Soft Computing Paradigms and Regression Trees in Decision Support Systems

Cong Tran, Ajith Abraham[*] and Lakhmi Jain

School of Electrical and Information Engineering
University of South Australia
Email: Tramcm001@students.unisa.edu.au, Lakhmi.Jain@unisa.edu.au
[*] School of Computer Science and Engineering,
Chung-Ang University, Seoul 156-756, Korea
Email: ajith.abraham@ieee.org

Abstract: Decision-making is a process of choosing among alternative courses of action for solving complicated problems where multi-criteria objectives are involved. The past few years have witnessed a growing recognition of Soft Computing (SC) technologies that underlie the conception, design and utilization of intelligent systems. In this chapter, we present different SC paradigms involving an artificial neural network trained using the scaled conjugate gradient algorithm, two different fuzzy inference methods optimised using neural network learning/evolutionary algorithms and regression trees for developing intelligent decision support systems. We demonstrate the efficiency of the different algorithms by developing a decision support system for a Tactical Air Combat Environment (TACE). Some empirical comparisons between the different algorithms are also provided.

1. Introduction

Several decision support systems have been developed mostly in various fields including medical diagnosis [5], business management, control system [41], command and control of defence and air traffic control [8] and so on. Usually previous experience or expert knowledge is often used to design decision support systems. The task becomes interesting when no prior knowledge is available. The need for an intelligent mechanism for decision support comes from the well-known limits of human knowledge processing. It has been noticed that the need for support for human decision makers is due to four kinds of limits: cognitive, economic, time and competitive demands [13]. Several artificial intelligence techniques have been explored to construct adaptive decision support systems. A framework that could capture imprecision, uncertainty, learn from the data/information and continuously optimize the solution by providing interpretable decision rules would be the ideal technique. Several adaptive learning frameworks for constructing intelligent decision support systems have been proposed [7][14][16][36]. Figure 1 summarizes the basic functional aspects of a decision support system. A database is created from the available data and human

knowledge. The learning process then builds up the decision rules. The developed rules are further fine tuned depending upon the quality of the solution using a supervised learning process.

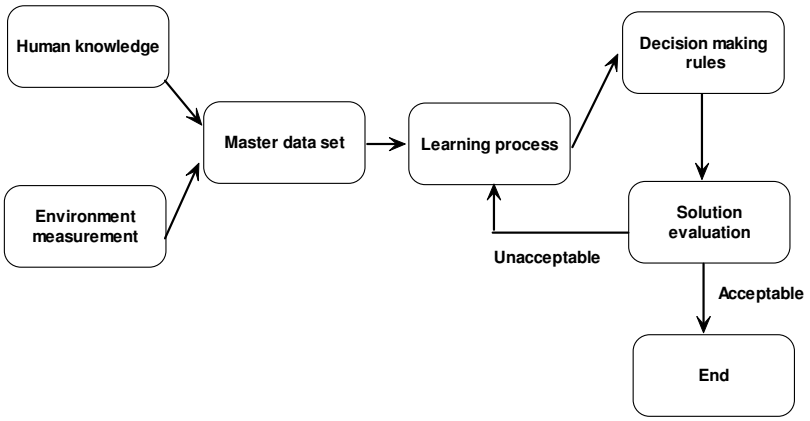


Figure 1. Database learning framework for decision support system

To develop an intelligent decision support system, we need a holistic view on the various tasks to be carried out including data management and knowledge management (reasoning techniques). The focus of this chapter is knowledge management, which consists of facts and inference rules used for reasoning.

Fuzzy logic, when applied to decision support systems, provides formal methodology to capture valid patterns of reasoning about uncertainty. Neural networks are popularly known as blackbox function approximators. Recent research work shows the capabilities of rule extraction from a trained network positions [30][31] neurocomputing as a good decision support tool. Recently Evolutionary Computation (EC) has been successful as a powerful global optimisation tool due to the success in several problem domains [2][42][43][44][45]. EC works by simulating evolution on a computer by iterative generation and alteration processes operating on a set of candidate solutions that forms a population. Due to the complementarity of neural networks, fuzzy inference systems and evolutionary computation, the recent trend is to fuse various systems to form a more powerful integrated system, to overcome their individual weakness.

Decision trees [6] have emerged as a powerful machine learning technique due to a simple, apparent, and fast reasoning process. Decision trees can be related to artificial neural networks by mapping them into a class of artificial neural networks or entropy nets with far fewer connections.

In Section 2, we present the complexity of the Tactical Air Combat Decision Support System (TACDSS), followed by some theoretical foundation on neural networks, fuzzy inference systems and decision trees in Section 3. In Sections 4 and 5, we present the different adaptation procedures for optimising fuzzy inference systems. A Takagi-Sugeno [33][41] and a Mamdani-Assilian [24] fuzzy inference system learned using neural network learning techniques and evolutionary computation is discussed. Experimentation results using the different connectionist paradigms are presented in Section 6. Detailed discussions of the different experimental results are given in Section 7 followed by conclusion towards the end.

2. Tactical Air Combat Decision Support Systems (TACDSS)

Implementation of a reliable decision support system involves two important factors: collection and analysis of prior information and the evaluation of the solution. The data could be an image or a pattern, real number, binary code or natural language text data depending on the objects of the problem environment. An object of the decision problem is also known as the decision factor. These objects can be expressed mathematically in the decision problem domain as a universal set where the decision factor is a set and decision data is an element of this set. The decision factor is a subset of the decision problem. If we call the Decision Problem (DP) as *X* and the decision factor (DF) as '*A*', then the decision data (DD) could be labelled as '*a*'. Suppose the set *A* has members a_1, a_2, ... , a_n then it can be denoted by $A = \{a_1, a_2, .., a_n\}$ or can be written as:

$$A = \{a_i \mid i \in R_n\} \tag{1}$$

where *i* is called the *set index*, the symbol '|' is read as 'such that' and R_n is the set of *n* real numbers. A subset '*A*' of *X*, denoted $A \subseteq X$, is a set of elements that is contained within the universal set *X*. For optimal decision-making, the system should be able to adaptively process the information provided by words or any natural language description of the problem environment.

To illustrate the proposed approach, we consider a case study based on a tactical environment problem. We aim to develop an environment decision support system for a pilot or mission commander in tactical air combat. We will attempt to present the complexity of the problem with some scenarios of the problem. In Figure 2 a typical scenario of an air combat tactical environment is presented. The Airborne Early Warning and Control (AEW&C) is performing surveillance in a particular area of operation. It has two hornets (F/A-18s) under its control at the ground base shown as "+" in the left corner of Figure 2. An air-to-air fuel tanker (KB707) "Í" is on station - the location and status of which are known to the AEW&C. One of the hornets is on patrol in the area of Combat Air Patrol (CAP). Sometime later, the AEW&C on-board sensors detect hostile aircraft(s) shown as "O". When the hostile aircraft enter the surveillance region (shown as a dashed circle) the mission

system software is able to identify the enemy aircraft and estimate its distance from the Hornets in the ground base or in the CAP.

The mission operator has few options to make a decision on the allocation of hornets to intercept the enemy aircraft:

- Send the Hornet directly to the spotted area and intercept,
- Call the Hornet in the area back to ground base or send another Hornet from the ground base.
- Call the Hornet in the area for refuel before intercepting the enemy aircraft.

The mission operator will base his/her decisions on a number of factors, such as:

- Fuel reserve and weapon status of hornet in the area,
- Interrupt time of Hornets in the ground base or at the CAP to stop the hostile,
- The speed of the enemy fighter aircraft and the type of weapons it possesses.

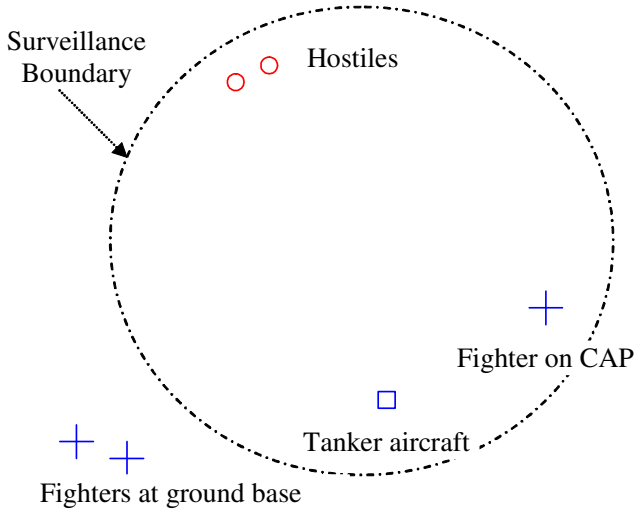


Figure 2. A Typical Air Combat Scenario

Table 1. Decision factors for the tactical air combat

Fuel reserve	Time Intercept	Weapon Status	Danger Situation	Evaluation Plan
Full	Fast	Sufficient	Very Danger	Good
Half	Normal	Enough	Danger	Acceptable
Low	Slow	Insufficient	Endanger	Bad

From the above scenario, it is evident that there are important decision factors of the tactical environment that might directly affect the air combat decision. For demonstrating our proposed approach, we will simplify the problem by handling only a few important decision factors such as "fuel status", "weapon possession status" and "interrupt time" (Hornet in the ground base and in the area of CAP) and the "Situation Awareness". These factors are tabulated in Table 1. The developed tactical air combat decision rules should be able to incorporate all the above-mentioned decision factors.

