Evolving Fuzzy Neural Networks in Adaptive Knowledge Bases to support Task-Oriented Decision Making for Sensor Management

Fook Wai Kong, Gee Wah Ng, Yuan Sin Tan, Chung Huat Tan
Advanced Analysis And Fusion Laboratory
DSO National Laboratories
20, Science Park Drive, Singapore 118230
{kfookwai, ngeewah, tyuansin, tchunghu}@dso.org.sg

Abstract – In the area of Process Refinement under Level 4 Data Fusion of the JDL model [1,2], high-level sensor management is often performed by human operators manning sensor systems who constantly have to monitor the situational and sensor picture for critical events and dynamically employ myriad sensors' functions to carry out mission-specific tasks. To assist the human operators in dealing better with the intense pressure to perform effectively in such environments, adaptive knowledge bases capable of capturing human operators' behavioural patterns can be harnessed to augment the task-oriented decision making process of sensor management. However, the unique problem domain in which human operators exercise sensor management functions has direct impact on obtainable training data and imposes several performance requirements on the adaptive knowledge bases. Several rule-learning algorithms [4-7] do not readily fulfil the identified requirements and selecting a more suitable alternative constitutes the focus of this paper. This paper selects the adaptive online-learning Evolving Fuzzy Neural Network (EFUNN) [8,9] and details two algorithmic and one qualitative contribution that enhance EFUNN's ability to realize the construction of adaptive knowledge bases. The two algorithmic contributions consist of modifications of EFUNN's original learning mechanism to handle training records with outlying inputs and those with contradictory class outputs that characterise the obtainable training data. The qualitative contribution suggests how multiple EFUNNs can be mapped to respective task-oriented rule-sets giving rise to adaptive knowledge bases that assist the human operators in selecting the right observation tasks.

Keywords: Data fusion, sensor management, adaptive online learning, knowledge capturing, knowledge management, unsupervised fuzzy c-means clustering, Mahalanobis distance measure

1 Introduction

In the JDL Data Fusion model, Level 4 Data Fusion or Process Refinement is the layer that actively observes the inputs from the "lower" level data fusion functions and determines what tasks are to be performed based on predefined goal settings dependent on the mission objectives and assigns the appropriate sensor resources to fulfil the pertinent tasks. Selecting the pertinent tasks to deal with fast-paced ad-hoc situations in dynamic environments requires substantial expertise in the form of highly trained human operators. If the knowledge or experience possessed by seasoned operators can be captured and made readily accessible in the form of human-comprehensible knowledge rules, it becomes possible to exploit and replicate the same successes that are only achievable with the presence of experienced operators.

To assist the operators in dealing better with the intense pressure to perform optimally in such demanding fast-paced environments, adaptive knowledge bases capable of capturing users' behavioural patterns may be harnessed to offload some of the operators' burden by first extracting and replicating the tacit decision making logic practised by seasoned human operators, and incorporating those logic into an adaptive knowledge base. The knowledge base subsequently can be queried based on the environmental conditions for recommendations that the man-in-the-loop can choose to adopt in dealing with new situations whose conditions are similar to those encountered by the seasoned operators.

This paper is structured into the following sections:

Section 2: explains the decision making process involved in Sensor Management and how adaptive knowledge bases can better assist the human operators in carrying out this role.

Section 3: examines the problem nature in the context of sensor management and lists the requirements expected from the adaptive knowledge bases.

Section 4: lists the possible machine learning techniques that may be exploited in the construction of such adaptive knowledge bases and explains how the different techniques measure up in terms of fulfilling the requirements of adaptive knowledge bases for decision-making in sensor management.
Section 5: explains why the EFUNN is selected and the modifications required to make it more suited to the construction of adaptive knowledge bases

Section 6: explains how an EFUNN can be mapped to a task-oriented rule-set and how multiple EFUNNs give rise to a standalone adaptive knowledge base.

Section 7: summarises the algorithmic and qualitative contributions of this paper.

2 Task-Oriented Decision-Making

An important function within sensor management is the decision-making process that is primarily concerned with the making of critical decisions to determine the right actions to direct and control based on the inputs coming from the lower fusion layers namely Situational Awareness and Threat Assessment [3].

