A Soft Computing Model to Counter Terrorism

Karthikeyan Marappan[†], Krishnan Nallaperumal[†], K. Senthamarai Kannan[††] and B.Bensujin[†]

[†]Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Tirunelveli, India
[††]Department of Statistics, Manonmaniam Sundaranar University, Tirunelveli, India

Summary
In the aftermath of September 11, the experts concluded that data mining could help it prevent future terrorist attacks. Experts are also concerned that in its zeal to apply technology to antiterrorism, the government could disrupt the crime-fighting processes of the agencies that are charged with finding and stopping terrorists before they act. The entire information or the evidence about a terrorist and the inclined behavior of some personalities are stored in interactive XML sheets (iXML), which are called as trifles, the piece of information. These trifles play a vital role in training the soft computing model and for pattern detection. These trifles in the form of iXML sheets are given in the network for pattern detection. The soft computing model used here is the Competitive Neural Tree (CNeT). The CNeT is the type of decision tree in which each node is compared and a decision will be taken to move to the next. In each stage the pattern recognition is done with the contents of the iXML nodes.

Key words:
CNet, decision tree, iXML, soft computing, and trifle.

1. Introduction

key aspiration of this paper is to develop techniques for organizing intelligence information to sculpt terrorism threats using data mining techniques [10]. Our goal can be stated quite clearly as follows: How well we organize existing evidence influences? How well we humans are able to engender new hypotheses as well as new evidential tests of all hypotheses we are considering? The process of organizing evidence is a decisive step in the process of discovery or investigation. A data-centered approach to organizing evidence [9], [11] is adopted, which allow us to create, justify, or negate hypotheses. This is accomplished through the creation of intelligent agents that act as conceptual magnets that attract trifles (or atomic pieces) of evidence. This attraction is triggered in one of three ways: 1) the evidence justifies an existing hypothesis, 2) the evidence negates an existing hypothesis, or 3) the evidence suggests that a new hypothesis be formed, which in turn becomes a new conceptual magnet. We propose a novel use of data mining, information retrieval and software agent technologies to enable this innovation.

Manuscript received May 5, 2008
Manuscript revised May 20, 2008

In this paper, we describe an architecture that facilitates the organizing of the enormous volume of evidence that an intelligence analyst has available. An essential point in our design is that of automating the process of hypothesis generation with human interaction. Humans, in contrast, have an amazing capacity of adaptation to new situations, and are capable of thinking of scenarios that have not occurred before. We are however, capable of organizing such a huge amount of evidence to corroborate or negate our hypothesis, and that is where our system strength resides through artificial neural networks. The network we used here is the Competitive neural tree (CNeT)[1] [2] [3][20]. The CNeT is a Tree network, in that there are well defined search techniques for spanning the tree is available. Neural network based pattern detection in an N-dimensional data space which consists of generating decision boundaries that can successfully distinguish the various classes in the feature space. The classification rules have been extracted from all these models in the form of lf-Then rules. Finally the extracted rules have been validated for their correctness.

The central piece of our design will be the support for queries, both ad-hoc and long standing, that will act as "magnets" attracting the relevant evidence that a human needs to estimate the validity of a hypothesis. The evidences are represented as an interactive XML file (trifle) and then fed to the soft computing model. The soft computing model is designed in such a way that, it accepts the given hypothesis, or negates the hypothesis, or it creates a new hypothesis. The proposed soft computing model, the Competitive neural tree is designed in such a way that it accepts an iXML file for training the network. The input to the network is an iXML file, it is read by individual nodes and the patterns are matched.

Neural tree architectures were recently introduced for pattern classification [1], [2], [4], [20] in an attempt to combine advantages of neural networks and decision trees. By applying the decision tree methodology, one difficult decision is split into a sequence of less difficult decisions.

The first decision determines which decision has to be made next and so on. From all the questions that are arranged in a tree like structure only some are asked in the process of determining the final answer. Neural tree architectures are decision trees with a neural network in each node. These networks perform either feature extraction from the input or make a decision. A decision must be made at internal nodes regarding the child-node to be visited next. Terminal nodes give the final answer.

DECISION trees have extensively been used to perform decision making *in pattern recognition* [6] [7] [8]. By applying the decision tree methodology, one difficult decision can be split into a sequence of less difficult decisions. The first decision determines which decision has to be made next by indicating which node of the tree should be visited. Because of the tree structure, only some of all possible questions are asked in the process of making the final decision. In fact, the final decision is made at a terminal node of the tree, which is reached by traversing the tree starting from the root as indicated by the decisions made at internal nodes.

