*World of Computer Science and Information Technology Journal (WCSIT)*
*ISSN: 2221-0741*
*Vol. 1, No. 5, 227-233, 2011*

# Hierarchical and Bayesian Scattered Data Taxonomy in Mobile Ad-hoc Networks

[1]Sunar Arif Hussain, [2]Dr K.E.Sreenivasa Murthy
[1]Prof. Dr. KVSRCEW., Dept .of ECE, Kurnool, Kurnool(Dt.), A.P.
[2]Principal of Sri Venkateswara Institute of Technology, Anantapur, Anantapur ( Dt.), A.P.

Abstract— MANETS promise an unprecedented opportunity to monitor physical environments via inexpensive wireless embedded devices. Given the sheer amount of sensed data, efficient taxonomy of them becomes a critical task in many sensor network applications. The Bayesian classifier is a fundamental taxonomy technique. We introduce two classifiers: Naive Bayes and a classifier based on class decomposition using K-means clustering. We consider two complementary tasks: model computation and scoring a data set. We study several layouts for tables and several indexing alternatives.

In this paper, we propose a novel decision-tree-based hierarchical scattered taxonomy approach, in which local classifiers are built by individual sensors and merged along the routing path forming a spanning tree. The classifiers are iteratively enhanced by combining strategically generated pseudo data and new local data, eventually converging to a global classifier for the whole network. We also introduce some control factors to facilitate the effectiveness of our approach. Through extensive analysis, we study the impact of the introduced control factors, and demonstrate that our approach maintains high taxonomy accuracy with very low storage and communication overhead. The approach also addresses a critical issue of heterogeneous data distribution among the sensors.

Keywords- Scattered Data Storage; Sensor Networks; Key Sharing; Key Management.

## I. Introduction

The recent advances in transceiver and embedded hardware designs have made massive production of inexpensive wireless sensors possible. A wireless sensor network (WSN) consists of a number of sensor nodes (few tens to thousands) storing, processing and relaying the sensed data, often to a base station for further computation. MANETS can be used in many applications, such as

wildlife monitoring, military target tracking and surveillance, hazardous environment natural disaster relief. Given the huge amount of sensed data, classifying them becomes a critical task in many of these applications. As an example, for wildlife monitoring, the sensor nodes continuously sense the physical phenomena such as temperature, humidity and sunlight, and meanwhile may also count the number of animals. The number reflects the suitability of the current environment for the animals, for example, if it is greater than a threshold, the environment is classified as suitable, and otherwise not. After learning the relation between the physical phenomena and the classes from such training data, we may later determine the suitability of the inquired environment from the external source. Taxonomy is typically done in two steps: first, a classifier is constructed to summarize a set of predetermined classes, by learning from a set of training data; then, the classifier is used to determine the classes of newly arrived data. Within this framework, there have been significant efforts in improving its speed and accuracy, most of which assume centralized storage and computation. The MANETS however pose a series of new challenges, particularly for the first step. First, the number of sensor nodes is huge, but each of them has only limited storage that can hardly accommodate all the training data of the whole network. Second, the sensor nodes are generally powered by non-rechargeable batteries, and energy efficiency is thus of paramount importance, which makes the straightforward solution of sending all the training data to the powerful base station quite inefficient. In short, conventional centralized solution is not directly applicable in this new type of network environment. To address the above challenges, in this paper, we present a novel decision-tree-based hierarchical scattered taxonomy in energy-constrained sensor networks.

Our approach organizes the sensor nodes and performs taxonomy in a localized, iterative and bottom-up manner, utilizing a decision tree method, in particular, an enhanced C4.5 algorithm. Starting from each leaf node, a classifier is built based on its local training data. An upstream node, upon receiving the classifiers from its children, will use them with its own data to build a new classifier. These local classifiers will be iteratively enhanced from bottom to top and finally reach the base station, making a global classifier for all the data scattered across the sensor nodes.

We propose a spanning tree constructing approach, in which each node selects a portion of the candidate children as the children to control the shape of the spanning tree. Another key difficulty lies in training a new classifier from a mix of downstream classifiers and the local training dataset, which cannot be directly accomplished by the existing learning algorithms that work on dataset only. We address this problem by generating a pseudo training dataset from each downstream classifier. We develop a smart generation algorithm, which ensures that the pseudo data closely reflect the characteristics of the original local data. We also introduce a control parameter that adaptively balances the recovery quality and the amount of the data.