The human operator performs the high level sensor management function by observing the situational and sensor picture, monitors dynamic changes that are constantly developing and carefully decide what observation tasks need to be performed to fulfil the mission objectives. Examples of observation tasks include Target-Search, Track-Update and Identify-Friend-Foe (IFF). This is followed by the human sensor manager assigning the most appropriate sensor functions to support the pertinent tasks in response to developments in the input picture. In this paper, the emphasis is placed solely on devising the adaptive knowledge base component to support the human operator in selecting the right observation tasks.

An adaptive knowledge base that is capable of supporting the human operator in selecting the right task has to be able to capture human decision logic into comprehensible knowledge rules that can be queried for recommendations to handle situations whereby human expertise is not readily available. The knowledge capturing also needs to be able to handle idiosyncratic behaviours exhibited by different human operators while they go about carrying out the mission objectives. For instance, experienced operators can anticipate critical events better and perform actions at the appropriate timing while the less experienced ones tend to over-react prematurely when suspicious activities are first detected.

Because different human operators will potentially respond differently to different scenarios, each human operator tends to use a unique and different set of decision logic. For long-term maintenance of the captured knowledge rules, it becomes important that the adaptive knowledge base offers knowledge management functions to enable different operators to share and exchange with one another their own expertise in the form of knowledge bases. This can be done by allowing the human operators to have personal ownership of the knowledge bases whose rules can be accessed, exchanged and modified subjected to security policies. This way, different operators can continue to contribute to a central repository of knowledge bases and the shared expertise can be better leveraged upon within a larger knowledge management framework hence ensuring the longevity of decision-making expertise and experiences used for sensor management purposes.

The next section elaborates on the unique problem nature and spells out the special requirements that are expected from the adaptive knowledge bases.

3 Problem Nature and Requirements

Due to the complexities of military operations, it can be both expensive and counter-productive to simulate all possible scenarios and have trained human operators to sit-through and provide the most appropriate responses to all possible cases. This characteristic of the problem domain therefore tends to limit the amount of collectable training data and causes the data to be sparse in nature.

In capturing behavioural patterns of the human operators, the adaptive knowledge base needs to do it responsively and allows ad-hoc querying of its rules as and when an external user wishes to view the internal contents of the knowledge base. Therefore the learning algorithm that is chosen to realize the knowledge base has to adopt the online and incremental learning approach and it should not require many iterations just to converge on the training data. This is particularly relevant in cases where the human operator wants to interactively check the decision logic newly captured by the knowledge base during the training session without the need to wait for offline processing.

The adaptive knowledge base also has to be able to cope with contradictory training data and still produces a consistent set of rules eventually. Otherwise, undesirables in the form of ambiguous rules will proliferate within the knowledge base. An effective knowledge base when queried should produce recommendations that are consistent without any ambiguity to avoid confusing the human operator.

For the purpose of subsequent maintenance of multiple adaptive knowledge bases, special functions such as combining rules from multiple knowledge bases contributed by different individuals into consistent ones will become essential in the longer term.

4 Machine Learning Techniques

When selecting the right learning algorithm, the following have been identified as key issues and requirements that need to be addressed and met by the prospective algorithm before it can be considered as suitable for the construction of adaptive knowledge bases:

a. training data issues – limited and sparse, contradictory training records (similar input conditions but opposing class outputs)
b. training pace – fast, online and incremental learning
c. maintenance functions – rules' extraction, aggregation and pruning

Machine learning techniques capable of generating human comprehensible rules are first examined [4-7] and their corresponding advantages or shortcomings in relation to the above issues are indicated below.

Decision tree algorithms namely ID3 and C4.5 have to be retrained whenever new training data are available. These algorithms do not readily have provisions for online and incremental learning.

Knowledge based neural networks based on the traditional multi-layered feed-forward neural network requires many iterations for convergence. Fuzzy neural networks also goes for global optimisation of its weights which requires multiple-pass weight updating that makes it less suitable for learning in an online and incremental manner.

The EFUNN [8,9] is an extension to the fuzzy neural approach with a flexible connectionist structure that dynamically adapts to the characteristics of the training data. The EFUNN operates in an online learning manner, offers rules' extraction, aggregation and pruning features and therefore is a suitable candidate but its ability to handle training records with outlying inputs and training records with contradictory class outputs can still be further enhanced. The next section elaborates on the original EFUNN algorithm and explains the modifications performed on its learning mechanism to better handle the above issues.

5 Evolving Fuzzy Neural Network

The EFUNN [9] is an enhancement on top of the traditional fuzzy neural networks that switches from global optimisation to local element tuning, from multiple-pass learning to one-pass learning, from a fixed connectionist structure to a fluctuating one that allows for growing through insertion of nodes and shrinking through node aggregation and pruning.

