OPTIMIZED FUZZY MIN-MAX NEURAL NETWORK: AN EFFICIENT APPROACH FOR SUPERVISED OUTLIER DETECTION

N. Upasani, H. Om**

Abstract: Fuzzy min-max neural network (FMN), proposed by Simpson is a well-known supervised neuro-fuzzy classifier that has been successfully used by many researchers for pattern recognition. However, the FMN represents the learned knowledge with exhaustive details in a 'fine-grained' manner that reduces its performance for pattern recognition in terms of the recall time per pattern. In this paper, we adapt the basic architecture of the FMN to represent the learned knowledge in a compact way that is in a 'coarse-grained' manner, which is closed to human thinking. The working of the proposed method that is fuzzy min-max neural network with knowledge compaction (FMN-KC) is illustrated using the Fisher Iris dataset. The potential of using the FMN-KC for supervised outlier detection is demonstrated using a time-series disk defect dataset published by NASA and KDD cup 99 dataset available in UCI repository. The proposed method achieves around 50 % gain in the recall time as compared to the original FMN and the recognition rate is also comparable. We strongly recommend using the proposed architecture FMN-KC for supervised outlier detection in the real time applications, where recall time per pattern is one of the key parameters.

Key words: *coarse-grained knowledge, fault detection, Fuzzy Min-Max Neural Network (FMN), intrusion detection, neuro-fuzzy systems, supervised outlier detection*

Received: June 3, 2016
Revised and accepted: August 10, 2018

DOI: 10.14311/NNW.2018.28.017

1. Introduction

An outlier is an observation that deviates from other observations in such a way that it creates a suspicion about the original method that was used to generate it [1]. The outliers may obstruct the process of data mining due to the incorrect knowledge discovery [2]. Outlier detection is an essential pre-processing task in diverse domains including automation and control in industry applications [2], computer network intrusion detection [3], fraud detection [4], suspicious or criminal activities detection in e-commerce [5], image processing [6], remote sensing [7], etc. Various

*Nilam Upasani – Corresponding author; Hari Om; Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad 826004, Jharkhand, India, E-mail: nilamupasani@gmail.com, hariom4india@gmail.com

methods have been discussed to catch the outliers and a detailed study can be found in [8–10]. There are three basic approaches for outlier detection that include unsupervised clustering, supervised classification, and semi-supervised recognition [10]. A general approach for detecting the outliers is to train the model for normality and then calculate the deviation of each input instance with respect to the trained normal data. The methods based on the supervised classification approach have used different methods to calculate the distance between an input pattern and the trained normal data. If the deviation of some input exceeds the pre-specified threshold, then that input is considered as an outlier. The threshold is initially set based on the application requirements [10]. The unsupervised clustering based methods do not have any prior knowledge about the data to determine the outliers and hence many times they are used for noise removal [11]. The semi-supervised based methods require either normal or abnormal patterns for training. The supervised methods use the characteristics of the known examples including the normal and abnormal patterns to sharpen the learning process to find more relevant outliers. The outlier detection systems have not been well-studied, explored, or applied as classification systems [12]. The supervised methods have been found more effective to get the meaningful outliers; however, these methods have not been explored much in this area. Gomez and Dasgupta have applied the supervised fuzzy classifiers for intrusion detection and they reported good detection rate [13]. The fuzzy systems however do not have the capability to construct the models solely based on the sample data for the target system. Zhang et.al. have employed a perceptron back propagation hybrid (PBH) neural network classifier for detecting the network–based attacks [14]. Hawkins et. al. have successfully employed the replicator neural networks for anomaly detection [15]. The works [14,15] have employed neural network approach for outlier detection that has advantages such as adaptive capability of learning new patterns, refining the existing ones, managing complex nonlinear decision boundaries, and faster recall. They however are not effective to handle the uncertainties involved in the outlier detection. Toosi, et.al. have used the adaptive neuro-fuzzy inference system (ANFIS) classifier to detect intrusions in computer networks [16].This integration of fuzzy and neural soft computing system for outlier detection is very promising as they exactly address the challenges associated with the outliers. The fuzzy logic handles the uncertainty involved in outlier detection and the parallel nature of neural networks achieves faster recall. A detailed survey on outlier detection techniques involving fuzzy and/or neural networks methods has been discussed in [17].

Fuzzy min-max neural network (FMN) is a well-known hybrid neuro-fuzzy classifier proposed by Simpson [18] that has been successfully used by many researchers for pattern recognition. It has many advantages such as on-line learning ability, learning with a single pass through data, learning any realistic and multidimensional data that have nonlinear decision boundaries. The parallel hardware can be used to accelerate the execution as implementation can be done using single precision arithmetic operations. Researchers have used this classifier for several applications such as medical diagnosis [19], fault diagnosis [20], detection of heart diseases [21], face recognition [22], speaker identification [23], Iris recognition [24], business intelligence [25], outlier detection [26–28], etc. The FMN is a three layer feed-forward neural network model that grows adaptively to meet the demands of

a given problem. It utilizes the fuzzy sets as pattern classes and each fuzzy set is a union of hyperboxes. A hyperbox(HB) is defined by min-max points that are updated and fine-tuned in the learning or training phase. The training algorithm determines the min-max points in a single pass through the data, which leads to creation of new HBs belonging to different classes and refining the existing ones (without retraining). The FMN allows overlap between the HBs belonging to the same class; however, overlap between the HBs belonging to different classes is not accepted. Each HB has an associated membership function, which states how much a particular pattern belongs to that HB. Some researchers have modified the FMN to make it more effective for specific type of problems. A fusion of fuzzy min-max classification and clustering algorithms, called general FMN, is discussed in [29], which can be used in hybrid clustering classification problems. Nandedkar and Biswas have updated the FMN by including the compensatory neurons to handle the overlapping and containment situations in FMN [30]. They have used this modified algorithm for outlier detection and obtained better performance than the existing well-known methods. Anas and Lim have modified the FMN to reduce the size of the rule set and used it for fault detection and classification [27]. They have reduced the number of HBs using a pruning strategy. The pruning method is based on a confidence factor which is calculated using the method discussed in [31]. The confidence factor recognizes the HBs that are frequently used and generally give high classification accuracy as well as the HBs that are highly accurate but rarely used. After training phase, the HBs having confidence factor less than the user defined threshold are pruned. Liu et al. have used the FMN with cancroids rather than the min-max points to improve the computation time and to obtain faster recall [32].