We also notice that, in practice, the data distribution across different sensor nodes is not necessarily homogeneous. We show strong evidence that such heterogeneity can easily lead to mistaxonomy, and we propose an enhanced C4.5 algorithm to mitigate its impact. To our knowledge, it is the first solution addressing this distribution issue.

## II. Taxonomy basics

### 2,1. Taxonomy

Taxonomy, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications. Data taxonomy is a two-step process. In the first step, a model is built describing a predetermined set of data classes or concepts, and the model is constructed by analyzing database samples described by attributes. Each sample is assumed to belong to a predefined class, as determined by one of the attributes, called the class label attribute.

The samples analyzed to build the model collectively form the training data set. Since the class label of each sample is provided, this step is also known as supervised learning. Typically, the learned model is represented in the form of decision trees, taxonomy rules, or mathematical formula. In the second step, the model is used for taxonomy. First, the predictive accuracy of the model (or classifier) is estimated. A simple way is to use a test set of class-labeled samples, which are randomly selected and are independent of the training samples. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. If the accuracy of the model is considered acceptable, the model can be used to classify future data of which the class label is not known.

### 2.2. Decision tree

Decision tree is one of the most important models for taxonomy, and also serves as the foundation for our taxonomy method. A decision tree is a mapping from observations about an item to conclusions about its target value. A decision tree has three types of nodes: (1) a root node that has no incoming edges, (2) internal nodes, each of which has one incoming edge and more than one outgoing edges, (3) leaf nodes, each of which has one incoming edge and no outgoing edge. In a decision tree, each leaf node is assigned a class label. The root and internal nodes, contain attribute test conditions to separate data that have different characteristics. Classifying a test data is straightforward once a decision tree has been constructed. Starting from the root node, the test condition is applied to the data and follows the appropriate branch based on the outcome of the test. Fig. 1 shows a decision tree example. For instance, it indicates that if the temperature is between 15 and 30 and the sunlight is weak, the environment is classified as suitable (Yes) for animals.

The challenge lies in constructing the decision tree is how to find the best attribute to split the sample data. The following describes two common criteria.

[Information gain:] The information gain is the simplest criterion. It uses the entropy measure, which can be calculated as Entropy where S is the dataset, c is the number of classes and pi is the proportion of each class. The information gain is then calculated as where V(A) is the set of all possible values for attribute A, and Sv is the subset of S for which attribute A has value v.

[Gain ratio:] There exists natural bias in information gain, as it favors attributes with many values. For example in weather forecast, the attribute "data" may have the highest information gain, but it will lead to a very broad decision tree of depth one and is inapplicable to any future data. There are some advanced criteria, such as gain ratio, which penalizes attributes by incorporating split information.

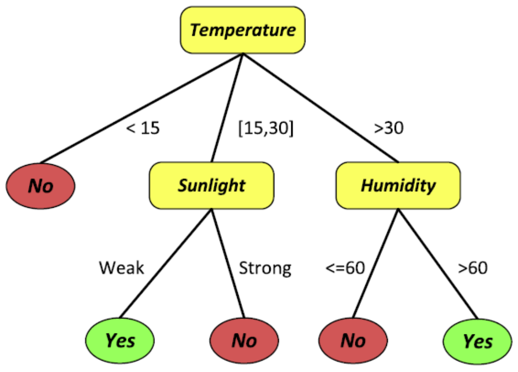


Fig. 1. An example of a decision tree.

Split Information

This information is sensitive to how broadly and uniformly the attribute splits the data. The gain ratio is calculated as Gain Ratio Information Gain Split Information Note that this ratio is not defined when the split information is zero, and the ratio many

tend to favor attributes for which the split information is very small. Consequently, to deal with this situation, the information gain is calculated for all attributes, and then the method only selects the best one. There are various methods to build a decision tree. Among them, ID3, proposed by Quinlan, is the simplest and most famous one. The ID3 algorithm recursively constructs a tree in a top-down divide-and-conquer manner. It uses information gain as the measure to determine the best attribute, and then creates a node for each possible attribute value, and partitions the training data into descendant nodes. There are three conditions to stop the recursion:

(1)All samples at a given node belong to the same class;

(2)No attribute remains for further partitioning;

(3)There is no sample at the node. Later, Quinlan proposed an extension of ID3 algorithm, C4.5. It performs similarly to 1D3 except using gain ratio to determine the best attribute. C4.5 algorithm also makes some improvements to ID3, including that it can handle numerical attributes by creating a threshold and splitting the data into those whose attribute value is above the threshold and those that are less than or equal to it. C4.5 can also prune the decision tree after creation, which reduces the size of the tree.