The design of decision trees is frequently performed in a *top-down* fashion [12]. The nodes are split during the design process according to some criterion. The existing splitting criteria include the impurity measure used in *classification and regression trees* [13], [17] and the mutual information measure employed by the *average mutual information gain* algorithm [16]. The terminal nodes are determined during the construction of the tree by freezing some of the nodes according to some stopping criterion or by growing a large tree and performing selective backward pruning. After the final tree structure is determined, the terminal nodes are frequently assigned class labels by using a majority rule.

2. Trifle Illustration

Before going into the details of our technical approach, we need to define a trifle as it forms the fundamental piece of evidence that will be used in hypothesis processing. Examples of trifles, or pieces of evidence, with which the intelligence community is currently inundated are: 1) an object identified in an image, 2) information in an intelligence report, 3) information in an open-source document (e.g., online newspaper article), or 4) a video clip (e.g., from Door-Dharshan, Al- Jazeera or CNN). This information could be stored in a structured database [14] or found from some unstructured location (e.g., web, Intelink). This information must be organized to support an existing query (e.g., Is chemical plant X producing weapons of mass destruction?) or to direct the analyst to perhaps consider a new hypothesis.

Outcome of this research work leads to a new way for intelligence analysts to interact with the intelligence data stream and can be described by the following. Intelligence analysts are currently overwhelmed with the amount of data that they must analyze. Most of this data is never analyzed at all, potentially leaving key pieces of information out of the analysis process. Our ability to collect data will only grow, further accentuating the problem. Our approach allows the analyst to interact with the intelligence stream in a novel way. This interaction can be viewed as a triangle with the Human, Hypotheses, and Trifles at each vertex. The interaction is not isolated at

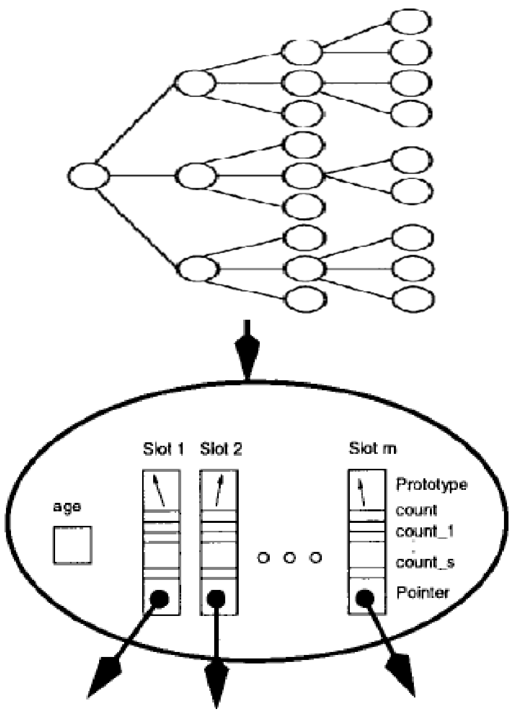

Fig. 1. (a) Tree structure and (b) Explanation of each node

each vertex of the triangle, however, and the interaction with all three is the key to the analyst making informed analysis that supports the intelligence decision-making process. The human also interacts with the trifles, not arbitrarily, but as a result of the hypothesis. From this processing, trifles are accumulating, forming a critical mass that may require the attention of the analyst. Perhaps a new hypothesis should be considered as a result of this.

3. Architecture

Competitive Neural Tree has a structured architecture. A hierarchy of identical nodes forms an m-ary tree as shown in Fig. 1(a). Fig. 1(b) shows a node in detail. Each node contains m slots $s_1, s_2, \ldots, s_m$ and a counter age that is incremented each time an example is presented to that node. The behavior of the node changes as the counter age increases. Each slot s_i stores a prototype pi, a counter count, and a pointer to a node. The prototypes $p_i \in P$ have the same length as the input vectors x. They are trained to match the patterns obtained from each node.

The slot counter count is incremented each time the prototype of that slot is updated to match an example. Finally, the pointer contained in each slot may point to a child-node assigned to that slot.

A NULL pointer indicates that no node was created as a child so far. In this case, the slot is called terminal slot or leaf. Internal slots are slots with an assigned child-node.