In this paper, we modify the basic architecture of FMN, calling it as fuzzy min-max neural network with knowledge compaction (FMN-KC), to organize the learned knowledge in a more compact and systematic manner. The aim is to provide faster recall and make it more effective for real time applications. It has been observed that the FMN creates more number of HBs even away from the decision boundary that is it represents the learned knowledge with exhaustive details in 'fine-grained' manner. Though the FMN is a well-known pattern recognizer, yet the 'fine-grained' representation of the learned knowledge may reduce its performance for pattern recognition in terms of the recall time per pattern. Here, we add a new phase, called knowledge compaction, after the regular training phase. The training phase creates the HBs and the knowledge compaction phase combines the HBs created away from the decision boundaries that belong to the same class, without creating any overlap with the HBs of other classes. This 'coarse-grained' representation of the knowledge achieves considerable improvement in the recall time than the regular FMN without reducing its recognition rate. Our modified architecture improves the recall time for outlier detection task. We use the Fisher Iris dataset [33] to explain the working of our proposed FMN-KC. The outlier detection performance of the FMN-KC is assessed on a 2-dimentional synthetic dataset, a time-series disk defect dataset published by NASA and KDD cup 99 dataset available in the UCI repository [34]. Our proposed method gives better performance than the original FMN in terms of the recall time per pattern and its recognition rate is also comparable.

The rest of the paper is organized as follows. Section 2 reviews the FMN. Section 3 introduces our proposed modified FMN-KC and Section 4 introduces it in an algorithmic form. Section 5 illustrates the proposed method with an example. Section 6 discusses the simulation results for outlier detection and finally Section 7 concludes the paper.

2. Review of Fuzzy Min-Max Neural Network (FMN)

The FMN is a three layer feed-forward neural network model that grows adaptively to meet the demand of a problem. It may be considered as an example of the hyperspherical attractor neural network [35]. It is a supervised learning algorithm proposed by Simpson [18] that creates the decision boundaries by creating subsets of a pattern space. The architecture of the FMN is given in Fig. 1.

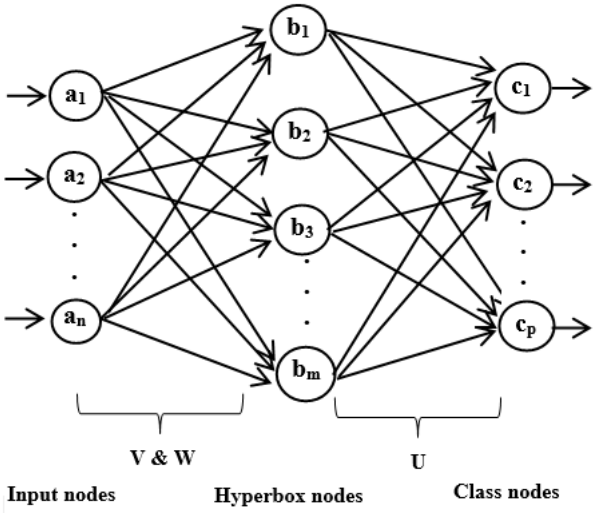

Fig. 1 *Architecture of FMN.*

An input pattern is defined as $(A_h\ , C_h)$, where $A_h = \{a_{h1}, a_{h1}, \cdots, a_{hn}\}$ is an input pattern in n-dimensional unit hypercube *(I-n)* and C_h, $h\epsilon\{1, 2, \cdots, p\}$, is a class-index that h-th pattern belongs to. The input patterns are represented as A or X in this paper. The FMN creates fuzzy sets, called HBs, and each pattern class is a fuzzy union of HBs. The HBs representing the hidden nodes in the three layer FMN model define a region of the n-dimensional pattern space that has patterns with full membership. A HB is defined by a fuzzy membership function and min-max points. The max and min refer to the farthest and closest points from the origin in a hyperspace, respectively. The j-th HB, denoted as B_j, is defined as follows.

$$B_j = \{X, V_j, W_j, b_j\,(X, V_j, W_j)\} \quad \forall X \epsilon I^n, \tag{1}$$

where matrices V_j and W_j store the min and max points, respectively, of B_j. The matrices V and W represent the connections between the input nodes and HB nodes, as shown in Fig. 1. The fuzzy membership function associated with a HB tells the degree to which an input pattern belongs to that HB. A fuzzy membership function b_j for j-th HB is given by

$$b_j(X_h) = \frac{1}{2n}\sum_{i=1}^{n} \max(0, 1 - \max(0, \gamma \min(1, x_{hi} - w_{ji}))) + \max(0, 1 - \max(0, \gamma \min(1, v_{ji} - x_{hi}))), \tag{2}$$

where γ is sensitivity parameter that regulates how fast the membership value changes. The fuzzy membership values 0 and 1 indicate no and full membership, respectively; and the values in between tell partial membership. The connections between the HBs and the output nodes that is the class nodes are stored in connection matrix U as binary values, which are given by

$$u_{jk} = \begin{cases} 1 & \text{if } b_j \text{ is a HB for class } c_k \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

We now briefly present the FMN training algorithm.

FMN training: The training process constructs HBs and learns the non-linear decision boundaries between different classes. New HBs are created and the existing are refined in a single pass through the data. The number of HBs created depends on a user defined value, called expansion coefficient that is the HB-size (θ), $0 \leq \theta \leq 1$. The less value of θ creates more number of HBs and gives better classification accuracy; it however badly affects the recall time. So, the value of θ must be selected such that it gives better classification accuracy with minimum possible number of HBs. There are four steps of training/learning algorithm, as discussed below.

1. Initialization: In this step, it creates the first HB using the first training pattern and its min-max point is identical to that pattern.