However, both original 1D3 and C4.5 algorithms cannot be used to directly combine the local classifiers, nor do they preserve the data attribute distribution to support pseudo data recovery. In this paper, we will present a substantially enhanced version to address these challenges.

## III. Decision-tree-based hierarchical scattered taxonomy

In this section, we first present the system overview, and then introduce our taxonomy approach in detail, showing how to construct the spanning tree, how to build the local decision tree, how to generate the pseudo data, and how to build the classifier in a bottom-up manner. We discuss the accuracy and the energy consumption afterwards.

### 3.1. System overview

We consider N sensor nodes n1, n2... nN scattered in a field. Each node covers an area of the field and is responsible for collecting data within the area. The data reporting follows a spanning tree rooted at the base station n0. The routing protocol for forming spanning tree will be presented in Section 3.2. Each sensor node ni first collects its local training data Di. If node ni is a leaf node, it builds a classifier Ci by a learning algorithm which we will illustrate in Section 3.3. The node then sends Ci to its parent node, say nj. We use a decision tree to represent the classifier, which, compared to the original data, is of a much smaller size.

### 3.2. Constructing spanning tree

Before we present the approach to construct the spanning tree, we first discuss how the shape of the tree affects the result of the taxonomy; specifically, how the height and width of the tree affect the energy consumption and the accuracy of the classifier. Given a certain amount of sensor nodes, it is intuitive that if the tree height is small, the tree width is large, and vice versa. Suppose in Fig. 2, node 0 is the base station and there are n other sensor nodes in one line with equal interval, and the distance between the base station and the last one is L. We assume there are two kinds of spanning tree, one is that each sensor node is the child of the other one that is closer to the base station, representing a high tree Tl, and the other is that all these sensor nodes are the child of the base station, representing a wide tree T2.

Therefore, to save more energy, a high tree is demanded. On the other hand, noise in the taxonomy is inevitable, and the noise will be accumulated along the spanning tree. Hence the larger the height is, the less accurate the classifier will be, and our evaluation demonstrates this. Therefore, we need an approach to control the shape of the spanning tree during its construction.

In our spanning tree constructing approach, the base station first broadcasts tree construction message containing the source (base station) and the depth (0). Upon receiving the message, the sensor nodes within the range of the base station send response messages back to the base station, and these nodes are candidate children.

**Table 1**
List of notations

| Notation | Explanation |
|---|---|
| $N$ | The number of sensor nodes, excluding the base station |
| $n_i$ | Sensor node ($i = 1, 2, \ldots, N$) |
| $n_0$ | Base station |
| $D_i$ | The training data collected by node $n_i$ |
| $D'_i$ | The pseudo data generated from classifier $C_i$ |
| $C_i$ | The local classifier built by node $n_i$ |
| $C_0$ | The global classifier built at the base station |

The base station selects all or some of these sensor nodes as its children, and the number of children will affect the shape of the spanning tree.

3.3. Building decision tree

In our system, the local decision trees are built by the widely used C4.5 algorithm [12]. The basic C4.5 algorithm, however, does not keep the information about the attribute distribution and the amount of the original training data, preventing pseudo data recovering from a classifier.

To solve this problem, we let each leaf node record the count of each class (i.e. the number of positive and negative labels in this application scenario). Therefore, we have the knowledge about the amount of the samples for building each branch of the decision tree, and thus we can make the distribution of the generated pseudo data resemble the original ones. Moreover, in the basic C4.5 algorithm, if all the samples belong to the same class, the recursion stops (e.g., when temperature is low in Fig. 1). Hence, the information of other attributes will be missing, which can cause problems with heterogeneous data distribution across different sensor nodes.

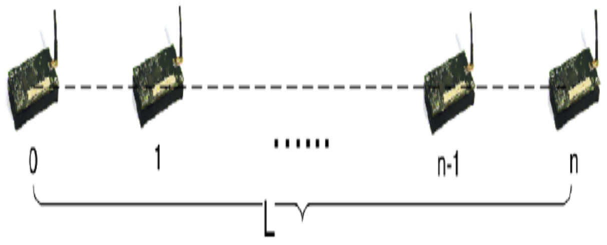


Fig. 2. Analysis of the spanning tree height.