The EFUNN comprises the lower layers that performs unsupervised clustering of the fuzzy input space and the upper layers that performs supervised learning or association of clusters to teacher-specified class outputs.

The unsupervised clustering attempts to tackle sparse training data and both the unsupervised and supervised learning require only single pass for the weights to be updated. The EFUNN also offers aggregation of rule nodes to facilitate merging of different rules as well as pruning of rule nodes in order to age out unwanted ones that are captured by the rule learning mechanism.

5.1 EFUNN's Network Structure

Figure 1 below illustrates the network structure of the EFUNN and this section explains the mathematical workings of the algorithm.

A is the vector of activation outputs emerging from J rule nodes in the rule layer. The individual rule node's activation output is based on:

$$a_j = f_1\left(D\left(X', w_j\right)\right) \tag{1}$$

where $j \in \{1 \ldots J\}$ and $D(d_1, d_2)$ is a distance function that quantifies the degree of difference between d_1 and d_2 fuzzy membership degree vectors. One such distance function is the local normalised fuzzy distance which is based on the Euclidean norm.

$$D(d_1, d_2) = \|d_1 - d_2\| / \|d_1 + d_2\| \tag{2}$$

The function f_1 in Equation 1 can take the form of a linear function such as $f_1(u) = 1 - u$. Y' is the vector of activation outputs emerging from K fuzzy output nodes in the fuzzy output layer.

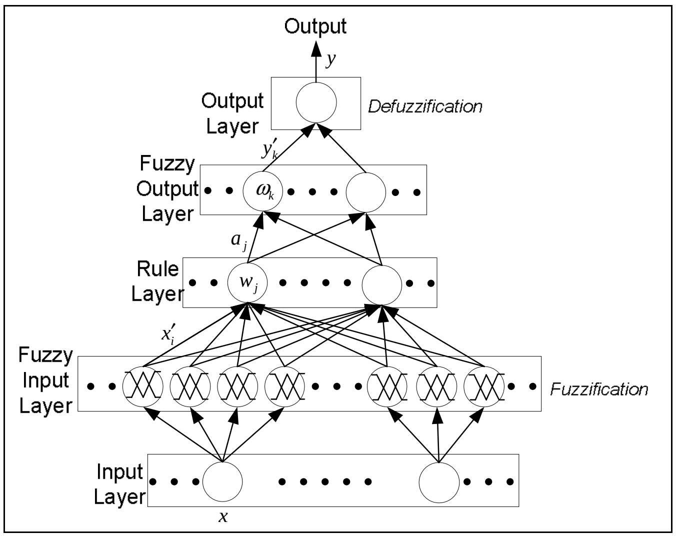

Figure 1

The individual fuzzy output node's activation is based on

$$y'_k = f_2\left(A^T \omega_k\right) \tag{3}$$

where $k \in \{1 \ldots K\}$. The function f_2 can take the form of a linear function such as $f_2(u) = u$. For every training record, the input x is fuzzified into X'_{train} form and its corresponding output y is fuzzified into Y'_{train} form as illustrated in Figure 2 below.

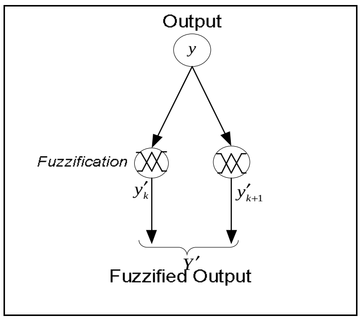

Figure 2

For a better understanding of the EFUNN algorithm, special attention needs to be paid to the ω_k weight vector whose dimension equals J number of rule nodes. This implies that the ω_k vector has a distinct weighting value $\omega_k(j)$ that associates with the j^{th} rule node in the lower rule layer.

5.2 EFUNN's Original Rule-Learning Mechanism

When a new training record (x, y) is available, it is fuzzified first into the (X'_{train}, Y'_{train}) fuzzy form before being introduced to the EFUNN. This is then passed to the EFUNN's weight-updating procedure which is summarised into the following:

Step 1: Determine if the training record has already been learnt by the network. This is achieved by first determining if any of its rules nodes satisfy two distinct criteria:

Criterion I

The first criterion requires the rule node's internal w_j and the fuzzified input X'_{train} to achieve a strong enough activation based on Equation 1 exceeding an adaptive sensitivity threshold S_j (refer to Equation 7 below).

