CHAPTER 2

METHODS OF DECISION MAKING
Chapter 2

METHODS OF DECISION MAKING

This chapter introduces the fundamental concepts, definitions and terminologies of the artificial intelligent (AI) methods of decision making namely decision tree, fuzzy logic, neural networks and genetic algorithms needed for later chapters.

2.1 Decision Tree - An Overview

Decision trees, generally defined, are trees whose internal nodes are tests (on input patterns) and whose leaf nodes are categories (of patterns). They are widely used in generic classification architectures of machine learning and data mining methods [80]-[81]. In the vast literature on decision trees, also known as classification trees, the two seminal papers that most cited are those by Breiman et al. [82] and Quinlan [83]. The former deals with in the field of statistical pattern recognition and describes a system, namely CART (Classification and Regression Trees); whereas, the latter tackles the experience gained by the machine learning experts and describes the ID3 algorithm. CART and ID3 are the discriminative learning algorithms working by recursive partitioning. Their basic ideas are the same: partition the sample space in a data-driven manner and represent the partition as a tree [84]. The major dissimilarity is that CART induces strictly the binary trees and uses re-sampling techniques for error estimation, whereas ID3 partitions according to attribute values [68]. ID3 is a powerful heuristic for generating crisp decision trees, which later developed into a new algorithm called C4.5 [83].

An important feature of DTs is their capability to break down a complex decision-making process into a collection of simpler decisions and thereby, providing an easily interpretable solution [80]. This divide and conquer approach of the DT chooses the attribute that best divides the input-output data into their classes and then classify the data according to the
information content of that attribute. In the tree structure of the DT, the leaves represent classifications and the branches represent conjunctions of features that lead to those classifications.

### 2.1.1 Design of a Decision Tree Classifier

Decision trees provide a powerful formalism for representing comprehensible and accurate classifiers [85]. A DT can be viewed as a partitioning of the instance space. Each partition represents a number of similar instances that belongs to the same class. The split points of ID3 [83], [86]-[87] are chosen according to the most information attributes of the data instances. The mathematical descriptions of decision tree generation based on the information measured by ID3 algorithm are as follows:

Let \( S \) contains a set of objects (instances) that belong to a mixture of classes and a test is chosen based on a single attribute that has one or more exhaustive outcomes \( \{ O_1, O_2, \ldots, O_K \} \), the target classes. The training set \( S \) is partitioned into subsets \( S_1, S_2, \ldots, S_K \), where \( S_j \) contains all the objects in \( S \) that have outcome \( O_j \) of the chosen set, where \( 1 \leq k \leq K \). Let \( O_j(S) \) is the set of data instances in \( S \) that are assigned to class \( O_j \). The information, measured in bits, contained in \( C_j(S) \) can be expressed as follows [87]:

\[
\text{Info}(p_j(S)) = - \log_2 \frac{|O_j(S)|}{|S|}
\]  

(2.1)

where \( 1 \leq j \leq K \) and \(|S|\) is the number of instances in \( S \). In general, the probability distribution for the data set \( S \) of training instances is the information conveyed by this distribution

\[
P(S) = \left( p_1(S) = \frac{|O_1(S)|}{|S|}, \ldots, p_K(S) = \frac{|O_K(S)|}{|S|} \right)
\]  

(2.2)

Therefore, entropy of \( S \), known as total entropy, is defined as

\[
\text{Entropy}(S) = - \sum_{j=1}^{K} p_j(S) \log_2(p_j(S))
\]  

(2.3)
where $\text{Entropy}(S)$ measures the average amount of information required to identify the classification accuracy of an instance of $S$ [88]. When the partition of the instances of $S$ is on the basis of the attribute $A$ into $K$ disjoint class regions $\{C_1, C_2, \ldots, C_k\}$, it can provide the information required for classification of the attribute with respect to the target classes. For this distribution, the expected information measurement can be found as the weighted sum over the subsets related to the value of the target attribute:

\[
\text{i.e., } \quad \text{Entropy}(A,S) = \sum_{i=1}^{K} \frac{|S_i|}{|S|} \text{Entropy}(S_i), \quad (2.4)
\]

where $S_i$, the subset of $S$, contains the elements in the $i^{th}$ class of the attribute $A$. The gain ratio [87] measures that how well a given attribute separates the training instances according to their target classification:

\[
\text{GainRatio}(A,S) = \frac{\text{Gain}(A,S)}{\text{SplitInfo}(A,S)}, \quad (2.5)
\]

According to Quinlan [83], $\text{Gain}$ or information gain is defined as the measure of information that is gained by partitioning $S$ in accordance with the attribute $A$, which is the expected reduction in entropy.

\[
\text{i.e., } \quad \text{Gain}(A,S) = \text{Entropy}(S) - \text{Entropy}(A,S), \quad (2.6)
\]

where, $\text{SplitInfo}(A,S) = -\sum_{i=1}^{K} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$ \quad (2.7)

Based on the gain ratio criterion, the ID3 algorithm selects the attribute that has the highest value of gain ratio as the superior node and assign it as the root node of the tree. The partitioning process is applied on the root node according to the attribute classes $\{C_1, C_2, \ldots, C_k\}$, and then find the root nodes corresponding to these classes by Eqs. (2.3)-(2.7). When this process is applied recursively to each node [89], the tree is grown in a fashion that starting from the most meaningful attribute and proceeding with other attributes to the lower nodes of the tree. In this top-down approach, the design of a decision tree classifier reduces to the following three tasks [90]:
Methods of Decision Making

(1) selection of a node splitting rule, (2) decision as to which nodes are terminal, and (3) assignment of each terminal node to a class label. Of these three tasks, the classification task is by far the easiest.

