Classifiability-Based Omnivariate Decision Trees

Yuanhong Li, *Student Member, IEEE*, Ming Dong, *Member, IEEE*, and Ravi Kothari, *Senior Member, IEEE*

***Abstract*—Top-down induction of decision trees is a simple and powerful method of pattern classification. In a decision tree, each node partitions the available patterns into two or more sets. New nodes are created to handle each of the resulting partitions and the process continues. A node is considered terminal if it satisfies some stopping criteria (for example, purity, i.e., all patterns at the node are from a single class). Decision trees may be univariate, linear multivariate, or nonlinear multivariate depending on whether a single attribute, a linear function of all the attributes, or a nonlinear function of all the attributes is used for the partitioning at each node of the decision tree. Though nonlinear multivariate decision trees are the most powerful, they are more susceptible to the risks of overfitting.**

In this paper, we propose to perform model selection at each decision node to build omnivariate decision trees. The model selection is done using a novel classifiability measure that captures the possible sources of misclassification with relative ease and is able to accurately reflect the complexity of the subproblem at each node. The proposed approach is fast and does not suffer from as high a computational burden as that incurred by typical model selection algorithms. Empirical results over 26 data sets indicate that our approach is faster and achieves better classification accuracy compared to statistical model select algorithms.

***Index Terms*—Bayes error, data complexity, data density, decision boundary, omnivariate decision trees.**

I. Introduction

DECISION trees implement a top-down divide-and-conquer approach to supervised classification [1]–[5]. The construction of decision trees begins by assigning all training examples to the root node. These training examples are partitioned and assigned to child nodes so as to increase the *purity* of the resulting child nodes. The procedure is repeated at each node until the leaf nodes have training examples of a single class, i.e., the leaf nodes are *pure*. Assuming that the input has d attributes and is represented as $x = [x_1, x_2, \dots, x_d]^T$, each internal node m of the decision tree implements a decision $f_m(x) > 0$, and each leaf node carries one class label t. Based on the characteristics of $f_m(x)$, decision trees can be grouped into four categories: univariate decision trees, linear multivariate decision trees, nonlinear decision trees, and omnivariate decision trees.

1) *Univariate Decision Trees:* The internal node uses only one attribute x_j of x to make a decision. If the attribute is numeric, the decision is of the form

$$f_m(x) : x_j + b_m > 0 \tag{1}$$

where b_m is a constant. This defines a hyperplane $x_j = b_m$ orthogonal to the axis x_j and partitions the input space into two parts.

If x_j is nominal with L possible values $a_1, a_2, \dots, a_L$, the decision is of the form

$$f_m(x) : x_j = a_i, \quad i = 1, 2, \dots, L. \tag{2}$$

This leads to L branches from the node.

Each univariate decision node has a single parameter though univariate decision trees often end up having a large number of nodes. ID3 [1], C4.5 [6], and neural networks [4] are some well-known methods for building univariate decision trees.

2) *Linear Multivariate Decision Trees:* The decision at each internal node is a linear combination of all attributes

$$f_m(x) : \sum_{j=1}^{d} w_{mj} x_j + b_m > 0. \tag{3}$$

This decision generates an arbitrary hyperplane that is more discriminating than one that is orthogonal to any particular axis. A linear multivariate node has $(d+1)$ parameters and is more complex than an univariate one. Such decision trees were first introduced with the CART system of Breiman *et al.* [2]. OC1 [7], LMDT [8], and FACT [9] are other popular algorithms to build linear multivariate decision trees.

3) *Nonlinear multivariate decision trees:* In the more general case, the decision can be a weighted sum of H nonlinear basis functions $\Phi_{mh}(x)$

$$f_m(x) : \sum_{h=0}^{H} w_{mh} \Phi_{mh}(x) + b_m > 0. \tag{4}$$

Nonlinear decision nodes can generate an arbitrarily complex decision boundary and provide the strongest discriminant power. However, each nonlinear node has $(H(d+1)+1)$ parameters and can be easily influenced by noise in the data. QDA [10] and neural networks [3] are some methods for generating such decision trees.

