Malaysian Journal of Computer Science, Vol. 14 No. 1, June 2001, pp. 68-82

MACHINE LEARNING: THE AUTOMATION OF KNOWLEDGE ACQUISITION USING KOHONEN SELF-ORGANISING MAP NEURAL NETWORK

Mohamed Khalil Hani, Sulaiman Mohd Nor, Sheikh Hussein and Nazar Elfadil
MiCE Department
Faculty of Electrical Engineering
University of Technology Malaysia
81310 UTM Skudai, Johor, Malaysia
Fax: +607-5566272
email: gazoli@mailcity.com

ABSTRACT

In machine learning, a key aspect is the acquisition of knowledge. As problems become more complex, and experts become scarce, the manual extraction of knowledge becomes very difficult. Hence, it is important that the task of knowledge acquisition be automated. This paper proposes a novel method that integrates neural network and expert system paradigms to produce an automated knowledge acquisition system. A rule-generation algorithm is proposed, whereby symbolic rules are generated from a neural network that has been trained by an unsupervised Kohonen self-organising map (KSOM) learning algorithm. The generated rules are evaluated and verified using an expert system inference engine. To demonstrate the applicability of the proposed method to real-world problems, a case study in medical diagnosis is presented.

Keywords: ***Kohonen self-organising maps, Machine learning, Knowledge acquisition, Expert system, Rule extraction***

1.0 BACKGROUND

There are two major approaches in machine learning, that is, symbolic and connectionist. Until the last decade both approaches progress independently. In the last five years, researchers have started investigating ways of integrating these artificial intelligence (AI) paradigms together [1]. Closer examination of the symbolic and connectionist (or adaptive processing) divide reveals that both approaches have a combination of advantages and limitations, and that integrating these different techniques can overcome their individual weaknesses [2]. In combining the paradigms, neural networks can be viewed as mechanisms for generating goals (learning), and expert systems as mechanisms for proving the goals (decision-making and explanation).

Basically, learning in a neural network is simply the problem of finding a set of connection strengths that allow the network to capture any regularities, statistical features, and probability distributions present in input data [3]. The learning capability of neural network is its strength, but this is limited by drawbacks, such as: (a) lack of a structured knowledge representation, (b) lack of inheritance, (c) inability to interact with conventional symbolic databases, and (d) inability to explain the reasons for conclusions reached. These apparent weaknesses can be overcome by integrating neural network with expert systems. But, expert systems have their limitations too. The main practical problem in building a conventional expert system is the construction and debugging of its knowledge base. It is usually difficult and expensive to get a human expert to express his/her knowledge in terms of the required IF-THEN rules, particularly in real-world problem cases. Also, once extracted, a set of rules is almost certain to be incomplete, inconsistent, and require tuning of the rules and confidence factors. Whilst expert systems' framework is symbolic, neural network representations are suited for numerical or statistical tasks, and they can be used in many different situations without the need for a detailed understanding of the problem [2]. Hence, the complementation of neural network and expert system in overcoming each other weaknesses.

Knowledge acquisition is the process of learning from one or more sources and passing on the knowledge acquired, via a suitable form, to someone else or to some other system. This process involves learning, reformalising, transferring, and representing the knowledge. Knowledge is not only sourced from human expert, books or databases, but also from behavioral patterns of real systems. For example, vibration data from machinery can provide knowledge about its maintenance status. Historical data of trends in the response times can lead to knowledge about possible component failure in computer networks. We will need tools to tap this information and

bring it to a level of abstraction where it can be used for decision making and planning. These tools will require machine-learning capabilities, more specifically, knowledge acquisition.

Loonely in [4] stated that automated knowledge acquisition methods can be divided into three major types: (a) Induction of Decision Trees (ID3), developed by Qunlan in 1986, (b) Clustering And Regression Tree (CART) developed by Bneiman in 1984, and (c) Chi- square statistic and tests of hypotheses, developed by Heayslet in 1976. Ultsch in [5] stated that ID3 is considered the better technique among the three, but it has serious shortcomings. ID3 uses a minimalisation criterion that seems to be unnatural for human expert in that, rules generated use only a minimal set of decisions to come to a conclusion. This is not quite the case in most real problems. For example, in the medical problem domain, the number of decisions is based on the types of disease. In simple cases, i.e. where the symptoms are unanimous, very few tests are made, while in difficult cases a diagnosis must be based on all available information [6, 7]. The Induction of Decision Trees method gave poor result in this latter case. What is needed is a rule generation algorithm that takes into account the significance of a symptom (range of a component value). Current work on automated knowledge acquisition suggests that integrating neural networks and inference systems can overcome the problems stated, and this concept is explored here.

However, not all problems are suitable for neural expert system integration [2, 3, 8]. The most suitable ones are those that seek to classify inputs into a small number of groups. Hence, candidate problems suitable for the proposed method include, among others, medical diagnosis, fault detection, process control, credit and loan, and network management. Another important factor in determining the applicability of a neural expert system approach is the availability of training examples; the more available are the training examples, the better results could be achieved. Also, it should be noted that training data might be typical cases given by a human expert. The success of the proposed technique is contingent on a rule extraction algorithm that overcomes the ID3 problem stated above.

This paper is organised as follows: Section 2 presents an overview of proposed method. Section 3 discusses the automated knowledge acquisition. In section 4 we demonstrate the experimental results. Finally, section 5 illustrates the conclusion and future work.

2.0 OVERVIEW OF PROPOSED METHOD

Fig. 1 illustrates the methodology of proposed method, which integrates neural network and expert system. Knowledge (connectionist) is extracted from data that have been clustered by a KSOM neural network. The knowledge at this stage is of the immediate-level concept rule hierarchy. A rule generation algorithm that is aided by an expert system inference engine is then applied to generate the final concept rule hierarchy. The knowledge generated at this point (symbolic) may be used in the construction of the symbolic knowledge base of an expert system.