2.1.2 ID3 Algorithm Strategy

The algorithm of an ID3 decision tree (here, multi-decision tree) is as follows [91]:

```
function ID3
    Input: (R: a set of non-target attributes, 
            C: the target attribute, 
            S: a training set) returns a decision tree;
    begin
        If S is empty, return a single node with value Failure;
        If S consists of records all with the same value for the target attribute, 
            return a single leaf node with that value;
        Else Let A be the attribute that best classifies the training set among 
            attributes in R;
            Decision Tree attribute for Root node = A;
            Let \{v_j\}_{j=1,2,..,K} be the values of attribute A;
            Let \{S_j\}_{j=1,2,..,K} be the subsets of S consisting respectively of 
                records with value v_j for A;
            If S(v_j) is empty, then below this new branch add a leaf node 
                with label = most common target value in S;
            Else below this new branch add the subtree ID3(S(v_j), C, R-\{A\})
        Return Root node
    end
```

Those attributes that have considered higher in the tree cannot appear in the lower level. Thus, an attribute involves at most once along any path through the tree. A node becomes a leaf when either its samples come from a unique class or when all attributes are used on the path [84]. If not, the node represents an unresolved node which cannot give any precise information about the knowledge base.

2.1.3 Sample Decision Tree

An example of a DT is illustrated in this section using a subset of the Coronary Artery Disease (CAD) database, which will be explained in Chapter 3. Consider a set of 20 input-output instances (Table 2.1) with seven attributes for input and the output attribute gives the risk factor. The input-output data set given here is in fuzzy format represented by their
corresponding labels such as low = 1, medium = 2, high = 3 and critical = 4. Age (AGE), Smoking (SMK), Systolic BP (SBP), Diastolic BP (DBP), LDL Cholesterol (LDL), Total Cholesterol (TC) and Diabetes Mellitus (DM) are the input attributes. A particular instance from the sample data might be described as

\[(AGE = 3) \& (SMK = 1) \& (SBP = 3) \& (DBP = 2) \& (LDL = 3) \& (TC = 2) \& (DM = 2)\] (2.8)

<table>
<thead>
<tr>
<th>SI.No</th>
<th>AGE</th>
<th>SMK</th>
<th>SBP</th>
<th>DBP</th>
<th>LDL</th>
<th>TC</th>
<th>DM</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3.</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4.</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5.</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6.</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>7.</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>8.</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>9.</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>10.</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>11.</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>12.</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>13.</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>14.</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>15.</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>16.</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>17.</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>18.</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>19.</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>20.</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2.1 Sample Data Set

With this CAD database, the ID3 algorithm provides the values of entropy and gain ratios related to each attribute.

\[
\text{Entropy} = 1.4166; \quad \text{GainRatio (AGE)} = 0.1932;
\]
\[
\text{GainRatio (SMK)} = 0.1619; \quad \text{GainRatio (SBP)} = 0.2349;
\]
\[
\text{GainRatio (DBP)} = 0.1818; \quad \text{GainRatio (LDL)} = 0.2897;
\]
\[
\text{GainRatio (TC)} = 0.3077; \quad \text{GainRatio (DM)} = 0.2052
\]

The attribute TC provides more information than the other attributes and so it is selected as the root node for extending the decision tree. The process is
progressed by partitioning \( \tau_C \) and finding the superior nodes along each class. Finally, a DT (Fig. 2.1) is obtained with 14 leaf nodes, where the leaf nodes represent the decision outputs corresponding to the target output “risk factor”. From the figure, it is obvious that there are three “unresolved” nodes. That are,

If (\( \tau_C \) is 2) \( \land \) (AGE is 2) \( \land \) (SYS is 2) then (Risk is \( C_2 \) or \( C_3 \))

If (\( \tau_C \) is 3) \( \land \) (SYS is 2) \( \land \) (DIAS is 2) \( \land \) (DM is 2) then (Risk is \( C_2 \) or \( C_3 \))

If (\( \tau_C \) is 4) then (Risk is \( C_3 \) or \( C_4 \))

The notations \( C_1, C_2, C_3 \) and \( C_4 \) stand for the decision outputs, whereas the numbers \( \{1, 2, 3, 4\} \) indicate the classes of the input attributes. Generally, a decision tree with the smallest number of leaf-nodes is usually called optimum [92], a most desirable representation of a database.

![Sample Decision Tree](image)

As indicated in Fig. 2.1, the tree represents a set of rules, recording the test outcomes as antecedents and the leaf-nodes as the consequent. A decision tree generated from the input-output database have the following properties: (1) The totality of instances \( \{e_1, e_2, \ldots, e_N\} \), for \( \forall e_i = [x_{i1}, x_{i2}, \ldots, \)
Methods of Decision Making

$\{x_n\}$, coming from a unique class is regarded as a leaf criterion and the positive highest information gain ratio ($GainRatio > 0$) is considered as the expanded attribute criterion; (2) Number of leaf nodes is identical with number of rules generated by the tree; (3) Every data point in the entire feature space is assigned into one of the possible classes; (4) Number of leaf nodes is less than the number of instances of the database.

Even though several optimization techniques are available in the literature [93] such as pre-pruning, post-pruning and so on to construct a DT with optimal rules, the proposed study exploits an alternative method neural network, which will be discussed in Section 4.3.