The difference between univariate, linear, and nonlinear decision nodes is shown through an illustrative example in Fig. 1. Note that there is an equivalence that can be established between decision trees and neural networks [11], [12]; thus decision trees could also be efficiently implemented in hardware [11].

4) *Omnivariate decision trees:* Typically, all nodes in a decision tree are univariate, linear multivariate, or nonlinear multivariate. Implicit in such decision trees is the assumption that the complexity of the decision to be made at each node is the same. Such an assumption is usually incorrect. Yıldız [13] proposed a new decision tree architecture called *omnivariate*

Manuscript received July 1, 2003; revised March 27, 2005. This work was supported in part by Wayne State University under a Faculty Research Award.

Y. Li and M. Dong are with the Machine Vision and Pattern Recognition Laboratory, Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: mdong@cs.wayne.edu).

R. Kothari is with the IBM-India Research Lab., New Delhi 110016, India.

Digital Object Identifier 10.1109/TNN.2005.852864

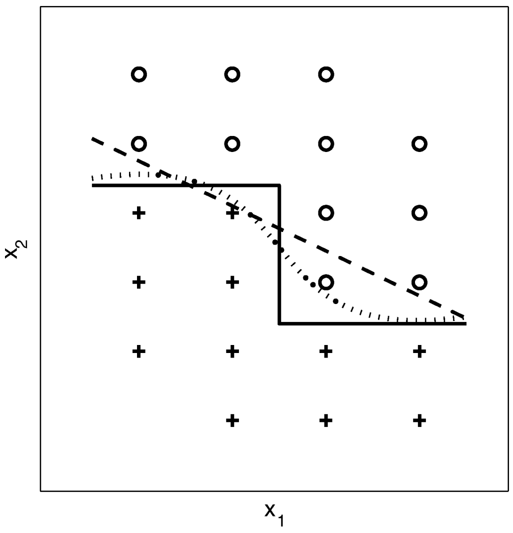

Fig. 1. Example of univariate (solid line), linear multivariate (dashed line), and nonlinear multivariate (dotted line) splits that separate instances of two classes.

decision trees, which is a hybrid of the three decision models mentioned above. Such trees utilize a combination of univariate, linear multivariate, or nonlinear multivariate decision nodes with the choice being made on the basis of some statistical model selection criteria.

One reason for looking at different types of internal nodes is to obtain better overall generalization. Within the same family of decision trees, one may prefer a tree with smaller depth compared to one with a larger depth. However, it is not very meaningful to compare the depths of trees of different families (one may replace the whole tree with a single node of arbitrarily large complexity). In general, however, constructing the smallest decision tree for a given data set is NP-hard [14]. Decision trees are based on a greedy search, i.e., a locally optimal decision is implemented at each node. For omnivariate decision trees, the most critical step is the model selection strategy at each node. As long as the appropriate split model (univariate, linear, or nonlinear) is selected, the node can then be partitioned by a mature algorithm corresponding to that model.

Statistical tests, such as the 5 $\times$ 2 cross-validation (cv) F test [15], can be used for model selection. Yıldız [13] proposed a methodology using 5 $\times$ 2 cv F test to choose the winner from three models. All three split models are induced independently at each node. If a 5 $\times$ 2 cv F test shows that a complex model is significantly better than a simple one, it will be chosen; otherwise the simple model will be chosen. For example, if both linear and nonlinear models are superior to univariate model but there is no significant difference between the linear and nonlinear model, then the linear model will be chosen. Experiments show that such a decision tree induction method generalizes better than trees with the same types of nodes everywhere, and induces smaller trees. However, its drawback is the long training time resulting from inducing all the considered models at each node. This computational burden makes the approach impractical for large data sets. Other measures, i.e., structure risk [16] and nearest neighbor rule-based implementation of structure risk [17], can also be employed for model selection in construction of omnivariate decision trees.

