Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree

Jayanta Basak and Raghu Krishnapuram

Abstract—In this paper, we propose a method for hierarchical clustering based on the decision tree approach. As in the case of supervised decision tree, the unsupervised decision tree is interpretable in terms of rules, i.e., each leaf node represents a cluster, and the path from the root node to a leaf node represents a rule. The branching decision at each node of the tree is made based on the clustering tendency of the data available at the node. We present four different measures for selecting the most appropriate attribute to be used for splitting the data at every branching node (or decision node), and two different algorithms for splitting the data at each decision node. We provide a theoretical basis for the approach and demonstrate the capability of the unsupervised decision tree for segmenting various data sets. We also compare the performance of the unsupervised decision tree with that of the supervised one.

Index Terms—Unsupervised decision tree, entropy, data set segmentation.

1 Introduction

The decision tree [1], [2], [3], [4], [5], [6], [7] is a widely used tool for classification in various domains such as text mining [8], speech [9], [10], bio-informatics [11], Web intelligence [12], [13], and many other fields that need to handle large data sets. One major advantage of the decision tree is its interpretability, i.e., the decision can be represented in terms of a rule set. The branching decision at each node is determined by the value of a certain attribute or combination of attributes, and the choice of the attribute(s) is based on a certain splitting criterion that is consistent with the objective of the classification process. Each leaf node of the tree represents a class and is interpreted by the path from the root node to the leaf node in terms of a rule such as: "If A_1 and A_2 and A_3, then class C_1," where A_1, A_2, and A_3 are the clauses involving the attributes and C_1 is the class label. Thus, each class can be described by a set of rules.

In the context of clustering, no supervised information in terms of class labels is available for the data. However, it is often necessary to interpret the clusters for knowledge discovery in various contexts such as customer profiling, text mining, and image and video categorization. In this paper, we propose a method to construct an unsupervised decision tree in which the embedded characteristics of the data set can be described by a set of rules analogous to the supervised decision tree.

Unsupervised decision trees are structurally similar to hierarchical clustering methods. Algorithms for hierarchical clustering are generally of two kinds, namely, top-down and bottom-up [1]. In the bottom-up algorithms for hierarchical clustering, each data point is considered to be a separate cluster and then these are progressively combined depending on certain criteria to generate the hierarchy. The structure generated by this process is commonly referred to as a dendrogram. Different distance measures give rise to different cluster structures at the end of the algorithm. In top-down hierarchical clustering algorithms, to begin with, all data points are considered to belong to the same cluster. The data set (consisting of a set of patterns) is then divided into a certain number of clusters at a coarse level. Then, each of these coarse clusters is further segmented into finer levels in the subsequent levels until a stopping criterion is satisfied. For example, in [14], the decision tree or a hierarchy representing the clusters was built based on the minimization of a criterion function that is generally used for clustering in the EM (expectation-maximization) and soft K-Means (Fuzzy c-Means) algorithms. In all these approaches for hierarchical clustering (either top-down or bottom-up), every level represents only a resolution or granularity of clusters, and in order to interpret the hierarchy, the clusters at each node need to be analyzed separately.

There have been a few attempts to build decision trees for data samples without any class labels. In [15], the authors propose a method to build a decision tree for text categorization where the information about the text clusters is used in conjunction with the supervised information about whether a document is useful or not with respect to a user. In [16], [17], noisy data points are first injected into the original data set. Then, a standard (supervised) decision tree is built by classifying the original data points and the noisy data points under the assumption that the original data points and the noisy data points belong to two different classes. In [17], the authors provide a guideline for injecting noise in the data set at each level of the decision tree. In these methods, the decision tree formed is a binary decision tree and it is hard to generalize it to an k-ary decision tree where k is the number of child nodes created at a node and is a variable that depends on the data at each node of the decision tree.

- *The authors are with the IBM India Research Laboratory, Block-I, Indian Institute of Technology, Hauz Khas, New Delhi 110016 India. E-mail: {bjayanta, kraghura}@in.ibm.com.*

Manuscript received 30 Apr. 2003; revised 3 Dec. 2003; accepted 17 Mar. 2004; published online 18 Nov. 2004.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0045-0403.

It may be mentioned here that there exist certain investigations about the divisive clustering algorithms [18], [19] to generate data clusters in a top-down hierarchical manner. In these algorithms, a particular cluster is tested by different criteria in order to ensure that the cluster can be further subdivided or not. If a cluster is selected for further partitioning by the criteria, then either the set of data records is partitioned by K-means or some other method. The resulting clusters need to be further analyzed to generate descriptive rules as in the case of bottom-up hierarchical clustering algorithms.

In this article, we propose an alternative method to build a decision tree for unlabeled data based on the attributes. At any given node, we choose an attribute in such a manner that if clustering is solely performed based on that attribute, then the resulting inhomogeneity is minimized. (Here, it can be mentioned that the objective of clustering the data set is to discover subsets of data records that possess homogeneous characteristics). Thus, the proposed method imparts a direct interpretability for the unsupervised decision tree in the context of clustering. We present a brief description of the overall algorithm in Section 2. In Section 3, we discuss two data set partitioning techniques and, in Section 4, we describe four different measures for attribute selection in the unsupervised mode. Certain theoretical analysis is provided in Section 5. We demonstrate the experimenatal results for various data sets in Section 6, and conclude the paper in Section 7.

2 Unsupervised Decision Tree

We form the unsupervised decision tree by stepwise clustering, where at every decision node, the data is split based on a single feature or a group of features. The candidate feature (or subset of features) is chosen based on a certain inhomogeneity measure. Thus, the following two aspects are crucial for building an unsupervised decision tree: 1) measuring the inhomogeneity of the data at a given node of the unsupervised decision tree, and selecting the attribute that has the maximum amount of inhomogeneity when the data is viewed in the dimension of the attribute (Section 4), and 2) partitioning the set of patterns or data records in the node into subsets based on the attribute and allocating each subset to the resultant child nodes (Section 3). We present the overall algorithm in Fig. 1. In the following, we provide the overall algorithm for constructing an unsupervised decision tree. We first explain the notation used:

- L : List of nodes in the decision tree. The root node is indexed as 1. Initialize the list as $L = (1)$.
- $NODE$: A data structure to store the data tags to indicate to which leaf node in the decision tree the data record belongs to. For example, $NODE(t) = N$ indicates that the tth data record belongs to the leaf node N of the decision tree. We initialize $NODE(t) = 1$ for all $t, t = 1, 2, \cdots, n$.
- L_F: List of attribute sets for each node N in the decision tree in the form $L_F(N) = \{F_N\}$, where F_N is the set of attributes associated with the node N of the decision tree. Initialize L_F as $L_F(1) = \{F\}$, where F is the entire set of features.
- DT: A data structure to store the decision tree. Since the number of child nodes can be different for different nodes in the decision tree, we store the decision tree as an array of links (or tree edges) such that $DT(p) = \{N_i, N_j, f_k, v_{kmin}, v_{kmax}\}$. Here, p is an index of the link, N_i is the index of the parent node, N_j is the index of the child node, f_k is the attribute associated with the pth link (here, we make the branching decision based on only one attribute f_k), and v_{kmin} and v_{kmax} are the minimum and maximum attribute values of attribute f_k in the data segement available to the child node N_j.
- D_N: The data set allocated to node N of the decision tree. For the root node, i.e., for $N = 1$, $D_1 = D$, where D is the entire data set. We store the data set allocated to a node by the indices of the data records. Thus, $D_N = \{k : NODE(k) = N\}$.

```
begin
  Initialize $L = (1)$, $DT = \phi$, $NODE(k) = 1$ for all data records $k$;
  Initialize $linkCounter = 1$; /* A tag to keep count of the links */
  for every $N \in L$ /* starting with the root node */
  begin
    delete $N$ from $L$;
    Get the set of attributes $F_N = L_F(N)$ associated with node $N$;
    Get the data set $D_N = \{k : NODE(k) = N\}$;
    Compute $j = argmax_{f_i \in F_N}\{H_i(f_i)\}$;
    if $H_j$ >certain user defined threshold then
    begin
      Segment the data set $D_N$ into a suitable number (say $K_N$)
      of clusters, and determine $v_{jmin_c}, v_{jmax_c}$ $\forall c \in \{1, \cdots, K_N\}$;
      /* $v_{jmin_c}$ and $v_{jmax_c}$ represent the minimum and maximum
      attribute values of the attribute $f_j$ in segment $c$ */
      Create $K_N$ child nodes for the node $N$ and index the
      nodes as $|L|+1, \cdots, |L|+K_N$;
      Append the indices of the new nodes to the tail of $L$;
      for $c = 1$ to $K_N$ do
      begin
        $DT(linkCounter) = \{N, |L|+c, f_j, v_{jmin_c}, v_{jmax_c}\}$;
        $linkCounter = linkCounter + 1$;
      endfor
      for every data record $k \in D_N$
      begin
        Find $c$, $c \in \{1, \cdots, K_N\}$ such that
         $v_{jmin_c} < x_k(f_j) \leq v_{jmax_c}$.
        /* $x_k$ is the data vector corresponding to data record index $k$ */
        Set $NODE(k) = |L| + c$.
      endfor
      (Optional Step)Update $L_F$ such that $L_F(|L|+c) = \{(F_N - f_j)\}$,
                                          $\forall c \in \{1, \cdots, K_N\}$;
    endif
  endfor
end
```