The proposed system, as depicted in Fig. 2, consists of four main phases: data preprocessing, learning and clustering, rules generation and knowledge verification. The data-preprocessing step essentially transforms the raw data into a format acceptable to the subsequent neural network phase. The second phase deals with neural network training using KSOM unsupervised learning algorithm, and clustering via the K-means algorithm. The rule generation phase extracts symbolic rules from the data that have been clustered by the neural network. The last task deals with the verification of acquired knowledge. The next section presents the proposed system in detail.

3.0 AUTOMATED KNOWLEDGE ACQUISTION

To explain the details of the various tasks in the proposed knowledge acquisition method, we will employ a simple but demonstrative example of a 7-segment display numeric recognition problem. This problem example contains data defining 10 decimal digits (0-9) on a 7-segment LED display as shown in Fig. 3. Each input pattern is composed of 7 binary attributes. The problem is to recognise the number displayed, given the input pattern of the seven segments. As the Kohonen neural net requires a large data set for learning effectively, the relevant data set was repeatedly fed to the network to simulate approximately 200 examples. No priori information is provided in the neural network training. Table 1 presents a sample of the input data that will be used in our discussions henceforth.

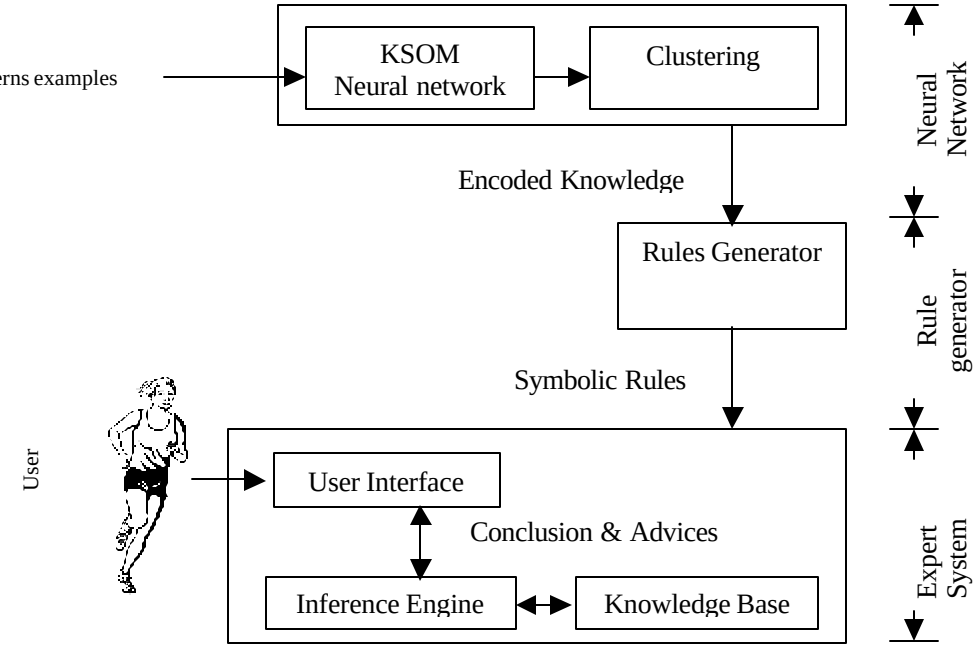


Fig. 1: Neural Expert system architecture

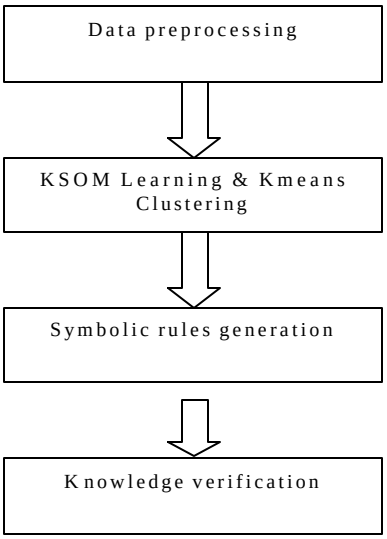


Fig. 2: The functional block diagram of knowledge acquisition system

3.1 Data Pre-Processing

The raw data must first be transformed into a new data set and format suitable for application in a neural network. All unsupervised methods merely illustrate some structures in the data set, and the features chosen to represent the data items ultimately determine the structures. The pre-processing step selects a set of features that are invariant with respect to some basic transformation groups of the input pattern. The pre-processing phase is composed of three main subtasks, namely: loading of data file, normalisation, and feature extraction.

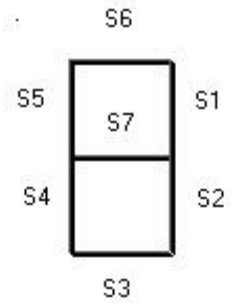


Fig. 3: 7-segment display unit

Table 1: Portion of 7 segment display unit data

Pattern Number	Pattern Syntax						
	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	1
3	1	1	0	0	1	0	1
4	1	1	0	0	0	0	1
5	1	0	1	1	0	1	1
6	1	1	1	0	0	1	1
7	1	1	0	0	1	0	1
8	0	1	1	0	1	1	1
9	0	1	1	1	1	1	1
10	1	1	1	1	1	1	1
11	1	1	1	1	1	1	0
12	1	1	1	0	1	1	1

Note:
S1 = Up_right, S2= Down_right, S3= Down_center,
S4= Down_left, S5= Up_left, S6= Up_center,
and S7= Mid_center

In feature extraction, the input data vectors are transformed into such representations that concisely best describe the problem from the analysis point of view. For example, in speech recognition, spectral data computed using the Fourier Transform are used as the feature vectors. In the 7_seg display case study, there is no necessity for feature extraction.