Knowledge of Tactical Air Combat Environment (TACE)

How human knowledge could be extracted to a database? Very often people express knowledge as natural (spoken) language or using letters or symbolic terms. The human knowledge can be analysed and converted into an information table. There are several methods to extract human knowledge. Some researchers use Cognitive Work Analysis (CWA) [29], others Cognitive Task Analysis (CTA) [26]. CWA is a technique to analyse, design and evaluate human computer interactive systems. CTA is a method used to identify cognitive skill, mental demands and needs to perform task proficiency. CTA focuses on describing the representation of the cognitive elements that defines goal generation and decision making. It is a reliable method to extract human knowledge because it is based on observations or an interview. We have used the CTA technique to set up the expert knowledge base for building the complete decision support system. For the TACE discussed previously, we have four decision factors that could affect the final decision options of “hornet in the CAP” or “hornet at the ground base”. These are “fuel status” being the quantity of fuel available to perform the intercept, the “weapon possession status” presenting the state of available weapons inside the hornet, the “interrupt time” which is required for the hornet to fly and interrupt the – hostile, and the “danger situation” providing information whether the aircraft is a friend or hostile.

Each of the above-mentioned factors has a different range of units, these being the fuel (0 to 1000 litres), interrupt time (0 to 60 minutes), weapon status (0 to 100 %) and the danger situation (0 to 10 points). The following are two important decision selection rules, which were formulated using expert knowledge:

- The decision selection will have a small value if the fuel is too low, the interrupt time is too long, the hornet has low weapon status and the Friend or Foe danger is high.
- The decision selection will have a high value if the fuel reserve is full, the interrupt time is fast enough, the hornet has high weapon status and the FOE danger is low.

In TACE, decision-making is always based on all states on all the decision factors. However sometimes a mission operator/commander can make a decision based on an important factor, such as the fuel reserve of the hornet is too low (due to high

fuel use), enemy has more powerful weapons, the quality and quantity of enemy aircraft. Table 2 shows the decision score at each stage of the TACE.

Table 2. Some prior knowledge of the TACE

Fuel status (litres)	Interrupt time (minutes)	Weapon status (percent)	Danger situation (points)	Decision selection (points)
0	60	0	10	0
100	55	15	8	1
200	50	25	7	2
300	40	30	5	3
400	35	40	4.5	4
500	30	60	4	5
600	25	70	3	6
700	15	85	2	7
800	10	90	1.5	8
900	5	96	1	9
1000	1	100	0	10

3. Soft Computing and Decision Trees

Soft computing paradigms can be used to construct new generation intelligent hybrid systems consisting of neural networks, fuzzy inference system, approximate reasoning and derivative free optimisation techniques. It is well known that the intelligent systems which provide human-like expertise such as domain knowledge, uncertain reasoning, and adaptation to a noisy and time varying environment, are important in tackling real world problems.

3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) have been developed as generalizations of mathematical models of biological nervous systems. A neural network is characterised by the network architecture, the connection strength between pairs of neurons (weights), node properties, and update rules. The updating or learning rules control the weights and/or states of the processing elements (neurons). Normally, an objective function is defined that represents the complete status of the network, and its set of minima corresponds to different stable states [40]. It can learn by adapting its weights to changes in the surrounding environment, can handle imprecise information, and generalise from known tasks to unknown ones. The network is initially randomised to avoid imposing any of our own prejudices about an application of interest. The training patterns can be thought of as a set of ordered pairs $\{(x_1, y_1), (x_2, y_2), ..,(x_p, y_p)\}$ where x_i represents an input pattern and y_i represents the output pattern vector associated with the input vector x_i. A valuable property of neural networks is that of generalisation, whereby a trained

neural network is able to provide a correct matching in the form of output data for a set of previously unseen input data. Learning typically occurs through training, where the training algorithm iteratively adjusts the connection weights (synapses). In the Conjugate Gradient Algorithm (CGA) a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. A search is made along the conjugate gradient direction to determine the step size, which will minimize the performance function along that line. A line search is performed to determine the optimal distance to move along the current search direction. Then the next search direction is determined so that it is conjugate to the previous search direction. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction. An important feature of CGA is that the minimization performed in one step is not partially undone by the next, as it is the case with gradient descent methods. An important drawback of CGA is the requirement of a line search, which is computationally expensive. The Scaled Conjugate Gradient Algorithm (SCGA) [27] was designed to avoid the time-consuming line search at each iteration, and incorporates the model-trust region approach used in the CGA Levenberg-Marquardt algorithm [2].

3.2 Fuzzy Inference Systems (FIS)

Fuzzy inference systems are a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. The basic structure of the fuzzy inference system consists of three conceptual components: a rule base, which contains a selection of fuzzy rules; a database, which defines the membership functions used in the fuzzy rule; and a reasoning mechanism, which performs the inference procedure upon the rules and given facts to derive a reasonable output or conclusion. Figure 3 shows the basic architecture of a FIS with crisp inputs and outputs implementing a non-linear mapping from its input space to its output [7].

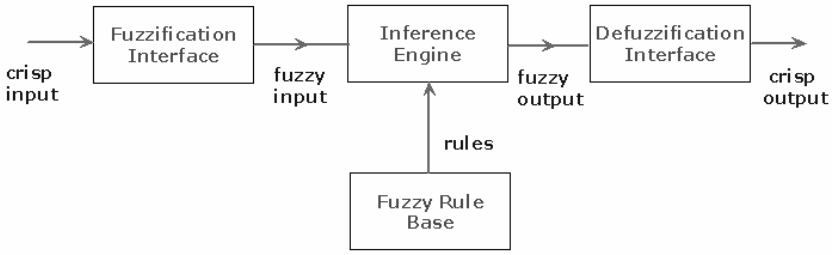


Figure 3. Block diagram of a fuzzy inference system

We now introduce two different fuzzy inference systems that have been widely employed in various applications. These fuzzy systems feature different consequents in their rules, and thus their aggregation and defuzzification procedures differ accordingly.

Most fuzzy systems employ the inference method proposed by Mamdani-Assilian in which the rule consequence is defined by fuzzy sets and has the following structure [24]:

$$If\ x\ is\ A_1\ and\ y\ is\ B_1\ then\ z_1 = C_1 \quad (2)$$

Takagi, Sugeno and Kang proposed an inference scheme in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of the crisp inputs rather than a fuzzy set, and which has the following structure [33]:

$$If\ x\ is\ A_1\ and\ y\ is\ B_1, then\ z_1 = p_1 x + q_1 y + r \quad (3)$$

Takagi-Sugeno FIS usually needs a smaller number of rules, because their output is already a linear function of the inputs rather than a constant fuzzy set [1].

3.3 Evolutionary Algorithms (EAs)

Evolutionary Algorithms are population-based adaptive methods, which may be used to solve optimization problems, based on the genetic processes of biological organisms [10],[42]. Over many generations, natural populations evolve according to the principles of natural selection and "survival-of-the-fittest", first clearly stated by Charles Darwin in "On the Origin of Species" [10]. By mimicking this process, EAs are able to "evolve" solutions to real world problems, if they have been suitably encoded. The procedure may be written as the difference equation [10]:

$$x[t+1] = s(v(x[t])) \quad (4)$$

where *x (t)* is the population at time *t*, *v* is a random operator, and *s* is the selection operator. The algorithm is illustrated in Figure 4.

1. Generate the initial population *P(0)* at random and set *i=0*;
2. Repeat until the number of iterations or time has reached or the population has converged.
   - *a.* Evaluate the fitness of each individual in *P(i)*
   - b. Select parents from *P(i)* based on their fitness in *P (i)*
   - c. Apply reproduction operators to the parents and produce offspring, the next generation, *P(i+1)* is obtained from the offspring and possibly parents.

