] allows that an object belongs to a set and a couple of objects belong to a relation are assigned with membership values, and rough set theory allows the objects in a set that are assigned with the membership values belonging to the lower and approximations. Recall that Eq. 2.14 represents fuzzy set in U . A couple of objects x and y in U , the relation R is a mapping $U \times U \rightarrow [0, 1]$ such that $R = \{((x, y), R(x, y)) | R(x, y) \in [0, 1]\}$. For each $y \in U$, the R -foreset of y is the fuzzy set, say R_y , and is defined as $R_y(x) = R(x, y)$, for all x in U . The similarity between the two objects is represented by a fuzzy similarity relation and it should satisfy the following properties:

$$\begin{aligned} R(x, x) &= 1 \quad (\text{reflexive}), \\ R(x, y) &= R(y, x) \quad (\text{symmetry}), \text{ and} \\ T(R(x, y)R(y, z)) &\leq R(x, z) \quad (T\text{-transitivity}), \end{aligned}$$

for all x, y and z in U . Given a T -norm, if R does not satisfy symmetry then R is called a fuzzy similarity relation (fuzzy tolerance relation), and if R does not satisfy symmetry and T -transitivity properties then R is called a fuzzy reflexive relation (fuzzy t -equivalence relation). Usually, the fuzzy T -equivalence relations constitute fuzzy T -equivalence class (fuzzy equivalence granule). The fuzzy T -equivalence

classes are exploited in approximating the sets (concepts) in U . In fuzzy rough set theory, the following fuzzy logical connectives [8] are typically used in generalization of the lower and upper approximations of a set. For all x and $y \in [0, 1]$, an operator T mapping from $[0, 1]^2$ to $[0, 1]$ satisfies $T(1, x) = x$. We use T_M and T_L to represent T -norms and these are defined as

$$T_M(x, y) = \min(x, y), \quad (2.23)$$

$$T_L(x, y) = \max(0, x + y - 1) \quad (\text{Lukasiewicz } t\text{-norm}). \quad (2.24)$$

for all x and $y \in [0, 1]$. On the other, a mapping $I : [0, 1]^2 \rightarrow [0, 1]$ such that $I(0, 0) = 1$, $I(1, x) = x$ for all $x \in [0, 1]$ where I is an implicator. For all x and $y \in [0, 1]$, the implicators I_M and I_L are defined as

$$I_M(x, y) = \max(1 - x, y), \quad (2.25)$$

$$I_L(x, y) = \min(1, 1 - x + y) \quad (\text{Lukasiewicz implicator}). \quad (2.26)$$

2.8.1 Concepts of Fuzzy Rough Sets: Crisp and Fuzzy Ways

In fuzzy rough sets, an information system is a pair $(U, \mathcal{A})$ where $U = \{x_1, x_2, \dots, x_s\}$ and $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ are finite non empty sets of objects and conditional attributes, respectively. The values of conditional attributes can be quantitative (real valued). Let a be an attribute in $\mathcal{A}$. The fuzzy similarity relation R_a between any two objects x and y in U with respect to the attribute a is defined as [7]

$$R_a(x, y) = \max \left(\min \left(\frac{a(y) - a(x) + \sigma_a}{\sigma_a}, \frac{a(x) - a(y) + \sigma_a}{\sigma_a} \right), 0 \right) \quad (2.27)$$

where σ_a denotes the standard deviation of the attribute a . It may be noted that the relation R_a does not necessarily satisfy the T -transitivity property.

A decision system $(U, \mathcal{A} \cup \{d\})$ is one kind of information system in which d ($d \notin \mathcal{A}$) is called a decision attribute and can be qualitative (discrete-valued). Based on these values, the set U is partitioned into non-overlapping sets/concepts corresponding to the decision concepts/decision classes, say $d_k, k = 1, 2, \dots, c$. Each decision class can be represented by a crisp set or a fuzzy set. The belongingness of a pattern to each of the classes can be represented by decision feature containing membership values for all the patterns in the class. For a qualitative attribute a in $\{d\}$, the decision classes are defined in the following two methods.

2.8.1.1 Method I: Crisp Way of Defining Decision Classes (Crisp Case)

$$R_a(x, y) = \begin{cases} 1, & \text{if } a(x) = a(y), \\ 0, & \text{otherwise,} \end{cases} \quad (2.28)$$

for all x and y in U . The crisp-valued decision class implies that objects in the universe U corresponding to the decision class would take values only from the set $\{0, 1\}$. A fuzzy set $A \subseteq U$ can be approximated only by constructing the lower and upper approximations of A with respect to crisp decision classes.

In real life problems, the data is generally ill defined, with overlapping class boundaries. Each pattern (object) in the set $A \subseteq U$ may belong to more than one class. To model such data, we extend the concept of a crisp decision granule into a fuzzy decision granule by inclusion of the fuzzy concept to crisp decision granules. The fuzzy decision classes are defined as follows:

2.8.1.2 Method II: Fuzzy Way of Defining Decision Classes (Fuzzy Case)

Consider a c -class problem domain where we have c decision classes of a decision attribute in a decision system. Let the n -dimensional vectors O_{kj} and V_{kj} , $j = 1, 2, \dots, n$, denote the mean and standard deviation of the data for the k th class in the decision system. The weighted distance of a pattern $\vec{F}_i$ from the k th class is defined [12] as

$$Z_{ik} = \sqrt{\sum_{j=1}^n \left[\frac{F_{ij} - O_{kj}}{V_{kj}} \right]^2}, \text{ for } k = 1, 2, \dots, c, \quad (2.29)$$

where F_{ij} is the value of the j th feature (attribute) of the i th pattern. The parameter $\frac{1}{V_{kj}}$ acts as the weighting coefficient, such that larger the value of V_{kj} is, the less is the importance of the j th feature in characterizing the k th class. Note that when the value of a feature for all the patterns in a class is the same, the standard deviation of those patterns along that feature will be zero. In that case, we consider $V_{kj} = 0.000001$ (a very small value, instead of zero for the sake of computation), so that the distance Z_{ik} becomes high and the membership value of i th pattern to k th class, denoted by $\mu_k(\vec{F}_i)$, becomes low where the membership value is defined using Eq. 2.20.

It may be noted that when a pattern $\vec{F}_i$ has different membership values corresponding to c decision classes then its decision attribute becomes quantitative, i.e., each pattern with varying membership value. The quantitative decision attribute can be made qualitative (as in Method I, crisp case) by two ways, namely, (i) by computing the average of the membership values over all the patterns in k th class to its own class, and assigning it to each pattern $\vec{F}_i$ in its k th decision class, and (ii) by computing the average of the membership values over all the patterns in k th class to the other classes, and assigning it to each pattern $\vec{F}_i$ in other decision classes (other

than the k th class). So the average membership value (qualitative) of all the patterns in the k th class to its own class is defined as

$$D_{kk} = \frac{\sum_{i=1}^{m_k} \mu_k(\vec{F}_i)}{|m_k|}, \text{ if } k = u, \quad (2.30)$$

and the average membership values of all the patterns in the k th class to other decision classes are defined as

$$D_{ku} = \frac{\sum_{i=1}^{m_k} \mu_u(\vec{F}_i)}{|m_k|}, \text{ if } k \neq u, \quad (2.31)$$

where $|m_k|$ indicates the number of patterns in the k th class and k and $u = 1, 2, \dots, c$.

For a qualitative attribute $a \in \{d\}$, the fuzzy decision classes are defined as

$$R_a(x, y) = \begin{cases} D_{kk}, & \text{if } a(x) = a(y), \\ D_{ku}, & \text{otherwise,} \end{cases} \quad (2.32)$$

for all x and y in U . Here D_{kk} corresponds an average membership value of all the patterns that belong to the same class ($k = u$), and D_{ku} corresponds to the average membership values of all the patterns from the classes other than k ($k \neq u$).