Criterion II

The second criterion requires the corresponding difference ξ between the network's resultant Y' versus the training record's desired output Y'_{train} to be below a fixed predefined error threshold. The difference is expressed as:

$$\xi = \left\| Y' - Y'_{train} \right\| \tag{4}$$

Note that criteria I and II therefore ensure the training pair of fuzzy input-output vectors exemplified in (X'_{train}, Y'_{train}) to be allocated to the j^{th} rule node if X'_{train} and Y'_{train} both fall into the j^{th} rule node's input and output receptive fields (hyper-spheres) respectively.

Step 2: Assuming that only one rule node is found to satisfy the two criteria, then the rule node's internal w_j and all the $\omega_k(j)$ weight values whose k ranges over $[1..K]$ and j pointing to the specific rule node will be updated based on the following where l_j is the learning rate associated with the j^{th} rule node.

$$w_j^{(t+1)} = w_j^{(t)} + l_j\left(w_j^{(t)} - X'\right) \tag{5}$$

$$\omega_k(j)^{(t+1)} = \omega_k(j)^{(t)} + l_j\left(y'_k - Y'_{train}(k)\right)a_j^{(t)}, k \in \{1..K\} \tag{6}$$

$$S_j^{(t+1)} = S_j^{(t)} - D\left(w_j^{(t+1)}, w_j^{(t)}\right) \tag{7}$$

If none of the existing rule nodes satisfies the two criteria above, the rule-learning mechanism will proceed to learn the new training record by memorizing it, which is achieved based on the following steps:

$$w_{J+1} = X'_{train} \tag{8}$$

where $J+1$ refers to the insertion of a new rule node into the existing rule layer.

$$\omega_k(J+1) = Y'_{train}(k) \tag{9}$$

where k ranges over $[1..K]$ and every ω_k weight vector embedded in the k^{th} fuzzy output node has its dimension expanded by one to its new $J+1$ dimension to accommodate the corresponding k^{th} value in the Y'_{train} vector.

5.3 Issues with EFUNN's Original Rule-Learning Mechanism

The issues underlying the rule learning mechanism have to do with its handling of training records with outlying inputs and training records with contradictory class outputs. These issues are elaborated in the following subsections.

5.3.1 Training Records with Outlying Inputs

It is possible that during the training stage, the rule learning mechanism may come across training records whose input values are superficially near to a rule node's w_j weight center but in reality, these training records' inputs may in fact be outliers that are not statistically close to the inputs of the past training records that were associated to this particular rule node's w_j weight center during the earlier stage.

According to Equations 1, 2 and 5, any training record whose input vector appears in the vicinity of a rule node's w_j weight center can assert an influence over it and this may be an undesirable outcome of the rule learning mechanism in that the particular rule node's w_j weight center may undergo unnecessary adjustments owing to Equation 5.

5.3.2 Training Records with Contradictory Class Outputs

Based on the weight updating steps mentioned in section 5.2, the EFUNN tends to end up memorizing multiple rules whose antecedents are near identical but whose consequents are contradictory. This results in the EFUNN generating rules with contradictory class outputs during the rules-extraction stage. This can be illustrated by way of a simple example.

Assuming that two training records with identical antecedents but contradictory consequents are fed into a fresh untrained binary-classed EFUNN classifier with no prior rule node. The two training records are represented as:

Training record 1:	($X'_{train(0)}, Y'_{train(0)}$)
Training record 1's input:	$X'_{train(0)} = [1 \quad 0]$
Training record 1's output:	$Y'_{train(0)} = [1 \quad 0]$
Training record 2:	($X'_{train(1)}, Y'_{train(1)}$)
Training record 2's input:	$X'_{train(1)} = X'_{train(0)}$
Training record 2's output:	$Y'_{train(1)} = [0 \quad 1]$

The first training record will result in the EFUNN memorizing the first training record. The second training record will result in the EFUNN also memorizing it because the two criteria highlighted in the weight-updating procedure are not satisfied. The final EFUNN ends up with two rule nodes whose associated values are $w_1 = [1 \quad 0]$, $\omega_1 = [1 \quad 0]$, $w_2 = [1 \quad 0]$ and $\omega_2 = [0 \quad 1]$ respectively.