2.1.4 Applications of Decision Tree Approach

A DT approach is one of the most common approaches in automatic learning [6]. They have been already used in decision support systems for many decision making purposes [65]. In data mining and machine learning, decision tree is a predictive model [46]; that is, a mapping from observations about an item to conclusions about its target value. Decision Trees are used successfully in many different areas such as expert systems, medical diagnosis, speech recognition, character recognition, remote sensing, and so on [90].

Decision tree provides high classification accuracy with a simple representation of gathered knowledge [11]. The strength of DTs is inspired mainly with their ease of construction and their use of information theory. An advantage is that there exist a number of efficient algorithms that are able to find near-optimal architectures for the tree. It should be able to process huge amount of data and to extract their decisions easier and more reliably. The graphic representation of a DT can represent decision alternatives, possible outcomes, and chance events schematically. One can quickly express complex alternatives clearly and can easily modify the DT when new information becomes available. When compared to other methods of
induction, the ease with which is DT can be extended to nonnumeric domains [94].

2.2 Fuzzy Logic

According to Zadeh’s seminal paper on fuzzy sets [95] published in 1965, the definition of fuzzy set is as follows: Let X be a space of points (objects), with a generic element of X denoted by x, i.e., X = {x}. A fuzzy set A in X is characterized by a membership function $f_A(x)$, which associates with each point in X a real number in the interval [0, 1], with the value of $f_A(x)$ at x representing the “grade of membership” of x in A. Thus, when the value of $f_A(x)$ is nearer to unity, the grade of membership of x in A is higher. The fuzzy set A is a subset of X that has no sharp boundary and that is completely characterized by the set of pairs

$$A = \{(x, f_A(x)), x \in X\} \quad (2.9)$$

A fuzzy set A can also be expressed as

$$A = \frac{f_1}{x_1} + \frac{f_2}{x_2} + \ldots + \frac{f_n}{x_n} = \sum_{i=1}^{n} \frac{f_i}{x_i} \quad (2.10)$$

where the “division operator” uses to link the elements $x_i$ with their grades of memberships $f_i$ (i = 1, 2, ..., n) and the “addition operator” indicates the collective form of the elements and their related membership grades.

Fuzzy set theory directly addresses the limitation of sharp boundary in classical set theory by allowing membership in a set to be a matter of degree [19]. They provide an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge based and quantitative expressions. Since fuzzy variables capture measurement of uncertainties as part of experimental data, they are more attuned to reality than crisp variables. Fuzzy sets can bridge the gap between the discrete world of reasoning and the continuity of reality [34]. This theory gives a mathematical basis to represent certain forms of vagueness associated with human cognitive processes like thinking and reasoning. Because of the rule-
based operation, any reasonable number of inputs can be processed and numerous outputs can be generated.

2.2.1 Linguistic Terms

The main contribution of fuzzy logic is a methodology for computing with words in which it involves a fusion of natural languages and computation with fuzzy variables [96]. A fuzzy linguistic variable \((V)\) is an attribute whose domain contains linguistic values (also called fuzzy terms). For example,

\[
IF \ [V_1 \text{ is } v_{1}] \land [V_2 \text{ is } v_{2}] \land \cdots \land [V_n \text{ is } v_{n}] \quad THEN \quad [V_c \text{ is } v_{c}], \quad (2.11)
\]

where \(\{V_1, V_2, \cdots, V_n, V_c\}\) are linguistic variables and \(\{v_{1}, v_{2}, \cdots, v_{n}, v_{c}\}\) are linguistic values or terms that are characterized by membership functions. That is, a linguistic variable is a variable whose value can be described: (1) qualitatively using an expression involving linguistic terms, and (2) quantitatively using a corresponding membership function. The linguistic term is used to express the concepts and knowledge in human communication, whereas membership function is useful for processing numeric input data [97]. Associating a fuzzy set to a linguistic term, it offers two important benefits: (1) the association makes it easier for human experts to express their knowledge using the linguistic terms; (2) the knowledge expressed in linguistic terms is easily comprehensible [2].

2.2.2 Basic Operations

In fuzzy logic, the set operations “intersection” and “union” correspond to logic operations “conjunction” (AND) and “disjunction” (OR), respectively. There are various choices for the fuzzy conjunction and the fuzzy disjunction operators. A common choice is to use “\(\text{min}\)” for conjunction and “\(\max\)” for disjunction. These fuzzy operators can be defined in terms of membership functions [21]:

\[
\text{for } \forall x \in X, \quad \mu_{A \cap B}(x) = \text{min}\{\mu_A(x), \mu_B(x)\} \quad (2.12)
\]
for $\forall x \in X$, $\mu_{A \cap B}(x) = \mu_A(x) \times \mu_B(x)$  \hspace{1cm} (2.14)

for $\forall x \in X$, $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x)$  \hspace{1cm} (2.15)

There are a number of other choices too. The complement of a fuzzy set $A$ is defined by the difference between one and the membership degree in $A$ and is defined as,

for $\forall x \in X$, $\mu_A'(x) = 1 - \mu_A(x)$  \hspace{1cm} (2.16)

The set of candidate fuzzy conjunction operators, called \textit{triangular norms} or \textit{t-norms}, is defined by a set of axioms. Similarly, the set of fuzzy disjunction operators called \textit{triangular conorms}, or \textit{t-conorms} or \textit{s-norms}, is defined by a set of dual axioms. The definition of \textit{t-norm} and \textit{t-conorm} operators can be found in [19].