In normalisation, the objective is to ensure that there is no data that dominates over the rest of input data vectors. Although normalisation computations incur speed cost and it is not always necessary, it was found that it significantly improves the overall numerical accuracy of statistical computations in connection with self-organising maps (SOM) algorithms [9]. Normally, it is advisable to normalise each attribute scale such that its variance taken over all the items is unity. There are several methods for normalisation. We conduct comprehensive trials of all the methods, and then make the choice of the most suitable to apply, based on the quality metrics of quantisation and topographic errors. The quantisation error is used as a measure of the resolution of the mapping, while topographic error calculates the error in the proportion of sample vectors for which two best matching weight vectors are not in adjacent units [10]. The advantage of these metrics is that the results are directly comparable between different mappings and even mappings of different data sets. Table 2 shows the result of the trials for different normalisation types in the case of the 7_segment display data. The range of normalisation is chosen as in the table.

Table 2: Quality measure of different normalisation (7-seg display case study)

No	Quantisation Error	Topographic Error	Normalisation Type
1	2.365	0.026	Variance
2	0.267	0.0004	Range
3	1.739	0.036	Logarithmic
4	0.632	0.026	Histogram Discrete
5	0.735	0.036	Histogram Continuous

3.2 Neural Network Learning and Clustering

This phase involves three main tasks namely: initialisation, training, and clustering. The process of learning can be categorised into supervised and unsupervised learning. Supervised learning requires a training set composed of input patterns with associated target or desired outputs. The target output acts like an external teacher to the network. In unsupervised learning the training set consists solely of input patterns. Hence, during learning, no comparison with predetermined desired responses on which to base subsequent modifications can be performed by the learning algorithm. There is therefore no external teacher in the sense described earlier. This situation is more akin to most real-world problems. The proposed method employs unsupervised learning, and this is the key contribution of this research. Once trained, the weights of the neural network are mapped to clusters, which facilitates the extraction of the knowledge, encoded in the net by the rule generation algorithm. Let us now discuss each of these tasks in turn.

3.2.1 Initialisation

The initialisation task involves three tasks namely: weight initialisation, topology initialisation, and neighborhood initialisation. The hexagonal lattice type is chosen as the map topology in 7-segment display problem examples. Fig. 4 illustrates the KSOM topology, which is composed of 150 output nodes and 7 input nodes. The choice of number of output nodes is done through comprehensive trials. (We have considered this issue, but its discussion is beyond the scope of this paper).

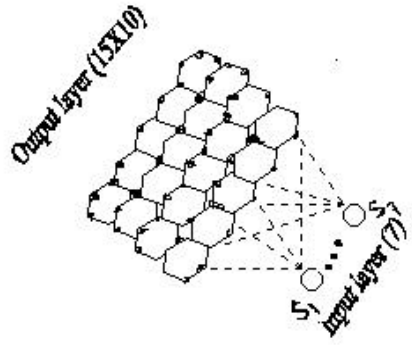


Fig. 4: KSOM topology

The weights of the neural network is initialised either by linear or random initialisation. The random initialisation technique is chosen here. Table 3 shows a portion of the resulting initial weight values. For the neighborhood function, Gaussian or Bubble is the typical choices. The bubble function is considered the simpler but adequate one, and it is applied here. In general, the topological relations and number of neurons are fixed from the beginning. The number of neurons are usually selected to be as large as possible, with the neighborhood size controlling the smoothness and generalisation of the mapping. If the neighborhood size is selected correctly, the mapping does not suffer significantly, even when the number of neurons exceeds the number of input vectors.

Table 3: A portion of the Neural Network weights after random initialisation
(7-segment display case study (15X10 output layer))

Output Nodes (Row, Column)	Input Nodes						
	S1	S2	S3	S4	S5	S6	S7
(1,1)	0.8533	0.8915	-0.1076	-0.1905	0.4809	0.0587	0.4623
(1,2)	0.8877	0.9713	0.0773	-0.0565	0.6208	0.2234	0.5417
(1,3)	0.9222	1.0511	0.2623	0.0775	0.7607	0.3880	0.6211
(1,4)	0.9567	1.1310	0.4473	0.2115	0.9006	0.5527	0.7005
(1,5)	0.9912	1.2108	0.6322	0.3455	1.0405	0.7173	0.7799
e: S1 = Up_right, S2 = Down_right, S3 = Down_center, S4 = Down_left, S5 = Up_left, S6 = Up_center, and S7 = Mid_center. (Row, Column) is used to represent a particular node in the 2D output layer which consist of 15X 10 nodes.							

3.2.2 KSOM Training

The training of the KSOM neural network may be considered to consist of two phases: rough training and fine-tuning. The rough training phase correspond to the first iteration stage in which, the initial formation of the order occurs. In this stage, the initial weight vectors, a large neighborhood radius and a large learning rate are applied. Rough training is normally short. The rest of the iteration stages constitute the fine-tuning phase in which, both learning rate and neighborhood radius begin with small values, and gradually reduced further at each iteration. Fig. 5 outlines the KSOM learning algorithm.

Step 1: initialise weight to small random values and set the initial neighborhood to be large.
Step 2: stimulate the net with a given input vector.
Step 3: calculate the Euclidean distance between the input and each output node and select the output with the minimum distance.

$$D(j) = \sum_i (w_{ij} - x_i)^2$$

D(j) is a minimum

Step 4: update weights for the selected node and the nodes within its neighborhood.

$$w_{ij}(new) = w_{ij}(old) + \alpha(x_i - w_{ij}(old))$$

Step 5: repeat from step 2 unless stopping condition.