Fig. 1. Algorithm for constructing unsupervised decision tree in the breadth-first order. We included an optional step of excluding the attribute f_j from L_F. However, in our implementation, we do not eliminate the attribute.

3 Partitioning the Data Set into Segments at Each Node

In this section, we present two different algorithms for splitting the data based on a single attribute.

3.1 Data Set Partitioning Based on Valley Detection

1. For categorical (or nominal) attributes, we partition the data set into a number of segments that is equal

to the number of different attribute values in the data set.

2. For numerical attributes, we compute the histogram of the data along the attribute dimension.
3. We then determine the valley points of the smoothed histogram by computing the change in direction of the gradient.
4. We evaluate each valley point as

$$e_i = \frac{\min\{q_i - v_i, q_{i+1} - v_i\}}{1 + \lambda v_i}, \tag{1}$$

   where v_i is the height of the valley, and q_i and q_{i+1} are heights of the peaks on either side of the valley. Based on the evaluated score e_i, we consider the top $k-1$ valleys (for k-ary tree) as potential cut-off points (if the number of detected valleys is less than k, then we consider all detected valleys).
5. Let $c_1, c_2, \cdots, c_{k-1}$ be the cut-off points. For every pair (c_i, c_{i+1}) of consecutive cut-off points, the data records for which the value of the attribute falls between c_i and c_{i+1} constitute the ith segment. If the number of data records in a segment is less than a certain predefined count, then we merge the records into the nearest segment.

The valley detection algorithm can be performed in $O(n)$ time where n is the number of data points. Other methods such as model selection, mode hunting algorithms in EM framework [20], [21], [22], [23], [24], or techniques from the image segmentation literature can also be used.

3.2 Binary Splitting of the Data Set

Let $H(D)$ represent the inhomogenity or information content of the data set D with respect to some attribute f_a. If we perform a binary partition of the data at a certain cut-off point c into two subsets D_1^c and D_2^c, then the resulting change in information is

$$\Delta H(c, D) = H(D_1^c) + H(D_2^c) - H(D). \tag{2}$$

If $\max_c \Delta H(c, D) > 0$, then we find the cut-off point c_0 as

$$c_0 = argmax_c\{\Delta H(c, D)\}. \tag{3}$$

Note that, in this partitioning process, we always split the data set into two segments with $O(n)$ complexity where n is the number of data points. However, it is possible to find a k partition using brute force search with $O(n^{(k-1)})$ complexity.

4 Attribute Selection

At each node of the unsupervised decision tree, we select an attribute in such a way that the inhomogeneity of the data set is maximum with respect to that attribute. We used various entropy measures in selecting the attribute at each level of the tree. Information theoretic measures have been used earlier in the context of clustering [25], [26]. We describe four different measures of inhomogeneity of the data set with respect to an attribute.

4.1 Measures of Inhomogeneity

Measure I. Let μ_{ij} be the degree of similarity [27] of two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the data set available at a given node. We define μ_{ij} as

$$\mu_{ij} = g\left(1 - \frac{d_{ij}}{d_{max}}\right), \tag{4}$$

where d_{ij} is the distance $d(\mathbf{x}_i, \mathbf{x}_j)$ between the data points $\mathbf{x}_i$ and $\mathbf{x}_j$ (not necessarily the Euclidian distance), d_{max} is the maximum of all interpoint distances, and the function $g(.)$ is a monotonically nondecreasing function,

$$g(x) = \begin{cases} x & \text{for } 0 \le x \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Equation (4) indicates that $\mu_{ij} \in [0, 1]$ for all i and j. The measure of importance for an attribute f_a is computed as [28]

$$H_a = -\left(\sum_{i,j} \mu_{ij}(1 - \mu_{ij}^a) + \mu_{ij}^a(1 - \mu_{ij})\right), \tag{6}$$

where μ_{ij}^a is the similarity between x_i and x_j in the reduced attribute space when the attribute f_a is removed from the set of attributes. We select the attribute for which H_a is maximum.

The measure of importance of an attribute can also be computed from the ditribution of interpoint distances. One such measure can be derived from Kullback-Leibler divergence such that

$$H_a = -\int p(d) \log\left(\frac{p(d)}{p(d^a)}\right) \delta d, \tag{7}$$

where $p(d)$ and $p(d^a)$ are the distributions of the interpoint distances in the original and reduced attribute space, respectively. Note that, other divergence measures such as Jeffrey-Shannon divergence can also be used for this purpose considering that there are a finite number of sample points.

Measure II. Instead of computing the loss of information when an attribute is dropped from the set of attributes, we can also compute ambiguity when an attribute is considered alone. In that case, the importance of an attribute is computed as

$$\hat{H}_a = -\sum_{i,j} \hat{\mu_{ij}^a}(1 - \hat{\mu_{ij}^a}), \tag{8}$$

where $\hat{\mu_{ij}^a}$ indicates the degree of similarity between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in terms of the attribute f_a only.

In terms of the distribution,

$$H_a = -\int p(d^a) \log(p(d^a)) \delta d^a. \tag{9}$$

Measure III. The time and space complexity for computing and storing distances between all pair of points $O(n^2)$. Instead of computing the distances between every pair of data points, it is possible to find the nature of the distribution along each dimension of the attribute space. If the information content of an attribute is high, or the inhomogeneity of the data set is high for the attribute, then the distribution will deviate from a uniform one. We, therefore, consider the following entropy measure:

$$H_a = -\int p(x(a)) \log(p(x(a))dx(a), \tag{10}$$

where $x(a)$ is the value for the attribute a and $p(x(a))$ is the density function of the data along that dimension. The measure is computed as

$$H_a = -\sum_i n_i \log n_i, \tag{11}$$

where n_1, n_2, $\cdots$, and n_b are the number of data points in the b bins of the histogram of $x(a)$ such that $\sum_i n_i = n$ is the total number of data records.

Measure IV. We can select the most significant attribute $f \in F$ such that

$$f = argmax_{\forall f_a \in F}\{H_a(D_1^{c_0^a}) + H_a(D_2^{c_0^a}) - H_a(D)\}, \tag{12}$$

where

$$c_0^a = argmax_c\{H_a(D_1^c) + H_a(D_2^c) - H_a(D)\}. \tag{13}$$

The measure reflects the increase in information or decrease in ambiguity of the data set when it is optimally partitined along the attribute dimension (as described in Section 3.2).

4.2 Handling Nonnumeric Attributes

In *Measure III* and *Measure IV*, for the categorical or nominal attributes, we count the number of records having the same attribute value. For example, let the data have colors (a categorical attribute), namely, *red*, *green*, *blue*, and *yellow*. We count the number of records that have the value *red*, the number of records that have the value *green*, and so on. The criterion measure is then computed exactly the same way as in (11), i.e.,

$$\begin{aligned} H = &- n_{red}\log(n_{red}) - n_{green}\log(n_{green}) \\ &- n_{blue}\log(n_{blue}) - n_{yellow}\log(n_{yellow}). \end{aligned} \tag{14}$$

In the case of *Measure I* and *Measure II*, it is difficult to handle categorical attributes as easily as in the case of *Measure III* and *Measure IV* since we need to define the distance between the data records. Ordinal attributes can be handled in the same way as numerical attributes in all four measures. As discussed in [27], various transformations can be applied selectively to handle categorical or nominal attributes.