For any $\mathcal{B} \subseteq \mathcal{A}$, the fuzzy $\mathcal{B}$ -indiscernibility relation $R_{\mathcal{B}}$ based on the fuzzy similarity relation R is induced by

$$R_{\mathcal{B}}(x, y) = R_a(x, y), a \in \mathcal{B}, \quad (2.33)$$

The fuzzy lower and upper approximations of the set $A \subseteq U$, based on fuzzy similarity relation R , are defined as [16]

$$(R_{\mathcal{B}} \downarrow R_d)(y) = \inf_{x \in U} I(R_{\mathcal{B}}(x, y), R_d(x)), \quad (2.34)$$

$$(R_{\mathcal{B}} \uparrow R_d)(y) = \sup_{x \in U} T(R_{\mathcal{B}}(x, y), R_d(x)), \quad (2.35)$$

for all y in U . Here the fuzzy logic connectives, Lukasiewicz T -norm T and implication I , are exploited in defining the lower and upper approximations. The fuzzy positive region, based on the fuzzy $\mathcal{B}$ -indiscernibility relation, is defined in [16] as

$$POS_{\mathcal{B}}(y) = \left(\bigcup_{x \in U} R_{\mathcal{B}} \downarrow R_d x \right)(y), \quad (2.36)$$

for all x and $y \in U$. Here, $POS_{\mathcal{B}}(y)$ characterizes the positive region with a maximum membership value of pattern y . Considering y belonging to a set A which is a subset of U , the fuzzy positive region Eq. 2.37 is simplified as [2]

$$POS_{\mathcal{B}}(y) = (R_{\mathcal{B}} \downarrow R_d x)(y). \quad (2.37)$$

The dependency degree of a set of attributes $\mathcal{B} \subseteq \mathcal{A}$, denoted by $\gamma_{\mathcal{B}}$, is defined as

$$\gamma_{\mathcal{B}} = \frac{\sum_{x \in U} POS_{\mathcal{B}}(x)}{|U|} \quad (2.38)$$

where $|\cdot|$ denotes the cardinality of a set U , and γ is $0 \leq \gamma \leq 1$.

It may be noted that the aforesaid procedure of determining the dependency factor of each conditional attribute with respect to the decision classes (either crisp case or fuzzy case) enables the fuzzy rough granular neural networks to define their initial weight parameters from the training samples. The initial weights in the fuzzy case (Method II) are seen to provide better performance in handling the overlapping class boundaries than those of the crisp case (Method I). These are explained in the following sections.

2.9 Configuration of the Granular Neural Networks Using Fuzzy Rough Sets

In this section, we explain the concept of granulation by partitioning data in the decision tables, computing equivalence relations using the fuzzy similarity relation and approximation of a set using rough approximations. Based on the granulation principle, the initial weights of the FRGNN1 and FRGNN2 are then determined; thereby providing a knowledge based network. Such a network is found to be more efficient than fuzzy MLP [15] and other similar types of granular neural networks [1, 3] as shown in experiments. During training, the networks search for a set of connection weights that corresponds to some local minima. Note that there may be a large number of such minima corresponding to various good solutions. Therefore, if we can initially set weights of the network so as to correspond one such solution, the searching space may be reduced and learning thereby becomes faster. Further, the architecture of the network can be made simpler by fixing the number of nodes in the hidden layer based on the class information. These are the characteristics that the fuzzy rough granular neural networks are capable to achieve. At first, the knowledge encoding procedure of FRGNN1 is described using the concepts of fuzzy rough set and fuzzy similarity relation. Then, it is also described for FRGNN2. The procedure of knowledge encoding is explained in Fig. 2.4 using block diagram.

2.9.1 Knowledge Encoding Procedures

Method of FRGNN1: The knowledge encoding procedure for FRGNN1 [4] is explained as follows. We use the aforesaid decision table $S = (U, \mathcal{A} \cup \{d\})$ with $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ its set of conditional attributes, and with decision attributes $\{d\}$

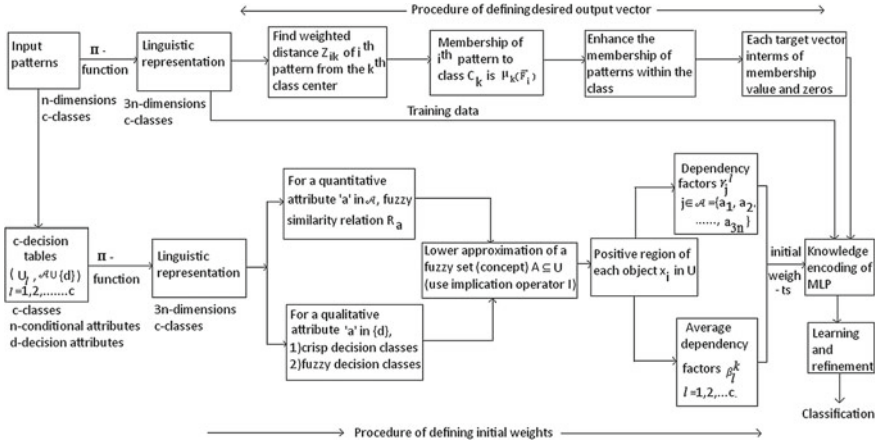


Fig. 2.4 Block diagram of knowledge encoding

where $U = \{x_1, x_2, \dots, x_s\}$, its set of objects, form c classes and the objects having labeled values corresponding to each n -dimensional conditional attribute. We split the decision table $S = (U, \mathcal{A} \cup \{d\})$ into c decision tables $S_l = (U_l, \mathcal{A} \cup \{d\})$, $l = 1, 2, \dots, c$, corresponding to c classes, and the objects are added to each decision table from all the c classes succession. Moreover S_l satisfies the following conditions:

$$(i) U_l \neq \emptyset, \quad (ii) \bigcup_{l=1}^c U_l = U, \quad (iii) \bigcap_{l=1}^c U_l = \emptyset. \quad (2.39)$$

The size of each S_l , $l = 1, 2, \dots, c$, is dependent on the available number of objects from all the classes. If all the classes have an equal number of patterns, then the number of objects that will be added to each S_l will be the same; otherwise, it will be different.

Let us consider the case of feature F_1 , when $j = 1$ (attribute a_1). The i th pattern $\vec{F}_i$ is mapped to a point in the 3D feature space of $\mu_{low(F_{i1})}(\vec{F}_i)$, $\mu_{medium(F_{i1})}(\vec{F}_i)$, $\mu_{high(F_{i1})}(\vec{F}_i)$ using Eq. 2.16. In this manner, an n -dimensional attribute valued decision table can be transformed into a $3n$ -dimensional attribute valued decision table. We apply the following procedure for the decision table S_l , $l = 1, 2, \dots, c$.

- S1: Define granulation structures, using the fuzzy similarity relation in Eq. 2.27, for each conditional attribute by generating a fuzzy similarity matrix.
- S2: Use the fuzzy similarity relations and compute lower approximations of Eq. 2.34 of each concept for each conditional attribute with respect to the decision classes using Eqs. 2.28 or 2.32.
- S3: Calculate the fuzzy positive region, using Eq. 2.37, of an object in the concept for each conditional attribute.

- S4: Calculate dependency degrees, using Eq. 2.38, for the conditional attributes, and the dependency degrees are initialized between the nodes of the input layer and the hidden layer of networks as connection weights.
- S5: Calculate the average of the dependency degrees of all conditional attributes with respect to every decision class and define the average values between the nodes of the hidden layer and the output layer as initial weights.

Let us now design the initial structure of FRGNN1. It contains three layers. The number of nodes in the input layer is set equal to $3n$ -dimensional attributes and it is represented by c classes in the output layer. The hidden layer nodes are modeled with the class information. We explain the procedure for initializing the initial weights of FRGNN1.