\textbf{Definition 1:} A \textit{t-norm} operator, denoted as $t(x, y)$, is a function mapping from $[0, 1] \times [0, 1]$ to $[0, 1]$ that satisfies the following conditions:

for any $w, x, y, z \in [0, 1]$,

1) $t(0, 0) = 0$, $t(x, 1) = t(1, x) = x$

2) $t(x, y) \leq t(z, w)$ if $x \leq z$ and $y \leq w$ (Monotonicity)

3) $t(x, y) = t(y, x)$ (Commutativity)

4) $t(x, t(y, z)) = t(t(x, y), z)$ (Associativity)

\textbf{Definition 2:} A \textit{t-conorm} operator, denoted as $s(x, y)$, is a function mapping from $[0, 1] \times [0, 1]$ to $[0, 1]$ that satisfies the following conditions:

for any $w, x, y, z \in [0, 1]$,

1) $s(0, 0) = 0$, $s(x, 1) = s(1, x) = x$

2) $s(x, y) \leq s(z, w)$ if $x \leq z$ and $y \leq w$ (Monotonicity)
3) \[ s(x, y) = s(y, x) \] (Commutativity)

4) \[ s(x, s(y, z)) = s(s(x, y), z) \] (Associativity)

The fuzzy set theory includes a number of other operations like these.

### 2.2.3 IF-THEN Rulebase

Fuzzy linguistic descriptions, the formal representations of systems, encode knowledge about a system in statements of the form “IF a set of conditions are satisfied THEN a set of consequents can be inferred”. Fuzzy rules have been used as a key tool to express knowledge in many fuzzy systems which are more adequate and flexible than the traditional IF-THEN rules. Using the rule-based structure of fuzzy logic, one can break down a complex problem into a series of “IF \( x \) and \( y \) THEN \( z \)” rules that define the desired system output response for the given system input conditions. The number of rules and their complexity depend on the number of input parameters that are to be processed and the number of fuzzy variables associated with each parameter. Among all the techniques developed in fuzzy sets, the fuzzy IF-THEN rule is the most visible one due to its wide range of successful applications.

The consequent of fuzzy rules can be classified into three categories [97]: crisp consequent, fuzzy consequent and functional consequent. In crisp consequent, it refers to

\[
\text{THEN } y = a \tag{2.17}
\]

where \( a \) is non-fuzzy numeric value or symbolic value. Generally, the fuzzy rules with crisp consequent can be processed more efficiently. The fuzzy consequent is of the form

\[
\text{THEN } y = A \tag{2.18}
\]

where \( A \) is a fuzzy set. A rule with a fuzzy consequent is easier to understand and more suitable for capturing imprecise human expertise. The functional consequent can be defined as
THEN \( y = a_0 + \sum_{i=1}^{n} a_i x_i \) \hspace{1cm} (2.19)

where \( a_0, a_1, \ldots, a_n \) are constants and \( x_1, x_2, \ldots, x_n \) are the objects of the fuzzy sets \( A_1, A_2, \ldots, A_n \) respectively. Rules with a functional consequent can be used to approximate complex nonlinear models using only a small number of rules. When a rule has multiple conditions, choose an appropriate fuzzy conjunction operator to combine these multiple conditions with their matching degree. The most commonly used conjunction operators in fuzzy logic are "minimum" and "product" operators denoted by "\text{min}" and "\text{prod}" respectively [97].

### 2.2.4 Fuzzification and Defuzzification Inferences

The purpose of the fuzzification is to interpret measurements of input variables, each expressed by a real number, as more realistic fuzzy approximations of the respective real numbers [19]. The fuzzification inference transforms crisp data into suitable linguistic values. In many real life situations, it is easier to take a crisp decision if the output is represented as a single scalar quantity. This conversion of a fuzzy set to single crisp value is called defuzzification. Several defuzzification methods are available in the literature such as centroid method, centre of sums, mean of maxima etc. Centroid method, also known as centre of gravity, obtains the centre of area \((x^*)\) occupied by the fuzzy set. For a continuous membership function, it is given by an expression with integral notation such as

\[
x^* = \frac{\int \mu(x) x \, dx}{\int \mu(x) \, dx}
\]

and for a discrete membership functions, the expression becomes

\[
x^* = \frac{\sum_{i=1}^{n} x_i \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}
\]
That is, the weighted strengths of each output member function are multiplied by their respective output membership function centre points and summed. Finally, this area is divided by the sum of the weighted member function strengths and the result is taken as the crisp output.

2.2.5 Fuzzy Inference Systems

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic [19]. The topology of the fuzzy systems consists of determining the number of membership functions in the input and output subspaces and the number of fuzzy rules to be constructed. Modelling of fuzzy systems [2] consists of three basic sub-tasks such as structure identification [98], parameter estimation and model validation. These tasks are discussed with the modelling process of the proposed system. Fuzzy systems can be broadly categorized into two families [99]: (1) Mamdani-type systems and (2) Sugeno type systems.