5 Theoretical Issues

In this section, we briefly analyze the computational complexities of different measures and the unsupervised decision tree construction algorithms. We also provide certain empirical justifications for selecting the measures.

5.1 Computational Complexity of Attribute Selection

Measure I. The distance computation for one pair of data records requires $O(m)$ time where m is the number of available attributes. Thus, for n such data records, we need $O(mn^2)$ time. Determining the most important attribute at a given node of the UDT requires a search over m attributes and it takes $O(m^2n^2)$ time. Note that it is possible to reduce the time complexity to $O(mn^2)$ if the distances between every pair of points along every attribute dimension are stored separately which requires an additional $O(mn^2)$ storage space.

Measure II. The distance computation for all pairs of data records with a single attribute requires $O(n^2)$ time. Determining the most important attribute at a given node of the UDT requires a search over m attributes and it takes $O(mn^2)$ time.

Measure III. For a single attribute, a histogram of the attribute values of data records can be obtained in $O(n)$ time. The measure from the histogram can be computed in $O(b)$ time, where b is the number of bins in the histogram. Since usually $b << n$, it takes $O(n + b) = O(n)$ time. Thus, searching over m attributes takes $O(mn)$ time.

Measure IV. We need to compute the entropy for all possible data set partitions over the discretized space of bins (as provided by (12)). This can again be performed in $O(b)$ time in the following way. Let $\{H_1, H_2\}$ denote the ambiguity of the two segments of a binary partition of the data set considering one cut-off point (say, bin c). As we move the cut-off point from bin c to bin $c+1$, we can update H_1 and H_2 by adding the information pertaining to bin c to H_1 and deleting that information from H_2 in $O(1)$ time. Thus, the total time for computing the importance of a single attribute is $O(n + b) = O(n)$, and for searching over m attributes, it takes $O(mn)$ time.

5.2 Complexity of Constructing a Binary Unsupervised Decision Tree

In this analysis, we assume that the attributes can be repeated at different levels of the hierarchy such that the number of candidate attributes to be evaluated is always m at any level of the unsupervised decision tree.

Measure I. The time required to find out the best binary data set partition for a given attribute is $O(n)$ (as described in the complexity analysis of *Measure IV*). Since attribute selection can be performed in $O(m^2n^2)$ time, the total time for constructing the unsupervised decision tree can be obtained as

$$\begin{aligned} T = &O(m^2n^2) + O(m^2n_{11}^2 + m^2n_{12}^2) \\ &+ O(m^2n_{21}^2 + m^2n_{22}^2 + m^2n_{23}^2 + m^2n_{24}^2) + \cdots, \end{aligned} \tag{15}$$

where n_{ij} is the number of data records available in the jth segement of level i of the binary unsupervised decision tree, where $n_{01} = n$. If the binary tree is balanced, then $n_{ij} = n/2^i$, and $\sum_j n_{ij}^2 = n^2/2^i$. Since $\sum_j n_{ij} \le n$ for all i, the overall time in such a special case can be shown to be $T = O(2m^2n^2)$. On the other hand, if the binary tree is skewed such that only a fraction of the data records form a leaf node (either the left or right child) and the rest of the data records are allocated to the other node to be subdivided at the next level, then $T = O(m^2n^2c)$, where c is the number of clusters (i.e., number of leaf nodes). In general, we have

$$O(2m^2n^2) \le T \le O(m^2n^2c). \tag{16}$$

TABLE 1
Estimates of the Time Complexities for Constructing Unsupervised Decision Trees with Different Measures with n Data Records, Each Having m Attributes

measure	Type of UDT	
	Binary	k-ary
Measure I	$O(2m^2n^2) \leq T \leq O(m^2n^2c)$	$O(m^2n^2) \leq T \leq O(m^2n^2c)$
Measure II	$O(2mn^2) \leq T \leq O(mn^2c)$	$O(mn^2) \leq T \leq O(mn^2c)$
Measure III	$O(mn\log_2(c)) \leq T \leq O(mnc)$	$O(mn\log_k(c)) \leq T \leq O(mnc)$
Measure IV	$O(mn\log_2(c)) \leq T \leq O(mnc)$	$O(mn\log_k(c)) \leq T \leq O(mnc)$

Measure II. Following a similar argument as in the case of *Measure I*, we can show that

$$O(2mn^2) \leq T \leq O(mn^2c). \tag{17}$$

Measure III and **Measure IV.** Since the computational complexity of the most important attribute is the same for both *Measure III* and *Measure IV*($O(mn)$), the time complexity for constructing binary UDT is the same for both these measures. The overall computational time can be shown to be $T = O(mnl)$, where l is the depth of the tree. In the case of a balanced decision tree, we have $l = \log_2(c)$, and for a skewed one, we have $l = c - 1$. Thus, the overall complexity is given as

$$O(mn\log_2(c)) \leq T \leq O(mnc). \tag{18}$$

5.3 Complexity of Constructing A *k*-ary Unsupervised Decision Tree

Measure I. The time complexity for constructing a k-ary unsupervised decision tree, k being the maximum number of child nodes of any node at any level, is given by

$$\begin{aligned} T = O(m^2n^2) + O(m^2(n_{11}^2 + n_{12}^2 + n_{13}^2 + \cdots + n_{1k}^2)) \\ + O(m^2(n_{21}^2 + n_{22}^2 + n_{23}^2 + \cdots + n_{2k^2}^2)) + \cdots. \end{aligned} \tag{19}$$

Following the same arguments as in the previous section, we can show that in the best possible case, $T = O(m^2n^2)$ for large values of k and c. In the worst case, we have $T = O(m^2n^2c)$. Thus, in general,

$$O(m^2n^2) \leq T \leq O(m^2n^2c). \tag{20}$$

Measure II. Following the same arguments as in the case of *Measure I*, we have

$$O(mn^2) \leq T \leq O(mn^2c). \tag{21}$$

Measure III and **Measure IV.** Following the arguments in the previous section, we have

$$T = O(mnl). \tag{22}$$

However, in the case of a k-ary balanced tree, we have $l = \log_k(c)$. Thus,

$$O(mn\log_k(c)) \leq T \leq O(mnc). \tag{23}$$

Note. We summarize the time complexities for constructing the different decision trees with different measures in Table 1. It should be noted here that if c (the number of clusters or leaf nodes obtained by the unsupervised decision tree) is the same as the actual number of clusters present in the data set, then the algorithm for unsupervised decision trees with *Measure III* or *IV* is more time efficient than the c-means algorithm.

5.4 Relative Behavior of the Measures

From previous section, it is observed that the construction of UDTs using *Measure I* is the most expensive among all four measures. This is because *Measure I* finds the most important attribute by considering the information content (or the ambiguity) of the attribute with respect to the entire attribute set. In general, this kind of feature selection techniques are subtractive methods where the importance of an attribute is judged based on the loss of information content when the attribute is dropped from the set of attributes. On the other hand, construction of UDT using *Measure II* is less costly than that using *Measure I*. This is due to the fact that *Measure II* finds the importance of an attribute by considering the information content of that attribute only and does not consider the information content with respect to other attributes. Thus, in a situation where data records with different class labels (note that, we do not use class labels in the construction of UDT) are highly overlapped, *Measure I* may provide certain better estimates for selecting the important attributes. Construction of UDT using *Measure III* and *Measure IV* are the most efficient in terms of time complexity. However, these measures do not take into account the distances between the data records.

It is difficult to predict theoretically the behavior of these measures for any general data distribution. In the Appendix, we provide an empirical analysis for a theoretical justification of the measures considering a mixture-of-Gaussian model for the data records. For a bimodal distribution, we observe that *Measure III* and *Measure IV* behave exactly in the same way. However, for construction of the binary unsupervised decision tree, *Measure IV* provides a more natural measure since it is directly derived from the data partitioning process. It may be mentioned here (as empirically studied in the Appendix) that *Measure II* becomes more insensitive if the modes of the bimodal distribution are very close to each other, as compared to *Measure III* or *Measure IV*. Moreover, both *Measure III* and *Measure IV* can handle nonnumeric attributes which is difficult in the case of *Measure I* and *Measure II*. On the other hand, *Measure II* does not require the computation of a histogram (unlike *Measures III* and *IV*) and, thus, it is independent of the bin size.