So when the network is consulted with an input $X'_{query} = [1 \quad 0]$, the resultant activation vector A becomes $[1 \quad 1]$ because both w_1 and w_2 have their weight centers or exemplars identical to X'_{query}. And because ω_1 and ω_2 already memorized two direct opposite class outputs based on Equation 9, the activation at the fuzzy output layer Y' then becomes $[1 \quad 1]$ which makes it difficult for a clear-cut decision to be made during the defuzzification stage. Such is the undesirable outcome of the rule learning mechanism that learns rules with near identical antecedents but with contradictory class outputs.

5.4 Modifications to EFUNN to Address Issues

In order for the EFUNN to better handle training records with outlying inputs and training records with contradictory class outputs, two modifications are introduced to address these issues and are elaborated in the next two sub-sections.

5.4.1 Modification to address Outlying Inputs

Instead of allowing training records with outlying inputs to assert undesirable influence on a rule node's w_j weight center, the original Equations 1, 2 and 5 need to be replaced with alternative substitutes.

The original rule learning mechanism of the EFUNN adopts the local normalised fuzzy distance measure in Equation 2, which is essentially a form of Euclidean distance. It is inherently not very resilient against training records whose input attributes may be correlated and may have different variances such as track's numerical speed versus track's ordinal/qualitative identification status. A more suitable distance measure will be the statistical Mahalanobis distance (MD) measure that takes into account the possible correlations among the attributes with their different variances.

The MD measure is expressed as $\left(x-\bar{x}\right)^t \Phi^{-1}\left(x-\bar{x}\right)$ where Φ is the covariance matrix, which is a square symmetric matrix whose individual element x_{ij} is the covariance between two attributes expressed as $\sum_{i=1}^{n}\frac{\left(x_i-\overline{x_i}\right)\left(x_j-\overline{x_j}\right)}{n-1}$ where n is the number of accumulated training records that are associated to the rule node's weight center.

Equation 5 does not fully take into account the past training records that have been associated to it during the training process. The way the w_j weight center embedded in the j^{th} rule node is being updated is based heavily on the difference between w_j and the X'_{train} fuzzified input of the training record. Therefore, it is susceptible to outliers since practically any fuzzified input accompanying a new training record can assert a direct influence on the w_j weight center as long as the activation output a_j is sufficiently strong. This issue is further exacerbated by Equation 7 because the sensitivity threshold decreases monotonically as training progresses and it becomes gradually easier for outliers in the vicinity to assert undesirable influence on the w_j weight center.

To augment the EFUNN with greater resilience against outliers, the Fuzzy c-means clustering (FCM) technique is proposed and its use with the EFUNN is explained in the next subsection.

5.4.1.1 Fuzzy c-means Clustering

Before explaining FCM, it is important to note that the learning process occurring in the fuzzy input layer and among the rule nodes' weight centers w_j in the original EFUNN is essentially achieved by unsupervised clustering that holds some resemblance to Self-Organizing Maps [10]. But as explained in section 5.3, the original learning method adopted by the EFUNN does not fully distinguish between statistically significant training data and outliers.

The new method however uses the unsupervised clustering method called Fuzzy c-means clustering (FCM) which is based on the concept of fuzzy c-partition [11] but with a minor alteration. It replaces the fuzzy input layer and the rule layer, including all the accompanying weight centers w_j. In their place, each rule node's new center is realized with a cluster-center c_j whose dimension equals the dimension of the input x. Each c_j in turn is dependent on its $[0..1]$-valued cluster-membership function $u_j(i)$ where i here refers to the i^{th} training record that has been associated to the cluster as opposed to *all the training records* from the training data pool based on the original definition of FCM. This distinction arises because in the original FCM, the number of clusters is pre-fixed and remains static throughout the entire clustering process and all the training data has to be available before the clustering commences. This is different from the unsupervised clustering portion of the EFUNN algorithm where the rule layer is flexible. A decision criterion is therefore needed to decide when to grow and when to reuse an existing rule node.

The c_j center is computed based on:

$$c_j^{(t+1)} = \frac{\sum_i^N u_j\left(x(i)\right)^m x(i)}{\sum_i^N u_j\left(x(i)\right)^m} \quad (10)$$

where N refers to the number of training records that are previously associated to that particular cluster and i ranges over $[1..N]$.