Let γ_j^l denote the dependency degree of j th conditional attribute in l th decision table S_l where $j \in \mathcal{A} = \{a_1, a_2, \dots, a_{3n}\}$ and $l = 1, 2, \dots, c$. The weight w_{lj} between an input node j and a hidden node l (see Fig. 2.2) is defined as follows: For instance, the decision table S_1 , when $l = 1$,

$$\gamma_j^1 = \frac{\sum_{x \in U} POS_j(x)}{|U|}. \quad (2.40)$$

Let β_l^k denote the average dependency degrees for all conditional attributes with respect to k th decision class d_k , $k = 1, 2, \dots, c$, in l th decision table S_l , $l = 1, 2, \dots, c$. The weight w_{kl} between the hidden node l and the output node k (see Fig. 2.2) is defined as follows: For instance, the decision table S_1 , when $l = 1$, and its k th decision class d_k , $k = 1, 2, \dots, c$,

$$\beta_1^k = \frac{\sum_{j=1}^{3n} \gamma_j^k}{|3n|}, \quad (2.41)$$

where γ_j^k is defined as

$$\gamma_j^k = \frac{\sum_{x \in A_k} POS_j(x)}{|A_k|}, \quad (2.42)$$

where $A_k \subseteq U$ represents k th set that contains objects with respect to k th decision class d_k and $POS_j(x)$ indicate the fuzzy positive membership value of an object x in A_k , corresponding to j th attribute.

Similar procedure is applied to the rest of the decision tables S_l ($l = 2, 3, \dots, c$). Then the dependency degrees of conditional attributes, γ_j^l , for $l = 1, 2, \dots, c$, are used as the initial weights between the nodes of the input layer and the hidden layer (see Step 4). The initial weights between the nodes of the hidden and the output layers are set as β_l^k (see Step 5). The connection weights, so encoded, are refined by training the network on a set of patterns supplied as input.

Method of FRGNN2: We use the aforesaid decision system $S = (U, \mathcal{A} \cup \{d\})$ where $U = \{x_1, x_2, \dots, x_s\}$ represents a set of objects from c -classes, and $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ & $\{d\}$ denote set of conditional attributes and decision attributes, respectively. The

objects in universe U are classified into k number of non-overlapping sets/concepts, A_k , corresponding to decision classes, $d_k, k = 1, 2, \dots, c$. The fuzzy reflexive relation R_a between two objects x and y in U corresponding to an attribute a is defined as [5]

$$R_a(x, y) = \begin{cases} \max \left(\min \left(\frac{a(y)-a(x)+\sigma_{a_k}}{\sigma_{a_k}}, \frac{a(x)-a(y)+\sigma_{a_k}}{\sigma_{a_k}} \right), 0 \right), \\ \quad \text{if } a(x) \& a(y) \in A_k, \\ \max \left(\min \left(\frac{a(y)-a(x)+\sigma_{a_u}}{\sigma_{a_u}}, \frac{a(x)-a(y)+\sigma_{a_u}}{\sigma_{a_u}} \right), 0 \right), \\ \quad \text{if } a(x) \in A_k, a(y) \in A_u, \\ \quad \text{and } k \neq u \end{cases} \quad (2.43)$$

where k and $u = 1, 2, \dots, c$, and σ_{a_k} and σ_{a_u} represent the standard deviation of sets (concepts) A_k and A_u , respectively, corresponding to an attribute a .

Let us consider the case of feature F_j (attribute a_j) in the decision table S . The i th representative pattern $\vec{F}_i$ is mapped to a point in the 3D granular space of $\mu_{low(F_{i1})}(\vec{F}_i)$, $\mu_{medium(F_{i1})}(\vec{F}_i)$ and $\mu_{high(F_{i1})}(\vec{F}_i)$ by using Eq. 2.16. In this manner an n -dimensional attribute valued decision table S is transformed into a $3n$ -dimensional attribute valued decision table S . The following steps are applied on the decision table S to determine the initial connection weights of FRGNN2.

- (S1) Develop the fuzzy reflexive relation matrix using Eq. 2.43, corresponding to every conditional attribute, where row in the matrix is a granulation structure.
- (S2) Use Steps 2 and 3 in Sect. 2.9.1.
- (S3) Calculate dependency degree using Eq. 2.38 for each conditional attribute with respect to the decision classes. The resulting values are determined as initial weights between nodes in the input layer and the hidden layer.
- (S4) At first, calculate the average of all the dependency degrees of all the conditional attributes with respect to every decision class. Then, divide the resulting average values by the number of nodes of the output layer, representing the number of classes, and assign those values between the nodes of the hidden and the output layers of the network as its initial connection weights.

Now we explain the knowledge encoding procedure of FRGNN2. Let γ_j^l denote the dependency degree of j th conditional attribute in decision table S where the number of decision classes in the decision table is c . The weight w_{lj} between the input node j and the hidden node l (see Fig. 2.2) is defined as

$$\gamma_j^l = \frac{\sum_{x \in A_l} POS_j(x)}{|A_l|}, \quad l = 1, 2, \dots, c, \quad (2.44)$$

where A_l represents l th concept corresponding to decision class d_l .

Let β_l denote the average dependency degree of all the conditional attributes with respect to l th decision class. It is computed as

$$\beta_l = \frac{\sum_{j=1}^{3n} \gamma_j^l}{|3n|}, \quad l = 1, 2, \dots, c. \quad (2.45)$$

The weight w_{kl} between node l in hidden layer and node k in the output layer is defined as $\frac{\beta_l}{|l|}$.

2.9.2 Examples for Knowledge Encoding Procedure

We explain the knowledge encoding procedure of FRGNN1 and FRGNN2 in crisp case (Method I) and fuzzy case (Method II) using data [7] in Table 2.1.

Example of FRGNN1 in crisp case:

The knowledge encoding procedure of FRGNN1 in crisp case (Method I) is explained as follows. Each conditional attribute in Table 2.1 is transformed into 3D attributes corresponding to *low*, *medium* and *high*. The 3D conditional attributes and the crisp decision classes are shown in Table 2.2.

We apply step S1 of Sect. 2.9.1 to Table 2.2. The fuzzy similarity matrix for a conditional attribute L_1 is provided as an example.

Table 2.1 Dataset

U	a	b	c	d
x_1	−0.4	−0.3	−0.5	1
x_2	−0.4	0.2	−0.1	2
x_3	−0.3	−0.4	0.3	1
x_4	0.3	−0.3	0	2
x_5	0.2	−0.3	0	2
x_6	0.2	0	0	1

Table 2.2 Decision table with 3-dimensional conditional attributes and decision attribute in crisp case

U	L_1	M_1	H_1	L_2	M_2	H_2	L_3	M_3	H_3	d
x_1	0.875	0.3950	0	0.9296	0.9243	0	0.125	0.3472	0	1
x_2	0.875	0.3950	0	0	0.2608	0.125	0	0.9861	0.3828	2
x_3	0.5	0.6975	0	0.3828	0.7391	0	0.125	0.875	0	1
x_4	0	0.3024	0.5	0.9296	0.9243	0	0	0.875	0.9296	2
x_5	0	0.6049	0.875	0.9296	0.9243	0	0	0.875	0.9296	2
x_6	0	0.6049	0.875	0	0.8132	0.125	0	0.875	0.9296	1

$$R_{L_1}(x, y) = \begin{pmatrix} 1 & 1 & 0.555 & 0 & 0 & 0 \\ 1 & 1 & 0.555 & 0 & 0 & 0 \\ 0.555 & 0.555 & 1 & 0.407 & 0.407 & 0.407 \\ 0 & 0 & 0.407 & 1 & 1 & 1 \\ 0 & 0 & 0.407 & 1 & 1 & 1 \\ 0 & 0 & 0.407 & 1 & 1 & 1 \end{pmatrix}$$

Similarly, the similarity matrices for the attributes $R_{M_1}(x, y)$, $R_{H_1}(x, y)$, $R_{L_2}(x, y)$, $R_{M_2}(x, y)$, $R_{H_2}(x, y)$, $R_{L_3}(x, y)$, $R_{M_3}(x, y)$, $R_{H_3}(x, y)$ are also determined. We then calculate the lower approximations, using step S2, for the concepts $A_1 = \{1, 3, 6\}$ and $A_2 = \{2, 4, 5\}$ for every conditional attribute with respect to the decision classes d_1 and d_2 under decision attribute column d where y belongs to A_1 & A_2 and x belongs to U .