In Algorithms 1 and 2 below, we describe the enhanced C4.5 algorithm for building the decision tree. Note that a brief illustration of the key steps of the basic C4.5 is in Section 2, and more details can be found in Quinlan's work.

### 3.4. Generating pseudo data

The pseudo data generation is one of the most important steps in our framework. A critical challenge here is to generate data that are as close to the original data as possible. In particular, the distribution of each attribute should closely resemble that of the original data.

**Algorithm 1.** LearningAlgorithm ($D$)

```
Require: training dataset $D$
    call EnhancedC45 ($root$, $D$, 0)
    prune decision tree $root$
    return $root$
```

Algorithm 3 summarizes our method to generate the pseudo data. For illustration, suppose the decision tree's leaf node represents a rule of when temperature is between 10 and 20, humidity is between 20 and 40, and sunlight is normal, there are 10 positive class labels and 90 negative ones. As such, the class label is negative.

We introduce two classifiers: Naive Bayes (NB) and a Bayesian Classifier based on K-means (BKM) that performs class decomposition. We consider two phases for both classifiers:

**Algorithm 2.** EnhancedC45 ($pt$, $D$, $depth$)

```
Require: pointer to the decision tree node $pt$, training dataset $D$,
  depth of the decision tree node $depth$
  if number of data $|D| = 0$ then
    return
  end if
  if $depth$ = number of attributes of the training data then
    get the class label with the most count in $D \rightarrow$ attribute of $pt$
    record the count of each class label for $pt$
    return
  end if
  calculate gain ratio for each attribute
  get the attribute with the greatest gain ratio $\rightarrow$ $target\_attribute$
  if $target\_attribute$ is category attribute then
    for all $target\_attribute$ value branches do
      add a child $pt'$
      set the value of the branch $v_i$
      partition $D \rightarrow D'$ with value $v_i$
      call EnhancedC45 ($pt'$, $D'$, $depth + 1$)
    end for
  else {//numerical category}
    for all $target\_attribute$ value branches do
      add a child $pt'$
      set the value range of the branch $[v_i, v_{i+1})$
      partition $D \rightarrow D'$ satisfying the value range $[v_i, v_{i+1})$
      call EnhancedC45 ($pt'$, $D'$, $depth + 1$)
    end for
  end if
```

(1) computing the model;
(2) scoring a data set based on the model. Our optimizations do not alter model accuracy.

**Naive Bayes:**

We consider two versions of NB: one for numeric attributes and another for discrete attributes. Numeric NB will be improved with class decomposition.

NB assumes attributes are independent and thus the joint class conditional probability can be estimated as the product of probabilities of each attribute. We now discuss NB based on a multivariate Gaussian. NB has no input parameters.

### 3.5. Hierarchical taxonomy

As mentioned above, we let the sensor nodes in the network be organized by a spanning tree, as illustrated in Section 3.2, and Fig. 3 shows an example. A leaf node builds the decision tree with the local sensed training data and sends the decision tree to the parent. The intermediate node periodically checks if there is any new classifier from children. If yes, it will generate a set of pseudo data for each

**Algorithm 3.** GeneratePseudoData $(C, pf)$

```
Require: decision tree received from one child C, preservation
  factor pf
  for all leaf nodes node of decision tree C do
    get rule R of node
    get class label counts → $c_1, c_2, \ldots, c_L$
    randomly generate $pf \cdot \sum_{i=1}^{L} c_i$ data satisfying R
    assign class label $l_k (k = 1, \ldots, L)$, to the data with probability
  as the proportion of the class label $c_k / \sum_{i=1}^{L} c_i$
    add these data to pseudo data set $D'$
  end for
  return pseudo data set $D'$
```

new classifier, and combines them with its local data. There are two situations here. (1) If the node has never built any classifier, which indicates that it has never received any classifier from its children, it will combine the generated pseudo data with its local sensed data and performs the learning algorithm. (2) If the node has once built a classifier, the previous received classifiers may have already been discarded (due to the memory constraint).

T We prefer a pull-based method, because if we utilize a push-based method, an update of the leaf node will trigger a series of re-computations and transmissions all the way to the root, consuming great energy.

In Algorithm 4, we summarize the detailed procedure of the bottom-up taxonomy.