Because it is important that given a single training record, all the respective cluster-membership degrees with respect to it need to sum to 1 in order to be consistent with the principle behind fuzzy logic, the following equation is used to formulate the cluster-membership function:

$$u_j(x) = \frac{1}{\sum_{l=1}^{J}\left(\frac{D\left(c_j^{(t)},x\right)}{D\left(c_l^{(t)},x\right)}\right)^{\frac{2}{m-1}}} \quad (11)$$

where D is based on the MD measure.

5.4.1.2 Use of FCM for Unsupervised Clustering in EFUNN

Replacing Equations 1, 2 and 5 of the EFUNN, the FCM technique is applied according to the following procedure:

Step 1: During the initialisation stage, existing Equations 8 and 9 are reused to incorporate an initial rule node into the rule layer. In the newly formed rule node, $u_j(i)$ is set to an arbitrary value close to 1.0. This signifies that given an initial sample point, the newly created cluster has its center placed near it.
Step 2: For the subsequent training record, the cluster-membership degree function in Equation 11 is used to compute the cluster membership degree value. If the cluster membership degree $u_j(x)$ exceeds a pre-fixed user-defined sensitivity parameter S'_j, the new training record will be associated to the cluster. Every time a new training record is associated to the same cluster, the cluster center c_j based on Equation 10 needs to be re-computed. If the membership degree is below the S'_j sensitivity parameter, repeat step 1 to insert a new rule node.

In the original unsupervised clustering method of the EFUNN, the dynamic sensitivity parameter in Equation 7 is decreased every time a new training record is associated to a rule node's w_j weight center. With such a monotonically decreasing sensitivity parameter, it becomes progressively easier for outliers to assert undesirable influence on the w_j center.

In the new method, a rule node's center c_j is based on past input sample points appropriately weighted with their corresponding cluster membership degrees. In addition, these cluster membership degrees are computed based on Equation 11, resulting in the rule node's new weight center to become more resilient against possible outliers.

While protecting the existing rule nodes' centers from training records with outlying inputs, it is still possible for a new rule node to be inserted into the rule layer to accommodate an outlier in Step 2 whereby none of the rule nodes' centers are statistically near it. However, this does not impose a significant problem because if it is indeed an outlier case, the simple lack of training records being associated to it will cause the rule node to age out and become pruned off after a prolonged time period via EFUNN's inherent pruning feature.

5.4.2 Modification to address Contradictory Class Output

The modification involves a simplified method that first checks whether the new training record's input X'_{train} is strongly similar to an existing rule node's weight center w_j and if so, perform the weight updating step in Equation 6 regardless of the size of ξ in Equation 4. In other words, the new method essentially focuses on updating the existing weights $\omega_k(j)$ in the fuzzy output layer to adapt to the contradictory class output instead of directly inserting a new rule node whose ω_k values are set directly just to memorize the contradictory class output whenever the error difference ξ exceeds the predefined fixed error threshold.

6 EFUNN for Knowledge Management

In order for the adaptive knowledge base to assist the human operator in deciding whether to exercise the respective observation tasks, the individual knowledge base needs to be equipped with multiple rule-sets whereby each rule-set governs the decision making behind each respective task. A rule-set therefore comprises multiple rules that need to be consistent with one another.

A single EFUNN can be mapped to one rule-set and the figure below illustrates the internal structure of an EFUNN capturing the rule-set that governs whether to exercise the Track-Update Task in the form of a binary classifier. Example rules are:

Track-Update Rule-Set's Rule 1:

If
the track's quality is DEGRADING,
the track's encroachment of critical assets is INFRINGING,
the track's identification is UNKNOWN
Then
Perform Track-Update task.

Track-Update Rule-Set's Rule 2:

If
the track's quality is NORMAL,
the track's encroachment of critical assets is NEGLIGIBLE,
the track's identification is FRIENDLY
Then
Refrain Track-Update task.

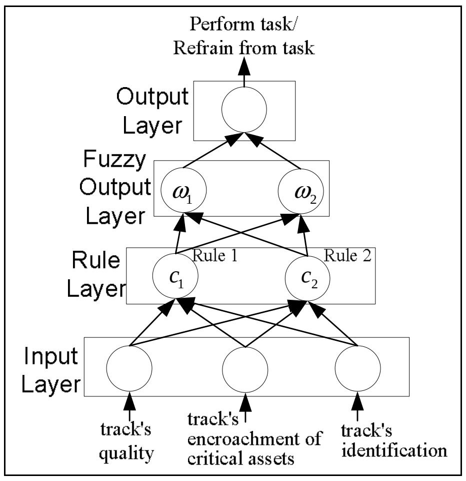

Figure 3

As for other observation tasks such as Target-Search and IFF, two separate EFUNNs are needed to represent the two rule-sets respectively. The multiple rule-sets or EFUNNs therefore make up one adaptive knowledge base.