In this paper, we propose to perform the model selection at each node based on a novel classifiability measure when building omnivariate decision trees. The classifiability measure captures the possible sources of misclassification with relative ease and is able to accurately reflect the complexity of the subproblem at each node. The proposed approach does not require time-consuming statistic model tests at each node and therefore does not suffer from as high a computational burden as typical model selection algorithms.

The remainder of the paper is organized as follows. Section II introduces the proposed classifiability measure and Section III discusses how to handle nominal attributes and missing values. Section IV addresses the relationship between the classifiability measure and three possible sources of misclassification: Bayes error, decision boundary complexity, and data sparsity. In Section V, we discuss how to choose the best split model based on the classifiability measure. In Section VI, the proposed classifiability-based omnivariate decision trees are evaluated based on 26 public data sets that are available from the UCI repository [18]. Conclusions are presented in Section VII.

II. A Classifiability Measure

When a d-dimensional classification problem is visualized in $(d+1)$ dimensions using the class label as the $(d+1)$th dimension, the class label may be viewed as defining a surface. Fig. 2 shows, for example, a two-dimensional classification problem and the corresponding visualization in three dimensions. The class label surface is rough in regions where classes are interlaced and smooth in regions where classes are noninterlaced. The smoothness of the class label surface thus provides an intuitive feel of the classifiability of data [19], [20].

The roughness of the class label surface can be captured by a co-occurrence matrix [21], [22], which provides the joint probability of a class occurring within a neighborhood of another class. Specifically, let N be the total number of patterns and c be the total number of classes denoted by $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$. Let y denote a pattern within the neighborhood of pattern z, i.e., $\|z-y\| \leq r$ (r is the neighborhood size). The co-occurence matrix W is a square matrix $(c \times c)$ with an element W_{ik} being

$$W_{ik} = \sum_y P(y)P(\omega_i|y, \omega_k|z) \tag{5}$$

where $P(y)$ is the probability of occurrence of y and the summation of y is over the neighborhood of pattern z. Since y and z are independent, (5) can be simplified to

$$W_{ik} = \sum_y P(y)P(\omega_i|y)P(\omega_k|z) \tag{6}$$

where $P(\omega_i|y)$ and $P(\omega_k|z)$ are the posterior probability.

We define the primary classifiability measure in the neighborhood of pattern z as the summation of the diagonal elements of co-occurrence matrix W.

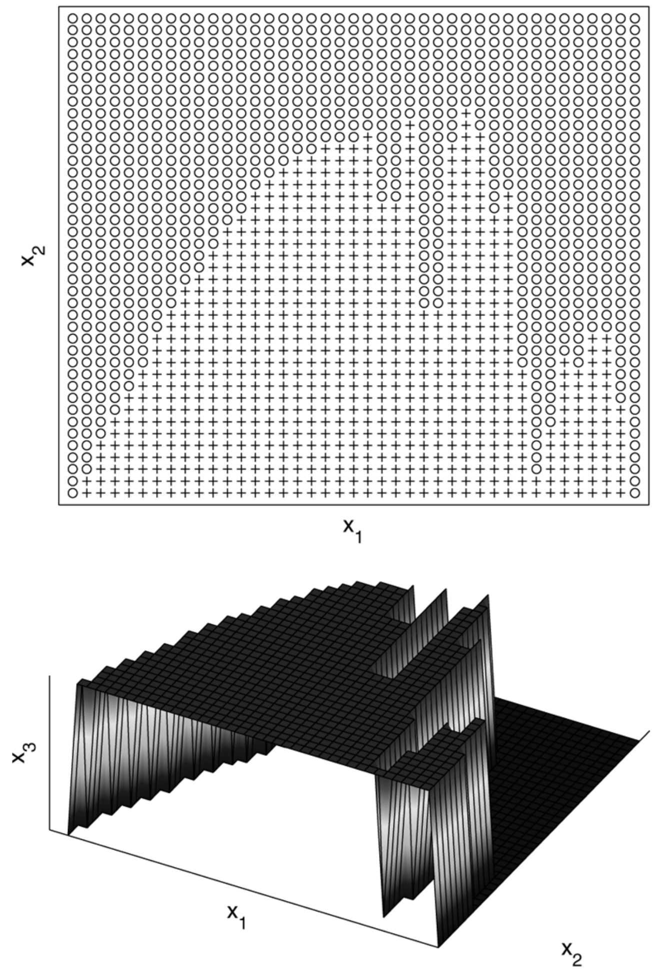

Fig. 2. A two-class classification problem (top panel) and the visualization in three dimensions (bottom panel).