We now apply step S2 in Sect. 2.9.1 to the concept A_1 ,

$$(R_{L_1} \downarrow R_d x)(y) = \inf_{x \in U} I\{R_{L_1}(x, y), R_d(x, y)\}.$$

For object x_1 , this is

$$\begin{aligned} (R_{L_1} \downarrow R_d x)(x_1) &= \inf_{x \in U} I\{R_{L_1}(x, 1), R_d(x, 1)\}, \\ &= \min \{I(1, 1), I(1, 0), I(0.555, 1), I(0, 0), I(0, 0), I(0, 1)\}, \\ &= 0.0. \end{aligned}$$

The lower approximation values for objects x_3 and x_6 are 0.444 and 0.0, respectively.

We also apply step S2 in Sect. 2.9.1 to the concept A_2 ,

$$(R_{L_1} \downarrow R_d x)(x_2) = 0.0, (R_{L_1} \downarrow R_d x)(x_4) = 0.0, (R_{L_1} \downarrow R_d x)(x_5) = 0.0.$$

The dependency degree of a conditional attribute, $\gamma_{\{L_1\}}$, is 0.074. The dependency degrees for the remaining conditional attributes are defined as

$$\begin{aligned} \gamma_{\{M_1\}} &= 0.031, & \gamma_{\{H_1\}} &= 0.074, \\ \gamma_{\{L_2\}} &= 0.077, & \gamma_{\{M_2\}} &= 0.506, \\ \gamma_{\{H_2\}} &= 0.000, & \gamma_{\{L_3\}} &= 0.042, \\ \gamma_{\{M_3\}} &= 0.111, \text{ and } & \gamma_{\{H_3\}} &= 0.233. \end{aligned}$$

These values are considered as the initial connection weights from nine input layer nodes to one hidden layer node of the FRGNN1 corresponding to class 1. We define the connection weights between the hidden layer node and the output layer nodes as follows:

For a qualitative attribute $a \in \{d\}$, the lower approximations are equivalent to positive degrees of objects in the concepts A_1 and A_2 . The positive membership values of objects in the concept $A_1 = \{1, 3, 6\}$ corresponding to the attribute L_1 are

$$POS_{L_1}(x_1) = 0.0, POS_{L_1}(x_3) = 0.444, POS_{L_1}(x_6) = 0.0.$$

The dependency degree is

$$\gamma_{\{L_1\}}(A_1) = \frac{0.0 + 0.444 + 0.0}{x_3} = 0.148.$$

The positive membership values of objects for the attribute L_1 are

$$POS_{L_1}(x_2) = 0.0, POS_{L_1}(x_4) = 0, POS_{L_1}(x_5) = 0.0.$$

The dependency degree is

$$\gamma_{\{L_1\}}(A_2) = \frac{0.0 + 0.0 + 0.0}{3} = 0.0.$$

The dependency degrees for a typical attribute M_1 , with respect to the concepts A_1 and A_2 , are given as follows:

$$\gamma_{\{M_1\}}(A_1) = 0.031, \quad \gamma_{\{M_1\}}(A_2) = 0.031.$$

The dependency values for the remaining attributes are also computed and the average dependency degrees of all the conditional attributes with respect to the decision classes are characterized by $\gamma(A_1) = 0.113$ and $\gamma(A_2) = 0.060$. The connection weights between the hidden layer node and two output layer nodes that correspond to classes 1 and 2 are initialized with 0.113 & 0.060, respectively.

Example of FRGNN1 in fuzzy case:

Table 2.3 provides 3D conditional attributes and fuzzy decision classes, obtained using Eqs. 2.30 and 2.31, under the decision attribute columns D_{kk} and D_{ku} , respectively.

We then calculate the lower approximations, using step S2, of the concepts $A_1 = \{x_1, x_3, x_6\}$ and $A_2 = \{x_2, x_4, x_5\}$ for every conditional attribute with respect to the fuzzy decision classes under the columns of fuzzy decision attributes D_{kk} and D_{ku} where y belongs to A_1 or A_2 and x belongs to U .

For the concept $A_1 = \{1, 3, 6\}$,

$$(R_{L_1} \downarrow R_d x)(y) = \inf_{x \in U} I\{R_{L_1}(x, y), R_d(x, y)\}.$$

Table 2.3 Decision table with 3D conditional attributes and decision attributes in fuzzy case

U	L_1	M_1	H_1	L_2	M_2	H_2	L_3	M_3	H_3	d	D_{kk}	D_{ku}
x_1	0.875	0.395	0	0.929	0.924	0	0.125	0.347	0	1	0.257	0.180
x_3	0.5	0.697	0	0.382	0.739	0	0.125	0.875	0	1	0.257	0.180
x_6	0	0.604	0.875	0	0.813	0.125	0	0.875	0.929	1	0.257	0.180
x_2	0.875	0.395	0	0	0.260	0.125	0	0.986	0.382	2	0.276	0.084
x_4	0	0.302	0.5	0.929	0.924	0	0	0.875	0.929	2	0.276	0.084
x_5	0	0.604	0.875	0.929	0.924	0	0	0.875	0.929	2	0.276	0.084

For object x_1 , this is

$$\begin{aligned}(R_{L_1} \downarrow R_d x)(x_1) &= \inf_{x \in U} I\{R_{L_1}(x, x_1), R_d(x, x_1)\}, \\ &= \min \{I(1, 0.257), I(1, 0.180), I(0.555, 0.257), \\ &\quad I(0, 0.180), I(0, 0.180), I(0, 0.257)\}, \\ &= 0.180.\end{aligned}$$

The dependency values for object x_3 and x_6 are obtained as 0.257 and 0.180, respectively. For the concept $A_2 = \{x_2, x_4, x_5\}$,

$$(R_{L_1} \downarrow R_d x)(x_2) = 0.084, (R_{L_1} \downarrow R_d x)(x_4) = 0.084, (R_{L_1} \downarrow R_d x)(x_5) = 0.084.$$

The dependency degree of the conditional attribute L_1 is $\gamma_{\{L_1\}} = 0.145$. Similarly, the dependency degrees for the remaining attributes are calculated. The resulting dependency degrees are used as the initial connection weights between the nodes in input layer and the hidden layer node of the FRGNN1 that corresponds to class 1. Similarly, one can determine from Table 2.3 the initial connection weights between the hidden layer node and the output layer nodes of FRGNN1. The resulting average dependency degrees of all the conditional attributes with respect to the decision classes d_1 and d_2 are characterized by $\gamma(A_1) = 0.209$ and $\gamma(A_2) = 0.113$. These values are initialized between one node in hidden layer and two nodes in output layer that correspond to classes 1 and 2.

So far we have discussed the procedure for determining the connection weights corresponding to one decision table. If the similar procedure is applied to the other decision table in both crisp and fuzzy cases then a fully connected FRGNN1 with initial weight parameters would be generated in both the cases.

Example of FRGNN2 in crisp case:

We use the data in Table 2.1 to explain the knowledge encoding procedure of FRGNN2. The 3D conditional attributes and the crisp decision classes under decision attribute columns are shown in Table 2.2. Step S1 of Sect. 2.9.1 is applied to Table 2.2 to define the fuzzy reflexive relational matrices for every conditional attribute. The fuzzy reflexive relational matrix for the attribute L_1 , $R_{L_1}(x, y)$ for x and y in U , is provided as an example.

$$R_{L_1}(x, y) = \begin{pmatrix} 1 & 0.938 & 0.184 & 1 & 0.194 & 0.194 \\ 0.938 & 1 & 0.246 & 0.938 & 0.256 & 0.256 \\ 0.184 & 0.246 & 1 & 0.194 & 1 & 1 \\ 1 & 0.938 & 0.184 & 1 & 0.194 & 0.194 \\ 0.184 & 0.246 & 1 & 0.194 & 1 & 1 \\ 0.184 & 0.246 & 1 & 0.194 & 1 & 1 \end{pmatrix}$$

Similarly, the fuzzy reflexive relational matrices $R_{M_1}(x, y)$, $R_{H_1}(x, y)$, $R_{L_2}(x, y)$, $R_{M_2}(x, y)$, $R_{H_2}(x, y)$, $R_{L_3}(x, y)$, $R_{M_3}(x, y)$, and $R_{H_3}(x, y)$ are also determined. The membership values of the patterns belonging to the lower approximations of the concepts $A_1 = \{x_1, x_3, x_6\}$ and $A_2 = \{x_2, x_4, x_5\}$ are recomputed using step S2 in Sect. 2.9.1.