Mamdani Fuzzy Systems

Mamdani's fuzzy inference method [100] was among the first control systems built using fuzzy set theory, which was based on Zadeh's 1973 paper [101] on fuzzy algorithms for complex systems and decision processes. It was proposed in 1975 by Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani models include linguistic models based on collections of IF-THEN rules, whose antecedents and consequents utilize fuzzy values. The Mamdani type of inference process [102] required to obtain a conclusion from the input space has three basic steps: (1) Convert numeric input values to linguistic terms according to the membership functions derived; (2) Match the linguistic terms with the decision rules to find the output groups; (3) Defuzzify the output groups to form the final decision. In Mamdani model, the knowledge is represented as

\[ R_i : \text{IF } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \ldots \text{ and } x_n \text{ is } A_{in}, \]

\[ \text{THEN } y_i \text{ is } B_i \]  \hspace{1cm} (2.22)
where $R_i$ denotes the $i$th fuzzy rule, $x_j$ is the $j$th input, $y_i$ is the output of the fuzzy rule $R_i$ and $A_{i1}, A_{i2}, \ldots, A_{in}, B_i$ are fuzzy membership functions usually associated with linguistic terms. Mamdani fuzzy models are intuitive, provide heuristic insight and are well suited in terms of interactions with humans [103].

**Takagi-Sugeno-Kang Fuzzy Systems**

The Takagi-Sugeno-Kang fuzzy model or simply the Sugeno fuzzy model was proposed by Takagi, Sugeno and Kang [104]-[105] in an effort to develop a systematic approach to generating fuzzy rules from a given input-output data set. This category uses a rule structure that has fuzzy antecedent and functional consequent parts. This can be viewed as the expansion of piece-wise linear partition represented as

$$R_i: \text{IF } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \cdots \text{ and } x_n \text{ is } A_{in},$$

$$\text{THEN } y_i = f(x_1, x_2, \cdots, x_n)$$

(2.23)

where $\{A_{i1}, A_{i2}, \ldots, A_{in}\}$ are fuzzy terms in the antecedent part, while $y_i = f(x_1, x_2, \cdots, x_n)$ is a crisp function in the consequent, the output of the fuzzy rule $R_i$. Usually $f(x_1, x_2, \cdots, x_n)$ is a polynomial in the input variables $x_1, x_2, \cdots, x_n$, but it can be any function as long as it can appropriately describe the output of the model within the fuzzy region specified by the antecedent of the rule. When $f(x_1, x_2, \cdots, x_n)$ is a first-order polynomial, the resulting fuzzy inference system is called a first-order Sugeno model [104]-[105]. When $f$ is a constant, it becomes a zero-order Sugeno model, which can be viewed either as a special case of the Mamdani fuzzy inference system, in which each rule’s consequent is specified by a fuzzy singleton. The overlap of the membership functions (MFs) in the consequent of a Mamdani model does not have a decisive effect on the smoothness, whereas, that of the antecedent MFs determines the smoothness of the resulting input-output behaviour. The TSK type of fuzzy models proposed in [104]-[105]
has attracted a great attention of the fuzzy modelling researchers due to their good performance in various applications.

2.2.6 Applications of Fuzzy Logic

Fuzzy logic has been applied to many disciplines such as control systems, decision making, pattern recognition, data classification, system modelling and so on. Industrial applications ranging from consumer products, robotics, manufacturing and process control to financial trading are other areas. The applications which may be generated from or adapted to fuzzy logic are wide-ranging, and provide the opportunity for modelling of conditions which are imprecisely defined, despite the concerns of classical condition. Fuzzy inference systems have been successfully applied in fields such as automatic control, expert systems, computer vision etc. Fuzzy logic also plays a critical role in medical applications ranging from diagnostic systems to medical imaging. It has been utilized in several different approaches of modelling the decision support systems in medical domain.

The representative power, good approximation ability and simplicity of its operation are the advantages of fuzzy approach. It is capable of describing a nonlinear system using sufficient rules and training data. Another important benefit of fuzzy model is that its rules are interpretable because they capture a local relationship between the model’s input and output. Fuzzy sets are considered to be advantageous in the logical field, and in handling higher order processing easily.

2.3 Artificial Neural Networks

Artificial Neural Networks or simply Neural Networks are information processing paradigm, which are inspired by the way the biological nervous systems such as the brain process information. They are highly simplified models of the human nervous system which mimic human’s ability to adapt to circumstances and learn from past experience. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of neural networks
Methods of Decision Making

(NNs) as well. Neural network contains layers of simple nodes of data that interact through weighted connection lines, so that they may process an outcome. The key element of this paradigm is the structure of the information processing system. A simple model of an artificial neuron is illustrated in Fig 2.2.

![Fig 2.2 Simple Model of an Artificial Neuron](image)

In this figure, $x_1, x_2, x_3, \ldots, x_n$ are the $n$ inputs to the artificial neuron and $w_1, w_2, w_3, \ldots, w_n$ are the weights attached to the input links. As the weights here are multiplicative factors of the inputs to account for the strength of the node, the total input $I$ of the artificial neuron is defined as

$$I = \sum_{i=1}^{n} w_i x_i$$

(2.24)

To generate the final output $y$, the sum is passed on to a non-linear filter $\phi$ called activation function. A very commonly used activation function is the threshold function defined by

$$y = \phi \left( \sum_{i=1}^{n} w_i x_i - \theta \right)$$

(2.25)

where "$\theta$" is a bias term. A single layer feedforward network comprises of two layers, an input layer and an output layer. The input layer receives the input signals and the output layer receives the output signals. As it is
feedforward, the synaptic links carrying the weights connected every input neuron to the output neuron but not vice-versa.

2.3.1 Feedforward and Feedback Networks

A feedforward neural network [106] consists of layers of processing units, each layer feeding input to the next layer in a feedforward manner through a set of connection weights. The simplest such network is a two layer network.