Figure 4. Pseudo Code of an Evolutionary Algorithm

A conventional fuzzy controller makes use of a model of the expert who is in a position to specify the most important properties of the process. Expert knowledge is often the main source to design the fuzzy inference systems. According to the performance measure of the problem environment, the membership functions and rule bases are to be adapted. Adaptation of fuzzy inference systems using evolutionary computation techniques has been widely explored [3][4]. In the following section, we will discuss how fuzzy inference systems could be adapted using neural network learning techniques.

3.4 Neuro- Fuzzy Computing

Neuro Fuzzy (NF) computing is a popular framework for solving complex problems. If we have knowledge expressed in linguistic rules, we can build a FIS, and if we have data, or can learn from a simulation (training) then we can use ANNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of a linguistic rule base will be advantageous from the viewpoint of ANN [1].

In a fused NF architecture, ANN learning algorithms are used to determine the parameters of the FIS. Fused NF systems share data structures and knowledge representations. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special ANN-like architecture. However the conventional ANN learning algorithm (gradient descent) cannot be applied directly to such a system as the functions used in the inference process are usually non differentiable. This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm. Two neuro-fuzzy learning paradigms are presented in Section 4 and 5.

3.5 Classification and Regression Trees (CART)

Tree-based models are useful for both classification and regression problems. In these problems, there is a set of classification or predictor variables (X_i) and a dependent variable (Y). The X_i variables may be a mixture of nominal and/or ordinal scales (or code intervals of equal-interval scale) and Y may be a quantitative or a qualitative (in other words, nominal or categorical) variable [6] [32].

The CART methodology is technically known as binary recursive partitioning. The process is binary because parent nodes are always split into exactly two child

nodes, and recursive because the process can be repeated by treating each child node as a parent. The key elements of a CART analysis are a set of rules for splitting each node in a tree:

- deciding when a tree is complete, and
- assigning each terminal node to a class outcome (or predicted value for regression)

CART is the most advanced decision-tree technology for data analysis, pre-processing and predictive modelling. CART is a robust data-analysis tool that automatically searches for important patterns and relationships and quickly uncovers hidden structure even in highly complex data. CART's binary decision trees are more sparing with data and detect more structure before further splitting is impossible or stopped. Splitting is impossible if only one case remains in a particular node, or if all the cases in that node are exact copies of each other (on predictor variables). CART also allows splitting to be stopped for several other reasons, including that a node has too few cases [32].

Once a terminal node is found we must decide how to classify all cases falling within it. One simple criterion is the plurality rule: the group with the greatest representation determines the class assignment. CART goes a step further: because each node has the potential for being a terminal node, a class assignment is made for every node whether it is terminal or not. The rules of class assignment can be modified from simple plurality to account for the costs of making a mistake in classification and to adjust for over- or under-sampling from certain classes.

A common technique among the first generation of tree classifiers was to continue splitting nodes (growing the tree) until some goodness-of-split criterion failed to be met. When the quality of a particular split fell below a certain threshold, the tree was not grown further along that branch. When all branches from the root reached terminal nodes, the tree was considered complete. Once a maximal tree is generated, it examines smaller trees obtained by pruning away branches of the maximal tree. Once the maximal tree is grown and a set of sub-trees is derived from it, CART determines the best tree by testing for error rates or costs. With sufficient data, the simplest method is to divide the sample into learning and test sub-samples. The learning sample is used to grow an overly large tree. The test sample is then used to estimate the rate at which cases are misclassified (possibly adjusted by misclassification costs). The misclassification error rate is calculated for the largest tree and also for every sub-tree.

The best sub-tree is the one with the lowest or near-lowest cost, which may be a relatively small tree. Cross validation is used if data are insufficient for a separate test sample. In the search for patterns in databases it is essential to avoid the trap of over fitting or finding patterns that apply only to the training data. CART's embedded test disciplines ensure that the patterns found will hold up when applied to new data. Further, the testing and selection of the optimal tree are an integral part of the CART algorithm. CART handles missing values in the database by

substituting surrogate splitters, which are back-up rules that closely mimic the action of primary splitting rules. The surrogate splitter contains information that is typically similar to what would be found in the primary splitter [32].

4 TACDSS Adaptation Using Takagi Sugeno FIS

We used the Adaptive Network based Fuzzy Inference System (ANFIS) framework [17] to develop the TACDSS based on a Takagi-Sugeno fuzzy inference system. The six-layered architecture of ANFIS is depicted in Figure 5.

Suppose there are two Input Linguistic Variables (ILV) X and Y and each ILV has three membership functions (MF) A_1, A_2 and A_3 and B_1, B_2 and B_3 respectively, then a Takagi-Sugeno type fuzzy *if-then* rule could be set up as

$$\text{Rule}_i : \textit{If } X \text{ is } A_i \text{ and } Y \text{ is } B_i \textit{ then } f_i = p_i X + q_i Y + r_i \qquad (5)$$

where i is an index i = 1,2..n and p, q and r are the linear parameters.

Some layers of ANFIS have the same number of nodes and nodes in the same layer have similar functions. Output of nodes in layer-l are denoted as $O_{l,i}$, where l is the layer number and i is neuron number of the next layer. The function of each layer is described as follows.

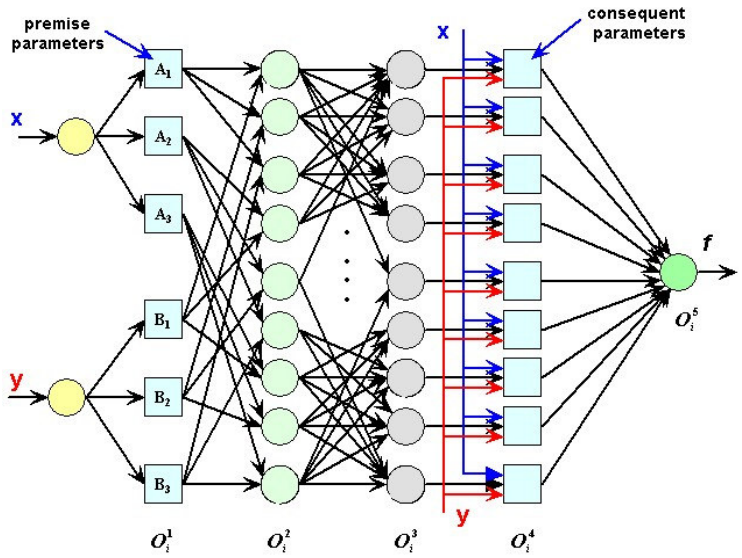


Figure 5. Architecture of ANFIS

Layer 1

The outputs of this layer is the input values of the ANFIS

$O_{1,x} = x$

$O_{1,y} = y$ (6)

For TACDSS the four inputs are *fuel status*, *weapons inventory levels*, *time intercept* and the *danger situation*.

Layer 2

The output of nodes in this layer are presented as $O_{l,ip,i,,}$ where *ip* is the ILV and *m* is the degree of membership function of particular MF.

$O_{2,x,i} = \mu_{Ai(x)}$ or $O_{2,y,i} = \mu_{Bi(y)}$ for $i = 1,2$ and 3 (7)

With three MFs for each input variable, "fuel status" has 3-membership functions: *full*, *half* and *low,* "time intercept" has *fast*, *normal* and *slow,* "weapon status" has *sufficient*, *enough* and *insufficient* and the "danger situation" has *very dangerous*, *dangerous* and *endangered*.

Layer 3

The output of nodes in this layer is the product of all the incoming signals, denoted by:

$O_{3,n} = W_n = \mu_{Ai}(x) \; x \; \mu_{Bi}(y)$ (8)

where $i = 1,2$ and 3, n is the number of the fuzzy rule. In general, any T-norm operator will perform the fuzzy 'AND' operation in this layer. With 4 ILV and 3 MFs for each input variable the TACDSS will have 81 ($3^4 = 81$) fuzzy *if-then* rules.

Layer 4

The nodes in this layer calculate the ratio of the i^{th} fuzzy Rule Firing Strength (RFS) to the sum of all RFS.

$$O_{4,n} = \overline{w_n} = \frac{w_n}{\sum_{n=1}^{81} w_n} \text{ where } n = 1,2,..,81 \quad (9)$$

The number of nodes in this layer is the same as the number of nodes in layer-3. The outputs of this layer are also called normalized firing strengths.

Layer 5

The nodes in this layer are adaptive, defined as:

$O_{5,n} = \overline{w_n} f_n = \overline{w_n} (p_n x + q_n y + r_n)$ (10)

where p_n, q_n, r_n are the rule *consequent parameters*. This layer also has the same number of nodes as layer-4 (81 numbers).