Fig. 5: KSOM learning algorithm

The stopping condition decides on the convergence achieved. We have two approaches, the choice of which depends on the size of input data and dimension of output layer. In the first approach, the winning node updates its parametric weight vector via the equation given in step 4 of the algorithm. All other neurons keep their old values. In the second approach, the strategy is to update positively all nodes that are close to the winning nodes, and update negatively all nodes that are farther away from the winner (i.e. lateral inhibition is applied). The first approach is used here. Table 4 illustrates a portion of the result of the KSOM training session. The required knowledge is encoded implicitly in the updated KSOM weights that link the input layer and output layer nodes.

Table 4: Portion of KSOM output (7-segment display case study)

Output Nodes (Row, Column)	Input Nodes						
	S1	S2	S3	S4	S5	S6	S7
(1,1)	1.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
(1,2)	1.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
(1,3)	1.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
(1,4)	1.0000	0.9975	0.0049	0.0025	0.0000	1.0000	0.0049
(1,5)	1.0000	0.5028	0.5028	0.4972	0.0000	1.0000	0.5028

3.2.3 Clustering

The next step is clustering the updated weights explicitly. The terminology comes from the appearance of an incoming sequence of feature vectors which arrange themselves into clusters, groups of points that are closer to each other and their own centers rather than to other groups. When a feature vector is input to the system, its distance to the existing cluster representatives is determined, and it is either assigned to the cluster, with minimal distance or taken to be a representative of a new cluster.

We have noted earlier that no priori information is provided about the clustering, hence the possible clusters (and even the number) are not known in advance. The K-means algorithm can perform the required clustering function. The K-means algorithm self-organises its input data to create clusters. We summarise the K-means algorithm as follows:

Step 1: From the given sample of feature vectors, select the first k sample vectors as initial centers.
Step 2: Assign each sample feature vector that is closest, in Euclidian distance, to center 1 to cluster 1, and which is closest to center 2 to cluster 2, and so on, to form k clusters.
Step 3: Obtain new optimal centers for each cluster by averaging the feature vectors contained in each k^{th} cluster.
Step 4: Assign each of the sample feature vectors again, to the cluster whose new center is closest to.
Step 5: Stop, when no clusters change further.

Fig. 6 shows a graphical representation for clustering session output. Each gray shade represents a certain cluster. Although the graphical representation is good in illustrating the clusters visually, it cannot be used to interpret explicitly the mapping that represents the encoded knowledge. We need to specify the links and weights associated with a cluster.

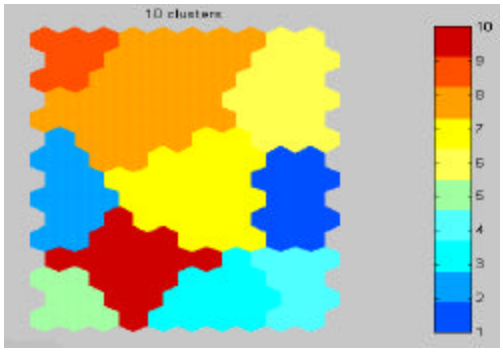


Fig. 6: Graphical representation of result of clustering session (7-seg display case study)

The step now is to find the codebook vector and the indices for each cluster. This data contains the weights that distinguish and characterise each cluster. Table 5 contains a portion of cluster 1 weights. The weight vector for an input unit in a clustering unit serves as a representative (exemplar) or codebook vector for the input patterns which the net has placed on that cluster.

Table 5: A part of cluster 1 weights or codebook vectors (7-segment display case study)

Output Nodes (Row, Column)	Input Nodes						
	S1	S2	S3	S4	S5	S6	S7
(2,3)	1.0000	0.6667	1.0000	1.0000	0.6667	1.0000	0.6667
(2,4)	1.0000	0.9895	1.0000	1.0000	0.9895	1.0000	0.0574
(2,5)	1.0000	0.9999	1.0000	1.0000	0.9999	1.0000	0.0475
(3,8)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0025
(3,9)	1.0000	0.9547	1.0000	0.9997	0.9544	1.0000	0.9547
Note: S1 = Up_right, S2 = Down_right, S3 = Down_center, S4 = Down_left, S5 = Up_left, S6 = Up_center, and S7 = Mid_center (Row, Column) is used to represent a particular node in the 2D output layer which consist of 15X 10 nodes.							

3.3 Symbolic Rule Generation

In this stage of the knowledge acquisition process, the extraction of a set of symbolic rules that map the input nodes into output nodes (with respect to each cluster) is performed. The antecedents of the rules that define these concepts consist of contributory and inhibitory input weights. The resulting rule base of this phase is an intermediate-level concept rule hierarchy.

Let us explain further. In a KSOM network, each output node is connected to every input node, with the strength of interconnection reflected in the associated weight vector. The larger the weight vector associated with a link, the greater is the contribution of the corresponding input node to the output node. The input with the largest weight link makes the largest contribution to the output node. To distinguish the contributory inputs from inhibitory inputs, we binarise the weights. If contributive, the real-valued weight is converted to 1, and is converted to 0 if inhibitory. There are two approaches to do this, namely: threshold or breakpoint technique [11]. We have chosen the threshold technique. The threshold is set at 50% (i.e. below 0.5 is considered as 0 and above 0.5 considered as 1). The mapping (a portion) produced by this binarisation step for 7-seg-display case study is given in Table 6.