**Algorithm 4.** BottomUpClassification ()

```
if node is leaf then
  periodically collect data D
  $C \leftarrow$ LearningAlgorithm(D)
  send C to its parent
else if node is not base station then
  periodically collect data D
  for all new classifiers $C_k$ from child k do
    $D'_k \leftarrow$ GeneratePseudoData($C_k$, a)
  end for
  $C \leftarrow$ LearningAlgorithm($D \cup (\bigcup D'_k)$)
  send C to its parent
else {// base station}
  if no classifier has been built then
    for all classifiers $C_j$ from child j do
      $D'_k \leftarrow$ GeneratePseudoData($C_k$, 1)
    end for
    $C \leftarrow$ LearningAlgorithm($\bigcup D'_k$)
  else if receive new classifier $C_k$ from child k then
    replace old classifier of child k with $C_k$
    for all classifiers $C_k$ from child k then
      $D'_k \leftarrow$ GeneratePseudoData($C_k$, 1)
    end for
    $T \leftarrow$ LearningAlgorithm($\bigcup D'_k$)
  end if
end if
```

## VI. PERFORMANCE EVALUATION

### 4.1. Configuration and dataset

We evaluate our framework with our customized simulator. We consider a square field consisting of m randomly deployed sensor nodes, with the base station being located in the center. We set m as 7 (totaling 49 nodes), and we control the spanning tree constructing approach to get the spanning trees that have heights of 3, 4 and 5, respectively. Fig. 4 shows an example with height 4. We have tested the topology with larger height, and the results show the same trend as the smaller height, thus we only provide the results of height being 3, 4 and 5.

This noise indicates the inaccuracy of a generative model representing the dataset, and we study two typical cases in the following. We generate 10 training datasets, each having 200 data. For each dataset, we make it into two versions, one is called heterogeneous data and the other is called homogeneous data. In the heterogeneous data set, the data distribution depends on the location of the sensor nodes, while the homogeneous data is independent on the location, and is randomly and uniformly scattered across sensor nodes, For each data in one dataset, we first calculate its coordinates according to the above, and then assign the data to the sensor node of that location if the training set of the node is not full (200 data). If that node is full, we then assign this data to another random node.

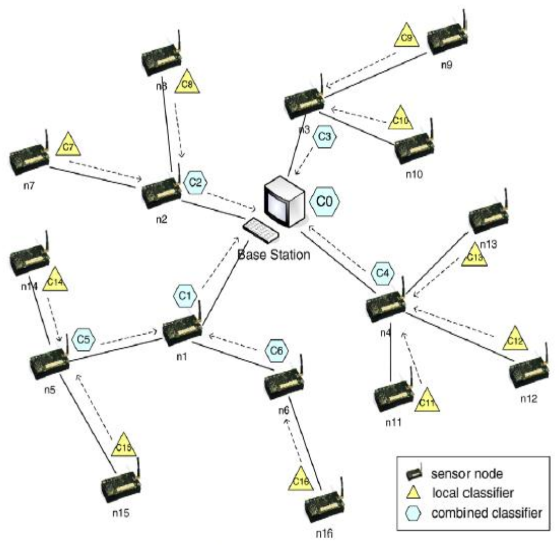


**Fig. 3.** The structure of the sensor network.

### 4.2. Comparison of enhanced and basic C4.5 algorithm

We next compare our enhanced C4.5 algorithm with the basic C4.5 algorithm, as well as the enhanced ID3 algorithm proposed in previous work. We clarify that we modify the basic C4.5 because it is not suitable for generating pseudo data in our approach for the heterogeneous data distribution. We again use the heterogeneous data with different noises, and plot the results in Figs. 6 and 7, respectively. We find that the enhanced C4.5 algorithm is not noticeably affected by the preservation factor when the factor is large enough, nor the height. On the other hand, the accuracy of basic C4.5 algorithm is much lower than ours, and is particularly lower when the preservation factor is smaller and the height is larger. To validate this, we also examine their performance with the homogeneous data, and the results show that the two algorithms perform almost the same. In comparison, C4.5 algorithm achieves better accuracy than the IDS algorithm, this is because C4.5 can deal with numerical attribute, whereas in ID3, the numerical attribute is considered as category attribute, thus reducing the accuracy in order to avoid the decision tree becoming too big.

### 4.3. Comparison of energy consumption

Finally, we investigate the energy consumption, which is one of the most important concerns in MANETS. We compare our hierarchical taxonomy with the ensemble method that transmits all the classifiers to the base station. The energy consumption consists of the computation energy consumption and the transmission energy consumption, which is significantly larger than the prior one, thus we neglect the computation energy consumption. The transmission energy consumption depends on the size of the transmitted data and the distance between the two nodes. Generally, it is proportional to the data size and square of the distance.