Layer 6

The single node in this layer is responsible for the defuzzification process using the center of gravity technique to compute the overall output as the summation of all the incoming signals:

$$O_{6,1} = \sum_{n=1}^{81} \overline{w_n} f_n = \frac{\sum_{n=1}^{81} w_n fn}{\sum_{n=1}^{81} w_n} \tag{11}$$

ANFIS makes use of a mixture of backpropagation to learn the premise parameters and least mean square estimation to determine the consequent parameters. Each step in the learning procedure comprises two parts: In the first part the input patterns are propagated, and the optimal conclusion parameters are estimated by an iterative least mean square procedure, while the antecedent parameters (membership functions) are assumed to be fixed for the current cycle through the training set. In the second part the patterns are propagated again, and in this epoch, backpropagation is used to modify the antecedent parameters, while the conclusion parameters remain fixed. This procedure is then iterated. Details are given below [17].

$$\begin{aligned}
\text{ANFIS output } f = O_{6,1} &= \frac{w1}{\sum_n wn} f1 + \frac{w2}{\sum_n wn} f2 + \ldots + \frac{wn}{\sum_n wn} fn \\
&= \overline{w_1}(p_1 x + q_1 y + r_1) + \overline{w_2}(p_2 x + q_2 y + r_2) + \ldots + \overline{w_n}(p_n x + q_n y + r_n) \\
&= (\overline{w_1} x)p_1 + (\overline{w_1} y)q_1 + \overline{w_1} r1 + (\overline{w_2} x)p_2 + (\overline{w_2} y)q_2 + \overline{w_2} r_2 + \ldots + \\
&\quad (\overline{w_n} x)p_n + (\overline{w_n} y)q_n + \overline{w_n} r_n
\end{aligned} \tag{12}$$

where n is the number of nodes in layer 5. From this, the output can be rewritten as

$$f = F(i,S) \tag{13}$$

where F is a function, i is the vector of input variables and S is a set of total parameters of consequent of the n^{th} fuzzy rule. If there exists a composite function H such that H o F is linear in some elements of S, then these elements can be identified by the least square method. If the parameter set is divided into two sets S_1 and S_2, defined as:

$$S = S_1 \oplus S_2 \tag{14}$$

where $\oplus$ represents direct sum and o is the product rule, such that H o F is linear in the elements of S_2, the function f can be represented as:

$$H(f) = H \text{ o } F(I,S) \tag{15}$$

Givens values of S_1, the S training data can be substituted into equation 15. $H(f)$ can be written as the matrix equation of $AX = Y$.
where X is an unknown vector whose elements are parameters in S_2.
If $|S_2| = M$ (M being the number of linear parameters) then the dimensions of matrices A, X and Y are $P \times M$, $M \times l$ and $P \times l$, respectively. This is a standard linear least-squares problem and the best solution of X that minimizes $||AX - Y||^2$ is the least square estimate (LSE) X^*

$$X^* = (A^T A)^{-1} A^T Y \tag{16}$$

where A^T is the transpose of A, $(A^T A)^{-1} A^T$ is the pseudo inverse of A if $A^T A$ is a non-singular. Let the i^{th} row vector of matrix A be a_i^T and the i^{th} element of Y be y_i^T, then X can be calculated as:

$$X_{i+1} = X_i + S_{i+1} a_{i+1} (y_i^T - y_{i+1}^T - a_{i+1}^T X_i) \tag{17}$$

$$S_{i+1} = Si - \frac{S_i a_{i+1} y_{i+1}^T - S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \; I = 0,1,\ldots, P - 1 \tag{18}$$

The LSE X^* is equal to X_p. The initial conditions of X_{i+1} and S_{i+1} are $X_0 = 0$ and $S_0 = \gamma I$, where γ is a positive large number and I is the identity matrix of dimension $M \times M$.

When hybrid learning is applied in batch mode, each epoch is composed of a forward pass and a backward pass. In the forward pass, the node output I of each layer is calculated until the corresponding matrices A and Y are obtained. The parameters of S_2 are identified by the pseudo inverse equation as mentioned above. After the parameters of S_2 are obtained, the process will compute the error measure for each training data pair. In the backward pass, the error signals (the derivatives of the error measure with respect to each node output) propagates from the output to the input end. At the end of the backward pass, the parameter S_1 is updated by the steepest descent method as follows:

$$\alpha = -\eta \frac{\partial E}{\partial \alpha} \tag{19}$$

where α is a generic parameter and η is a learning rate and E is an error measure .

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha}\right)^2}} \tag{20}$$

where k is the step size.

For the given fixed values of parameters in S_1, the parameters in S_2 are guaranteed to be global optimum points in the S_2 parameters space due to the choice of the

squared error measure. This hybrid learning method can decrease the dimension of the search space using the steepest descent method, and can reduce the time needed to reach convergence. The step size k will influence the speed of convergence. Observation shows that if k is small, the gradient method will closely approximate the gradient path; convergence will be slow since the gradient is being calculated many times. If the step size k is large, convergence will initially be very fast. Based on these observations the step size k is updated by the following two heuristic rules[17]:

- If E undergoes four continuous reductions then increase k by 10%, and
- If E undergoes continuous combinations of increase and decrease, then reduce k by 10%.

5 TACDSS Adaptation Using Mamdani FIS

We have made use of the Fuzzy Neural Network (FuNN) framework [18] for learning the Mamdani-Assilian fuzzy inference method. A functional block diagram of the FuNN model is depicted in Figure 6 [19]; it consists of two phases of learning.

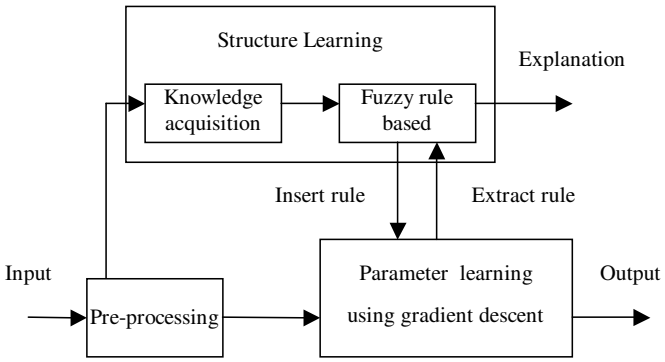


Figure 6. A general schematic diagram of the hybrid fuzzy neural network

The first phase is the structured learning (*if-then* rules) using the knowledge acquisition module. The second phase is the parameter learning for tuning membership functions to achieve a desired level of performance. FuNN uses a gradient descent learning algorithm to fine-tune the parameters of the fuzzy membership functions. In the connectionist structure, the input and output nodes represent the input states and output control-decision signals, respectively, while in the hidden layers, there are nodes functioning as quantification of membership functions (MFs) and *if-then* rules. We used the simple and straightforward method proposed by Wang and Mendel [38] for generating fuzzy rules from numerical input-output training data. The task here is to generate a set of fuzzy rules from the desired input-output pairs and then use these fuzzy rules to determine the complete structure of the TACDSS.

Suppose we are given the following set of desired input (x_1, x_2) and output (y) data pairs (x_1, x_2, y): (0.6, 0.2; 0.2), (0.4, 0.3; 0.4). In TACEDSS, the input variable *fuel reserve* has a degree of 0.8 in *half*, a degree of 0.2 in *full.* Similarly, the input variable *time intercept* has a degree of 0.6 in e*mpty* and 0.3 in *normal*. Secondly, assign x_1^i, x_2^i, and y^i to a region that has maximum degree. Finally, obtain one rule from one pair of desired input-output data, for example:

(x_1^1, x_2^1, y^1) => [x_1^1 (0.8 in *half*), x_2^1 (0.2 in *fast*), y^1 (0.6 in *acceptable*)],

- R_1: if x_1 is *half* and x_2 is *fast*, then y is *acceptable* (18)

(x_1^2,x_2^2,y^2), => [x_1(0.8 in *half*), x_2(0.6 in *normal*),y^2(0.8 in *acceptable*)],

- R_2: if x_1 is *half* and x_2 is *normal*, then y is *acceptable* (19)

Assign a degree to each rule. To resolve a possible conflict problem, that is, rules having the same antecedent but a different consequent, and to reduce the number of rules, we assign a degree to each rule generated from data pairs and accept only the rule from a conflict group that has a maximum degree. In other words, this step is performed to delete redundant rules, and therefore obtain a concise fuzzy rule base. The following product strategy is used to assign a degree to each rule. The degree of the rule denoted by:

Ri : if x_1 is A and x_2 is B, then y is $C(w_i)$ (20)