In computing *Measure III* and *Measure IV*, the bin size of the histogram should be small enough so that the measures capture the small variations in the distribution. At the same time, bin size of the histogram should not be so small that

TABLE 2
A Summary of the Data Sets That We Used in Our Experiment

Data Set	no. of samples	no. of attributes	no. of class	source
Synthetic	238	5	4	Generated
Iris	150	4	3	UCI
Pageblock	5473	10	5	UCI
Diagnostic Breast Cancer (wdbc)	569	30	2	UCI
Liver Disease (bupa)	345	7	2	UCI
Spambase	4601	57	2	UCI

The data sets are obtained from the UCI data repository [29].

very few points are contained in a bin. We observed that so long as the number of bins is approximately less than one-fifth of the number of available data records, the behavior of the measure is stable (we empirically observed this behavior for independently sampled data sets from a normal distribution). This provides a guideline for automatic selection of the number of bins.

6 Experimental Results

In Table 2, we describe the data sets that we used in our experiments involving the proposed algorithm for constructing unsupervised decision trees. We implemented the algorithms for constructing unsupervised decision trees on a Pentium IV machine with Matlab 5.4. The size of the decision tree is always controlled by the minimum possible size of a cluster represented by a leaf node. For relatively smaller data sets such as IRIS, WDBC, Bupa, and the synthetic data set, we consider leaf nodes that have a minimum of 10 data records. For relatively larger data sets such as Spambase and Pageblock data sets, we consider leaf nodes that consist of a minimum of 50 data records.

It may be mentioned here that the construction of a binary tree does not depend on any user specified parameter for a given minimum size of a cluster. The data set is partitioned if there is a reduction in ambiguity in the resulting partitioned data set. In the case when the k-ary tree based valley detection technique is used, we find the best $k-1$ valleys (selected according to (1) with $\lambda = 1$) as cut-off points. Thus, the structure of a decision tree can be controlled by the value of k and the minimum allowable size of a cluster. However, even if k is chosen to be very large, we observed that for all the data sets in Table 2, a node does not branch out into more than six nodes in the next level.

We implememted all four measures and constructed both binary and k-ary decision trees with these four measures for the IRIS and synthetic data sets. We found that all these four measures provide the same structure of the decision tree in all four cases. Since *Measure I* and *Measure II* are relatively costlier (in terms of CPU time) than *Measures III* and *IV*, we used the later two measures in constructing UDTs for the other data sets. In the rest of this section, we report our results based on *measures III* and *IV*.

As an example of the data partitioning process, in Fig. 3, we illustrate the partitions generated by the unsupervised decision tree (Fig. 2) on the synthetic data set. The number inside each region in Fig. 3 represents the node of the decision tree to which the corresponding subset of data records are allocated by the algorithm. We plot the data in the space of two attributes (two-dimensional space), one is the attribute selected at the top level (see Fig. 2) and the other is the attribute at the next level. Since there are two nodes at the second level of the decision tree, we illustrate the partitions in two figures showing the corresponding data contained in the next levels.

Fig. 4 shows the unsupervised decision trees generated for Iris data set. The corresponding rules that we obtain by interpreting the decision tree are listed in Table 3 as another illustration. A subset of the rules obtained by construction of the tree for the Pageblock data set is shown in Table 4. We explain the rules obtained during the construction of decision trees in the respective table captions.

We evaluated the quality of the clusters obtained from the unsupervised decision trees by considering the class label information as follows: We assigned each leaf node of an unsupervised decision tree the class label that occurs most often with the data records in the leaf node. Thus, each class is represented by a set of leaf nodes, i.e., by a set of rules. The class label can be predicted from these set of rules for data records that were not used in the construction of the tree. We obtain a 10-fold cross-validation score by repeating this process. We then compare these scores with those obtained from other clustering algorithms such as c-means and Ward's algorithm (a hierarchical clustering algorithm) [30]. Table 5 summarizes the results of the cross-validation

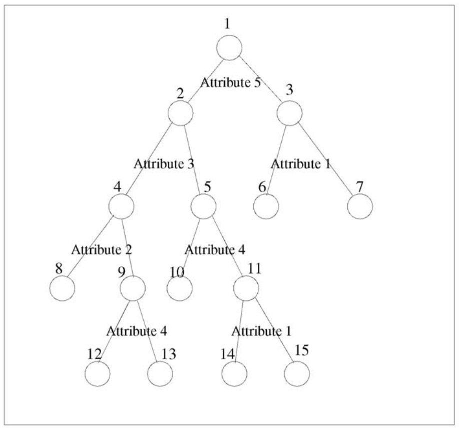


Fig. 2. The unsupervised binary decision tree obtained for the synthetic data set.

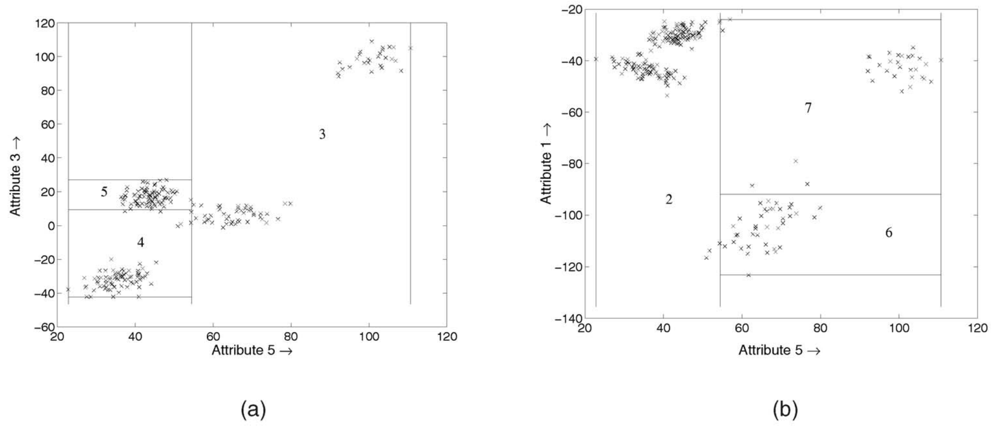


Fig. 3. Partition generated for the synthetic data set by the binary unsupervised decision tree (Section 3.2) with *Measure III* for attribute selection.

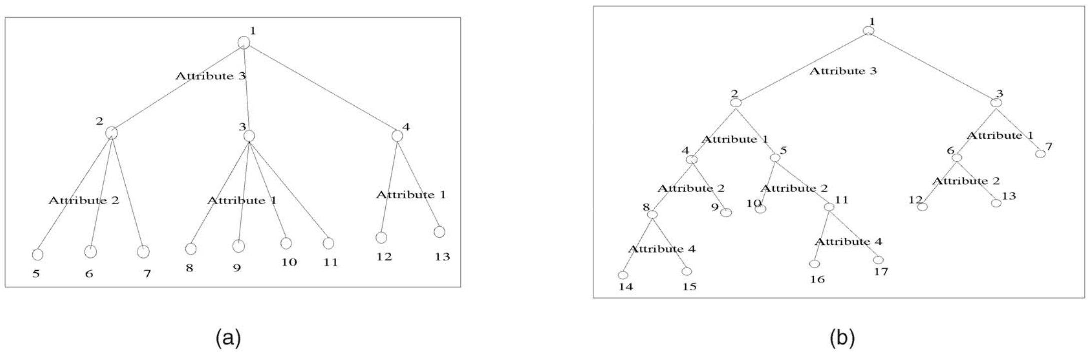


Fig. 4. (a) The *k*-ary unsupervised decision tree generated from the IRIS data set. The data set is partitioned by using the valley detection algorithm. The decision tree constructed is the same for *Measure I*, *Measure II*, and *Measure III*. (b) The binary unsupervised decision tree generated from the IRIS data set.

scores along with those obtained with a supervised decision tree (C4.5). It can be seen from the table that the unsupervised decision tree outperforms the *c*-means algorithm and Ward's algorithm in terms of the classification score. We can also see that, even without using the class label information, the performance of the unsupervised decision tree is almost as good as that of the supervised decision tree. This indicates that the quality of clusters generated by the algorithm is good.