Here, the lower approximations are equal to the positive membership values. The dependency values of the attributes, based on the positive membership values, with respect to the decision classes, are computed using step S3 in Sect. 2.9.1. The dependency values of attribute L_1 with respect to class1, $\gamma_{\{L_1\}}(A_1)$, and with respect to class2, $\gamma_{\{L_1\}}(A_2)$, are given as examples.

$$\gamma_{\{L_1\}}(A_1) = 0.106, \quad \gamma_{\{L_1\}}(A_2) = 0.$$

The weights between nine input nodes that correspond to nine attributes ($L_1, M_1, H_1, L_2, M_2, H_2, L_3, M_3$ and H_3) and one hidden node are initialized with the dependency values with respect to class 1 (0.106, 0.030, 0.0, 0.107, 0.089, 0.0, 0.083, 0.175 and 0.239), and other hidden node are initialized with the dependency values with respect to class2 (0, 0.030, 0.106, 0.0, 0.158, 0.0, 0.0, 0.035 and 0.107).

The average values of the dependency values with respect to classes 1 & 2 are 0.092 & 0.048, respectively. The weights between two hidden nodes & one output node that corresponds to class 1 are initialized with 0.046 & 0.046 (= 0.092/2), and the other output node that corresponds to class 2 are initialized with 0.024 & 0.024 (0.048/2). The procedure for determining the connection weights of FRGNN2 in fuzzy case (Method II) are also determined in similar way to that of weights in crisp case (Method I).

2.10 Experimental Results

The FRGNN1 and FRGNN2 algorithms are implemented in C-language. The characteristics of several real life data sets are provided in Table 2.4. The first four data sets in the table are used for FRGNN1 and the rest in the table are used for FRGNN2. The data sets, except Telugu vowel, are collected from the UCI Machine Learning

Table 2.4 Characteristics of the data sets for FRGNN1 and FRGNN2

Dataset name	Num of patterns	Features	Classes
<i>Data used for FRGNN1</i>			
Telugu vowel	871	3	6
Sonar	208	60	2
Lymphography	148	18	4
Thyroid	215	5	3
<i>Data used for FRGNN2</i>			
Telugu vowel	871	3	6
Image segmentation	2310	19	7
Sonar	208	60	2
Spam base	4601	57	2

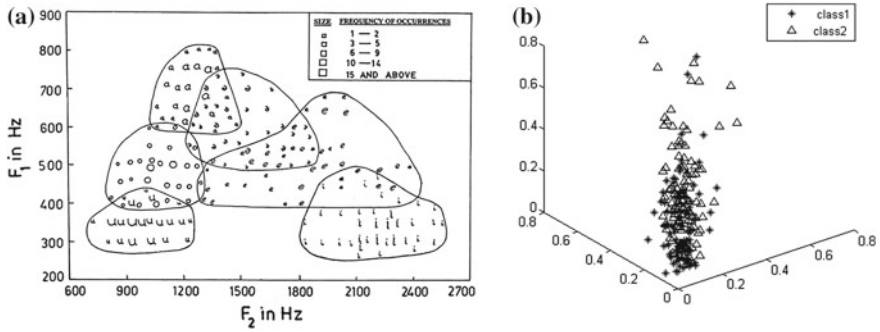


Fig. 2.5 **a** Vowel data in the F_1 - F_2 plane. **b** Sonar data in the F_1 - F_2 - F_3 space

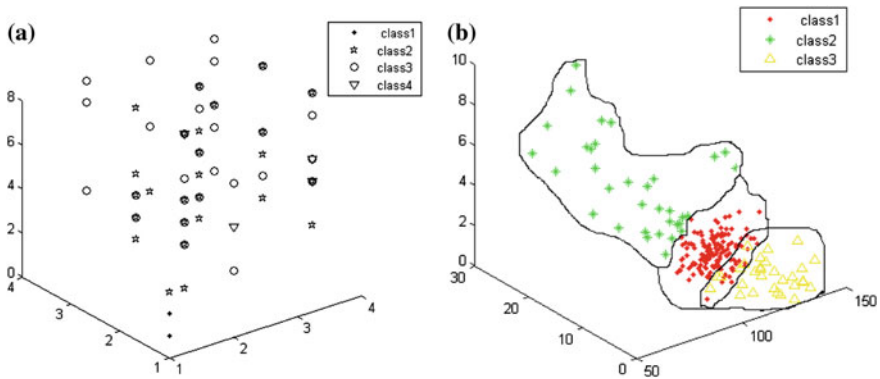


Fig. 2.6 **a** Lymphography data in the F_1 - F_2 - F_3 space. **b** Thyroid data in the F_1 - F_2 - F_3 space

Repository [11]. Figure 2.5a shows a two-dimensional (2D) projection of the 3D feature space of the six vowel classes in the F_1 - F_2 plane. Figure 2.5b shows a 3D projection of the 60D feature space of two classes of sonar data in the F_1 - F_2 - F_3 space. Figure 2.6a shows a 3D projection of the 18D feature space of the four classes of lymphography data in the F_1 - F_2 - F_3 space. Figure 2.6b shows a 3D projection of the 5D feature space of the three classes of thyroid data in the F_1 - F_2 - F_3 space.

Results of FRGNN1

During learning of FRGNN1, we have used an v -fold cross validation design with stratified sampling. The training is done on $(v - 1)$ folds of the data selected randomly from each of the classes. The remaining one fold data is considered as the test set. This is repeated v times, and the overall performance of the FRGNN1 is computed taking an average over v sets of folds. In the experiment, we considered $v = 10$ or 9 based on the size of the data sets. The parameters in Eq. 2.20 were chosen as $f_d = 6$ and $f_e = 1$ for all the data sets. The momentum parameter α , learning rate η , and the bias b of the FRGNN1 traversed a range of values between 0 and 1, and had different values depending on the folds used for learning. For example, in case

of the vowel data, the appropriate values were $\alpha = 0.98$, $\eta = 0.0858$, $b = 0.00958$ in crisp case, and $\alpha = 0.958$, $\eta = 0.06558$, $b = 0.0958$ in fuzzy case. It is observed that the FRGNN1 converges to a local minimum at the 1500th epoch for vowel data.

Before providing the performance of the FRGNN1 on the test sets for all the data, we explain the implementation process for its training on Telugu vowel data, as an example. Here, the FRGNN1 has nine nodes in the input layer, and six nodes in each of hidden and output layers. The data is first transformed into a 3D granular space using Eq. 2.16. The appropriate values of the center C and scaling factor λ for each feature of granules, e.g., *low*, *medium* or *high*, are determined as $C_{low_1} = 368.932$, $C_{medium_1} = 470.482$, $C_{high_1} = 583.616$, $C_{low_2} = 1110.323$, $C_{medium_2} = 1514.684$, $C_{high_2} = 2047.021$, $C_{low_3} = 2359.322$, $C_{medium_3} = 2561.021$, $C_{high_3} = 2755.891$, and $\lambda_{low} = 586.666$, $\lambda_{medium} = 1300.000$, $\lambda_{high} = 666.666$. Then, the 3-dimensional patterns have numerical components which are mapped into a 9D granular space with components $L_1, M_1, H_1, L_2, M_2, H_2, L_3, M_3, H_3$, while the desired vector has components $d_1, d_2, d_3, d_4, d_5, d_6$ corresponding to the six vowel classes. Table 2.5 provides some examples of a granular input vector $\vec{F}_i$, and the desired output vector d_k , $k = 1, 2, \dots, 6$, for a set of sample patterns.