The rule weight is defined as:

$$w_i = \mu_A(x_1)\mu_B(x_2)\mu_c(y) \quad (21)$$

For example in the TACE, R_1 has a degree of

$$W_1 = \mu_{half}(x_1)\ \mu_{fast}(x_2)\ \mu_{acceptable}(y) = 0.8 \text{ x } 0.2 \text{ x } 0.6 = 0.096 \quad (22)$$

and R_2 has a degree of

$$W2 = \mu_{half}(x_1)\ \mu_{normal}(x_2)\ \mu_{acceptable}(y) = 0.8 \text{ x } 0.6 \text{ x } 0.8 = 0.384 \quad (23)$$

Note that if two or more generated fuzzy rules have the same preconditions and consequents, then the rule that has maximum degree is used. In this way, assigning the degree to each rule, the fuzzy rule base can be adapted or updated by the relative weighting strategy: the more task-related the rule becomes, the more weight degree the rule gains. As a result, not only is the conflict problem resolved, but also the number of rules is reduced significantly. After the structure-learning phase (*if-then* rules), the whole network structure is established, and the network enters the second learning phase to optimally adjust the parameters of the membership functions using a gradient descent learning algorithm to minimise the error function:

$$E = \frac{1}{2}\sum_{x}\sum_{l=1}^{q}(d_1 - y_l)^2 \quad (24)$$

where d and y are the target and actual outputs for an input x. This approach is very similar to the *MF* parameter tuning in ANFIS.

5.1 Membership Function Parameter Optimisation Using EAs

We have investigated the usage of evolutionary algorithms (EAs) to optimise the number of rules and fine-tune the membership functions [35]. Given that the optimisation of fuzzy membership functions may involve many changes to many different functions, and that a change to one function may affect others, the large possible solution space for this problem is a natural candidate for a EA based approach. This has already been investigated in [25], and has been shown to be more effective than manual alteration. A similar approach has been taken to optimise membership function parameters. A simple way is to represent only the parameter showing the centre of MF's to speed up the adaptation process and to reduce spurious local minima over the center and width.

The EA module for adapting FuNN is designed as a stand-alone system for optimising the MF's if the rules are already available. Both antecedent and consequent MF's are optimised. Chromosomes are represented as strings of floating-point numbers, rather than strings of bits. In addition, mutation of a gene is implemented as a re-initialisation, rather than an alteration of the existing allegation. Figure 7 shows the chromosome structure including the input and output MF parameters. One point crossover is used for the chromosome reproduction.

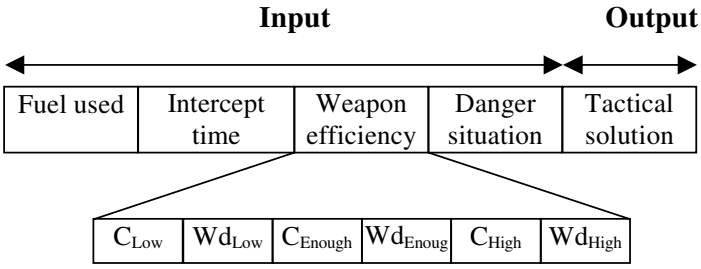


Figure 7. The chromosome of the centres of inputs and output MF's

6. Experimental Results for Developing the TACDSS

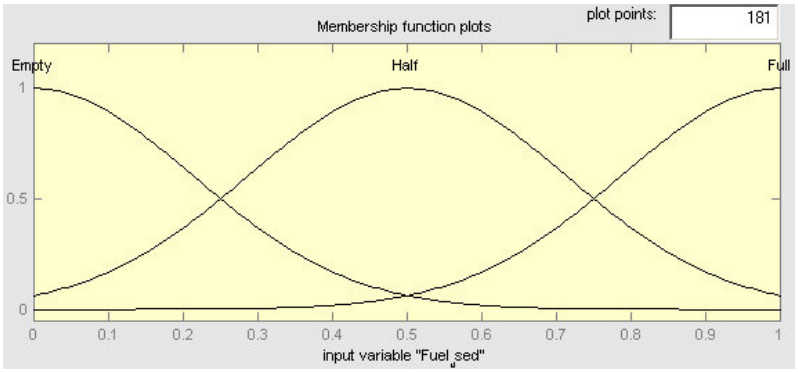
Our master data set comprised 1000 numbers. To avoid any bias on the data, we randomly created two training sets (Dataset A - 90% and Dataset B- 80%) and test data (10% and 20 %) from the master dataset. All experiments were repeated three times and the average errors are reported here.

6.1 Takagi Sugeno FIS

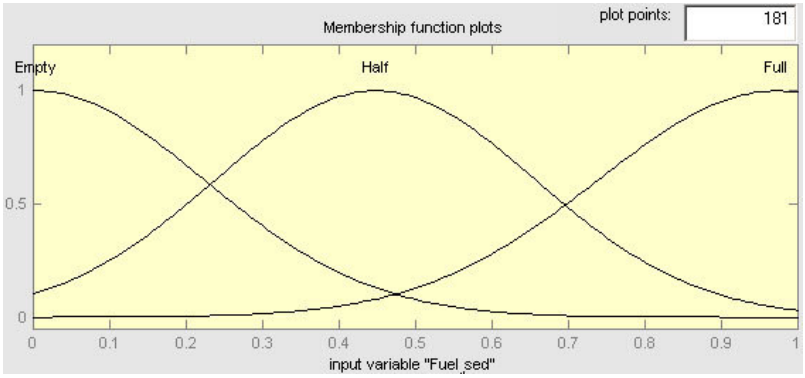
In addition to the development of the Takagi Sugeno FIS, we also investigated the behaviour of TACEDSS for different membership functions (shape and quantity per ILV). We also explored the importance of different learning methods for fine-tuning the rule antecedents and consequents. Keeping the consequent parameters constant, we fine-tuned the membership functions alone using the gradient descent technique (backpropagation). Further, we used the hybrid learning method wherein the consequent parameters were also adjusted according to the least squares algorithm. Even though backpropagation is faster than the hybrid technique, learning error and decision scores were better for the latter technique. We used three Gaussian MFs for each ILV. Figure 8 shows the three MFs for the "*fuel reserve*" ILV before and after training. The fuzzy rule consequent parameters before training was set to zero and the parameters were learned using the hybrid learning approach.

Comparison of the shape of membership functions of FIS

In this section, we demonstrate the importance of the shape of membership functions. We used the hybrid-learning technique and each ILV had three MFs. Table 3 shows the convergence of the training RMSE during the 15 epoch learning using four different membership functions for 90% and 80% training data. 81 fuzzy if-then rules were created initially using a grid-partitioning algorithm. We considered Generalised bell, Gaussian, trapezoidal and isosceles triangular membership functions. Figure 9 illustrates the training convergence curve for different MF's.



(a)



(b)

Figure 8. The membership function of the "fuel reserve" ILF (a) before and (b) after learning

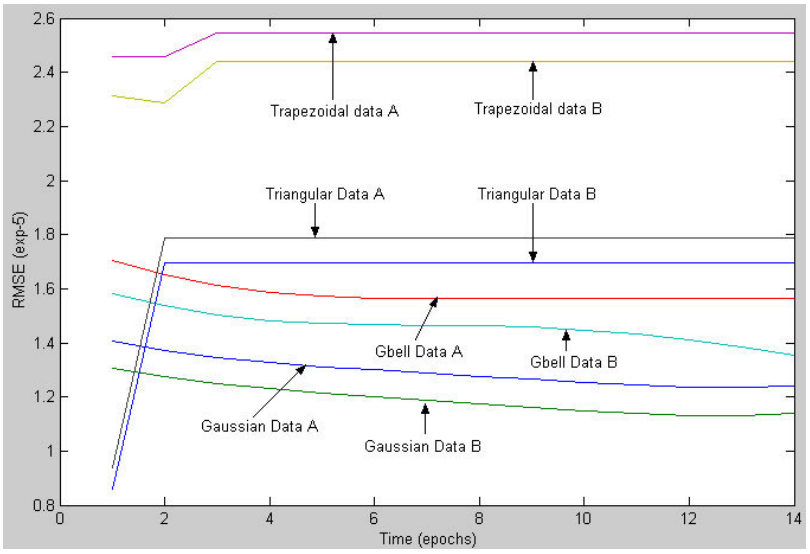


Figure 9. Effect on training error for the different membership functions

Table 3. Learning performance showing the effect of the shape of MF

Root Mean Squared Error (E- 05)								
	Gaussian		**Gbell**		**Trapezoidal**		**Triangular**	
Epochs	**Data A**	**Data B**	**Data A**	**Data B**	**Data A**	**Data B**	**Data A**	**Data B**
1	1.406	1.305	1.706	1.581	2.459	2.314	0.9370	0.8610
2	1.372	1.274	1.652	1.537	2.457	2.285	1.789	1.695
3	1.347	1.249	1.612	1.505	2.546	2.441	1.789	1.695
4	1.328	1.230	1.586	1.483	2.546	2.441	1.789	1.695
5	1.312	1.214	1.571	1.471	2.546	2.441	1.789	1.695
6	1.300	1.199	1.565	1.466	2.546	2.441	1.789	1.695
7	1.288	1.186	1.564	1.465	2.546	2.441	1.789	1.695
8	1.277	1.173	1.565	1.464	2.546	2.441	1.789	1.695
9	1.265	1.160	1.565	1.459	2.546	2.441	1.789	1.695
10	1.254	1.148	1.565	1.448	2.546	2.441	1.789	1.695
11	1.243	1.138	1.565	1.431	2.546	2.441	1.789	1.695
12	1.236	1.132	1.565	1.409	2.546	2.441	1.789	1.695
13	1.234	1.132	1.565	1.384	2.546	2.441	1.789	1.695
14	1.238	1.138	1.565	1.355	2.546	2.441	1.789	1.695
Test RMSE	1.44	1.22	1.78	1.36	2.661	2.910	1.8583	1.8584

As evident from Table 3 and Figure 9, the lowest training and testing error was obtained using Gaussian MF.