TABLE 3
The Set of Rules Obtained by Interpreting the Unsupervised Decision Tree (See Fig. 4) Generated for the Iris Data Set

(petal length < 2.475) & (sepal width < 3.21) $\rightarrow$ *cluster* 1
(petal length < 2.475) & ($3.21 \leq$ sepal width < 3.67) $\rightarrow$ *cluster* 2
(petal length < 2.475) & (sepal width ≥ 3.67) $\rightarrow$ *cluster* 3
($2.475 \leq$ petal length < 5.33) & (sepal length < 5.6) $\rightarrow$ *cluster* 4
($2.475 \leq$ petal length < 5.33) & ($5.6 \leq$ sepal length < 5.775) $\rightarrow$ *cluster* 5
($2.475 \leq$ petal length < 5.33) & ($5.775 \leq$ sepal length < 6.44) $\rightarrow$ *cluster* 6
($2.475 \leq$ petal length < 5.33) & (sepal length ≥ 6.44) $\rightarrow$ *cluster* 7
(petal length ≥ 5.33) & (sepal length < 6.55) $\rightarrow$ *cluster* 8
(petal length ≥ 5.33) & (sepal length ≥ 6.55) $\rightarrow$ *cluster* 9

The Iris data set consists of four numeric attributes, namely, sepal length, sepal width, petal length, and petal width (all are in cm). It is observed that attribute 3, i.e., petal length, plays the most important role in deciding the groups in the data set. Within each group of data records, the subgroups are decided based on the values of either sepal width or sepal length.

We now describe the application of the proposed unsupervised decision tree for automatically generating a taxonomy from a text data set. Automatic taxonomy generation is an important problem in summarizing search results [33]. We considered the *Classic3* data set [34] consisting of 3,893 documents from three different classes, namely, *medicine*, *aerosystems*, and *literature* with a dictionary size of 5,717 words. We generated the taxonomy by constructing binary and *k*-ary unsupervised decision trees with *Measure IV* as the attribute selector. In the text data set, we considered the minimum number of records that can be present at a leaf node to be 50. In information retrieval, TF-IDF (text frequency-inverse document frequency) [35] method is commonly used to rank words based on their discriminating power. Therefore, as a comparison, we generated the same decision trees with TF-IDF [35] as the attribute selector where the TF-IDF value of a word is used as the attribute selection criterion. Figs. 5 and 6 illustrate the different taxonomies generated by the different techniques for generating unsupervised decision tree. In the figures, the leftmost branch represents a low occurrence of the word and the rightmost branch represents a high occurrence of the word. The branches between the leftmost and the rightmost represent various grades of occurrence of the word. As in the case of the other data sets, we tested the classification performance of different unsupervised decision trees on the text data set. Table 6 demonstrates that the unsupervised decision trees constructed with *Measure IV* as

TABLE 4
A Subset of Rules Obtained by Interpreting the *k*-ary Unsupervised Decision Tree Generated from the Page Block Data Set

$(pblack < 0.83)$ & $(pand < 0.81)$ & $(length < 230.33) \rightarrow cluster\ 1$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(230.33 \leq length < 294.27)$ & $(wbtrans < 473.87) \rightarrow cluster\ 2$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(230.33 \leq length < 294.27)$ & $(wbtrans \geq 473.87) \rightarrow cluster\ 3$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(294.27 \leq length < 358.20)$ & $(wbtrans < 728.33) \rightarrow cluster\ 4$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(294.27 \leq length < 358.20)$ & $(wbtrans \geq 728.33) \rightarrow cluster\ 5$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(358.20 \leq length < 422.13)$ & $(blackpix < 1061) \rightarrow cluster\ 6$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(358.20 \leq length < 422.13)$ & $(1061 \leq blackpix < 3042) \rightarrow cluster\ 7$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(358.20 \leq length < 422.13)$ & $(blackpix \geq 3042) \rightarrow cluster\ 8$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(length \geq 422)$ & $(wbtrans < 270) \rightarrow cluster\ 9$
$(pblack < 0.83)$ & $(pand < 0.81)$ & $(length \geq 422)$ & $(wbtrans \geq 270) \rightarrow cluster\ 10$

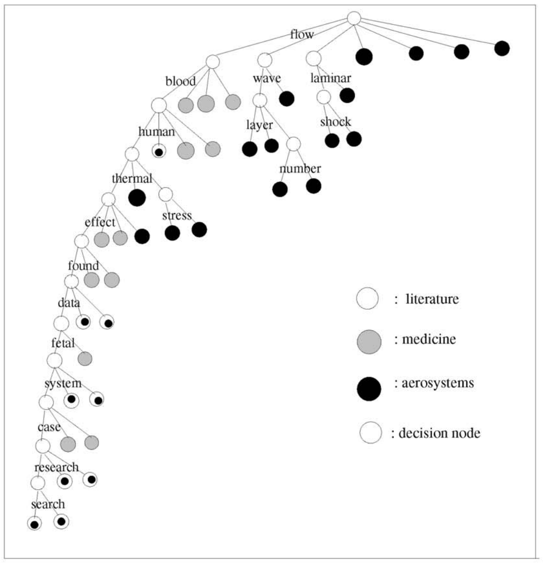
The page block data set has five different classes of page layout, namely, text, horizontal line, picture, vertical line, and graphic, and 10 different attributes, namely, 1) height: height of the block, 2) length: length of the block, 3) area: area of the block (height $\times$ length), 4) eccen: eccentricity of the block (length/height), 5) pblack: percentage of black pixels within the block (blackpix/area), 6) pand: percentage of black pixels after the application of the Run Length Smoothing Algorithm (RLSA) (blackand/area), 7) meantr: mean number of white-black transitions (blackpix/wbtrans), 8) blackpix: total number of black pixels in the original bitmap of the block, 9) blackand: total number of black pixels in the bitmap of the block after the RLSA, and 10) wbtrans: number of white-black transitions in the original bitmap of the block. We observe that the percentage of black pixels is the most important attribute in determining the groups of data records. It is also observed that the attribute length also plays an important role in deciding the characteristics of the page layout.

TABLE 5
Performance of the Proposed Unsupervised Decision Tree with the Traditional Supervised Decision

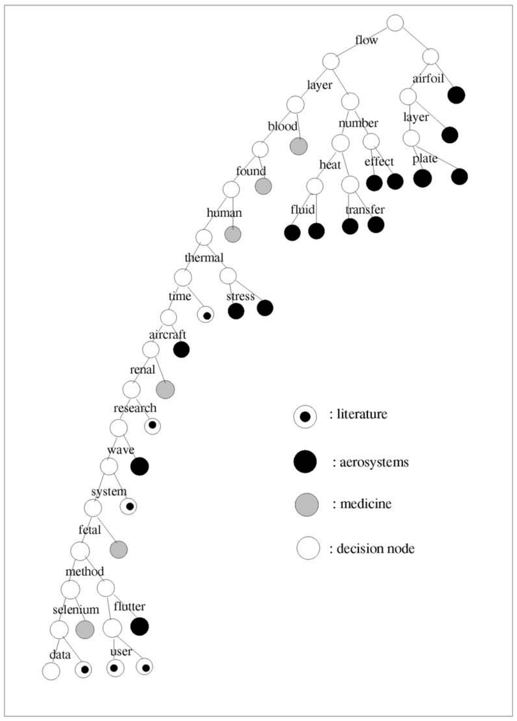
			Data Set			
			Pageblock	Wdbc	Bupa	Spambase
Validation Error (10-fold)	UDT	UDT1	8.24	8.75	40.88	21.06
		UDT2	8.31	7.50	34.41	22.54
		UDT3	8.79	7.32	36.47	23.50
	SDT		3.05	7.56	33.62	7.28
	c-means (# clusters)	8	10.13	10.71	42.05	31.11
		10	9.87	8.75	42.05	30.63
		12	9.71	8.21	42.05	30.30
		14	9.62	8.57	42.05	30.43
		16	9.41	7.50	42.05	30.87
		18	9.21	8.57	42.05	30.59
		20	9.07	8.21	42.05	30.87
	Ward's Algorithm (# clusters)	8	10.04	11.43	41.76	31.41
		10	9.96	10.71	41.76	31.11
		12	9.96	10.18	41.76	30.67
		14	9.96	10.00	41.76	30.69
		16	9.91	10.00	41.76	30.43
		18	9.91	10.00	41.76	30.43
		20	9.89	10.00	41.76	30.48
No. of nodes (average)	UDT1		118.5	32.5	44.8	131.3
	UDT2		297.2	76	48.2	259
	UDT3		297.8	99.4	48.6	272.8
	SDT		87	25	51	207
No. of leaves (average)	UDT1		68.2	19.1	26.5	90.6
	UDT2		149.1	38.5	24.6	130
	UDT3		149.4	50.2	24.8	136.9
	SDT		44	13	26	104