3.1 Learning at the Node-Level

In the learning phase [5] [18] [19], the tree grows starting from a single node, the root. The prototypes of each node form a minuscule competitive network. All prototypes in a node compete to attract the examples arriving at this node. These networks are trained by competitive learning. When an example $x \in \chi$ arrives at a node, all of its prototypes $p_1, p_2, \ldots, p_m$ compete to match it. The closest prototype to x is the winner. If $d(x, p_j)$ denotes the distance between x and p_j, the prototype p_k is the winner if $d(x, p_k) < d(x, p_j)$ for all values of $j \neq k$.

The distance measure used in this paper is the squared Euclidean norm, defined as

$$d(x, P_j) = \|x - p_j\|^2 \qquad (1)$$

The competitive learning scheme used at the node level resembles an unsupervised learning algorithm proposed to generate crisp c- partitions of a set of unlabeled data vectors [4], [5]. According to this scheme, the winner p_k is the only prototype that is attracted by the input x arriving at the node. More specifically, the winner p_k is updated according to the equation

$$P_k^{new} = P_K^{old} + \alpha(x - P_k^{old}) \qquad (2)$$

where α is the learning rate. The learning rate α decreases exponentially with the age of a node according to the equation

$$\alpha = \alpha_0 \exp(-\alpha_d age) \qquad (3)$$

where α_0 is the initial value of the learning rate and α_d determines how fast α decreases. The update equation (2) will move the winner p_k closer to the example x and therefore decrease the distance between the two. After a sequence of example presentations and updates, the prototypes will respond each to examples from a particular region of the input space. Each prototype p_j attracts a cluster of examples R_j.

The prototypes split the region of the input space that the node sees into sub regions. The examples that are located in a sub region constitute the input for a node on the next level of the tree that may be created after the node is mature. A new node will be created only if a splitting criterion is TRUE.

3.2 Life Cycle of Nodes

Each node goes through a life cycle. The node is created and ages with the exposure to examples. When a node is mature, new nodes can be assigned as children to it. A child-node is created by copying properties of the slot that is split to the slots of the new node. More specifically, the child will inherit the prototype of the parent slot. Right after the creation of a node, all its slots are identical. As soon as a child is assigned to a node, that node is frozen. Its prototypes are no longer updated in order to keep the partition of the input space for the child-nodes constant. A node may be destroyed after all of its children have been destroyed.

4. Training Procedure

The generic training procedure is described below:

Do while stopping criterion is FALSE:

- Select an I XML file.
- Traverse the tree starting from the root to find a terminal prototype p_k that is close to x. Let n_l and s_k be the node and the slot that p_k belongs to, respectively.
- If the node n_l is not frozen, then update the prototype p_k according to equation (2).
- If a splitting criterion for slot s_k is TRUE, then assign a new node as child to s_k and freeze node n_l.
- Increment the counter count in slot s_k and the counter age in node n_l.

Depending on how the search in the second step is implemented, various learning algorithms can be developed. The search method is the only operation in the learning algorithm that depends on the size of the tree. Hence, the computational complexity of the search method determines the speed of the learning process. Among the various search methods available, the full search method achieves the best performance and it s detailed in the following section.

Sample pseudo code used for training the network

```
/*Assigning inputs to each neuron*/
    • Set the initial value i=0
    • For all neuron In Input
    • Neuron.Output = Inputs(i)
    • i = i + 1
    • End
/*Calculating the weight of each neuron*/
    • For all input neuron connected to This Neuron
    • netValue = netValue + (Weight Associated With
      InputNeuron * Output of InputNeuron)
    • End
/*Calculating the error value */
    • Delta = Neuron.Output * (1 - Neuron.Output) *
      ErrorFactor
/*Calculating the output */
    • For each layer in Input layers
    • neuron.Update(Input* Weight)
    • End
/*Calculating the Bias Value */
    • Set netValue As Single = bias
    • For all input neuron connected to ThisNeuron
    • netValue = netValue + (Weight Associated With
      InputNeuron * Output of InputNeuron)
    • End
```

4.1 Full Search Method

The search method determines the speed of learning and recall as well as the generalization ability of the trained CNeT. A feature vector x constitutes the input for the search. An exhaustive search of the tree is guaranteed to return the closest prototype v_k to the input vector x. Because of the computational and time requirements associated with an exhaustive search, alternative search methods can be employed for determining a terminal prototype v_k that is close, but not necessarily the closest, to the input x. During learning, any terminal prototype v_j Є V is a candidate to be selected by the search method. In contrast, only the prototypes that responded during learning to at least one example are candidates to be selected in the recall phase. Fig. 2 shows which nodes are visited and expanded in a complete binary tree by the search methods described in this section.