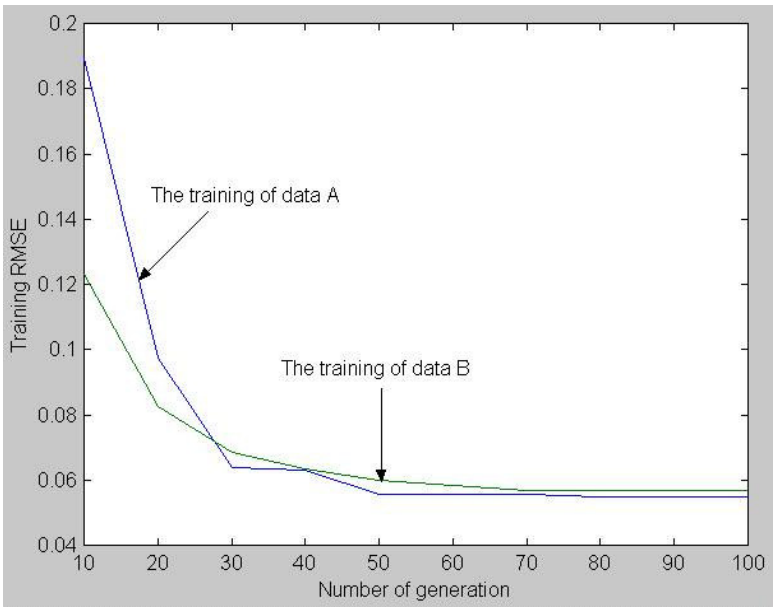


Figure 10. Convergence of training using evolutionary algorithms

6.2 Mamdani Fuzzy Inference System

We used the FuzzyCOPE [39] to investigate the tuning of membership functions using backpropagation and evolutionary algorithms. The learning rate and momentum were set at 0.5 and 0.3 respectively for 10 epochs. We obtained training RMSE of 0.2865 (Data A) and 0.2894 (Data B). We further improved the training performance using evolutionary algorithms. The following settings were used for the evolutionary algorithm parameters.

Population size = 50
Number of generations = 100
Mutation rate = 0.01

We used the tournament selection strategy and Figure 10 illustrates the learning convergence during the 100 generations for Datasets A and B. 54 fuzzy *if-then* rules were extracted after the learning process. Table 4 summarizes the training and test performance.

Table 4. Training and test performance of Mamdani FIS using EAs

Root Mean Squared Error (RMSE)			
Data A		Data B	
Training	Test	Training	Test
0.0548	0.0746	0.0567	0.0612

6.3 Neural Networks

We used 30 hidden neurons for Data A and 32 hidden neurons for Data B. We used a trial-and-error approach to finalize the architecture of the neural network. We used the scaled conjugate gradient algorithm to develop the TACEDSS. Training was terminated after 1000 epochs. Figure 11 depicts the convergence of training during 1000 epochs learning. Table 5 summarizes the training and test performance.

6.4 Classification and Adaptive Regression Trees

We used the CART [9] simulation environment to develop the decision trees. We selected the minimum cost tree regardless of tree size. Figures 12 and 13 illustrate the variation of error with reference to the number of terminal nodes for Datasets A and B. For Data A, the developed tree has 122 terminal nodes as shown in Figure 14 while for Data B the rest of tree had 128 terminal nodes as depicted in Figure 15. Training and test performance are summarized in Table 5.

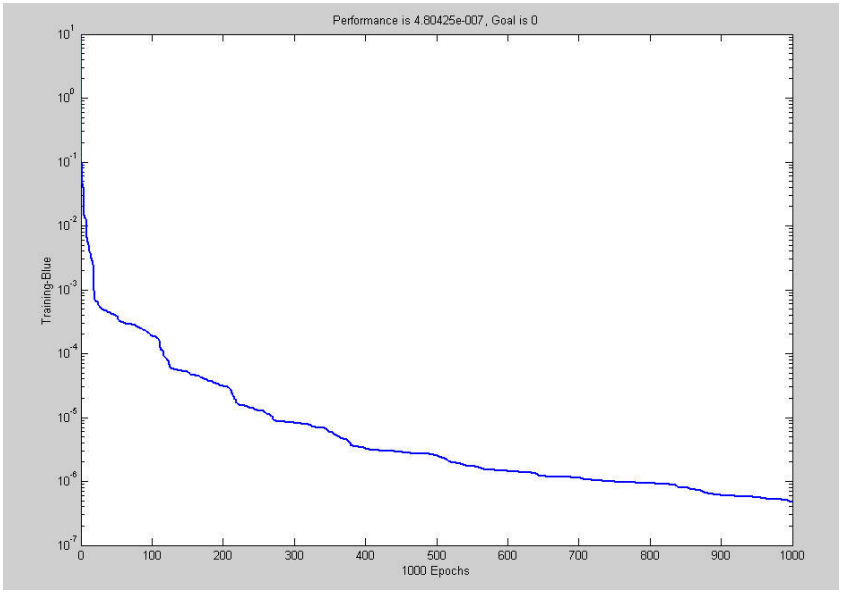


Figure 11. Neural network training using SCGA

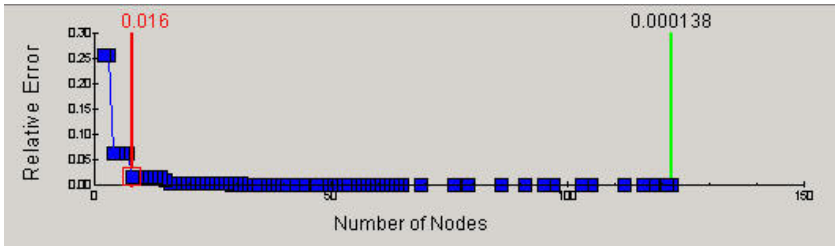


Figure 12. Dataset A: Variation of relative error for the number of terminal nodes

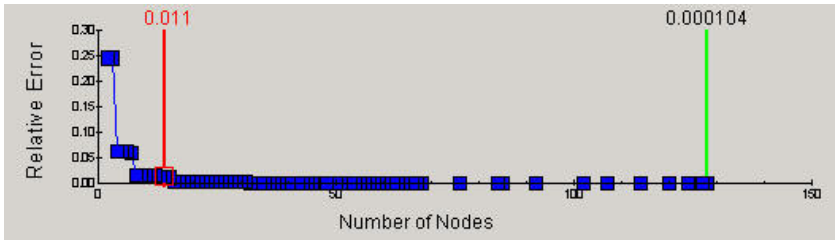


Figure 13. Dataset B: Variation of relative error for the number of terminal nodes

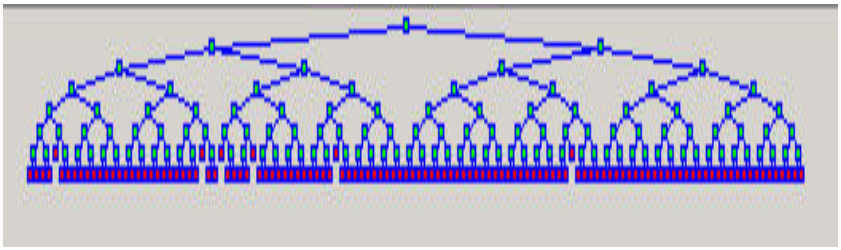


Figure 14. Dataset A: Developed decision tree with 122 nodes

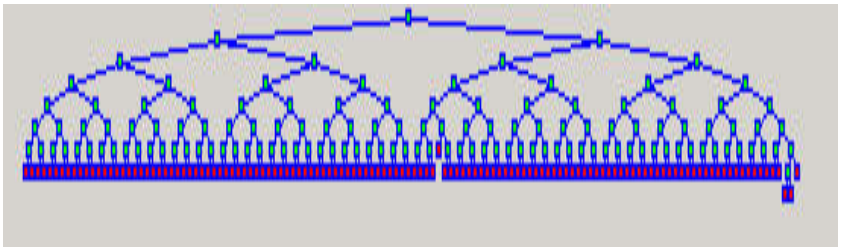


Figure 15. Dataset B: Developed decision tree with 128 nodes

Table 5. Training and test performance of neural networks and decision trees

	Data A		**Data B**	
	Training	**Testing**	**Training**	**Testing**
	RMSE			
CART	0.00239	0.00319	0.00227	0.00314
Neural Network	0.00105	0.00095	0.00041	0.00062

Figure 16 compares the performance of the different intelligent paradigms used in developing the TACDSS (for clarity we have chosen only 20% of the test results of Dataset B).