Performance is measured in terms of the 10-fold cross-validation score. UDT1 represents the k-ary unsupervised decision tree where the data set is segmented at each node using the valley detection algorithm. The attributes in UDT1 are selected using Measure III. UDT2 represents the binary unsupervised decision tree with Measure III for attribute selection. UDT3 represents the binary unsupervised decision tree constructed with Measure IV. SDT represents the supervised decision tree (J4.8 [31], [32] which is the same as C4.5 [5], [6] in the Weka software). Performance of the c-means algorithm for different values of c from 8-20 are presented as a comparison. Performance of a hierarchical clustering algorithm (Ward's algorithm) (for values of c from 8 to 10) is also shown here. The size of the hierarchy in Ward's algorithm is decided by the number of clusters.

the attribute selector performs better than that with the TF-IDF measure. From the results in Figs. 5 and 6, we see that the taxonomy with *Measure IV* retrieves meaningful words at the top of the hierarchy. In all these taxonomies, the word "flow" is at the top level. This is a discriminating word between the categories *medicine*, *aerosystems*, and *literature*. In the category *literature*, usually, occurrence of the word "flow" is very low. We mark the three different categories using three different types of nodes in Figs. 5 and 6. One interesting finding is that the documents belonging to *medicine* and *aerosystems* appear at the top of the hierarchy, whereas the documents belonging to *literature* appear at a relatively lower level. This is due to the fact that, without any supervised information, the category *literature* does not exhibit any discriminating set of words as compared to *medicine* and *aerosystems*. For example, the word "blood" is highly discriminating, and it is able to distinguish the *medicine* category from the other two. Similarly, the words "heat" and "thermal" are quite significant in the context of *aerosystems*, and they appear

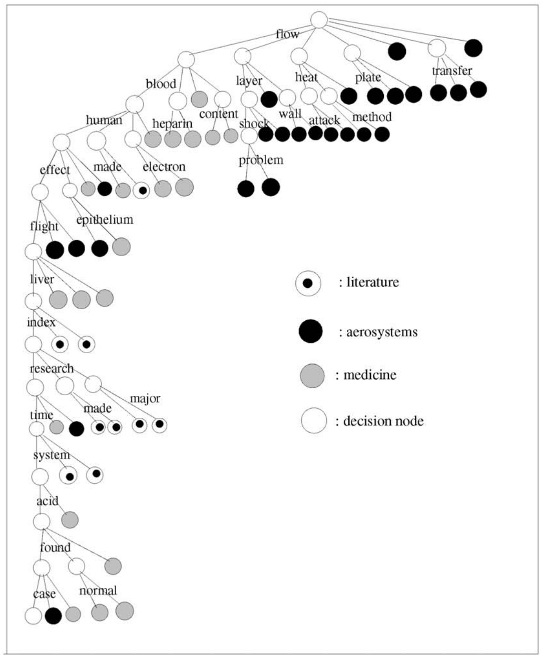


(a)

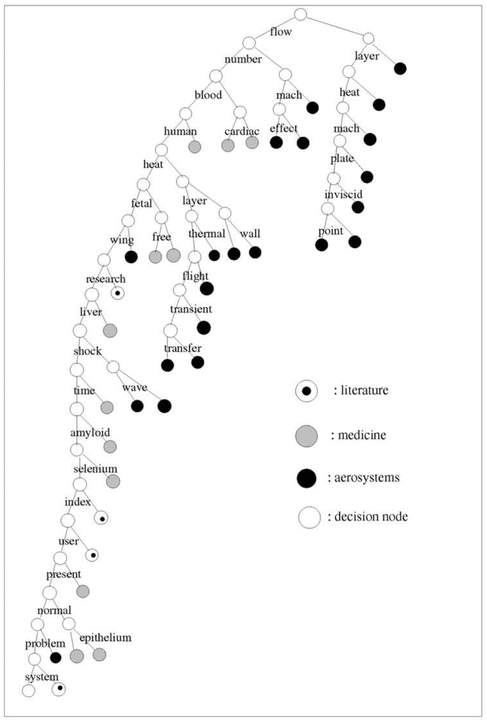


(b)

Fig. 5. (a) The *k*-ary unsupervised decision tree generated with TF-IDF measure as the attribute selector. The leftmost and rightmost branches at each level represent the low and high occurrence of the corresponding word, and the intermediate branches represent various grades of medium occurrence. (b) The binary unsupervised decision tree generated with TF-IDF as the attribute selector. The left and right branches at each level represent low and high occurrence of the corresponding words.



(a)



(b)

Fig. 6. (a) The *k*-ary unsupervised decision tree generated with *Measure IV* as the attribute selector. (b) The binary unsupervised decision tree generated with *Measure IV* as the attribute selector. The leftmost and rightmost branches at every level of both the figures have the same interpretation as in Fig. 5.

TABLE 6
Classification Performance of the Four Different Unsupervised Decision Trees on the *Classic3* Data Set

Type of decision tree	% Classification accuracy
k-ary UDtree with tf-idf	72.59
binary UDtree with tf-idf	78.42
k-ary UDtree with *Measure IV*	82.27
binary UDtree with *Measure IV*	82.02

as discriminating words in the top level. Thus, the unsupervised decision tree can be considered as a potential tool for obtaining the taxonomies without using any supervised information (or where the supervised information is not easily available).

7 Conclusions and Discussion

We have presented an approach to clustering based on constructing an unsupervised decision tree where clusters or groups of data records represented by the leaf nodes can be directly interpreted in terms of rules. We have experimantally validated the quality of the clusters generated by the unsupervised decision trees on different real-life data sets. We have also demonstrated the effectiveness of the unsupervised decision tree in generating a taxonomy from text data.

In constructing an unsupervised tree, we considered only one attribute at a time, i.e., the most important one. As in the case of the supervised decision tree [7], we can consider a combination of the attributes at each node and perform the clustering in higher-dimensional space. However, this would make the results harder to interpret. For example, algorithms such as C4.5 (which considers one attribute at a time) are relatively more popular than oblique decision trees mainly due to "interpretability." We interpret the unsupervised decison tree based on the paths to reach the leaf nodes. The presence of noise in the data can have an adverse effect on the results of supervised decision tree. This is due to the inherent bias in the approach. A judicious pruning of the tree can improve the performance of the algorithm. A similar approach can be used to deal with the noise problem in the unsupervised case. Overall, the proposed method provides a way of mining interpretable clusters from the data, which can be further improved by incorporating external knowledge about the data set.

In the proposed method, we have used a lower bound on the minimum number of data points at a node as the stopping criterion. Various other stopping criteria such as depth of the decision tree and minimum description length can also be used in combination with the required minimum number of data points. The data set partitioning process can also be controlled by defining a level-dependant threshold such that if the reduction in inhomogeneity is greater than the threshold only, then the data set is partitioned. A level-dependant threshold will allow top-level attributes to play a more important role in the data set partitioning process. Finally, we have shown an example of text data clustering where *Measure IV* provides better results in terms of classification score as compared to the TF-IDF measure. In general, the proposed method can be applied to different cases such as automatic image and video data categorization and generation of corresponding descriptions, modeling user interests in various categories and designing catalogs in e-commerce.