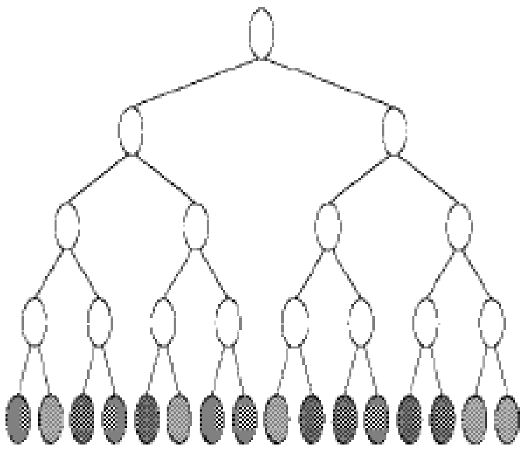
Fig. 2. Shaded nodes visited by the full search method.

The full search method explained in fig. 2 is based on conservative exhaustive search. To guarantee that the prototype p_k with the minimum distance to a given feature vector x is returned, it is necessary to compute the distances d(x, p_j) between the input vector x and each of the terminal prototypes p_j Є P. The prototype p_k with the minimum distance is returned. The time requirement for the full search method is of the order O(n). However, the full search method guarantees the return of the closest prototype to the input vector.

4.2 Pattern Detection

All the networks are trained and stored in the form of XML files and can be used later. During the detection process, the Interactive XML sheets are uploaded to the network. As per the learning done on the neural tree, the tree recognize each node and perform a full search on the tree. When the searching is done on the neural tree, the tree does not grows, it still remain constant and can be compared with the nodes.

The algorithm for the pattern detection is given below,

```
Do While Terminal Node is True
    • Read the root node of the iXML file.
    • Compare the node with the existing trained
      network which was already trained with effective
      data supplied.
    • If the root node presents the availability, then
      move to the child node to dig the information and
      store in the nodes of the new network.
    • If the delta value provides an output which is
      greater than higher threshold, justify the
      hypothesis.
    • Else if the delta value provides an output which is
      less than lower threshold, negate the hypothesis.
    • If the hypothesis is neither justified nor negated
      the new hypothesis is generated.
    • The tree will be again trained with the new
      hypothesis.
```

The above algorithm receives an input in the form of an iXML file and feed it to the network. The network matches the pattern with previously trained elements. If the input matches with the pattern then a justification on the hypothesis is made or the hypothesis is negated else a new hypothesis will be generated.

5. Generalization of Network

The CNeT is generalized to classify the vectors of same features. The generalization ability is measured by the *accuracy* with which it makes these classifications. The network accepts complete set of nonlinear input. One of the major advantages of CNeT is their ability to generalize. This means that a trained net could classify data from the same class as the learning data that it has never seen before. To reach the best generalization, the dataset should be split into three parts:

- The ***training set*** is used to train a CNeT. The error of this dataset is minimized during training.
- The ***validation set*** is used to determine the performance of a neural network on patterns that are not trained during learning.
- A ***test set*** for finally checking the over all performance of a neural net.

Fig. 3 shows a typical error development of a training set (lower curve) and a validation set (upper curve).

The learning should be stopped at the minimum of the validation set error. At this point the net generalizes best. When learning is not stopped, overtraining occurs and the performance of the net on the whole data decreases, despite the fact that the error on the training data still gets smaller. After completing the learning phase, the net should be finally checked with the third data set, the test set.

The generalization is not achieved, sometimes, when the training set error is low. Two different estimate have been conducted for calculating the error rate from the input which are not in the training set but drawn from the same underlying distribution as the training set. The error rate we call it as out-of-sample-set error rate. Perhaps the simplest technique we adopted is to divide the input vectors available for training into two disjoint sets, considering one as the training set and the other as the validation set**.** Based on the two set of data's the error rate is calculated. If the number of vectors in both sets is large, the error rate on the validation set is a reasonable estimate of generalization accuracy.