Knowledge extraction procedure for Telugu vowel data:

Let the decision table $S = (U, \mathcal{A} \cup \{d\})$ be used to represent the entire training data set. The decision table S is then divided into six decision tables corresponding to six vowel classes, namely, $S_l = (U_l, \mathcal{A} \cup \{d\})$. Every S_l , $l = 1, 2, \dots, 6$, need not be of the same size. Each S_l is transformed into 3D granular space using Eq. 2.5. We apply the knowledge encoding procedure (which was explained before to the data in Table 2.1) for each decision table S_l , and the resulting domain knowledge is then encoded into the FRGNN1 in the form of initial connection weights. Table 2.6 provides the initial connection weights of the FRGNN1 in fuzzy case (see Method II of Sect. 2.8). Values of the fuzzifiers f_d and f_e in Eq. 2.20 were considered to be 1. The complete FRGNN1 architecture thus generated for the vowel data is shown in Fig. 2.7. The network is trained for refining the initial weight parameters in the presence of training samples. The trained network is then used to classify the set of test patterns.

Note that, if the number of training patterns in any class is less than the number of classes, then the average dependency factors of all the conditional attributes with respect to a decision class may not possible to define. To avoid this, we include some training patterns with zero attribute value to that particular class so as to make the number of training patterns in that class equal to the number of classes.

Performance of the FRGNN1 on Test Sets:

The experimental results of the FRGNN1 for Telugu vowel, sonar, lymphography and thyroid are shown in Table 2.7. Results correspond to three types of the initial weights of the FRGNN1. These are (i) random numbers in $[-0.5, 0.5]$, (ii) crisp case (Method 1 of Sect. 2.8) and (iii) fuzzy case (Method 2 of Sect. 2.8). One may note that considering the random initial weights (i.e., type (i)) makes the FRGNN1 equivalent to a fuzzy MLP [15].

Table 2.5 Examples of input and target vectors for FRGNN1. A pattern with three input features is mapped into $3 \times 3D$ granular space with components $L_1, M_1, H_1, L_2, M_2, H_2, L_3, M_3, H_3$, while the desired vector has components $d_1, d_2, d_3, d_4, d_5, d_6$ corresponding to the 6 vowel classes

Input features			Input vector									Desired vector					
F_1	F_2	F_3	L_1	M_1	H_1	L_2	M_2	H_2	L_3	M_3	H_3	d_1	d_2	d_3	d_4	d_5	d_6
700	1500	2600	0.37	0.94	0.93	0.22	0.99	0.06	0.66	0.99	0.89	0.9	0.0	0.0	0.0	0.0	0.0
650	1200	2500	0.54	0.96	0.98	0.95	0.88	0.00	0.88	0.99	0.70	0.0	0.9	0.0	0.0	0.0	0.0
300	2100	2600	0.97	0.96	0.63	0.00	0.59	0.98	0.66	0.99	0.89	0.0	0.0	0.9	0.0	0.0	0.0
400	1150	2500	0.99	0.99	0.84	0.99	0.84	0.00	0.88	0.99	0.70	0.0	0.0	0.0	0.9	0.0	0.0
500	2000	2750	0.90	0.99	0.96	0.00	0.72	0.99	0.22	0.95	0.99	0.0	0.0	0.0	0.0	0.9	0.0
400	900	2700	0.99	0.99	0.84	0.74	0.55	0.00	0.35	0.97	0.98	0.0	0.0	0.0	0.0	0.0	0.9
550	1800	2600	0.01	0.97	0.98	0.03	0.95	0.87	0.79	0.99	0.86	0.98	0.0	0.0	0.0	0.0	0.0
600	1400	2600	0	0.92	0.99	0.73	0.99	0.24	0.79	0.99	0.86	0.99	0.0	0.0	0.0	0.0	0.0

Table 2.6 Initial connection weights of FRGNN1 for Telugu vowel data in fuzzy case

Input to hidden layer (w_{ij})							Hidden to output layer (w_{kl})						
0.0747	0.0832	0.0759	0.0849	0.0937	0.0892	0.0936	0.0648	0.0362	0.0166	0.1588	0.07930		
0.0634	0.06202	0.0661	0.0713	0.0761	0.0757	0.0970	0.0615	0.0472	0.0082	0.1616	0.0675		
0.0642	0.0655	0.0741	0.0868	0.0737	0.0798	0.1214	0.0881	0.0477	0.0237	0.1601	0.0904		
0.1117	0.0948	0.1024	0.1079	0.1104	0.0862	0.1347	0.1101	0.0603	0.0172	0.1636	0.0780		
0.0778	0.0893	0.0981	0.1114	0.1232	0.0962	0.1294	0.0813	0.0532	0.0197	0.1629	0.0836		
0.1044	0.0890	0.1286	0.1106	0.1047	0.1266	0.1371	0.095	0.0463	0.0179	0.1663	0.0844		
0.0681	0.0768	0.0868	0.0899	0.0763	0.0925								
0.0629	0.0620	0.0824	0.0697	0.0695	0.0845								
0.0768	0.0713	0.0870	0.0941	0.0765	0.0855								

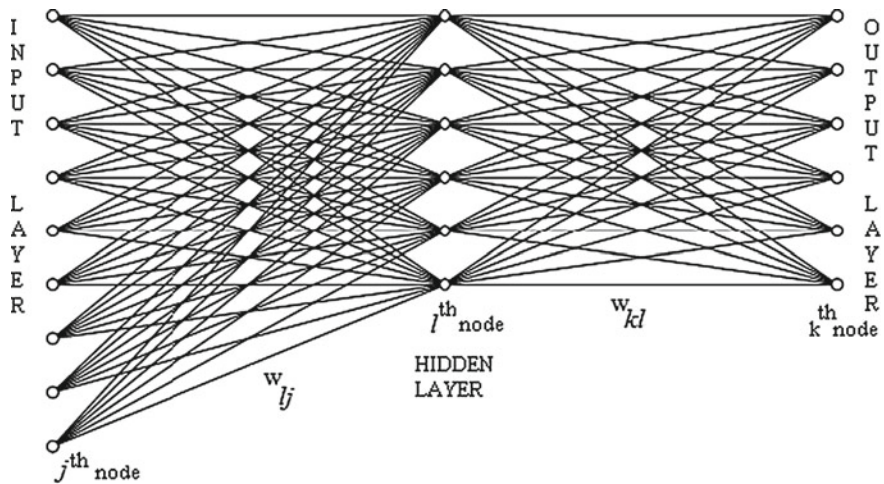


Fig. 2.7 The architecture of FRGNN1 for Telugu vowel data

The performance of the FRGNN1 is seen to vary over different folds. For example, in the 10-fold cross validation, for Telugu vowel data with initial weights in random case, crisp case and fuzzy case, the recognition scores of FRGNN1 are seen to vary between 79.01% (minimum) and 88.51% (maximum) with an average accuracy of 84.75%; 81.61% (minimum) and 90.80% (maximum) with an average accuracy of 86.55%; and 85.06% (minimum) and 94.25% (maximum) with an average accuracy of 87.71%, respectively. Similar type of recognition scores of FRGNN1 for remaining three data sets can also be found.

The generalization ability of the FRGNN1 is seen to depend on the characteristics of the data. If the correlation among input variables is high, the generalization ability is not so high and vice versa. For example, the generalization ability of the FRGNN1 for Telugu vowel, sonar and lymphography is below 90% because of the high correlation in the feature space of input vectors whereas it is above 90% for thyroid.