## V. Conclusion

In this paper, we have proposed a novel decision-tree-based hierarchical scattered taxonomy approach in MANETS. In particular, we consider a practical scenario in which the data distribution is heterogeneous, which is seldom studied before. In our solution, the sensor nodes are organized in a spanning tree, and local classifiers are built by individual sensor nodes and merged along the routing path. The classifiers are iteratively enhanced by combining strategically generated pseudo data with new local data, eventually converging to a global classifier for the whole network. We demonstrate that our approach maintains high taxonomy accuracy with very low storage and communication overhead. With the framework, an intelligent wildlife monitor can be developed. It will not only sense and track the animal, but also perform taxonomy that mines the environment suitability. However, there could be many possible enhancements. We are particularly interested in designing a more effective pseudo data generating algorithm that makes the generated data as close to the original as possible with limited side information. We are also interested in evaluating the performance with real sensor test beds.

## References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102-114.

[2] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Computer Networks 52 (12) (2008) 2292-2330.

[3] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, MANETS for habitat monitoring, in: Proceedings of ACM International Workshop on MANETS and Applications (WSNA), 2002, pp. 88-97.

[4] X. Liu, Q. Huang, Y. Zhang, Combs, Needles, haystacks: balancing push and pull for discovery in large-scale sensor networks, in: Proceedings of International Conference on Embedded Networked Sensor Systems (SenSys), 2004, pp. 122-133. [5] G. Werner-Allen, K. Lorincz, M. Welsh, 0. Marcillo, J. Johnson, M. Ruiz, J. Lees, Deploying a wireless sensor network on an active volcano, IEEE Internet Computing 10 (2) (2006) 18-25.

[5] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, J. Lees, Deploying a wireless sensor network on an active volcano, IEEE Internet Computing 10 (2) (2006) 18–25.

[6] E. Cayirci, T. Coplu, SENDROM: sensor networks for disaster relief operations management, Wireless Networks 13 (3) (2007) 409–423.

[7] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, in: Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), 1999, pp. 263–270.

[8] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Communications 1 (4) (2002) 660–670.

[9] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison Wesley Publishers, 2005.

[10] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman Publishers, 2006.

[11] J.R. Quinlan, Induction of Decision Trees, in: Machine Learning, Kluwer Academic Publishers., 1986, pp. 81–106 (Chapter 1).

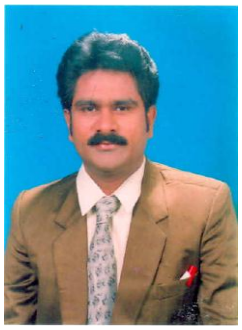
[12] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman Publishers, 1993.

[13] F. Bouhafs, M. Merabti, H. Mokhtar, A node recovery scheme for data dissemination in wireless sensor networks, in: Proceedings of IEEE International Conference on Communications (ICC), 2007, pp. 3876–3881.

[14] X. Cheng, J. Xu, J. Pei, J. Liu, Hierarchical distributed data classification in wireless sensor networks, in: Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), 2009.

[15] T.G. Dietterich, Ensemble methods in machine learning, in: Proceedings of the First International Workshop on Multiple Classifier Systems (MCS), 2000, pp. 1–15.

AUTHORS PROFILE



1]. Prof. S. Arif Hussain did his B.Tech in Gulbarga University and M.Tech from JNTU Anantapur. Now Pursuing Ph.D form JNTU Hyderabad. My interest areas are Ad-Hoc Sensor Networks in Mobile Computing, I attended 3 National level Conferences and I attend so many workshops National and International level. I publish a 3 National Level publications (Journals). I have 16 years experience of Teaching in various colleges. Presently working in Dr. K.V. Subba Reddy College of Engineering For Women , Kurnool. As a Professor & HOD of ECE Dept.

2]. Dr. K.E.Sreenivasa Murthy, did his B.Tech(ECE) and M.Tech from Sri Venkateswara University, Tirupati and obtained his Ph.D from Sri Krishnadevaraya University, Anantapur. His interest areas are microprocessors, computer hardware and Digital Signal Processing. Under his guidance two M.Phil and One Ph.D was awarded. At present eight students are working under his guidance for their Ph.Ds in JNTU, SKU etc. He is having around 20 publications to his credit. At present he is working as Principal of Sri Venkateswara Institute of Technology, Anantapur.