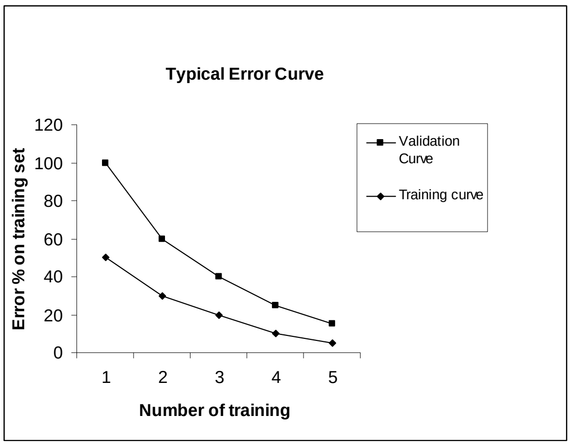

Fig. 3. Error curve.

The second method adopted to estimate the error rate, generalization accuracy, is the cross validation**.** The available training vectors are divided in to k disjoint subsets, called folds. A single fold is selected as validation set and the remaining k-1 set are selected as training set. The procedure has been repeated for k times, each time selecting a different fold as a validation set and its complement as the training set. The error rate is computed for each validation set and the average of these error rates is taken and estimated as the out-of-sample error.

6. Training Patterns

A sample iXML file / trifle is illustrated below. Thousands of such trifles are used for training the network. The training set and validation set are classified from the thousand number of trifles used for training.

6.1. Sample iXML File / Trifle.

```
<?xml version="1.0" ?>
 <doc>
   <assembly>
    <name>ixml</name>
    <version>1.0.2346.43108</version>
    <fullname>nxml,             Version=1.0.2346.43108,
   Culture=neutral, PublicKeyToken=null</fullname>
   </assembly>
/* Terrorist Details*/
<Terrorist>
            <Name> Name of the terrorist </Name>
            <Age> Age of the terrorist </Age>
```

```
        <Gender>Sex of the terrorist </Gender>
        <Place of Birth> Birth Place</Place of Birth>
    <Previous Activities>Type of attack done by the
    terrorist </Previous Activities>
    <Location>Place where the terrorist attack has been
    happened </Location>
    <Number of participants> Total number of people
    involved in the activity</Number of participants>
</Terrorist>
</doc>
<?xml version="1.0" ?>
 <doc>
   <assembly>
    <name>ixml</name>
    <version>1.0.2346.43108</version>
    <fullname>nxml,            Version=1.0.2346.43108,
   Culture=neutral, PublicKeyToken=null</fullname>
   </assembly>
/* Details of evidence*/
<Terrorist Attack>
    <Location>Place in which the terrorist attack took
    place </Location>
    <Type of Attack> The type of attack conducted by the
    terrorist </Type of Attack>
    <Members in the group> The members participated in
    the activity</Members>
    <Loss>Number of people died in the attack</Loss>
</Terrorist Attack>
```

Trifles are generated automatically based on the data available in various formats, which includes heterogeneous databases and information provided by intelligence agencies.

6.2. Hypothesis

The system is designed to accept any type of hypothesis given to it. The CNeT try to match the hypothesis and attracts all the trifles associated to the hypothesis. Based on the volume of the trifles attracted by the submitted hypothesis, the system either justify the hypothesis or negate the hypothesis, else suggests the way for generation of new hypothesis.

7. Conclusion and Summary

This paper introduces an innovative methodology to organize the evidences of terrorism and discover knowledge with the help of data mining technique. The technique presented here works on interactive XML files which overcomes the problem of mining the heterogeneous data. The soft computing model presented here is also capable of providing a solid inculcation with the training of the network and the detection of patterns.