2. HB expansion/creation: A HB is expanded to accommodate an input pattern in the training set that belongs to the same class. The expansion criterion, given in (4), should be satisfied to include a pattern x_h in the HB B_j

$$n\theta \geq \sum_{i=1}^{n} (\max(w_{ji}, x_{hi}) - \min(v_{ji}, x_{hi})). \tag{4}$$

New min and max points for the expanded HB are calculated using (5) and (6), respectively, as given below.

$$v_{ji}^{\text{new}} = \min\left(v_{ji}^{\text{old}}, x_{hi}\right) \quad \forall\, i = 1, 2, \cdots, n, \tag{5}$$

$$w_{ji}^{\text{new}} = \max\left(w_{ji}^{\text{old}}, x_{hi}\right) \quad \forall\, i = 1, 2, \cdots, n. \tag{6}$$

If the expansion criterion in (4) is not fulfilled, then a new HB is created.

3. Overlap test: A pattern classifier must be proficient enough to form the decision boundaries that can avoid overlapping of the classes to minimize the misclassification. The overlaps of the HBs that belong to the same class do not create any class formation problem and hence allowed. The overlap of the HBs that belong to different classes must be eliminated to avoid misclassification. The overlap of an expanded or a newly created HB is checked using the overlap tests as given in [18]. If any overlap is detected, then a contraction process is used to eliminate the overlap.

4. HB contraction: The overlap is removed by minimally adjusting one of the dimensions of both the HBs. The minimal adjustment is done to assure a very negligible change in the shape of the HBs, using the rules defined in [18].

After training, the learned knowledge gets stored in the form of HBs.

Test Phase: Here, the fuzzy membership value of a test pattern is calculated in each HB, using (2). A transfer function is implemented at the output node to calculate the degree to which a test pattern belongs to each class. The transfer function for k-th class is a fuzzy union of the corresponding HB fuzzy membership values that is given by

$$c_k = \max_{i=1}^{m} (b_i u_{ik}) , \tag{7}$$

where m is number of HBs that are connected to class c_k using matrix U.

3. Proposed architecture

The classification accuracy of the FMN is mainly affected by the HB-size (θ), which is a user defined parameter. If the value of θ is less, more number of small-sized HBs is created that increases the classification accuracy. However, these small-sized HBs are created even away from the decision boundary that need not be stored separately as this fine-grained representation of the learned knowledge with exhaustive details reduces its performance for pattern recognition in terms of the recall time per pattern. These HBs cannot be completely removed as it will reduce the recognition accuracy; however, big-sized HBs can be formed by combining the HBs away from the decision boundary. Here, the basic architecture of the FMN is modified by adding a new phase, called knowledge compaction phase, which organizes the learned knowledge in a systematic and compact manner. The main aim of adding this phase is to combine the HBs away from the decision boundary rather than removing them. Our proposed modified architecture FMN-KC is given in Fig. 2.

The FMN is first trained according to the FMN training algorithm as described in Section 2. The FMN creates K number of HBs with the corresponding min-max points (V, W) and a connection matrix U. The matrices V, W and U are given as input to the KC-phase. This step reduces the count of the HBs to fit the knowledge in a smaller system. New sets of HBs are represented with the revised min-max points G, H and the updated connection matrix as F. The KC-phase inherits the knowledge learned in the training phase and shows a comparable level of detection rate with faster recall after the compaction.

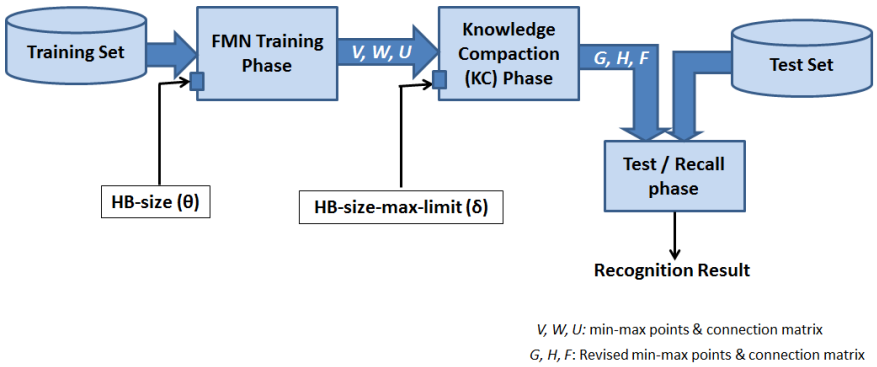

Fig. 2 *Modified Architecture: Fuzzy Min-Max Neural Network with Knowledge Compaction (FMN-KC).*

In order to avoid misclassification, an upper bound is set on the maximum size of the HB while combining them, called HB-size-max-limit (δ). It is a user-defined parameter that should be greater than the value of θ, i.e. ($\delta > \theta$) to create big-sized HBs and reduce the count of HBs to achieve better compaction. If ($\delta < \theta$), then it would not reduce the count as the HBs are not combined together. Same is the case for equal values. The compaction process is described below.

The KC-phase tries to combine the K number of HBs created by FMN, considering δ as the maximum limit on the size of HB. For each HB, p in K, find the number of HBs belonging to the same class as that of the p-th HB, denoted as (t). The KC-phase tries to merge p-th HB with each of the HBs in t. After combining p-th HB with q-th HB in t, the min-max points (g, h) of newly created r-th HB are calculated using (8) and (9), respectively.

$$g_{ri} = \min\left(v_{pi}, v_{qi}\right) \quad \forall i = 1, 2, \cdots, n, \tag{8}$$

$$h_{ri} = \max\left(w_{pi}, w_{qi}\right) \quad \forall i = 1, 2, \cdots, n. \tag{9}$$

The overlap of a newly created r-th HB with the existing set of HBs belonging to other classes is checked using the overlap test given in [18]. If no overlap is detected, then delete p-th and q-th HBs, that is, the min-max points of the corresponding HBs from the existing set (V, W). Then add the newly created r-th HB, that is, the new min-max points calculated using (8) and (9) to the existing set of HBs. An overlap of r-th HB with the HBs belonging to the same class, that is, the HBs in t is allowed. However, if it creates an overlap with the HBs belonging to different classes, then we do not add it to the system and retain the existing set of HBs.

The compaction process is executed for all the K number of HBs created by FMN. The KC-phase creates L number of HBs with the revised min-max points (G, H), where the value of L is substantially less than K. Finally, the connections between the revised set of HBs and class nodes are calculated using (3) and stored in matrix F. The KC-phase changes the representation of internal knowledge before test phase to get faster recall than the original FMN. We now present the algorithm to implement our proposed method.