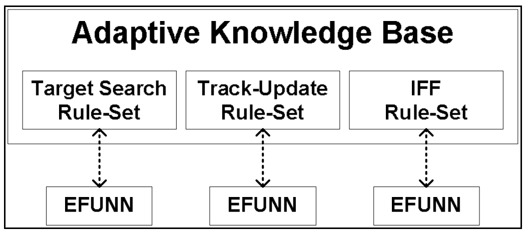

Figure 4

To enable the EFUNNs to start capturing the decision logic exhibited by the human operators, training records need to be made available first. Whenever the human operator interacts with a test scenario and reacts to changes in the dynamic situational picture, there should be system provisions to capture the situational picture inputs and the corresponding human operator's responses as outputs. Reusing the above example based on the Track-Update task, a sample training record's inputs can be *track quality, track's encroachment of critical assets* and *track identification*. The corresponding output can be *perform task* or *refrain from task.*

Each EFUNN is therefore responsible for capturing the decision logic for one observation task. A single knowledge base therefore will have multiple rule-sets with one-to-one correspondence to an equal number of EFUNNs. These EFUNNs observe in the background the different behavioural patterns exhibited by the human operator's interaction with a C2 system. As human operators interact with more test scenarios, the EFUNNs become better attuned and eventually able to replicate some of the decision-making logic of the human operators. Because the human operators can also view the rule-sets that are learnt by the EFUNNs, the operators can check whether the learnt rules are consistent with actual operational requirements.

When a less experienced operator undergoes experiential training using the same set of test scenarios, the operator can choose to be assisted by a knowledge base that is previously trained by more seasoned human operators. During the running of the test scenario, the EFUNNs governing the rule-sets that make up the adaptive knowledge base will be queried periodically to determine which rule-sets are to be triggered. When a rule-set is triggered with the outcome indicating *perform task*, a non-obstructing visual display recommending the performance of the observation task can be shown to the human operator to prompt for further action.

To further promote the sharing of operators' expertise, each personnel can contribute to a single knowledge base. A knowledge base owned by a particular individual can be further refined (or trained) using training records contributed by other individuals through EFUNN's rule-learning feature augmented with the FCM technique. By leveraging on the existing EFUNN's rule extraction and aggregation features, it is possible to perform viewing and merging functions on the different knowledge bases.

6.1 Refine (Training) function

By collecting the training records explained in section 6, these records can be used to train the human operator's knowledge base. Each operator can own a set of training records that log down the responses asserted by the operator during the operator's interaction with a test scenario. The operator can then train one's own knowledge base with the logged training records so that the experiences can possibly be captured and translated into knowledge rules by the EFUNNs.

A human operator's experience can vary greatly from others. An operator upon seeing one's own set of rules would like to learn from other operators in terms of how they handle the various critical situations in test scenarios. Because the other operators' training records can be captured and stored offline, it is possible for the first operator to take the second operator's training records and use them to train one's own knowledge base.

After the operator has trained up one's own knowledge base, the operator will want to inspect the captured knowledge in human comprehensible rules. The viewing function of the EFUNN is explained in the next section.

6.2 View function

The EFUNN comes readily with a rule extraction feature that allows its learnt rules to be examined. Examples of learnt rules come in the form of Rule 1 and Rule 2 in section 6.

With the option to view the rules learnt by the EFUNNs, the human operators can check the correctness of the rules and upon coming across interesting ones contributed by others, the human operator can attempt to merge one's own knowledge base with knowledge bases from others. The rule-aggregation feature is an inherent part of the EFUNN but with the introduction of the FCM technique, the original rule-aggregation needs to undergo certain modification which is explained in the next section.

6.3 Merge function

The merge function is facilitated by EFUNN's rule aggregation feature to merge or combine two dissimilar knowledge bases by allowing rule-sets from one knowledge base to merge with those found in the second knowledge base in strict accordance to their task-types. An EFUNN can only merge with another EFUNN that represents the same rule-set type because it is not possible to combine rule nodes whose antecedents or consequents differ in the attributes' makeup.