References

[1] Sven Behnke and Nicolaos B. Karayiannis, “CNeT: Competitive neural trees for pattern classification,” in Proc. IEEE Int. Conf. Neural Networks, Washington, D.C., June 3–6, 1996, pp. 1439–1444.
[2] Sivanandan,Shanmugam,sumathi,”Development of Soft Computing Models For Data Miming”, IE(I) journal Vol 86, May 2005 - CP PP 22 -31.
[3] Sven Behnke and Nicolaos B. Karayiannis, “ Competitive neural trees for pattern classification”.
[4] L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, J. Connor, D. Park, M.El-Sharkawi, and R. J. Marks II, “A performance comparison of trained multilayer perceptrons and trained classification trees,” Proc. IEEE, vol. 78, no. 10, pp. 1614–1619, 1990.
[5] L. Fang, A. Jennings, W. X. Wen, K.Q.-Q. Li, T. Li \Unsupervised learning for neural trees," Proceedings International Joint Conference on Neural Networks, vol 3, pp. 2709{2715, 1991.
[6] S. Rasoul Safavian and David Landgrebe, “A Survey of Decision Tree Classifier Methodology”
[7] Safavian, S.R. Landgrebe, D., “A survey of decision tree classifier methodology”Proceedings in the IEEE, Vol 21, issue 3.
[8] Xiangyang Li, Nong Ye, “Decision tree classifiers for computer intrusion detection”,Published in Nova Science Publishers, Inc. Commack, NY, USA.
[9] Cousins, D.B.; Weishar, D.J.; Sharkey, J.B. , “Intelligence collection for counter terrorism in massive information content” Aerospace Conference, 2004. Proceedings. 2004 IEEE Volume 5, Issue , 6-13 March 2004 Page(s): 3273 – 3282.
[10] http://www.cio.in/govern
[11] www.homelandsecurity.org
[12] R. Nowak, “Decision Trees” IEEE Trans. Systems, Man, & Cybernetics, May 1991.
[13] L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees. Belmont, CA: Wadsworth, 1984.
[14] http://www.tkb.org
[15] Michael J. Andrews, “ An Information Theoretic Hierarchical Classifier for Machine Vision” WORCESTER POLYTECHNIC INSTITUTE, May 1999.
[16] I. K. Sethi and G. P. R. Sarvarayudu, “Hierarchical classifier design using mutual information,” IEEE Trans. Pattern Anal. Machine Intell.vol. 4, pp. 441–445, 1982.
[17] P. A. Chou, “Optimal partitioning for classification and regression trees,” IEEE Trans. Pattern Anal. Machine Intell., vol. 13, pp. 340–354, 1991.
[18] S. Behnke and N. B. Karayiannis, “Competitive neural trees for vector quantization,” Neural Network World, vol. 6, no. 3, pp. 263–277, 1996.
[19] “CNeT: Competitive neural trees for pattern classification,” in Proc. IEEE Int. Conf. Neural Networks, Washington, D.C., June 3–6,1996, pp. 1439–1444.
[20] Sven Behnke and Nicolaos B. Karayiannis, “Competetive neural trees for vector quantization,in Neural Network World, 6(3): 263-277,1996.
[21] M. Karthikeyan, Krishnan Nallaperumal, K.Senthamaraikannan, K.Velu and B.Bensujin, “A Soft Computing Model for Knowledge Mining and Trifle Management”, International Journal of Imaging Science and Engineering (IJISE), vol.1, No.4, Dec 2007, GA, USA. ISSN:1934-9955, pp. 132-138.

Nallaperumal Krishnan received M.Sc., degree in Mathematics from Madurai Kamaraj University, Madurai, India in 1985, M.Tech degree in Computer and Information Sciences from Cochin University of Science and Technology, Kochi, India in 1988 and Ph.D., degree in Computer Science & Engineering from Manonmaniam Sundaranar University, Tirunelveli. Currently, he is heading Centre for Information Technology and Engineering of Manonmaniam Sundaranar University, Tirunelveli. His research interests include Signal and Image Processing, Remote Sensing, Visual Perception, Mathematical Morphology Fuzzy Logic, Data mining and Pattern recognition. He has authored three books, edited 18 volumes and published 25 scientific papers in Journals. He is a Senior Member of the IEEE..

K. Senthamarai Kannan is currently working as a Professor in Statistics, at the Manonmaniam Sundaranar University, Tirunelveli, Tamilnadu. He has more than 18 years of teaching experience at post-graduate level. He has published more than 30 research papers in international and national journals and authored four books. He has visited Turkey, Singapore and Malaysia. He has been awarded TNSCST Young Scientist Fellowship and SERC Visiting Fellowship. His area of specialization is 'Stochastic Processes and their Applications'. His other research interests include stochastic modeling in the analysis of birth intervals in human fertility, bio-informatics, data mining and precipitation analysis.

Karthikeyan Marappan received B.E degree in Electronics and Communication Engineering from Bharathiar University, Coimbatore, India in 1990, M.Tech., degree in Computer and Information Technology from Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Tirunelveli, India. Currently, he is doing Ph. D., in Computer and Information Technology at Faculty of Engineering, Centre for Information Technology and Engineering, Manonmaniam Sundaranar University Tirunelveli. His research interests include Data mining and Image Processing. He is a Member of the IEEE.