4. Our proposed method in algorithmic form

It has three phases that include training phase, knowledge compaction phase, and test phase, as discussed below.

4.1 Phase 1: Training phase

Denote,
P: Number of patterns in the training dataset, N: Dimension of pattern vector, θ: HB-size
Use training dataset to train the neural network using the learning algorithm [18] that is described in Section 2. This phase provides the learned knowledge in terms of HBs. Let K be the number of HBs.

4.2 Phase 2: Knowledge compaction phase

In this phase, we combine the HBs belonging to the same class, which are away from the decision boundary, without creating any overlap with the HBs of other classes. The algorithm for knowledge compaction is given in Algorithm 1.

We keep increasing the value of δ and execute the KC-algorithm (Algorithm 1) to combine the HBs. Increase the value of δ till it does not drop the classification accuracy that we achieved in the training phase. Select the maximum possible value of δ that creates the minimum number of big-sized HBs. A revised set of HBs, denoted by (L), with the corresponding min-max points (G, H) and a revised matrix F are given as inputs to the test phase. The KC-phase organizes the learned knowledge in a compact manner prior to the test phase. We get different levels of compaction based on the value of δ. It also depends on the way the HBs have been created in the first step of training. We can achieve better compaction if more number of small-sized HBs is created away from the decision boundaries in the training phase.

4.3 Phase 3: Test phase

The test set is given as input to the fully-trained classifier. The fuzzy membership value of each test pattern is calculated in the revised set of HBs (G, H), using (2). The HB giving the maximum membership to a test pattern is identified as the winner. The class to which this HB belongs is detected as the class of a test pattern. We achieve faster recall after compaction as the fuzzy membership value of a test pattern is calculated for L number of HBs instead of K, where the value of L is substantially less than K.

The basic aim of this work is to represent the learned knowledge in a coarse-grained manner using less number of big-sized HBs. The classifier is first trained using the FMN training algorithm, where the value of θ is finalized and then the HBs created for the finalized value of θ are combined using the KC-algorithm. The compaction is done till it does not drop the accuracy achieved by the classifier in the training phase. The compact knowledge represented with the less number of big-sized HBs is used for the test phase. The big-sized HBs after compaction will fit in all the training patterns which were earlier covered by the small-sized HBs. This

Algorithm 1 Knowledge compaction algorithm.

Set an upper bound on compaction, called HB-size-max-limit (δ)
{Find HBs belonging to the same class as that of p-th HB}
for p =1 to K **do**
  for q =1 to K **do**
    if p != q AND classof[p] = classof[q] **then**
      Add index q to t
    end if
  end for
  {Combine p-th HB with all the HBs in t iff it does not create any overlap with the HBs of other classes}
  for q =1 to sizeof(t) **do**
    Calculate min-max points (g, h) using (8) and (9) for newly created r-th HB, generated after combining p-th and q-th HBs
    size-of-newly-created-HB (δ^{n}) = Euclidean distance (g, h)
    Test overlap of r-th HB with all existing, K number of HBs using overlap test [18]
    {Combine p-th and q-th HB if no overlap detected and size of r-th HB is less than the upper bound (δ)}
    if No overlap detected AND $\delta^{\mathrm{n}} < \delta$ **then**
      Delete the min-max points of p-th and q-th HBs from (V, W)
      Add new min-max points, g and h to (V, W)
    end if
  end for
end for
{After compaction, we get L number of HBs with the revised min-max points G, H}
$G = V$
$H = W$
Update matrix U using (3) for revised set of HBs, calling it as F
return G, H, F

compact representation reduces the computation time for the recall phase without compromising the accuracy. Next section illustrates our proposed method with an example.

5. Illustration of proposed method

We use the Fisher Iris dataset [33] to illustrate our proposed method. The Fisher Iris dataset is a standard dataset used in a wide range of classification techniques. It has three classes of Iris Plants: Iris-setosa, Iris-versicolor, and Iris-virginica. This dataset consists of a total of 150 patterns, 50 patterns of each class, where each pattern is a four-dimensional feature vector. We use all three phases of our method on this dataset, as follows.

5.1 Phase 1: Training phase

We select 100 patterns randomly from the dataset and for simplicity use only the first two dimensions of the Iris dataset for training. In order to make the computation simpler, we rescale the 2-dimensional input pattern space to the 2-dimensional unit cube I^2. The training dataset after pre-processing is shown in Fig. 3.

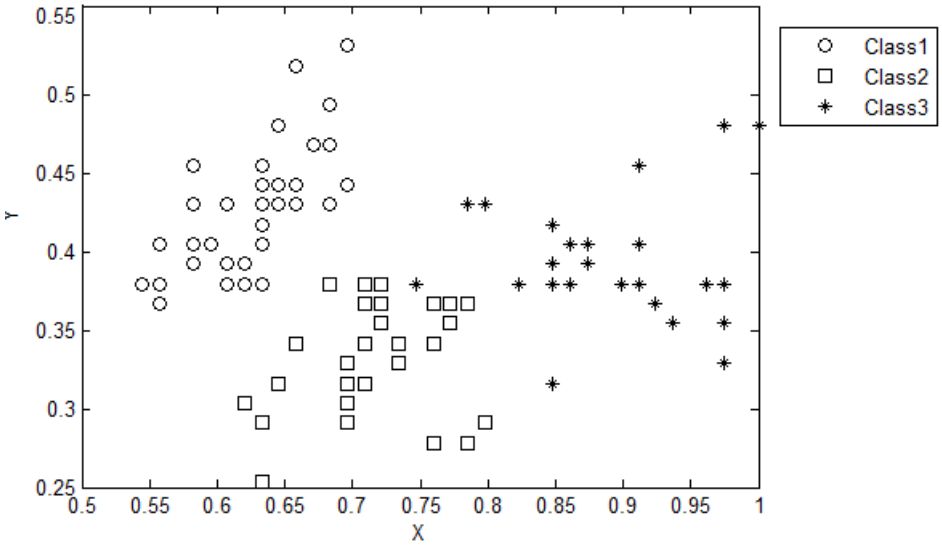

Fig. 3 *Scatter plot: IRIS data converted into 2-dimensional unit cube (I^2) used for training.*

The FMN is trained using the training dataset as shown in Fig. 1 by varying the value of θ and the size that gives nearly 100 % classification with the least number of HBs is finalized for the KC-phase. Fig. 4 shows the HBs and non-linear decision boundaries created for the three classes of the Iris dataset after training phase.