Table 6: Portion of cluster 1 weights after using threshold technique (7-segment display case study)

Output Nodes (Row, Column)	**Input nodes**						
	S1	S2	S3	S4	S5	S6	S7
(2,3)	1	1	1	1	1	1	1
(2,4)	1	1	1	1	1	1	0
(2,5)	1	1	1	1	1	1	0
(3,8)	1	1	1	1	1	1	0
(3,9)	1	1	1	1	1	1	1
(3,10)	1	1	1	1	1	1	1
(5,4)	1	1	1	1	1	1	1
(2,3)	1	1	1	1	1	1	0

The final sets of antecedents in each cluster usually contain some duplicated patterns. This redundancy is now removed, and Table 7 shows the number of patterns covered by each cluster before and after redundancy. We can now map symbolically the antecedents to each cluster and obtain the rules for each cluster. The symbolic rule extraction algorithm is an inductive learning procedure. The algorithm as is provided in Fig. 7 is self-explanatory. The result of this step is a set of production rules (of the intermediate-level concept rule hierarchy), an example of which, for cluster 1 we have:

1. **IF** [(S1)& (S2)& (S3)& S4)& (S5)& (S6)& (S7)] **THEN** {cluster 1}
2. **IF** [(S1)& (S2)& (S3)& (S4)& (S5)& (S6)& (¬S7)] **THEN** {cluster 1}

Table 7: Number of patterns covered by each cluster before and after redundancy

Redundancy	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12	Cluster 13
Before	17	1	12	12	6	6	23	13	10	11	12	25	2
After	2	1	2	1	3	2	2	1	1	1	2	2	1

Step 0: with each U ∈ {all output layer's units}
Let W be the weight of unit U
Step 1: determine groups of positively weighted links (contributors) U such that (Link weights of group) = 1
{Call this set of groups *contributor*.}
Step 2: determine groups of negatively weighted links (inhibitors) U such that (Link weights of group) = 0
{Call this set of groups *inhibitor*.}
Step 3: Create a rule with the following form
If ∀ contributors and ¬∀ inhibitors Then (name of U)
Step 4: remove any duplicate rules

Fig. 7: Symbolic rule extraction algorithm

Intermediate-level concept rules may have one cluster containing more than one production rule, and this will require post-processing to be performed, to obtain the final concept rule hierarchy. The end user requests the post-processing stage. Based on the information provided by the user concerning the kind of rules that should be produced, further clustering with respect to produced clusters is performed to resolve the rule base to the final form. For example, in the case of the 7-segment display case study, 10 digits should be recognised (i.e. digit 0 – 9), and there is some clusters contain more than one rule (e.g. it is possible that digit 8 & 0 clustered into cluster 1), the user will request further clustering to decompose those clusters.

Fig. 8 shows the result of the post-processing session in the 7-segment display problem. The knowledge encoded in the neural network has now been interpreted into symbolic rule knowledge. Given an input pattern of the LED display, the digits can now be recognised using this rule base.

1	(S1)&(S2)&(¬S3)&(¬S4)&(¬S5)&(¬S6)&(¬S7);⇒ {digit 1}	
2	(S1)&(S2)&(S3)&(S4)&(S5)&(S6)&(¬S7);	⇒ {digit 0}
3	(S1)&(S2)&(S3)&(¬S4)&(S5)&(S6)&(S7);	⇒ {digit 9}
4	(S1)&(S2)&(S3)&(S4)&(S5)&(S6)&(S7);	⇒ {digit 8}
5	(S1)&(S2)&(¬S3)&(¬S4)&(¬S5)&(S6)&(¬S7); ⇒ {digit 7}	
6	(S1)&(S2)&(S3)&(¬S4)&(¬S5)&(S6)&(S7);	⇒ {digit 3}
7	(S1)&(S2)&(¬S3)&(¬S4)&(S5)&(¬S6)&(S7); ⇒ {digit 4}	
8	(¬S1)&(S2)&(S3)&(¬S4)&(S5)&(S6)&(S7);⇒ {digit 5}	
9	(¬S1)&(S2)&(S3)&(S4)&(S5)&(S6)&(S7);	⇒ {digit 6}
10	(S1)&(¬S2)&(S3)&(S4)&(¬S5)&(S6)&(S7);	⇒ {digit 2}

Fig. 8: Final concept rule base & recognised digits

3.4 Knowledge Verification and Evaluation

For the sake of testing and evaluating the final produced symbolic rules, an expert system was developed using the C language. We then evaluated the rules using the expert system inference engine. Fig. 9 shows the expert system inference engine user interface window.