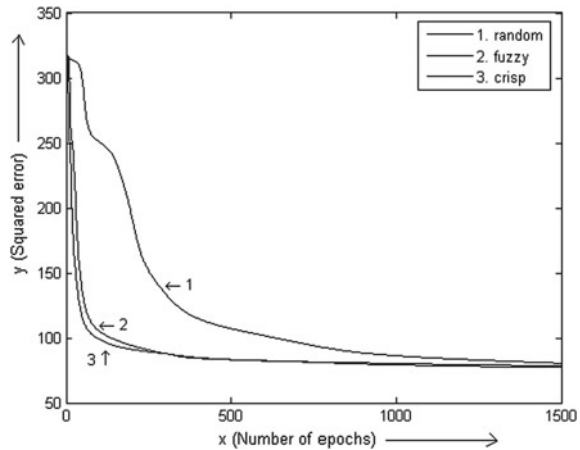
From Table 2.7, we can conclude that the performance of the FRGNN1 with initial connection weights corresponding to both fuzzy and crisp cases is superior to fuzzy MLP (i.e., FRGNN1 with initial weights corresponding to the random case) for all the data sets. Weight selection by Method II of Sect. 2.8 (fuzzy case) results in better performance than Method I of Sect. 2.8 (crisp case). Further, their difference is more apparent for overlapping classes of lymphography data (see Fig. 2.6a) where the difference is seen to be more than 3%.

Figure 2.8 presents the variation of the squared error with the number of epochs carried out during training of the FRGNN1 with initial weights in random, crisp, and fuzzy cases. This comparative result is shown, as an example, only for Telugu vowel data. Here, the results correspond to one fold over ten folds. As expected, the squared error decreases with the number of epochs. The error drops significantly

Table 2.7 Experimental results of FRGINNI

Dataset name	Initial weights case	Accuracy of fold										Average accuracy
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	
Telugu vowel	Random	82.76	85.06	83.91	85.06	85.06	86.21	86.21	79.01	88.51	85.06	84.75
	Crisp	85.06	85.06	90.8	87.36	87.36	86.21	85.06	81.61	88.51	88.51	86.55
	Fuzzy	86.21	86.21	94.25	89.66	86.21	86.21	86.21	85.06	88.51	88.51	87.71
Sonar	Random	86.96	86.96	73.91	86.96	91.3	82.61	82.61	86.96	86.96	–	85.03
	Crisp	91.3	91.3	73.91	86.96	91.3	82.61	82.61	86.96	91.3	–	86.47
	Fuzzy	95.65	91.3	78.26	86.96	100	86.96	78.26	86.96	95.65	–	88.89
Lymphography	Random	87.5	81.25	75	75	81.25	81.25	75	87.5	75	–	79.86
	Crisp	87.5	87.5	75.00	75.00	87.5	81.25	81.25	93.75	81.25	–	83.33
	Fuzzy	100	87.5	75	81.25	93.75	81.25	81.25	100	81.25	–	86.81
Thyroid	Random	95.45	95.45	95.45	95.24	95.45	86.36	90.48	100.00	100.00	100.00	95.39
	Crisp	95.45	95.45	95.45	95.24	95.45	90.48	95.24	100.00	100.00	100.00	96.28
	Fuzzy	95.45	95.45	95.45	95.24	95.45	90.48	95.24	100.00	100.00	100.00	96.28

Fig. 2.8 Comparison of squared errors of the FRGNN1 with initial weights in 1 Random case, 2 Crisp case, and 3 Fuzzy case for Telugu vowel data for various epochs



at lower number of epochs for the crisp and fuzzy cases than the random case of weight selection, because the former two cases enable the FRGNN1 to start learning from a much better position than the latter case. Among the crisp and fuzzy cases, although the crisp one is seen to provide slightly better results for the particular fold of vowel data (considered in Fig. 2.8), in overall performance, the fuzzy case of weight selection is the best for all the data sets.

Comparison of FRGNN1 with rough fuzzy MLP:

In order to compare the performance of FRGNN1 against an existing well known network of a rough fuzzy MLP (rfMLP) [1], we consider Telugu vowel data, as an example.

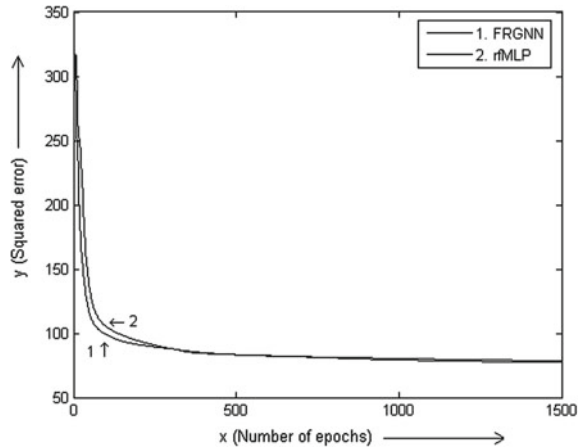
Table 2.8 depicts the performance of the rough fuzzy MLP, with one hidden layer as in the FRGNN1 with weights in crisp and fuzzy cases, for classification of vowels after 1500 epochs for a typical set of 10 folds. Parameters of the membership function (f_d and f_e) were assigned to the same values as in the FRGNN1 for the purpose of fair comparison. The values of learning rate (η), momentum parameter (α), and bias b are chosen by trail and error method. In rfMLP, we have obtained one reduct for each class representing its decision rule; thereby generating six decision rules for six vowel classes. By comparing the results of FRGNN1 with weights in both crisp case and fuzzy case, we can say that the performance of the FRGNN1 is better than rough fuzzy MLP.

In Fig. 2.9, we present the comparison of squared errors of the FRGNN1 (with initial weights in fuzzy case) and the rough fuzzy MLP for one of the 10-folds, considered in Table 2.8, for which both the FRGNN1 and rough fuzzy MLP have given the best classification accuracy within their respective sets of folds for vowel data. In the FRGNN1, the number of nodes in the hidden layer is equal to the number of classes which is six in case of vowel data. Interestingly, the same number of nodes was also obtained automatically in rough fuzzy MLP using its rough dependency factors. Again, the minimum values in both the cases were reached at 1500th epoch.

Table 2.8 Comparison of FRGNN1 and rough fuzzy MLP for Telugu vowel data

Dataset name	Initial weights case	Accuracy of fold										Average accuracy
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	
Telugu vowel	FRGNN1											
	crisp case	85.06	85.06	90.8	87.36	87.36	86.21	85.06	81.61	88.51	88.51	86.55
	FRGNN1											
	fuzzy case	86.21	86.21	94.25	89.66	86.21	86.21	86.21	85.06	88.51	88.51	87.71
	rFMLP	81.61	83.91	87.36	88.51	87.36	87.36	87.36	83.91	88.51	89.66	86.55

Fig. 2.9 Comparison of squared errors of the FRGNN1 and rough fuzzy MLP for various epochs



Considering the variation of average accuracy rate, Fig. 2.9 further supports the earlier findings concerning the superiority of FRGNN1 over rough fuzzy MLP for Telugu vowel data.

Salient Points of Difference between the FRGNN1 and rough fuzzy MLP:

In the rough fuzzy MLP, rough sets are integrated with a fuzzy MLP for encoding domain knowledge in network parameters whereas it is on fuzzy rough sets which are integrated with a fuzzy MLP in FRGNN1.

As stated in Method of rough fuzzy MLP, a threshold value (Tr) is used to eliminate the noisy patterns from input data. In contrast, no such threshold value is required in the FRGNN1 and the problem of removing noisy patterns is resolved by incorporating the concept of fuzzy rough sets in determining initial weights of the network.

In the rough fuzzy MLP, the decision rules are generated for each reduct of the reduct set and the dependency factors of these rules are mapped into the connection weights of the network. In contrast, no such attribute reducts are generated in the FRGNN1 and the dependency factors of all the conditional attributes and the average value of all the dependency factors of all conditional attributes, with respect to the decision classes, are defined in the form of initial connection weights of the network.

In the rough fuzzy MLP, network architecture is modeled for each reduct of the reduct set and every such network gives a recognition score for the test set. The maximum recognition score computed over them is then considered as the performance measure of the rough fuzzy for the concerned test set. In contrast, only one network architecture is modeled in the FRGNN1 corresponding to the entire training data, and the network is seen to perform better than the rough fuzzy MLP for the same test set.