From Fig. 4, we observe that the FMN creates many HBs even away from the decision boundaries. That is, the classifier stores the knowledge in a fine-grained manner. Normally, the KC-phase described in next section tries to combine the HBs away from the decision boundaries as the HBs near the decision boundaries could easily create overlap after merging and cause misclassification. However, the HBs near the decision boundaries could also be combined if they do not create any overlap with the HBs belonging to different classes after merging.

5.2 Phase 2: Knowledge compaction phase

The min-max points of the HBs created in the training phase are given as input to the knowledge compaction phase. For combining HBs, we set the value of HB-size-max-limit (δ) greater than the HB-size (θ) that was set in the training phase.

We execute the knowledge compaction phase by increasing the value of δ till we get maximum possible or nearly 100 % classification and finalize the best possible

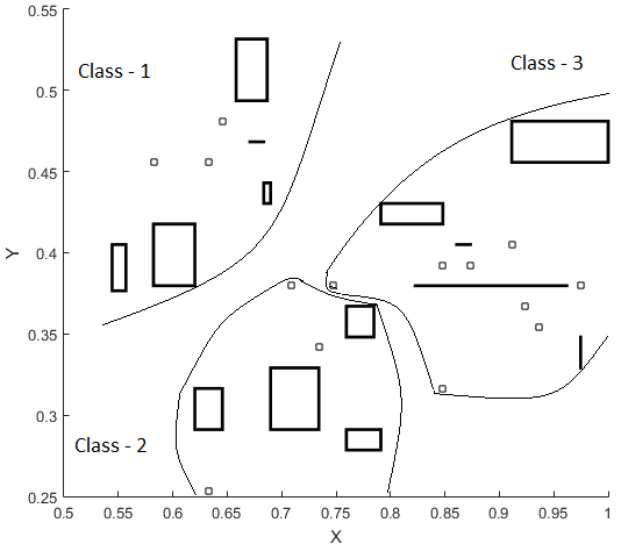

Fig. 4 *HBs and Decision boundaries created for three classes of Iris dataset after training phase of FMN (HB-size, θ = 0.07).*

maximum value for δ with least number of HBs. Fig. 5 shows the HBs and decision boundaries created for three classes of the Iris dataset after executing the knowledge compaction phase.

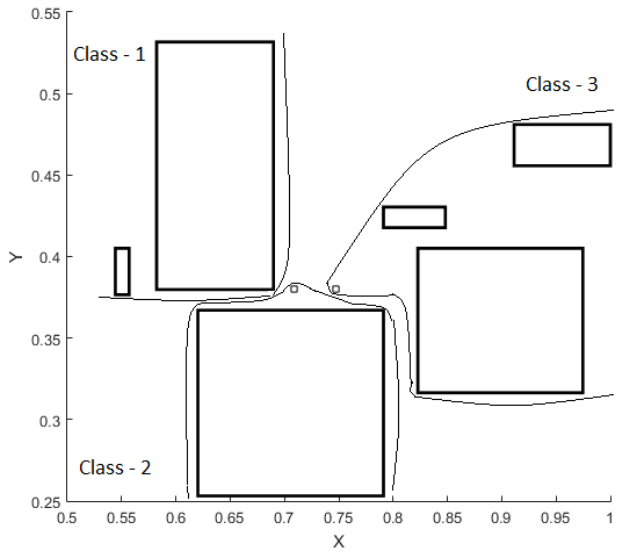

Fig. 5 *HBs and decision boundaries created for 3-classes of Iris dataset after executing Knowledge compaction phase (HB-maximum-limit (δ) = 0.21).*

5.3 Phase 3: Test phase

Tab. I. shows the comparative performance of the original FMN and our FMN-KC for the Iris dataset.

Classifier	HB-size (θ) / (δ)	Number of HBs (K) / (L)	Detection rate (%)	Recall time per pattern (ms)
Original FMN	0.07	28	100	0.72
FMN-KC	0.21	8	100	0.30

Tab. I *Comparative performance of the original FMN and FMN-KC for Iris dataset.*

It can be observed from Tab. I. that the proposed method achieves more than 50 % gain in the recall time while having the comparable detection rate. We have used a small-sized dataset to understand how the HBs are combined in the knowledge compaction phase. For the large-sized datasets, our modified architecture would provide considerable improvement in the recall time. In next Section, the simulation of the proposed method has been done for the large-sized datasets.

6. Simulation results for supervised outlier detection

We employ our proposed FMN-KC for outlier detection. Experiments have been done using two benchmark datasets, a time-series disk defect dataset and a KDD cup 99 dataset. We use the detection rate, false alarm, and recall time per pattern as the standard metrics to evaluate the performance of the proposed algorithm. The detection rate is the total number of predictions that are correct. The false alarm is the ratio of the number of normal patterns that are classified as outliers to the total number of normal patterns.

6.1 Outlier detection using synthetic dataset

Here, we use a 2-dimentional synthetic dataset containing some regular patterns and few outlier patterns for experiment, as shown in Fig. 6.

Class1 and Class 2, both representing regular patterns, are shown in Fig. 6. We use 80 % patterns selected from each regular and outlier category for training and the remaining 20 % patterns for testing. We select the value of θ for training and δ for the knowledge compaction phase that give nearly 100 % classification with least number of HBs. The results of the original FMN and our FMN-KC for outlier detection on the test dataset are shown in Tab. II.

As evident from Tab. II, our FMN-KC reduces the recall time per pattern almost half of the original FMN.