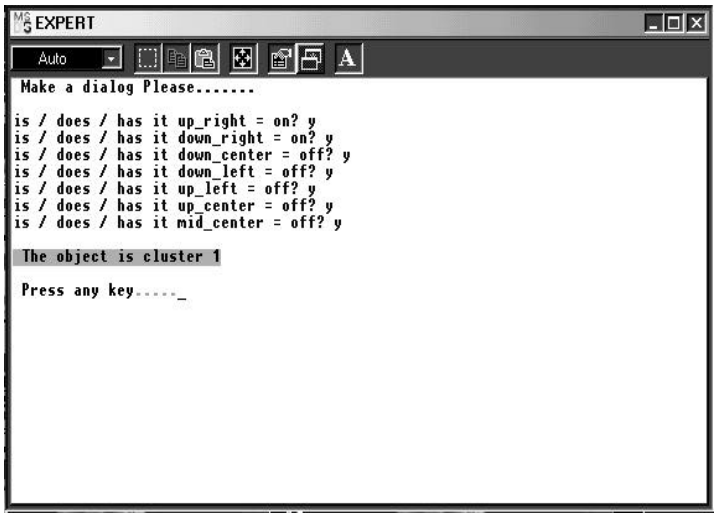


Fig. 9: Knowledge verification user interface

4.0 EXPERIMENTAL RESULTS

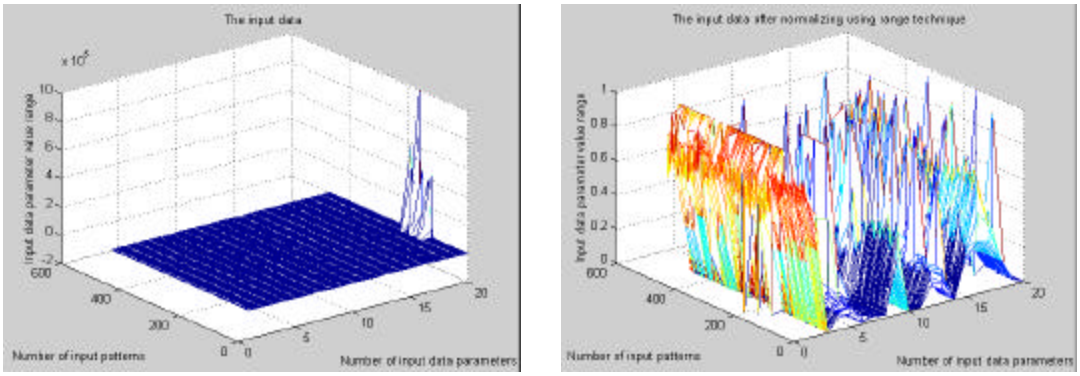
To further illustrate the potential of the proposed automated knowledge acquisition method, we present a case study of a real-world problem in the medical domain. A subset of medical blood test data is collected from 440 patients. For each patient, 20 medical inspection values are obtained. The data is to diagnose the patient into one of 5 different hepatitis diseases. Our system generates, from this data set, the symbolic rule base of the diagnosis rules.

A set of training examples in the form of matrix in which, rows represent patient cases (or patterns) and columns represent the feature data, is formatted. Table 8 shows a portion of input data, with each pattern (row) comprising of 20 entities Q1 to Q20, where Q_i are medical inspection parameters. Fig. 10(a) illustrates the graphical distribution of the input data, indicating the dominance of certain data.

Table 8: A portion of input data (medical case study)

Number of Patterns	Medical Data Input Parameters																			
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
1	7.8	6	0.5	40	42	29	91	4.7	399	15	15	12	3.8	0	19.6	34	270	64.7	1.8	2
2	8	6	0.4	50	61	12	87	4.5	284	15	18	16	3.6	0	21.9	22	214	61.1	2.8	1
3	8.4	5	0.5	40	34	6	90	4.7	257	10	10	10	2	0	23.7	45	280	60.9	3	2
4	7.3	7	0.7	43	61	19	94	4.5	454	23	12	19	5.1	0	20.8	21	198	61.3	2.6	2
5	7.8	7	0.7	43	61	19	94	4.5	454	23	12	19	5.1	0	22.6	23	342	65.5	2.3	1
Note: (Q1 –Q20) represented blood test data concerning five group of patients; namely Cluster 1, Hepatoma, Acute hepatitis, Chronic hepatitis, and Liver cirrhosis patients.																				

The feature vector values are further preprocessed. Intervals of values are quantised to integer values, using the data in Table 9. These intervals are derived empirically. The resulting input data is shown in Table 10, with its graphical representation presented in Fig. 11.



(a) Before normalisation (b) After normalisation

Fig. 10: Graphical representation of input data distribution (Medical case study)

Table 9: Shows the division of medical test data of hepatitis diseases

No	Medical Inspections	Integers of Attribute Values					
		1	**2**	**3**	**4**	**5**	**6**
1	SP	~5.5	5.6~6.5	6.6~7.5	7.6~8.5	8.6~	
2	II	~4	5~6	7~9	10~		
3	Tbil	~1.0	1.1~5.0	5.1~10	10.1~20	20.1~	
4	Dbil	~40	41~60	61~80	81~100		
5	Alp	~80	81~200	201~300	301~400	401~	
6	G.GTP	~30	31~100	101~200	201~300	301~	
7	LDH	~100	101~250	251~500	501~1000	1000~	
8	Alb-G	~2	2.1~3.0	3.1~4	4.1~5	5.1~	
9	ChE	~100	101~150	151~200	201~250	251~500	501~
10	GPT	~25	26~100	101~200	201~500	501~1000	1001~
11	GOT	~20	21~100	101~200	201~500	501~1000	1001~
12	BUN	~9	10~20	21~30	31~40	41~	
13	UrA	~2.7	2.8~8.5	8.6~			
14	Retic	~1.5	1.6~3.0	3.1~6	6.1~		
15	Plt	~1.0	1.1~5.0	5.1~10	10.1~15	15.1~35	35.1~
16	Lympho	~20	20.1~40	40.1~60	60.1~		
17	Fibrino	~200	201~400	401~			
18	Alb%	~45	45.1~65	65.1~			
19	Al%	~2.5	2.6~3.7	3.8~5	5.1~		
20	AFP	~20	21~100	101~200	201~1000	1001~	

Table 10: Input data after pre-processing (medical case study)

Number of Patterns	Medical Data Input Parameters																			
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
1	4	2	1	1	1	1	4	5	1	1	2	2	1	5	2	2	2	1	1	1
2	4	2	1	2	1	1	4	5	1	1	2	2	1	5	2	2	2	2	1	1
3	4	2	1	1	1	1	4	5	1	1	2	1	1	5	3	2	2	2	1	1
4	3	3	1	1	1	1	4	5	1	1	2	2	1	5	2	1	2	2	1	1
5	4	3	1	2	1	1	4	5	1	1	2	2	1	5	2	2	3	1	1	1

Note: (Q1 –Q20) represented blood test data concerning five group of patients; namely Cluster 1, Hepatoma, Acute hepatitis, Chronic hepatitis, and Liver cirrhosis persons.