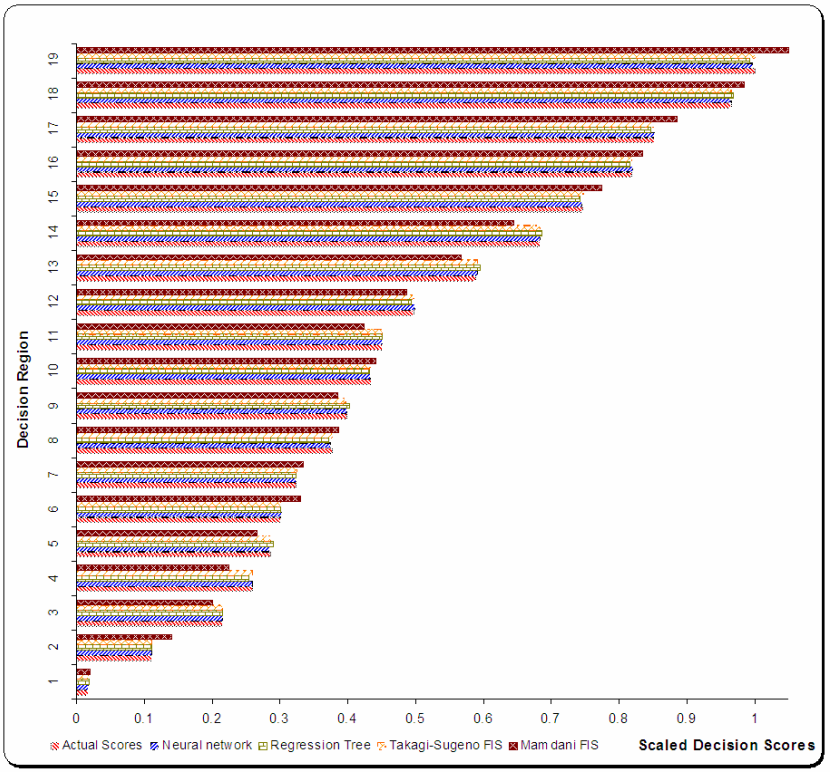


Figure 16. Test results illustrating the efficiency the different intelligent paradigms for developing the TACEDSS.

7. Discussions

The focus of this research is to create accurate and highly interpretable (using rules or tree structures) decision support systems for a tactical air combat environment problem.

Experimental results using two different datasets revealed the importance of fuzzy inference system to construct accurate decision support systems. As expected, by providing more training data (90% of the randomly choosed master data set), the models were able to learn and generalize more accurately. Takagi-Sugeno fuzzy inference system has the lowest RMSE on both test datasets. Since learning involves a complicated procedure, the training process of the Takagi-Sugeno fuzzy inference system took longer compared to Mamdani-Assilian fuzzy inference method - hence there is a compromise between performance and computational complexity (training time). Our experiments using different

membership function shapes also reveal that Gaussian membership function is the 'optimum' shape for the constructing accurate decision support systems.

Neural networks can no longer be considered as 'black boxes'. Recent research has revealed that it is possible to extract rules from trained neural networks. In our experiments we used a neural network trained using the scaled conjugate gradient algorithm. Results depicted in Figure 5 also reveal that the trained neural network could not learn and generalize accurately compared with the Takagi Sugeno fuzzy inference system. The proposed neural network outperformed Mamdani-Assilian fuzzy inference system and CART.

Two important features of the developed classification and regression tree are its easy interpretability and low complexity. Due to its one pass training approach; the CART algorithm also has the lowest computational load. For Dataset A, the best results were achieved using 122 terminal nodes (relative error = 0.00014). As shown in Figure 12, when the numbers of terminal nodes were reduced to node 14, the relative error increased to 0.016. For Dataset B, the best results could be achieved using 128 terminal nodes (relative error = 0.00010). As shown in Figure 13, when the terminal nodes were reduced to node 14, the relative error increased to 0.011.

8. Conclusions

In this Chapter, we have presented different soft computing and machine learning paradigms for developing a tactical air combat decision support system. The techniques explored were a Takagi-Sugeno fuzzy inference system trained using neural network learning techniques, a Mamdani-Assilian fuzzy inference system trained using evolutionary algorithms and neural network learning, feedforward neural network trained using the scaled conjugate gradient algorithm, and classification and adaptive regression trees.

The empirical results clearly demonstrate that all these techniques are reliable and could be used for constructing more complicated decision support systems. Experiments on the two independent data sets also reveal that the techniques are not biased on the data itself. Compared to neural networks and regression trees, the Takagi-Sugeno fuzzy inference system has the lowest RMSE and the Mamdani-Assilian fuzzy inference system the highest RMSE. In terms of computational complexity, perhaps regression trees are best since they use a one pass learning approach when compared to the many learning iterations required by all other considered techniques. An important advantage of the considered models is fast learning, easy interpretability (*if-then* rules for fuzzy inference systems, m-of-n rules from a trained neural network [30] and decision trees), efficient storage and retrieval capacities and so on. It may also be concluded that fusing different intelligent systems knowing their strengths and weakness could help to mitigate the limitations and take advantage of the opportunities to produce more efficient decision support systems than those built with stand alone systems.

Our future work will be directed towards optimization of the different intelligent paradigms [2], which we have already used and also to develop new adaptive reinforcement learning systems that can update the knowledge from data especially when no expert knowledge is available.

Acknowledgements

Authors would like to thank Professor John Fulcher for the editorial comments which helped to improve the clarity of this chapter.

References

[1] Abraham, A., Neuro-Fuzzy Systems: State-of-the-Art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Springer-Verlag Germany, Jose Mira and Alberto Prieto (Eds.), Granada, Spain, pp. 269-276, 2001.

[2] Abraham, A., Optimization of Evolutionary Neural Networks Using Hybrid Learning Algorithms, IEEE International Joint Conference on Neural Networks (IJCNN'02), 2002 IEEE World Congress on Computational Intelligence, Hawaii, IEEE Press, Volume 3, pp. 2797-2802, 2002.

[3] Abraham, A. and Nath, B., Evolutionary Design of Neuro-Fuzzy Systems - A Generic Framework, Proceedings of The 4-th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems (JA2000 - Japan), Published by the National Defence Academy (Japan) and University of New South Wales (Australia), Akira Namatame et al (Editors), Japan, pp. 106-113, 2000.

[4] Abraham, A. and Nath, B., Evolutionary Design of Fuzzy Control Systems - An Hybrid Approach, Proceedings of The Sixth International Conference on Control, Automation, Robotics and Vision, (ICARCV 2000 - Singapore), (CD ROM Proceeding), Wang J.L. (Editor), Singapore, 2000.

[5] Adibi, J., Ghoreishi, A., Fahimi, M. and Maleki, Z., Fuzzy logic information theory hybrid model for medical diagnostic expert system, Proceedings of the Twelfth Southern Biomedical Engineering Conference, Tulane University, New Orleans, Louisiana, pp. 211-213, April 1993.

[6] Breiman, L., Friedman, J., Olshen, R., and Stone, C. J, Classification and Regression Trees, Chapman and Hall, New York, 1984.

[7] Cattral R., Oppacher F. & Deogo D (1999), Rule acquisition with a genetic algorithm, Proceedings of the Congress on Evolution computation, IEEE Press, Washington, DC USA, Volume 1, pp. 125-129, 6-9 July 1999.