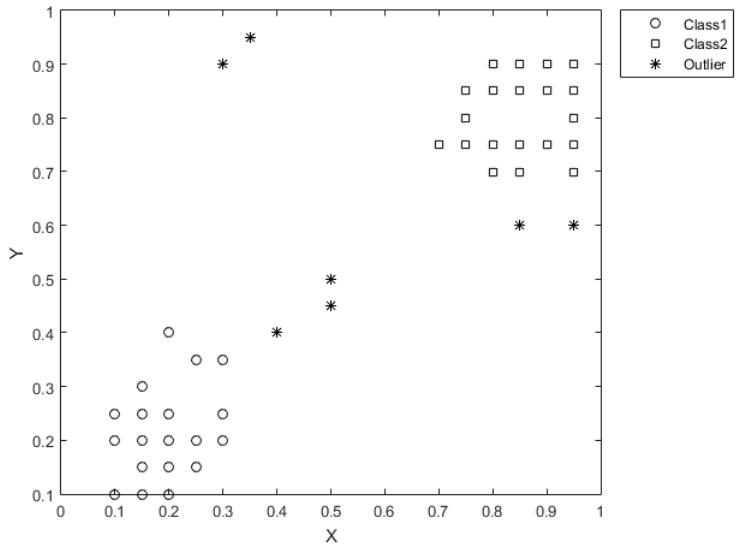

Fig. 6 *Synthetic dataset.*

Classifier	HB-size (θ) / (δ)	Number of HBs (K) / (L)	Detection rate (%)	Recall time per pattern (ms)
Original FMN	0.15	13	100	0.19
FMN-KC	0.30	7	100	0.10

Tab. II *Comparative performance of original FMN and FMN-KC for outlier detection using synthetic dataset.*

6.2 Fault detection using time-series disk defect dataset

The Rotary Dynamics Laboratory at NASA Glenn Research Centre acquired a time series data on the simulated engine disks to detect cracks in it. These anomalies could be detected in advance to prevent accidents. The data was recorded in three different states: normal, notch, and large notch at three different speeds measured in rpm. The class-distribution of patterns in disk defect dataset when disk moved at different speeds is shown in Tab. III.

Speed	Normal	Fault	Severe fault	Total
3krpm-3min-50sec	8997	8998	8998	26993
4krpm-3min-50sec	11998	11997	11998	35993
5krpm-3min-50sec	14998	14998	14998	44994
	Total			107980

Tab. III *%-distribution of patterns in Disk defect dataset.*

Higher rpm runs have more samples since there are more revolutions over the same period of time. The normal or healthy state represents the regular data, the faulty state indicates anomalous data with small notch or crack on the disk, and the severe fault indicates anomalous data with large notch or crack. Each pattern vector has been represented with 38 parameters that include clock time, measured RPM, maximum gap across all gap sensors, average gap across all gap sensors, minimum gap across all gap sensors, and measured gap for total 32 blades, and the last parameter is the state to which the pattern belongs. All these parameters have been sampled once per revolution. We have performed experiment on total 10,000 patterns randomly selected from the dataset when disk is moving at a speed of 3krpm with 33.33 % patterns from each category including the normal, faulty, and severe faulty. The training dataset is prepared with 80 % patterns selected from each class and remaining 20 % patterns are used for testing as shown in Tab. IV.

Class	Training dataset (80 % patterns)	Test dataset (20 % patterns)	Total patterns (%)
Normal	2668	666	33.34
Fault / Notch	2666	667	33.33
Severe fault / Large notch	2666	667	33.33
Total	8000	2000	100.00

Tab. IV *Number of patterns selected for experiment from Disk defect dataset when disk is moving at a speed of 3krpm.*

The data is pre-processed to make the computations simple. All the features are normalized in the range [0, 1] using (10).

$$x = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{10}$$

where x is a numerical value of a feature. The maximum and minimum values of feature that x belong to are represented as $x_{\max}$ and $x_{\min}$, respectively.

After pre-processing, the FMN is trained on the training dataset to get nearly 100 % classification. The time taken to train the classifier is given as
Time taken for training = 1488.5 seconds

The test dataset is given as input and the performance of the original FMN for fault detection has been measured with K number of HBs as shown in Tab. V. The HBs created in the training phase are given as input to the knowledge compaction phase. We have set the HB-size-max-limit (δ) such that it is the maximum possible value of the HB giving 100 % classification and is greater than the HB-size (θ) that was finalized in the training phase. This phase has created L number of HBs of size δ by combining the HBs away from the decision boundary. The time taken for the compaction of knowledge is given as
Time taken for knowledge compaction = 37114.0 seconds

The test dataset is given as input and the performance of the FMN-KC for fault detection is measured and the results are shown in Tab. V.

It can be observed from Tab. V. that the proposed FMN-KC reduces the number of HBs for the disk defect dataset, which in turn improves the recall time per pattern

Classifier	HB-size (θ) / (δ)	Number of HBs (K) / (L)	False Alarm	Detection rate (%)	Recall time per pattern (ms)
Original FMN	0.000001	5385	0.172	83.65	704.5
FMN-KC	0.000300	2362	0.111	83.80	215.9

Tab. V *Performance of original FMN and FMN-KC for fault detection on a time-series disk defect dataset.*

as compared to the original FMN, with comparable detection rate. The following formula has been used to calculate the % gain in time achieved by the proposed FMN-KC.

$$\%\text{Gain in time} = \frac{\text{Recall time for Original FMN} - \text{Recall time for proposed FMNKC}}{\text{Recall time for Original FMN}}.100 \tag{11}$$

The proposed FMN-KC has achieved 69.4 % gain in the recall time for the disk defect dataset as compared to the original FMN.

6.3 Signature based intrusion detection using KDD cup 99 dataset [34]

The KDD cup 99 intrusion detection dataset is based on the DARPA 98 dataset containing a wide variety of intrusions simulated in a military network environment. There are three different components of KDD 99 intrusion detection dataset. The 10-percent dataset is a more concise version of the Whole KDD dataset. Additional 14 types of attack are available in Corrected KDD dataset that also provides different statistical distributions than 10-percent KDD or Whole KDD. We use the corrected KDD dataset for experiments. There are total of 311029 single connection vectors, each described by 41 features, labelled as either normal or attack with a specific type of attack. There are total 37 types of attacks, belonging to one of the following four categories:

i. Denial of Service (Dos): Attacker tries to prevent a legitimate user from using service.

ii. Remote to Local (R2l): unauthorized access from a remote machine.

iii. User to Root (U2r): Attacker is having local access to the victim's machine and tries to get the super-user privileges.

iv. Probe: Attacker tries to capture information about the target host for circumventing its security controls.