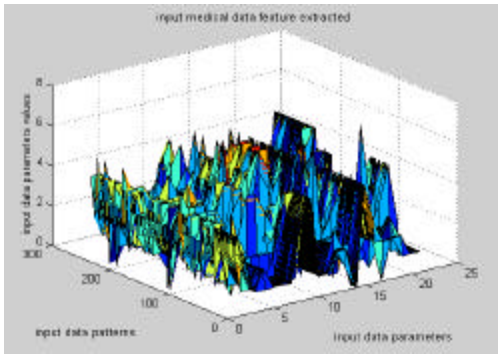


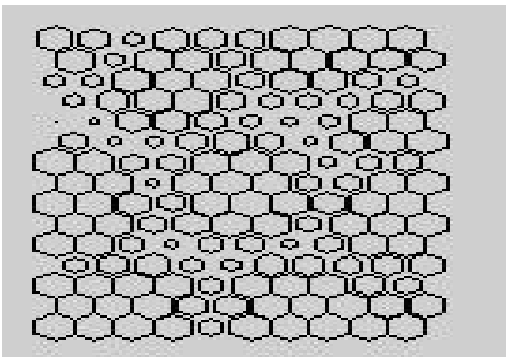
Fig. 11: Input data after pre-processing

Normalisation by the Range technique, chosen by quality measure computation recorded in Table 11, removes this dominance, as shown in Fig. 10(b).

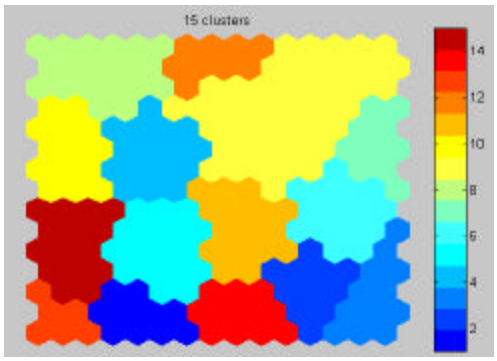
Table 11: Quality measure of different Normalisation types

No.	Input Data Size	Quantisation Error	Topographic Error	Normalisation Type
1	[440 20]	2.365	0.026	Variance
2	[440 20]	0.267	0.028	Range
3	[440 20]	1.739	0.036	Logarithmic
4	[440 20]	0.632	0.026	Histogram Discrete
5	[440 20]	0.735	0.036	Histogram Continuous

In the KSOM learning stage, the following initialisations were performed. The topology was initialised using a hexagonal lattice with 15X15 output layer dimension, random weights initialisation and bubble function for neighborhood initialisation. The result of KSOM training and K-mean clustering is illustrated in Fig. 12(a) and 12(b), respectively. Redundancy is now removed, and Table 12 illustrates the number of rules covered by each cluster before and after the redundancy process. The result is an intermediate-level concept rule base for the medical diagnosis problem. Fig. 13 shows a portion of the rule base, in which the cluster 1 symbolic rules are listed.



(a) KSOM training output



(b) K-means clustering output

Fig. 12: Visualisation of the clusters (Medical case study)

Table 12: Number of rules covered by each class before and after redundancy (Medical case study)

Redundancy	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster11	Cluster 12	Cluster 13	Cluster 14	Cluster 15
Before	20	11	12	16	17	15	10	19	44	14	16	9	6	11	15
After	4	9	6	6	7	8	3	4	7	5	7	2	3	5	2

1 (Not Q1)& (Q2)& (Q3)& (Q4)& (Q5)& (not Q6)& (not Q7)& (not Q8)& (not Q9)& (not Q10)& (Q11)& (not Q12)& (Q13)& (not Q14)& (Q15)& (not Q16)& (Q17)& (not Q18)& (not Q19)& (Q20);⇒ {Cluster 1}

2 (Q1)& (Q2)& (Q3)& (Q4)& (Q5)& (Not Q6)& (Not Q7)& (not Q8)& (Q9)& (Not Q10)& (Q11)& (Not Q12)& (Q13)& (Not Q14)& (Q15)& (Not Q16)& (Q17)& (Not Q18)& (Not Q19)& (Q20); ⇒ {Cluster 1}

3 (Not Q1)& (Q2)& (Q3)& (Q4)& (Q5)& (Not Q6)& (Not Q7)& (Not Q8)& (Not Q9)& (Not Q10)& (Not Q11)& (Not Q12)& (Q13)& (Not Q14)& (Q15)& (Not Q16)& (Q17)& (Not Q18)& (Not Q19)& (Q20); ⇒{Cluster 1}

4 (Not Q1)& (Q2)& (Q3)& (Q4)& (Q5)& (Not Q6)& (Not Q7)& (Not Q8)& (Not Q9)& (Not Q10)& (Not Q11)& (Not Q12)& (Q13)& (Not Q14)& (Q15)& (Not Q16)& (Not Q17)& (Not Q18)& (Not Q19)& (Q20); ⇒ {Cluster 1}

Fig. 13: Cluster 1 symbolic rules

With the data set given, our knowledge acquisition system produced a rule base of 65 rules contained in 15 clusters. (Compare this with an equivalent statistical approach which will generate 2^{20} (1048576) possible rules.) The rule base is now submitted into the knowledge base of an expert system. This knowledge should contain all the required information to carry out the reasoning process. Via consultation with a physician (i.e. the expert) who validates and verifies the rule base, the inference engine discriminates the rules found in the clusters such that the final-concept rules are obtained. For example, let us take pattern number 4 from table 8 and feed it to the inference engine (the chosen pattern has been recognised by the physician as hepatoma symptom). Through query and answer session with the rule base the physician submits his diagnosis to be hepatoma. Thus, the system managed to link each cluster to certain group of diagnosis. Fig. 14 illustrates the runtime trace of the resulting reasoning process in our system. Then, from cluster level recognition, the process is repeated until all the 65 rules are linked to specific diseases. The final-concept rule base now contains the knowledge to perform the medical diagnosis, that is we have the various clustered symptoms linked to a disease. (Note that each cluster is linked to a certain type of disease.)