Results of FRGNN2:

We perform 10-fold cross validation with stratified sampling. The FRGNN2 is trained using 9-folds of the data selected randomly from each of the classes and remaining one fold data is considered as the test set. This is repeated for 10 times

Table 2.9 Results of FRGNN2

Dataset name	Initial weights case	Accuracy of fold										Average accuracy
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	
Image segmentation	Random	85.71	85.71	76.19	76.19	76.19	80.95	90.48	90.48	80.95	85.71	82.99
	Crisp	85.71	85.71	80.95	90.48	76.19	80.95	90.48	85.71	90.48	71.43	83.81
	Fuzzy	90.48	85.71	85.71	85.71	85.71	85.71	90.48	90.48	85.71	85.71	87.14
Sonar	Random	82.35	88.24	76.47	82.35	70.59	100	64.71	82.35	82.35	76.47	80.59
	Crisp	82.35	94.12	76.47	82.35	70.59	100	64.71	82.35	82.35	76.47	81.18
	Fuzzy	82.35	94.12	76.47	82.35	70.59	100	70.59	82.35	82.35	82.35	82.35
Spam base	Random	86.27	87.36	87.8	87.36	88.02	85.62	87.36	85.84	89.98	86.27	87.19
	Crisp	89.98	90.85	87.36	90.85	91.07	87.58	88.45	87.80	90.85	88.02	89.28
	Fuzzy	89.98	91.29	89.76	90.41	91.94	89.32	90.63	85.62	90.85	88.02	89.78

Table 2.10 Comparison of FRGNN2 and rough fuzzy MLP for Telugu vowel data

Dataset name	Initial weights case	Accuracy of fold										Average accuracy
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	
Telugu vowel	FRGNN2											
	fuzzy case	88.24	85.88	89.41	88.24	83.53	84.71	85.88	85.88	88.24	87.06	86.71
	rfMLP	81.61	83.91	87.36	88.51	87.36	87.36	87.36	83.91	88.51	89.66	86.55

such that the entire data is employed in training and testing. The accuracies of 10-test sets and their average accuracy are provided in Table 2.9. The results of FRGNN2, as compared to FRGNN2 with weights in crisp case and fuzzy MLP [15] (FRGNN2 with weights chosen randomly within -0.5 to 0.5), are provided in Table 2.9.

The results from Table 2.9 show that FRGNN2 with weights in fuzzy case has achieved the best average classification accuracies for all data sets.

The comparative results of Telugu vowel data from (Table 2.10) reveal that the performance of FRGNN2 is superior to rfMLP. When the results of FRGNN2 are compared to FRGNN1 in fuzzy case (Table 2.8), the performance of FRGNN2 is inferior to FRGNN1. Therefore, it can be concluded that the FRGNN1 with weights in fuzzy case performs better than FRGNN2 and rfMLP when they are used for classification of patterns arising in the overlapping regions between the classes.

2.11 Conclusion

This chapter dealt with the problems of classification (supervised) in the framework of granular neural networks. Two models, namely FRGNN1 and FRGNN2, are described in detail. The concept of granulation is incorporated in both the input level and the initial connection weights of the network. The networks accept input in terms of fuzzy granules *low*, *medium* and *high*, and provides output decisions in terms of class membership values and zeros. They use the dependency factors of the conditional attributes, instead of random numeric connection weights, as the initial connection weights. Here, the dependency factors are determined using the concepts of fuzzy rough set. This investigation not only demonstrates a way of integrating fuzzy rough sets with a fuzzy neural network, but also provides a methodology that is capable of generating granular neural networks and improving their performance. The fuzzy rough sets provide an advantage that by which a discrete real-valued noise data can be effectively reduced without the need for any user supplied information (thresholds).

Two special types of granular computations are examined. One of them is induced by *low*, *medium* and *high* fuzzy granules and the other has classes of granulation structures induced by a set of fuzzy equivalence granules based on the fuzzy similarity relation/fuzzy reflexive relation. With respect to the classes of granulation structures, stratified fuzzy rough set approximations are obtained to determine the dependency factors of all conditional attributes to obtain the initial weights of the FRGNN1 and FRGNN2. The incorporation of granularity at various levels of the conventional MLP helps the resulting FRGNN1 and FRGNN2 to efficiently handle uncertain and ambiguous input information. This is demonstrated with extensive experimental results on a several real life data sets with varying dimension and size. The performance of FRGNN1 is found to be superior to rough fuzzy MLP which uses rough sets, rather than fuzzy rough sets, for knowledge encoding. Although, the comparative result is shown only for vowel data, the same observation holds for all other data sets considered in the experiment. The FRGNN1 and FRGNN2 mod-

els reflect the useful applications of granular computing to real world classification problems. Though their effectiveness has been demonstrated here on speech data, sonar data and medical data, these can be made well applicable to other real life problems such as gene expression analysis, web mining, social networks analysis, image and video analysis. Further, the information granules characterizing the fuzzy lower approximate regions of classes are used here for forming the basic networks. The effect of size and shape of the information granules on the network performances may therefore need to be investigated.

References

1. Banerjee, M., Mitra, S., Pal, S.K.: Rough fuzzy MLP: knowledge encoding and classification. *IEEE Trans. Neural Netw.* **9**(6), 1203–1216 (1998)
2. Cornelis, C., Jensen, R., Hurtado, G., Slezak, D.: Attribute selection with fuzzy decision reducts. *Inf. Sci.* **180**(2), 209–224 (2010)
3. Dick, S., Kandel, A.: Granular computing in neural networks. In: Pedrycz, W. (ed.) *Granular Computing: An Emerging Paradigm*, pp. 275–305. Physica Verlag, Heidelberg (2001)
4. Ganivada, A., Dutta, S., Pal, S.K.: Fuzzy rough granular neural networks, fuzzy granules, and classification. *Theor. Comput. Sci.* **412**, 5834–5853 (2011)
5. Ganivada, A., Pal, S.K.: A novel fuzzy rough granular neural network for classification. *Int. J. Comput. Intell. Syst.* **4**(5), 1042–1051 (2011)
6. Jang, J.S.R.: ANFIS: adaptive-network-based fuzzy inference systems. *IEEE Trans. Syst. Man Cybern.* **23**(3), 665–685 (1993)
7. Jensen, R., Shen, Q.: New approaches to fuzzy-rough feature selection. *IEEE Trans. Fuzzy Syst.* **17**(4), 824–838 (2009)
8. Klir, G.J., Folger, T.: *Fuzzy Sets, Uncertainty and Information*. Prentice Hall, N.J. (1988)
9. Mandal, D.P., Murthy, C.A., Pal, S.K.: Determining the shape of a pattern class from sampled points in R^2 . *Int. J. Gen. Syst.* **20**, 307–339 (1992)
10. Mitra, S., De, R.K., Pal, S.K.: Knowledge-based fuzzy MLP for classification and rule generation. *IEEE Trans. Neural Netw.* **8**(6), 1338–1350 (1997)
11. Newman, D.J., Hettich, S., C. L. Blake, C.J.M.: UCI repository of machine learning databases, irvine. Department of Information and Computer Science, University of California, CA (1998). <http://archive.ics.uci.edu/ml/>
12. Pal, S.K., Majumder, D.D.: Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Trans. Syst. Man Cybern.* **7**(8), 625–629 (1977)
13. Pal, S.K., Majumder, D.D.: *Fuzzy Mathematical Approach to Pattern Recognition*. Wiley, New York (1986)
14. Pal, S.K., Mandal, D.P.: Linguistic recognition system based on approximate reasoning. *Inf. Sci.* **61**, 135–161 (1992)
15. Pal, S.K., Mitra, S.: Multilayer perceptron, fuzzy sets, and classification. *IEEE Trans. Neural Netw.* **3**(5), 683–697 (1992)
16. Radzikowska, A.M., Kerre, E.E.: A comparative study of fuzzy rough sets. *Fuzzy Sets Syst.* **126**(2), 137–155 (2002)

Granular Neural Networks, Pattern Recognition and
Bioinformatics

Pal, S.K.; Ray, S.S.; Ganivada, A.

2017, XIX, 227 p. 54 illus., 31 illus. in color., Hardcover

ISBN: 978-3-319-57113-3