Tab. VI. shows the number of pattern vectors available in the corrected KDD dataset under five different categories, viz. Normal, DoS, Probe, R2L, and U2R, and their percentage distribution.

Class	Number of Patterns	% distribution
Normal	60593	19.48
R2L	16347	5.25
DoS	229853	73.90
Probe	4166	1.34
U2R	70	0.03
Total	311029	100.00

Tab. VI *%-distribution of patterns in corrected KDD dataset.*

The KDD-99 dataset contains features having all forms of data including symbolic, continuous, and discrete. In pre-processing phase, the symbolic features like protocol-type, service, and flag are mapped to integer values ranging from 0 to N-1, where N is number of symbolic values available in the corresponding feature. For example, the protocol-type feature with three different symbols, namely, UDP, TCP, and ICMP have been mapped to three discrete numeric values 1, 2 and 3, respectively. The features available in continuous or discrete form have been used in the original form. All the features have been normalized between 0.0 and 1.0 using (10). We have performed the experiment with a total of10000 patterns, selected from the dataset with similar percentage distribution of patterns from all the categories including the normal and attack types, as shown in Tab. VII. The 80 % patterns have been selected from each class to prepare a training dataset and the remaining 20 % patterns for test, as shown in Tab. VII.

Class	Training dataset (80 % patterns)	Test dataset (20 % patterns)	Total patterns (%)
Normal	1600	400	20.00
R2L	554	138	6.92
DoS	5600	1400	70.00
Probe	240	60	3.00
U2R	6	2	0.08
Total	8000	2000	100.00

Tab. VII *Number of patterns selected for experiment from KDD corrected dataset.*

As shown in Tab. VII, the time taken by the FMN for training on the training dataset is given as

$$\text{Time taken in training} = 1431.63 \text{ seconds}$$

In the test phase, the test dataset as shown in Tab. VII is given as input and the performance of the original FMN for signature-based intrusion detection is measured with K number of HBs as shown in Tab. VIII. The HBs created in the training phase are given as input to the knowledge compaction phase and the value

of δ is set as the maximum possible value of the HB giving 100 % classification, creating L number of HBs. The time taken for the compaction of knowledge is given as

$$\text{Time taken for knowledge compaction} = 24780.05 \text{ seconds}$$

The performance of our FMN-KC has been tested for signature-based intrusion detection with L number of HBs and the results are shown in Tab. VIII.

Classifier	HB-size (θ) / (δ)	Number of HBs (K) / (L)	False Alarm	Detection rate (%)	Recall time per pattern (ms)
Original FMN	0.000000009	3831	0.0075	99.4	491.5
FMN-KC	0.00003	2374	0.0075	99.4	254.9

Tab. VIII *Performance of the original FMN and FMN-KC for signature-based intrusion detection on KDD corrected dataset.*

It can be observed from Tab. VIII. that the FMN-KC has reduced the number of HBs for KDD dataset, that is, the knowledge has been represented in a compact form. The FMN-KC has achieved the gain in the recall time as compared to the original FMN and the recognition rate is also comparable. For the KDD corrected dataset, the proposed FMN-KC has achieved 48.1 % gain in the recall time as compared to the original FMN, calculated using (11). As evident from Tab. V. and Tab. VIII, the proposed FMN-KC has obtained a significant gain in the recall time as compared to the original FMN without dropping the detection accuracy of the classifier that proves the success of the proposed method.

7. Conclusions

The neuro-fuzzy approach is very promising for outlier detection as it exactly addresses the challenges associated with outliers, such as uncertainty involved in it, false alarm, and faster recall. A fuzzy min-max neural network (FMN) proposed by Simpson is a well-known supervised neuro-fuzzy classifier used for pattern recognition. The FMN creates more number of HBs even away from the decision boundary and stores the learned knowledge with extensive details in a fine-grained manner. In this paper, we have added a knowledge compaction phase in the basic architecture of the FMN, calling it as fuzzy min-max neural network with knowledge compaction (FMN-KC), to represent the knowledge in a coarse-grained manner. We have evaluated its performance for the supervised outlier detection. The KC-phase combines the HBs of the same class that are away from the decision boundary without creating any overlap with the HBs of other classes. This phase changes the representation of the internal knowledge before the test phase. This compact and coarse-grained representation of the learned knowledge improves the recall time per pattern. Experimentally, we have shown that the proposed FMN-KC achieves around 50 % gain in the recall time as compared to the original FMN. That is, the FMN-KC requires almost half of the time that of the original FMN. This significant

gain in the time without dropping the detection accuracy of the classifier proves the success of the proposed method. The FMN-KC however requires additional time for compaction of the knowledge. Though the training activity needs to be done only for few times (generally once) as opposed to the recall activity which is a recurring activity, the extra time needed for FMN-KC should not be recognized as serious handicap of this architecture. The FMN-KC outperforms the existing FMN in terms of the recall time per pattern. So, the proposed FMN-KC can be used for supervised outlier detection in real time applications where the recall time per pattern is one of the key parameters.

References

[1] HAWKINS D.M. Identification of outliers, *London: Chapman and Hall*, Vol. 11, Springer, 1980.

[2] CATENI S., COLLA V., VANNUCCI M. Outlier detection methods for industrial applications. In: *Advances in Robotics, Automation and Control*, InTech, 2008, doi: 10.5772/5526.

[3] LANE T., BRODLEY C.E. Temporal sequence learning and data reduction for anomaly detection, *ACM Transactions on Information and System Security (TISSEC)*, 2(3), 1999, pp. 295–331. doi: 10.1145/288090.288122.

[4] BOLTON R.J., HAND D.J. Statistical fraud detection: A review, *Statistical science*, 2002, pp. 235–249.

[5] CHIU A.L.M., FU A.W.C. Enhancements on local outlier detection, in: Database Engineering and Applications Symposium, 2003. *Proceedings. Seventh International, IEEE*, 2003, pp. 298–307, doi: 10.1109/IDEAS.2003.1214939.

[6] AL-ZOUBI M.B., KAMEL A.M., RADHY M.J. Techniques for image enhancement in the spatial domain, *WSEAS Transactions on Computers*, 5(5), 2006, pp. 1047–1052.