Appendix

Theoretical Analysis to Provide Insight into the Proposed Measures

We can view the data as random sample generated from a mixture distribution

$$p(\mathbf{x}) = \sum k_i \exp(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{i})^T \Sigma_i^{-1}(\mathbf{x}-\mu_\mathbf{i}), \tag{24}$$

where k_i are the normalizing constants such that

$$\int p(\mathbf{x})d\mathbf{x} = 1. \tag{25}$$

For the sake of simplicity, we assume that there are only two clusters in the mixture distribution. Further, let the data vectors be projected on some dimension x_j (in the sequel, we omit the subscript j without loss of generality) such that

$$p(x) = k_1 \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) + k_2 \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right). \tag{26}$$

For the sake of further simplicity, let us assume that $\sigma_1 = \sigma_2 = \sigma$ and $k_1 = k_2 = \frac{1}{2\sqrt{2\pi}\sigma}$. The entropy measure can be written as

$$\begin{aligned} H(x) = &-\frac{1}{2\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty}\left(\exp\left(-\frac{-(x-\mu_1)^2}{2\sigma^2}\right)\right.\\ &\left.+\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)\log\left(\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)\right.\\ &\left.+\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)dx. \end{aligned} \tag{27}$$

For $x < \frac{1}{2}(\mu_1+\mu_2)$, $\log(p(x))$ can be simplified as

$$\begin{aligned} \log(p(x)) &= \log\left(\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right) + \exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)\\ &= -\frac{(x-\mu_1)^2}{2\sigma^2} + \exp\Big(-\Big(\frac{\mu_1-\mu_2}{\sigma^2}\Big)\\ &\quad \Big(x - \frac{1}{2}(\mu_1+\mu_2)\Big)\Big). \end{aligned} \tag{28}$$

A similar approximation can be derived for the case when $x > \frac{1}{2}(\mu_1+\mu_2)$. After simplification, the entropy $H(x)$ (see (27)) can be written as:

$$\begin{aligned} H(x) =& \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{(\mu_1+\mu_2)/2} \frac{(x-\mu_1)^2}{2\sigma^2}\left(\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)\right. \\ &\left.+\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)dx \\ &-\frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{(\mu_1+\mu_2)/2}\exp\left(-\frac{\mu_1-\mu_2}{\sigma^2}\left(x-\frac{1}{2}(\mu_1+\mu_2)\right)\right) \\ &\left(\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)+\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)dx. \end{aligned} \tag{29}$$

Evaluating the first integral and after simplification, we obtain

$$\begin{aligned} I_1 =& \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{(\mu_1+\mu_2)/2} \frac{(x-\mu_1)^2}{2\sigma^2}\left(\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)\right. \\ &\left.+\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)dx \\ =& 1+\frac{1}{\sqrt{2\pi}}\left(\frac{\mu_2-\mu_1}{\sigma}\right)\exp\left(-\frac{(\mu_2-\mu_1)^2}{8\sigma^2}\right) \\ &+\frac{(\mu_2-\mu_1)^2}{2\sigma^2}\left(1-erf\left(\frac{\mu_2-\mu_1}{2\sigma}\right)\right). \end{aligned} \tag{30}$$

Similarly, the second part of the integral can be evaluated as

$$\begin{aligned} I_2 =& -\frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{(\mu_1+\mu_2)/2}\exp\left(-\frac{\mu_1-\mu_2}{\sigma^2}\left(x-\frac{1}{2}(\mu_1+\mu_2)\right)\right) \\ &\left(\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)+\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)dx \\ =& 1+\exp\left(\frac{(\mu_1-\mu_2)^2}{\sigma^2}\right). \end{aligned} \tag{31}$$

From (30), it is evident that I_1 increases as $\mu_2-\mu_1$ increases, i.e., the distance between the two clusters in the projected dimension increases. Similarly, from (31), the second integral I_2 increases with $\mu_2-\mu_1$. Thus, the entropy

$$H(x) = I_1(x) + I_2(x) \tag{32}$$

increases with the increase in $\mu_2-\mu_1$. Note that, (30) and (31) indicate an unbounded increase in the integrals, although the entropy function is bounded in $[0,1]$. The error is due to the first order approximation of $\log(p(x))$ in (28). The mathematical analysis only shows a trend of the change in the entropy function as the distance between the centers increases when the centers are in the vicinity of each other. When the centers are further apart, i.e., $(\mu_2-\mu_1)/\sigma$ is large, the entropy is relatively insensitive to the increase in the distance between the centers.

Thus, empirically, *Measure III* can be written as

$$Measure\ III = I_1(x) + I_2(x). \tag{33}$$

In computing *Measure IV*, it can be shown that the cut-off point in the distribution is $(\mu_1+\mu_2)/2$. Again, we have

$$\int_{-\infty}^{(\mu_1+\mu_2)/2} p(x)dx = \int_{(\mu_1+\mu_2)/2}^{\infty} p(x)dx = \frac{1}{2}. \tag{34}$$

Thus, the distribution on the left half of the cut-off point is

$$r_1(x) = \begin{cases} 2p(x) & \text{for } -\infty < x \le (\mu_1+\mu_2)/2 \\ 0 & \text{for } x > (\mu_1+\mu_2)/2. \end{cases} \tag{35}$$

Similarly, the distribution on the right half of the cut-off point is

$$r_2(x) = \begin{cases} 0 & x \le (\mu_1+\mu_2)/2 \\ 2p(x) & \text{for } x > (\mu_1+\mu_2)/2. \end{cases} \tag{36}$$

Thus, *Measure IV* can be computed exactly the same way as *Measure III* and can be written as

$$\begin{aligned} Measure\ IV &= 2\log 2 + 2(I_1+I_2) - (I_1+I_2) \\ &= I_1 + I_2 + 2\log 2. \end{aligned} \tag{37}$$

We observe from the empirical analysis that *Measure III* and *Measure IV* behave exactly in the same way except for a constant factor which does not affect the relative importance measures of the attributes.

In empirically computing *Measure II* from the same distribution, first we need to derive the distribution of the interpoint distances, and then need to obtain the entropy measure for this derived distribution. Let X_1 and X_2 represent two independent random variables drawn from the same distribution $p(x)$, such that $Z = |X_2 - X_1|$ represents the distance between the two points. Then,

$$P(z) = Prob(0 \le Z \le z) = Prob(-z \le (X_2 - X_1) \le z) \tag{38}$$

represents the cumulative distribution of the distance between two points from the same distribution. Therefore, $P(z)$ can be written as

$$P(z) = \int_{-\infty}^{\infty} p(x_2)\left(\int_{x_2-z}^{x_2+z} p(x_1)dx_1\right)dx_2, \tag{39}$$

which after simplification becomes

$$\begin{aligned} P(z) =& \frac{1}{2\sqrt{2\pi}\sigma}\sum_{i,j}\kappa_i\kappa_j\int_{-\infty}^{\infty}\left[erf\left(\frac{x-\mu_i+z}{\sqrt{2}\sigma}\right)\right. \\ &\left.-erf\left(\frac{x-\mu_i-z}{\sqrt{2}\sigma}\right)\right]\exp\left(-\frac{-(x-\mu_j)^2}{2\sigma^2}\right)dx, \end{aligned} \tag{40}$$

where μ_i, μ_j are the means of the kernels and κ_i and κ_j are the weights of the respective kernels in the distribution. The density function for z can be derived as

$$p(z) = \frac{\partial P(z)}{\partial z}. \tag{41}$$

Note that $P(z)$ is not differentiable at $z = 0$, however, only the directional derivative exists at $z = 0$. Algebraically, it can be shown that $p(z)$ has the form

$$\begin{aligned} p(z) =& K\left[2\exp\left(-\frac{z^2}{2\sigma^2}\right)+\exp\left(-\frac{(z-(\mu_1-\mu_2))^2}{2\sigma^2}\right)\right. \\ &\left.+\exp\left(-\frac{(z-(\mu_2-\mu_1))^2}{2\sigma^2}\right)\right], \end{aligned} \tag{42}$$

where K is a normalizing constant and $z \geq 0$. The density function $p(z)$ is again a Gaussian mixture in nature, having modes at $z = 0$ and $z = \mu_2 - \mu_1$. Note that $\mu_1 - \mu_2$ does not represent a mode in the distribution since $\mu_1 - \mu_2 < 0$. Analogous to the previous analysis, it is observed that the entropy $H(z)$ increases as $|\mu_1 - \mu_2|$ increases. Also, looking at the expression for $p(z)$, it is observed that the mode at $z = 0$ is higher than that at $\mu_2 - \mu_1$. Thus, *Measure II* becomes less sensitive when compared with *Measure III* and *Measure IV* when $\mu_2 - \mu_1$ is relatively small.