![A Simple Feedforward Network](image)

A simple feedforward network looks like the diagram given in Fig. 2.3. A feedback network [106] consists of a set of processing units, the output of each unit is fed as input to all other units including the same unit. With each link connecting any two units, a weight is associated which determines the amount of output a unit feeds as input to the other unit. Feedback neural networks are used mainly for pattern storage tasks.

2.3.2 Multilayer Feedforward Networks

A network, made up of multiple layers, the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network, is known as multilayer feedforward network. The architecture of a multilayer feedforward network have an input layer, an output layer and one or more intermediary layers called hidden layers. The hidden layer is performed useful intermediary computations before directing the input to the output layer. Any number of hidden layers are possible.
within a feedforward network but one is usually enough to suffice for most problems to tackle. There can be any number of neurons in each layer, it all depends on the problem to be solved. The data that is feed for the network is called a training set. Characteristics of NNs are the following:

- NNs exhibit mapping capabilities, that is, they can map input patterns to their associated output patterns
- NNs learn by examples. Thus, NN architectures can be ‘trained’ with known examples of a problem. They can, therefore, identify new objects previously untrained.
- NNs possess the capability to generalize. Thus, they can predict new outcomes from past trends.
- NNs are robust systems and are fault tolerant. They can, therefore, recall full patterns from incomplete, partial or noisy patterns.
- NNs can process information in parallel, at high speed, and in a distributed manner.

The NNs have adaptive learning property, in which the learning rule specifies how the parameters of the network change to minimize the error measure. They have the advantage of self-organization, where a NN can create its own organization or representation of the information it receives during learning time. The higher flexibility is a characteristic feature of neural nets produced by learning, and hence this suits data-driven processing better. The NN is composed of a large number of highly interconnected processing elements working in unison to solve specific problems.

2.3.3 Learning Methods

Learning methods in Neural Networks can be broadly classified into three basic types [106]: supervised, unsupervised and reinforced.

**Supervised learning:** In this learning method, every input pattern that is used to train the network is associated with an output pattern, which is the target or the desired pattern. This target is assumed to be present during the learning process, when a comparison is made between the network's
computed output and the target output to determine the error. To improve the performance of the network, the error can be used to change the network parameters.

**Unsupervised learning:** In this learning method, the target output is not presented to the network. Hence, the system learns of its own by discovering and adapting to structural features in the input patterns.

**Reinforced learning:** In this learning, a teacher though available, does not present with the expected answer but only indicates if the computed output is correct or incorrect. The information provided helps the network in its learning process.

Supervised and unsupervised learning methods are most popular forms of learning, whereas, reinforced learning is not one of the popular forms. Some of the widely used learning rules [106] are Hebbian learning, Gradient descent learning, Competitive learning and Stochastic learning. Gradient descent and Stochastic are supervised, whereas, Hebbian and Competitive are unsupervised learning methods. Once the neural network has been created it needs to be trained. One way of doing this is to initialize the neural net with random weights and then feed it a series of inputs. There are many different ways of adjusting the weights; the most common type of this problem is called “backpropagation” [107].

### 2.3.4 Backpropagation Learning Rule

Backpropagation [107] or generalized delta learning rule is a systematic method of training multilayer NNs with gradient descent learning. A multilayer feedforward network with backpropagation learning is sometimes called multilayer perceptron because of its similarity to perceptron networks with more than one layer. The hidden layer allows the network to develop its own internal representation of this mapping. The training of a backpropagation NN involves two passes known as a *forward pass* and a *backward pass*. In the forward pass, the input signals propagate from the network to the output. In the reverse pass, the calculated error
signals propagate backwards through the network where they are used to adjust the weights.

Learning rate coefficients (a small positive value) determine the size of the weight adjustments made at each iteration and hence influences the rate of convergence. Poor choice of the coefficient can result in a failure in convergence. If the learning-rate is too large, the search path may oscillate or more slowly converge than a direct descent. When the coefficient is too small, the descent will progress in small steps that significantly increase the time to converge. If the learning coefficient is zero, no learning takes place and hence, the learning coefficient must be positive.

2.3.5 Applications of Neural Networks

Neural networks are applicable to solve a great variety of real world tasks due to their properties of learning by examples [108]. Some of the examples are the following: The NNs model have been used to predict the risk of the diseases. Neural networks features are ideal in recognising diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. In medical modelling and diagnosing, this implies that even though each sensor in a set may be sensitive only to a specific physiological variable, NNs are capable of detecting complex medical conditions by fusing the data from the individual biomedical sensors. A NN is configured for pattern recognition or data classification through a learning process. They have also been successfully applied in many industries.

2.4 Genetic Algorithms

Genetic Algorithms (GAs) are computer programs which create an environment where populations of data can compete and only the fittest survive [109]. They are the optimization methodology that works with a population of randomly generated individuals and continue in generations [55]. This characteristic, associated with their stochastic nature, enables GAs to deal with large search spaces randomly and efficiently [110]. For solving a
single objective problem, GA designs many solutions until no further improvement (no increase in fitness) can be achieved or some predetermined number of generations has evolved. When solving multi-objective problems, GA gives out many satisfactory solutions in terms of the objectives, and then allows the decision maker to select the best alternative.

Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. A standard representation of the solution, known as chromosomes, might be strings of bits (0s and 1s) where each bit represents a different object. A typical chromosome may look like this:

```
1001010111010100101001
```

Arrays of other types and structures can be used in essentially the same way. A chromosome encodes a parameter set for a set of variables being optimized. Each encoded parameter in a chromosome is called a gene. A set of chromosomes forms a population, which is evaluated and ranked by a fitness evaluation function. This function plays a critical role in GA because it provides information about how good each candidate solution is. The evolution from one generation to the next involves mainly the following steps [55]: (1) Fitness Evaluation, (2) Selection, (3) Crossover and (4) Mutation.

### 2.4.1 Fitness Evaluation

A typical fitness function, which is always problem dependent, is defined over the genetic representation and measures the quality of the represented solution [111]. At each generation, the current population is evaluated using the fitness evaluation function and then ranked based on their fitness values [77]. There are at least two types of fitness evaluation functions: fitness-based and performance-based. A fitness-based function calculates how well a model using the parameter set in a chromosome fits a
set of input-output training data. That is, the function computes the error between the target’s output and the model’s output. A performance-based evaluation function determines how well a system using the parameter set in a chromosome achieves a set of performance objectives. The goal of fitness-based GA evaluation is similar to that of supervised learning. The spirit of performance-based GA evaluation is similar to that of reinforcement learning. Therefore, GA can be used as an alternative to both of these two neural network learning techniques.

2.4.2 Selection

Selection is a stage of the GA in which individual genomes are chosen from a population for later breeding (recombination or crossover). It is a process of copying the individual strings according to their objective function values, which means that strings with a higher value have a higher probability of contributing one or more offspring in the next generation [77]. GA stochastically selects “parents” from the current population, which could be accomplished using a selection probability that is determined by the fitness value or the ranking of a chromosome. There are several generic selection algorithms, such as tournament selection, fitness proportionate selection, and roulette wheel selection. Among these selection methods, the most common type is roulette wheel selection [112].

Roulette Wheel Selection: In this method, individuals are given a probability of being selected that is directly proportionate to their fitness. Each chromosome represents a portion on the wheel whose size is proportional to its fitness value. The wheel is then tossed as many times as parents are needed to create the next generation, and each winning individual is selected and copied into the new population. This method allows an individual to be selected more than once and the ‘death’ of some individuals as well. To construct a roulette wheel for a population, first to calculate the roulette probability of each chromosome, that is, the ratio of its fitness to the fitness of the entire population.
Roulette Probability = \frac{\text{Fitness}}{\text{Total Fitness}} \tag{2.26}

A population of four chromosomes (individuals) and their fitness values in terms of sum of “1” bits are given in Fig 2.4.

<table>
<thead>
<tr>
<th>CHROMOSOME</th>
<th>FITNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 0 1 0</td>
<td>3</td>
</tr>
<tr>
<td>1 1 1 1 0 1</td>
<td>5</td>
</tr>
<tr>
<td>1 0 0 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0 0 0 0</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 2.4 4 - Chromosome Population

The roulette wheel is designed on the basis of the probabilities of the fitness value. Fig 2.5 depicts the roulette wheel applied to the population given in Fig 2.4. The individual with the highest probability is selected, the higher the fitness of an individual, the larger the portion of the roulette wheel to be assigned this individual and thus the higher its probability of being selected.

The next step is to generate a second generation population of solutions from those selected parents through genetic operators: crossover and/or mutation.
2.4.3 Crossover

In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. The crossover operation produces offspring by exchanging information between two parent chromosomes. While there are many different kinds of crossover, the most common type is single point crossover. In single point crossover, one chooses a locus at which he/she swaps the remaining alleles (one of two or more DNA sequences occurring at a particular gene locus) from one parent to the other. In this particular method, the point at which the chromosome is broken depends on the randomly selected crossover point because only one crossover point exists. The probability of crossover occurring is usually 60% to 80%. The chances that the crossover operation applies to a chromosome are controlled by the crossover probability.

![Diagram of Single-Point Crossover](image)

Fig 2.6 Single-Point Crossover for a Pair of Chromosome of Length \( l = 8 \)

Crossing-over proceeds basically in three steps [112]:

1. Two strings \( a = a_1a_2 \cdots a_l \) and \( b = b_1b_2 \cdots b_l \) are selected from the current population \( P \).
2. A number \( r \) indicating the crossover point is randomly selected from \( \{1, 2, \cdots, l - 1\} \).
3. Two new strings are formed from \( a \) and \( b \) by exchanging the set of attributes to the right of position \( r \), yielding \( a' = a_1 \cdots a_r b_{r+1} \cdots b_l \) and \( b' = b_1 \cdots b_r a_{r+1} \cdots a_l \).
These new chromosomes (strings) \( a' \) and \( b' \) are the children of \( a \) and \( b \). This single-point crossover is illustrated in Fig 2.6 for two strings of length \( l = 8 \).

### 2.4.4 Mutation

Mutation is a genetic operator used to maintain the genetic diversity from one generation of a population of chromosomes to the next. The mutation operation produces offspring from parents through a random modification of the parents. The purpose of mutation in GAs is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. After selection and crossover, a new population of individuals is generated. Some are directly copied, and others are produced by crossover.

In order to ensure that the individuals are not all exactly the same, the mutation operation applies to a chromosome are controlled by the mutation probability. This is usually a very low value for binary encoded genes, say 0.001. Each string in the population \( P \) is operated as follows:

1. The numbers \( r, \ldots, u \) indicating the positions to undergo mutation are determined by a random process where each position has a small probability “\( pm \)” of undergoing mutation, independently of the other positions.