[7] YANG J., ZHAO Z., ET. AL. A fast geometric rectification of remote sensing imagery based on feature ground control point database, *WSEAS Transactions on Computers*, 8(2), 2009, pp. 195–204.

[8] KNOX E.M., NG R.T. Algorithms for mining distancebased outliers in large datasets. In: *Proceedings of the international conference on Very Large Data Bases*, Citeseer, 1998, pp. 392–403.

[9] KNORR E.M., NG R.T., TUCAKOV V. Distance-based outliers: algorithms and applications, *The VLDB Journal—The International Journal on Very Large Data Bases*, 8(3-4), 2000, pp. 237–253.

[10] HODGE V., AUSTIN J. A survey of outlier detection methodologies, *Artificial intelligence review*, 22(2), 2004, pp. 85–126.

[11] AGGARWAL C.C. An introduction to outlier analysis. In: *Outlier analysis, Springer*, 2013, pp. 1–40, doi: 10.1007/978-1-4614-6396-2_1.

[12] FAN W., MILLER M., STOLFO S., LEE W., CHAN P. Using artificial anomalies to detect unknown and known network intrusions, *Knowledge and Information Systems*, 6(5), 2004, pp. 507–527.

[13] GOMEZ J., DASGUPTA D. Evolving fuzzy classifiers for intrusion detection. In: *Proceedings of the 2002 IEEE Workshop on Information Assurance*, Vol. 6, New York: IEEE Computer Press, 2002, pp. 321–323.

[14] ZHANG Z., LI J., MANIKOPOULOS C., JORGENSON J., UCLES J., Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In: *Proc. IEEE Workshop on Information Assurance and Security*, 2001, pp. 85–90.

[15] HAWKINS S., HE H., WILLIAMS G., BAXTER R. Outlier detection using replicator neural networks. In: *International Conference on Data Warehousing and Knowledge Discovery*, Springer, 2002, pp. 170–180, doi: 10.1007/3-540-46145-0_17.

[16] TOOSI A.N., KAHANI M., MONSEFI R. Network intrusion detection based on neuro-fuzzy classification. In: *International Conference on Computing and Informatics (ICOCI'06)*, IEEE, 2006, pp. 1–5.

[17] UPASANI N., OM H., Outlier detection: A survey on techniques involving fuzzy and/or neural approaches. In: *IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions*, IIT Kanpur, India, July 2013, pp. 28–32.

[18] SIMPSON P.K. Fuzzy min-max neural networks – Part 1: classification, *IEEE transactions on neural networks*, 3(5), 1992, pp. 776–786. doi: 10.1109/72.159066.

[19] QUTEISHAT A., LIM C.P. Application of the fuzzy min-max neural networks to medical diagnosis. In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Springer, 2008, pp. 548–555. doi: 10.1007/978-3-540-85567-5-68.

[20] PANICKER S.S., DHABE P., DHORE M. Fault diagnosis using fuzzy min-max neural network classifier, *Artificial Intelligent Systems and Machine Learning*, 2(7), 2010, pp. 95–101.

[21] MOHAMMADI M., PAWAR R.V., DHABE P.S. Heart diseases detection using fuzzy hyper sphere neural network classifier, *CiiT International Journal of Artificial Intelligent Systems and Machine Learning*, Issue Jul, 2010.

[22] POTEY M.A., UPASANI N.V. Generalized ring averaging: a new method for left and right directional illumination invariant face recognition for frontal poses and their small variants. In: *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, ACM, 2010, pp. 206–210. doi: 10.1145/1741906.1741951.

[23] JAWARKAR N., HOLAMBE R., BASU T. Use of fuzzy min-max neural network for speaker identification. In: *International Conference on Recent Trends in Information Technology (ICRTIT)*, IEEE, 2011, pp. 178–182. doi: 10.1109/ICRTIT.2011.5972455.

[24] CHOWHAN S., SHINDE G. Iris recognition using fuzzy min-max neural network, *International Journal of Computer and Electrical Engineering*, 3(5), 2011, pp. 743. doi: 10.7763/IJCEE.2011.V3.414.

[25] SUSAN S., KHOWAL S.K., KUMAR A., KUMAR A., YADAV A.S. Fuzzy min-max neural networks for business intelligence. In: *International Symposium on Computational and Business Intelligence (ISCBI)*, IEEE, 2013, pp. 115–118. doi: 10.1109/ISCBI.2013.31,115-11.

[26] MENEGANTI M., SAVIELLO F.S., TAGLIAFERRI R. Fuzzy neural networks for classification and detection of anomalies, *IEEE Transactions on Neural Networks*, 9(5), 1998, pp. 848–861. doi: 10.1109/72.712157.

[27] QUTEISHAT A., LIM C. P. A modified fuzzy min–max neural network with rule extraction and its application to fault detection and classification, *Applied Soft Computing*, 8(2), 2008, pp. 985–995. doi: 10.1016/j.asoc.2007.07.013.

[28] UPASANI N., OM H. Evolving fuzzy min-max neural network for outlier detection, *Procedia computer science*, 45, 2015, pp. 753–761. doi: 10.1016/j.procs.2015.03.148.

[29] GABRYS B., BARGIELA A. General fuzzy min-max neural network for clustering and classification, *IEEE transactions on neural networks*, 11(3), 2000, pp. 769–783.

[30] NANDEDKAR A.V., BISWAS P.K. A fuzzy min-max neural network classifier with compensatory neuron architecture, *IEEE transactions on neural networks*, 18(1), 2007, pp. 42–54.

[31] CARPENTER G.A., TAN A.-H. Rule extraction: From neural architecture to symbolic representation, *Connection Science*, 7(1), 1995, pp. 3–27.

[32] LIU J., HE X., YANG J. A fuzzy min-max neural network classifier based on centroid. In: *Control Conference (CCC)*, 30th Chinese, IEEE, 2011, pp. 2759–2763.

[33] DHEERU D., KARRA TANISKIDOU E. UCI machine learning repository 2017. http://archive.ics.uci.edu/ml.

[34] KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[35] KENNEDY R. The hyperspherical attractor neural network: concepts, theory and application. In: *Proc. WNNAIND*, 91, 1991, pp. 241–271.