REFERENCES

[1] R. Duda, P. Hart, and D. Stork, *Pattern Classification,* second ed. New York: Wiley, 2001.
[2] J. Durkin, "Induction via ID3," *AI Expert,* vol. 7, pp. 48-53, 1992.
[3] U.M. Fayyad and K.B. Irani, "On the Handling of Continuous-values Attributes in Decision Tree Generation," *Machine Learning,* vol. 8, pp. 87-102, 1992.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* New York: Chapman & Hall, 1993.
[5] J.R. Quinlan, *Programs for Machine Learning.* San Fransisco: Morgan Kaufmann, 1993.
[6] J.R. Quinlan, "Improved Use of Continuous Attributes in C4.5," *J. Artificial Intelligence,* vol. 4, pp. 77-90, 1996.
[7] C.E. Brodley and P.E. Utgoff, "Multivariate Decision Trees," *Machine Learning,* vol. 19, pp. 45-77, 1995.
[8] Y. Yang and J.O. Pedersen, "A Comparatative Study on Feature Selection in Text Categorization," *Proc. 14th Int'l Conf. Machine Learning (ICML '97),* pp. 412-420, 1997.
[9] M. Riley, "Some Applications of Tree Based Modeling to Speech and Language Indexing," *Proc. DARPA Speech and Natural Language Workshop,* pp. 339-352, 1989.
[10] J. Chien, C. Huang, and S. Chen, "Compact Decision Trees with Cluster Validity for Speech Recognition," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* pp. 873-876, 2002.
[11] S. Salzberg, A.L. Delcher, K.H. Fasman, and J. Henderson, "A Decision Tree System for Finding Genes in DNA," *J. Computational Biology,* vol. 5, pp. 667-680, 1998.
[12] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," *Research and Development in Information Retrieval,* pp. 46-54, 1998.
[13] Y.H. Cho, J.K. Kim, and S.H. Kim, "A Personalized Recommender System Based on Web Usage Mining and Decision Tree Induction," *Expert Systems with Applications,* vol. 23, pp. 329-342, 2002.
[14] M. Held and J.M. Buhmann, "Unsupervised On-Line Learning of Decision Trees for Hierarchical Data Analysis," *Proc. Advances of the Neural Information Processing Systems (NIPS),* 1997.
[15] P. Bellot and M. El-Beze, "Clustering by Means of Unsupervised Decision Trees or Hierarchical and k-Means Like Algorithms," *Proc. RIAO 2000 Conf.,* pp. 344-363, 2000.
[16] B. Liu, Y. Xia, and P. Yu, "Clustering through Decision Tree Construction," Technical Report RC 21695, IBM Research Report, IBM, 2000.
[17] B. Liu, Y. Xia, and P.S. Yu, "Clustering through Decision Tree Construction," *Proc. Conf. Information and Knowledge Management (CIKM),* pp. 20-29, 2000.
[18] D. Boley, "Hierarchical Taxonomies Using Divisive Partitioning," Technical Report TR-98-012, Dept. of Computer Science, Univ. of Minnesota, Minneapolis, 1998.
[19] S.M. Savaresi, D.L. Boley, S. Bittanti, and G. Gazzaniga, "Choosing the Cluster to Split in Bisecting Divisive Clustering Algorithms," CSE Report TR 00-055, Univ. of Minnesota, 2000.
[20] M.H. Law, A.K. Jain, and M.A.T. Figueiredo, "Feature Selection in Mixture-Based Clustering," *Proc. Advances in Neural Information Processing,* vol. 15, 2003.
[21] F. Markowetz and A. v. Heyderbreck, "Class Discovery in Gene Expression Data: Characterizing Splits by Support Vector Machines," *Proc. 26th Ann. Conf. Gesellschaft für Klassifikation,* pp. 1-8, 2002.
[22] A. v. Heydebreck, W. Huber, A. Poustka, and M. Vingron, "Identifying Splits with Clear Separation: A New Class Discovery Method for Gene Expression Data," *Bioinformatics,* vol. 17, pp. 107-114, 2001.
[23] A. Ben-Dor, N. Friedman, and Z. Yakhini, "Class Discovery in Gene Expression Data," *Proc. Eighth Ann. Int'l Conf. Research in Computational Molecular Biology,* pp. 31-38, 2001.
[24] A. Hinneburg and D.A. Keim, "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering," *Proc. 25th Very Large Data Bases Conf.,* pp. 506-517, 1999.
[25] H.H. Bock, "Probabilistic Aspects in Cluster Analysis," *Conceptual and Numerical Analysis of Data,* O. Opitz, ed., pp. 12-44, Heidelberg: Springer-Verlag, 1989.
[26] H.H. Bock, "Information and Entropy in Cluster Analysis," *Proc. First US/Japan Conf. Frontiers of Statistical Modeling,* 1994.
[27] F.B. Baulieu, "A Classification of Presence/Absence Based Dissimilarity Coefficients," *J. Classification,* vol. 6, pp. 233-246, 1989.
[28] J. Basak, R.K. De, and S.K. Pal, "Unsupervised Feature Selection Using Neuro-Fuzzy Approach," *Pattern Recognition Letters,* vol. 19, pp. 997-1006, 1998.
[29] http://www.ics.uci.edu/mlearn/MLRepository.html, 2003.
[30] A. El-Hamdouchi and P. Willett, "Hierarchical Document Clustering Using Ward's Method," *Proc. Ninth Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 149-156, 1986.
[31] S. Garner, "Weka: The Waikato Environment for Knowledge Analysis," *Proc. New Zealand Computer Science Research Students Conf.,* pp. 57-64, 1995.
[32] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations,* chapter 8, Morgan Kaufmann, 2000.
[33] R. Krishnapuram and K. Kummamuru, "Automatic Taxonomy Generation: Issues and Possibilities," *Proc. Fuzzy Sets and Systems (IFSA '03),* pp. 52-63, 2003.
[34] ftp://ftp.cs.cornell.edu/pub/smart, 2004.
[35] G. Salton, *Cluster Search Strategies and the Optimization of Retrieval Effectiveness.* Englewood Cliffs, N.J.: Prentice Hall, 1971.

Jayanta Basak received the ME (computer science and engineering) degree from the Indian Institute of Science (IISc), Bangalore, and the PhD degree from the Indian Statistical Institute (ISI), Calcutta, in 1989 and 1995, respectively. He served as a computer engineer on the KBCS project of ISI from 1989 to 1992, and as a faculty member of the same institute from 1992 to 2000. He was a visiting scientist at the Robotics Institute at Carnegie Mellon University in 1991-1992 and a researcher in the RIKEN Brain Science Institute, Japan, in 1997-1998. Since September 2000, he has been a research staff member at IBM India Research Lab, Delhi. He has received young scientist awards from the Indian Science Congress Association (ISCA), the Indian National Science Academy (INSA), the International Neural Networks Society (INNS), USA, in 1994, 1996, and 2000, respectively. His research interests include neural networks and soft computing, pattern recognition, and vision and image analysis.

Raghu Krishnapuram received the PhD degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, in 1987. From 1987 to 1997, he was on the faculty of the Department of Computer Engineering and Computer Science at the University of Missouri, Columbia. From 1997 to 2000, Dr. Krishnapuram was a full professor in the Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado. Since then, he has been at at IBM India Research Lab, New Delhi. Dr. Krishnapuram's research encompasses many aspects of fuzzy set theory, neural networks, pattern recognition, computer vision, image processing, text mining, and e-commerce. He has published more than 150 papers in journals and conferences in these areas. Dr. Krishnapuram is an associate editor of the *IEEE Transactions on Fuzzy Systems* and he is a coauthor (with J. Bezdek, J. Keller, and N. Pal) of the book *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*.