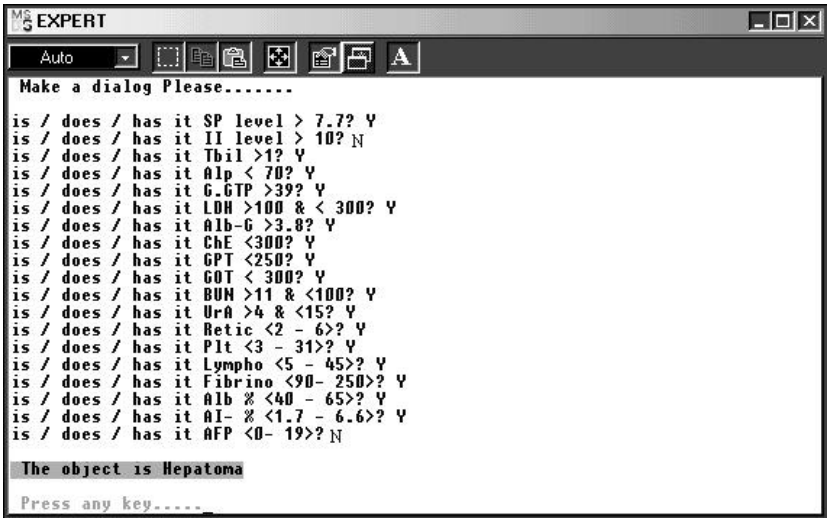


Fig. 14: Inference engine for testing pattern

5.0 CONCLUSION

The 7-segment display problem and the medical case study show that the proposed automated knowledge acquisition method can successfully extract knowledge in the form of production rules from numerical data set representing the salient features of the problem domain. This study has demonstrated that symbolic knowledge extraction can be performed using unsupervised learning KSOM neural networks, where no target output vectors are available during training. The system is able to learn from examples via the neural network section. The extracted knowledge can form the knowledge base of an expert system, from which explanations may be provided, and it is quite possible to diagnose new unknown disease. Large, noisy and incomplete data set can be handled. The system proves the case of the viability of integrating neural network and expert system to solve real-world problems.

AKNOWLEDGEMENT

The research is supported in part by an IRPA grant (grant number 72112).

REFERENCES

[1] M. Ishikawa, "Neural Networks Approach to Rule Extraction". *IEEE Expert*, 1995, pp. 6-9.

[2] M. S. Kurzyn, "Expert Systems and Neural Networks: A Comparison". *IEEE Expert*, 1993, pp. 222-223.

[3] N. DeClaris, M. Su, "A Neural Network Based Approach to Knowledge Acquisition and Expert Systems". *IEEE Expert*, 1993, pp. 645-650.

[4] Carl G. Looney, *Pattern Recognition Using Neural Network*. Oxford University Press, New York. USA 1997.

[5] A. Ultsch, D. Korus, Kleine, and T. O., "Integration of Neural Networks and Knowledge-Based Systems in Medicine", in *5th. Conf. Artificial Intelligence in Medicine Europe AIME'95*, Pavia, Italy, June 1995, in Barahona, P., Stefanelli, M., Wyatt, J.: Artificial Intelligence in Medicine, *Lectures in Artificial Intelligence 934*, Springer 1995, pp. 425-426.

[6] A. Ultsch, S. Farsch, H. Li, "Automatic Acquisition of Medical Knowledge from Data Sets with Neural Networks", in *Proc. KI'95, Bielefeld/Germany, Advances in Artificial Intelligence*, Springer 1995, pp. 258-260.

[7] A. Ultsch, "Knowledge Extraction from Self-organizing Neural Networks", in *O. Opitz, B. Lausen and R. Klar (Eds.): Information and Classification*, Berlin, Springer Verlag, pp. 301-306.

[8] A. Ultsch, "The Integration of Connectionist Models with Knowledge-based Systems: Hybrid Systems", in *Proceedings of the IEEE SMC 98 International Conference*, San Diego, 11. - 14. October 1998, pp. 1530-1535.

[9] Teuvo Kohonen, *Self-Organizing Maps*. Springer, *1995, USA*.

[10] Juha Vesanto, "Data Mining Techniques Based on the Self Organizing Maps". *Helsinki University of Technology. Thesis M. Sc.*, 1997.

[11] Sabrina Sestito, *Automated Knowledge Acquisition*. Prentice Hall, 1994, Australia.

BIOGRAPHY

Mohamed Khalil Hani received his B.Eng (Electrical) in Communications from the University of Tasmania, M. Eng (Electrical) in Computer Architecture from Florida Atlantic University, and Ph.D. in Digital Systems and Computer Engineering from Washington State University. He is currently the Vice-Dean at the Faculty of Electrical Engineering, Universiti Teknologi Malaysia. His research interests include digital system design, computer architecture, VLSI, artifical intelligence (neural network, fuzzy logic, and expert system), hardware design of cryptography systems, and rapid-prototyping CAD with FPGA.

Suliaman Mohd Nor received his B.Eng. in Electrical engineering from University of Sheffield, M.Sc. in computing systems from Cranfield Institute of technology, and Ph.D. in from Universiti Teknologi Malaysia. His expertise and research interests include computer systems, computer networks and protocols, microprocessor systems, and digital systems.

Sheikh Hussain Shaikh Salleh received his B.Sc. from University of Bridgeport, M.EE. from Universiti Teknologi Malaysia, and Ph.D. from University of Edinburgh. His research interests include neural network, microprocessor, speech processing, and digital signal processing.

Nazar Elfadil received his first degree in science and technology from University of Gizera (Sudan), M.Sc. in computer science from Universiti Teknologi Malaysia. Currently, he is Ph.D candidate in the Faculty of Electrical Engineering, Universiti Teknologi Malaysia. His research interest includes artificial intelligence (neural network, fuzzy logic, and expert system), software engineering, automated knowledge acquisition, machine learning, and computer engineering. He is an affiliate member of IEEE computer soceity since 1998.