2. A new string \( a' = a_1 \cdots a_r \cdots a_u \cdots a_l \) is generated where \( a_r \cdots a_u \) are drawn at random from the set of alleles for each gene. In the case of
Methods of Decision Making

bit-strings, if a position has an allele "0", then it becomes "1", or if it is originally "1", then it becomes "0".

Fig 2.7 illustrates the mutation for a chromosome having length \( l = 8 \). Whenever chromosomes are chosen from the population, the algorithm first checks to see if crossover should be applied and then the algorithm iterates down the length of each chromosome mutating the bits if applicable. The process continues until a new population of solutions of appropriate size is generated. These processes ultimately result in the next generation population of chromosomes that is different from the initial generation.

2.4.5 Termination Criteria

The cycle of evaluation, selection, crossover and mutation terminates when an acceptable solution is found, when a convergence criterion is met, or when a predetermined limit on the number of iterations is reached. The generational process is repeated until a termination condition has been reached. Common terminating conditions are

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching
- Combinations of the above.

2.4.6 Applications of Genetic Algorithms

Genetic algorithms are a very effective way of quickly finding a reasonable solution to a complex problem and are most effective in a search space for which little is known. They produce solutions that solve the problem in ways one may never have even considered. Genetic algorithms find application in biogenetics, computer science, engineering, chemistry, mathematics, physics, economics, manufacturing, and other fields. They have been used to solve many different types of business problems in functional areas such as finance, marketing, information systems, and
production. GAs have been used for many machine-learning applications, including classification and prediction. They have also been used to design neural networks, to evolve rules for learning classifier systems or symbolic production systems, and to design and control robots.

The genetic algorithms' strength come from the implicitly parallel search of the solution space that it performs via a population of candidate solutions and this population is manipulated in the simulation. Genetic operators could then be applied to improve the performance of the population of behaviours. One cycle of testing all of the competing behaviour is defined as a generation, and is repeated until a good behaviour is evolved. The good behaviour is then applied to the real world.

2.5 Hybrid Systems

Hybrid systems are those for which more than one technology is employed to solve the problem. The objective of the hybridization has been to overcome the weakness in one technology during the application, with the strengths of the other by appropriately integrating them. Hybridization should only be performed for the purpose of investigating better methods of problem solving. Neuro-fuzzy [113], neuro-genetic [77], fuzzy-genetic [114], fuzzy-neural network [72] and fuzzy ARTMAP [115] are some of the hybrid systems reported in literature. Neuro-fuzzy hybridization [97] is done broadly in two ways: a neural network equipped with the capability of handling fuzzy information, known as fuzzy-neural network (FNN) and a fuzzy system augmented by neural networks to enhance some of its characteristics like flexibility, speed and adaptability, known as neural-fuzzy system (NFS) [99].

2.5.1 Hybrid Learning Rules

Generally, there are two ways to adjust the weights using back-propagation in a hybrid learning mechanism: (1) Pattern mode, (2) Batch mode. Pattern mode or on-line approach adjusts the weights based on the error signal of an input-output pair in the training data. These adjustments are
made immediately after each training datum is fed into the neural network. The other approach “batch mode” or “off-line” learning adjusts the weights based on the error signal of the entire training set. Therefore, weights are adjusted once only after all the training data have been processed by the NN.

### 2.5.2 Neuro-Fuzzy Hybrid Systems

Neural Networks and Fuzzy logic represent two distinct methodologies to deal with some common features such as distributed representation of knowledge, model-free estimation, ability to handle data with uncertainty and imprecision etc. Fuzzy logic has tolerance for imprecision of data, while neural networks have tolerance for noisy data. Neural network’s learning ability provides a good way to adjust expert’s knowledge and it automatically generates additional fuzzy rules and membership functions to meet certain specifications. On the other hand, the fuzzy logic approach possibly enhances the generalization capability of a neural network by providing more reliable output. Fuzzy logic systems address the imprecision of inputs and outputs directly by defining them using fuzzy sets and allow for a greater flexibility in formulating system descriptions at the appropriate level of detail [116]. The significance of this integration becomes even more reasonable by considering their similarities and disparities.

### 2.6 Need of Developing a New Model

Neuro-fuzzy approach has a profound impact in a wide variety of applications of modelling and decision making problems, whereas working with such a system is not viable if the number of input parameters and membership functions are considerably large. Being a hybrid model, neuro-fuzzy systems have been widely considered and successfully applied to classification task, rule-based process controls, pattern recognition problems, modelling problems and so on. A long-standing problem in fuzzy rule-based modelling is the “curse of dimensionality”, which occurs due to the exponential increase of rules when the number of input variables increases.
In this aspect, the major problem of a neuro-fuzzy system is the partition of input space with a sufficiently large number of inputs. To deal with this problem, an appropriate method should be used to reduce the number of rules.

Several approaches using singular value decomposition (SVD) to overcome this problem have been reported in the literature, an example is [118]. Fuzzy rules optimization with genetic algorithm is another efficient method to reduce the number of rules, which is reported by several authors [110], [119]. Some other methods based on some similarity measure are presented in [120] to reduce the number of fuzzy rules. Recently, we have published a paper [79] to overcome this complexity by integrating fuzzy decision tree and neuro-fuzzy systems, in which rules are extracted from the decision trees. It becomes more efficient to represent a considerably big database with a few rules. Besides rules reduction, a generalized hybrid system with high efficiency is the key goal of this study.