Due to the adoption of the FCM technique, the merging of the rule nodes requires modification. The original method used in merging the weight centers is based on the average of the rule nodes' centers individually weighted by the number of training records associated to the cluster and is expressed as:

$$w_{aggregated} = \frac{1}{\sum_{j} num_j} \sum_j w_j num_j \tag{12}$$

where j ranges over those rule nodes whose weight centers are considered near to each other based on the original normalized fuzzy distance measure.

With the introduction of the FCM technique, the following procedures are adopted instead:

Step 1: Go through all the rule nodes' centers to first partition them into groups with high similarities. Two centers c_j and c_l can be compared by feeding the latter into Equation 11 to see the score achieved by $u_j(c_l)$. The two centers are considered similar and deemed as belonging to the same partition if the score exceeds a pre-fixed user-defined parameter S'_j.

Step 2: Assuming c_j and c_l belong to the same partition, retrieve every training record x from cluster center c_l and attempt to transfer and associate it to cluster center c_j. If the $u_j(x)$ score exceeds the S'_j parameter, apply Equation 6 to update the $\omega_k(j)$ weights in case the class output learnt by cluster center c_j is different from the one learnt by cluster center c_l. Equation 10 is then used to update c_j since the training record from c_l has just been associated to the cluster center c_j. Those training records whose $u_j(x)$ scores did not exceed the S'_j parameter will be left intact within the c_l cluster. After all the training records from c_l have been considered and if there are still remaining training records within c_l, apply Equation 10 to update the c_l cluster center.

Step 3: If all the training records previously associated to cluster center c_l are completely assimilated by the cluster center c_j, c_l and as well as all the $\omega_k(l)$ elements among the K fuzzy output nodes can be removed. Repeat step 2 until all the cluster centers in the current partition have been considered for merging. Repeat steps 2 and 3 for the remaining partitions.

7 Conclusion

This paper spells out the advantages of employing adaptive knowledge bases to support the decision-making processes of the human operators when dealing with dynamic situation/sensor picture in the context of sensor management. The EFUNN learning algorithm has been identified to be suitable for this purpose and this paper has suggested the use of the FCM technique to enhance the EFUNN's clustering ability coupled with modifications to the weight-updating procedures to better handle training records with outlying inputs and training records with contradictory class outputs.

This paper also suggests an innovative use of the EFUNN in representing a rule-set, which is an important component in the make-up of a knowledge base. Since each EFUNN comes readily with features namely training, rule-extraction, rule-aggregation and rule-pruning, these functions fulfil the goals of adaptive knowledge bases whereby individual rule-set (embodied in an EFUNN) can be refined, viewed and merged with other rule-sets.

References

[1] "Data Fusion Lexicon", U.S. Department of Defense, Data Fusion Subpanel of the Joint Directors of Laboratories, Technical Panel for C3, 1991

[2] James Linas, Christopher Bowman, Galina Rogova, Alan Steinberg, Ed Waltz, Frank White, "Revisiting the JDL Data Fusion Model II", Proceedings of the 7th International Conference on Information Fusion, 2004

[3] Fok Bolderheij, Piet Van Genderen, "Mission Driven Sensor Management", Proceedings of the 7th International Conference on Information Fusion, 2004

[4] J. R. Quinlan, "Induction on decision trees", Machine Learning, Vol 1, pp. 81-106, 1986

[5] J. R. Quinlan, "C4.5: Programs for Machine Learning", San Mateo, CA: Morgan Kaufmann, 1993

[6] Geoffrey G. Towell, Jude W. Shavlik, "Knowledge-Based Artificial Neural Networks", Artificial Intelligence, Volume 69/70, 1994

[7] Zhihua Zhou, Shifu Chen, Zhaoqian Chen, "Mining Typhoon Knowledge with Neural Networks", Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, Evaston, IL, 1999, pp. 325-326

[8] Nikola K. Kasabov, "Evolving connectionist and fuzzy connectionist systems - theory and applications for adaptive, on-line intelligent systems", Neuro-Fuzzy Techniques for Intelligent Information Processing, Heidelberg, Physica Verlag, 1999, pp. 111-114

[9] Nikola K. Kasabov, "On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks", Neurocomputing, Elsevier, 2000

[10] Kohonen, "The Self-Organizing Map", Proceedings of the IEEE, v.78(9), 1990, pp. 1464-1480

[11] Ruspini, E. H. "Numerical methods for fuzzy clustering", Information Sciences, 2(3), pp. 319-350, 1970