[8] Chappel, A. R., Knowledge-based reasoning in the Paladin tactical decision generation system. Proceedings of the 11th AIAA Digital Avionics Systems Conference, Seattle, WA USA, pp. 155-160, October 5-81992.

[9] Classification and Regression Trees: CART®, http://www.salford-systems.com/products-cart.html (accessed on 20 July 2002)

[10] Fogel, D., Evolutionary Computation: Towards a New Philosophy of Machine Intelligence, 2nd Edition, IEEE press, Piscataway, NJ, 1999.

[11] Gorzalczany, M. B, An idea of the application of fuzzy neural networks to medical decision support systems, Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE '96, Warsaw Poland, Volume 1, pp. 398-403, 17-20 June 1996.

[12] Holland, J.H., Kaufmann M. and Altos L., Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning: An Artificial Intelligence Approach, Volume II, pp. 593--623. Morgan Kaufmann, San Mateo, CA., 1986.

[13] Holsapple C.W. and Whinston A.B., Decision Support Systems: A Knowledge Based Approach, West Publishing Company, Minneapolis, 1996.

[14] Hung, C. C. November, Building a Neuro-Fuzzy Learning Control System, AI Expert, pp. 40-49, 1993.

[15] Ichimura, T., Takano, T. and Tazaki, E., Reasoning and learning method for fuzzy rules using neural networks with adaptive structured genetic algorithm, IEEE International Conference on Systems, Man and Cybernetics. Intelligent system for the 21st century, Vancouver, Canada, Volume 4, pp. 3269-3274, 8-11 October 1995.

[16] Jagielska I., Linguistic rule extraction from neural networks for descriptive datamining, The proceedings of second conference on knowledge-based intelligent electronic systems, KES'98, IEEE Press, Adelaide South Australia, Volume 2, pp.89-92, 21-23 April 1998.

[17] Jang, R, Neuro-Fuzzy Modeling: Architectures, Analyses and Applications, PhD Thesis, University of California, Berkeley, July 1992.

[18] Kasabov, N., Kim. J. S. & Gray, A. R., FUNN – A fuzzy neural network architecture for adaptive learning and knowledge acquisition, Information Sciences, Volume 101, Issue 3, pp. 155-175, 1996.

[19] Kasabov, N., Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems, Fuzzy Sets and Systems, Vol.82, pp. 135-149, 1996.

[20] Kasabov, N., Evolving Fuzzy Neural Networks for Supervised/Unsupervised On-Line Knowledge-Based Learning, IEEE Transaction of Systems, man and Cybernetic, Part B – Cybernetic, Vol. 31, Iss. 6, December 2001

[21] Kearney, D. A. and Tran, C. M., Optimal fuzzy controller design for minimum rate of change of acceleration in a steel uncoiler, Control95 Meeting the Challenge of Asia Pacific Growth, University of Melbourne Australia, Vol. 2, pp. 393-397, September 1995.

[22] Lee, C. C., Fuzzy logic control systems: Fuzzy Logic Controller- Part I & II, IEEE Transactions on systems, man, and cybernetics, vol. 20, no. 2, pp. 404-435, 1990.

[23] Lin, T. Y., Cercone, N., Rough sets and data mining – Analysis of imprecise data, Kluwer Academic publishers, 1997.

[24] Mamdani E H and Assilian S, An experiment in Linguistic Synthesis with a Fuzzy Logic Controller, International Journal of Man-Machine Studies, Vol. 7, No.1, pp. 1-13, 1975.

[25] Mang, G., Lan, H. and Zhang, L., A genetic-base method of generating fuzzy rules and membership function by learning from examples, Proceedings of International Conference on Neural Information (ICONIP'95), Beijing, Vol. 1, pp. 335-338, 30 Oct - 3 Nov 1995.

[26] Militallo, L. G. Hutton, R. J. B., Applied cognitive task analysis (ACTA): A practitioner's toolkit for understanding cognitive, Ergonomics, Vol. 41, Issue 11, pp 1618-1642, 1998.

[27] Moller, A F., A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, Neural Networks, Volume (6), pp. 525-533, 1993.

[28] Perneel, C. and Acheroy, M., Fuzzy reasoning and Genetic Algorithm for decision making problems in uncertain Environment, Industrial Fuzzy control and Intelligent Systems Conference and NASA joint technology workshop on Neural Networks and Fuzzy Logic NAFIPS/IFIS/NASA 94, San Antonio, Texas, pp. 115-120, 12-13 December 1994.

[29] Sanderson, P. M., Cognitive work analysis and the analysis, design, evaluation of human computer interactive systems, Proceedings of the Annual Conference of the Computer-Human Interaction Special Interest Group (CHISIG) of the Ergonomics Society of Australia (OzCHI98). Adelaide, Australia, pp. 40-45, 29 Nov- 4 Dec 1998.

[30] Setiono, R, Extracting M-of-N rules from trained neural networks, IEEE Transactions on Neural Networks, Vol. 11, No. 2, pp. 512-519, 2000.

[31] Setiono, R., Leow, W.K. and Zurada J.M., Extraction of rules from artificial neural networks for nonlinear regression, IEEE Transactions on Neural Networks, Vol. 13, No. 3, pages 564-577, 2002.

[32] Steinberg, D. and Colla, P. L., CART: Tree-Structured Nonparametric Data Analysis, San Diego, CA: Salford Systems, 1995.

[33] Sugeno, M, Industrial Applications of Fuzzy Control, Elsevier Science Pub Co., 1985.

[34] Tran C. and Zahid, Q, Intelligent techniques for decision support in tactical environment, Land Warfare Conference 2000, Proceedings of the Warfighting in Complex Terrain, published by Australia Defence Science & Technology Organisation, Melbourne Australia, pp. 403-411. September 2000.

[35] Tran, C., Jain, L., and Abraham, A., Adaptation of Mamdani Fuzzy Inference System Using Neuro - Genetic Approach for Tactical Air Combat

Decision Support System, 15th Australian Joint Conference on Artificial Intelligence (AI'02), Canberra, Australia, Springer Verlag Germany, pp. 402-410, December 2002.

[36] Tran C., Jain, L. and Abraham, A., Adaptive database learning in decision support system using evolutionary fuzzy systems: A generic framework, Hybrid Information Systems, Advances in Soft Computing, Abraham A. and Koppen M. (Eds.), Physica Verlag Germany, pp. 237-252, 2002.

[37] Tran, C., Jain, L. and Abraham, A., TACDSS: Adaptation of a Takagi – Sugeno Hybrid Neuro-Fuzzy System, Seventh Online World Conference on Soft Computing in Industrial Applications (WSC7), Springer Verlag Germany, 2002.

[38] Wang L. X. and Mendel J. M., Generating Fuzzy Rules by Learning from Examples, IEEE Transcation on System, Man and Cybernetics, Vol. 22, Issue 6, pp. 1414-1427, 1992.

[39] Watts, M., Woodford, B. & Kasabov, N., FuzzyCOPE: A Software Environment for Building Intelligent Systems – The Past, The Present and the Future, ICONIP/ANZIIS/ANNES'99 Workshop, Dunedin/Queenstown, New Zealand, pp. 188-192, 22-24 November 1999.

[40] Zurada, J.M., Introduction to Artificial Neural Systems, West Publishing Company, 1992.

[41] Takagi, T. and Sugeno, M., Derivation of fuzzy control rules from human operator's control actions, Proceeding of the IFAC Symposium. on Fuzzy Information, Knowledge representation and decision analysis. Marseilles, France, pp. 55-60, 1983.

[42] Tan, K. C., Yu, Q., Heng, C. M. and Lee T. H., Evolutionary computing for knowledge discovery in medical diagnosis, Artificial Intelligence in Medicine, Volume 27, Issue 2, pp. 129-154, February 2003.

[43] Tan, K. C. and Li, Y., Performance-based control system design automation via evolutionary computing, Engineering Applications of Artificial Intelligence, Volume 14, Issue 4, Pages 473-486, August 2001,.

[44] Cortés, P., Larrañeta, J., Onieva, L., García, J. M. and Caraballo, M. S., Genetic algorithm for planning cable telecommunication networks, Applied Soft Computing, Volume 1, Issue 1, pp. 21-33, June 2001,.

[45] Ponnuswamy, S., Amin, M. B., Jha, R. and Castañon, D. A., A C3I Parallel Benchmark Based on Genetic Algorithms Implementation and Performance Analysis, Journal of Parallel and Distributed Computing, Volume 47, Issue 1, pp. 23-38, 25 November 1997.

[46] Abraham, A. and Nath, B., A Neuro-fuzzy Approach for Modelling Electricity Demand in Victoria, Applied Soft Computing, Volume 1, Issue 2, pp. 127-138, August 200.