Definition 1: The primary classifiability measure for patterns distributed in the neighborhood of a pattern z is defined by

$$C(z) = \sum_{i=1}^{c} \sum_{y} P(y)P(\omega_i|y)P(\omega_i|z). \tag{7}$$

Based on (7), the overall primary classifiability can be obtained by integrating the local classifiability.

Definition 2: The primary classifiability measure L_0 for the entire data is defined by

$$L_0 = \sum_{z} C(z). \tag{8}$$

Operationally, the classifiability can be computed as follows:

- Define the neighborhood size r. r should be large enough such that each instance has a few instances in its neighborhood. r also should be small enough to keep the calculation of co-occurrence matrix local (see next step).
- For each pattern $x^{(i)}$, obtain a co-occurrence matrix $W(x^{(i)})$ of size $c \times c$. Elements of $W(x^{(i)})$, $W(x^{(i)})_{jk}$ are the total number of patterns of class ω_k that occur within a circular neighborhood of radius r of an instance of class ω_j, i.e.,

$$W\left(x^{(i)}\right)_{jk} = \sum_{m=1}^{N_{\omega_k}} f\left(x^{(i)}, x^{(m)}\right) \tag{9}$$

where $x^{(m)}$ denotes a pattern of class ω_k, $f(\cdot)$ is an indicator function that is one if $\|x^{(i)} - x^{(m)}\| \leq r$ and $x^{(i)}$ belongs to ω_j, and N_{ω_k} is the number of patterns from class ω_k.

- The co-occurrence matrix of entire data can be computed by summing all the individual co-occurrence matrices together, i.e.,

$$A = \sum_{i=1}^{N} W\left(x^{(i)}\right). \tag{10}$$

One can normalize A such that the sum of its elements a_{ij} is one.

- We thus obtain classifiability measure $\mathcal{L}$ as

$$\mathcal{L} = \alpha L_0 \tag{11}$$

where L_0 is primary classifiability measure and can be computed as follows:

$$L_0 = \sum_{j=1}^{c} a_{jj} \tag{12}$$

and α, the data sparsity factor, is of the form

$$\alpha = f(N, d, c) \tag{13}$$

where d is the dimensionality of the input space, c is the number of classes, and N is the number of examples. Our motivation for introducing α is to incorporate the complexity caused by the data density. In this paper, we propose to use this extended version of classifiability measure $\mathcal{L}$ as the model selection criterion when constructing omnivariate decision trees. The sparsity factor α is discussed in greater detail later in this paper.

III. Handling Nominal Attributes and Missing Values

The classifiability measure proposed above depends on the Euclidean distance between instances (9), which requires the calculation of the differences between two instances on each and every attribute. It is straightforward to do so on a numerical attribute. For a nominal attribute, the difference between two instances is set at zero if and only if the two instances have the same nonmissing value in that attribute; otherwise the difference is set at one.

Most data sets encountered in practice contain missing values. Different machine learning schemes may deal with it in different ways [23]. For example, the instance with missing attributes might be simply discarded; the missing attributes may be replaced by estimates, and so on. When we compute the classifiability measure, we adopt some common methods of handling missing values for the calculation of the Euclidean distance of two instances [23].

- For nominal attributes, assume that a missing feature is maximally different from any other feature value.

Thus if either (or both) of the values is/are missing, or if the values are different, the difference between them is taken as one; the difference is zero only if they are not missing and both are the same.

- For numeric attributes, the difference between two missing values is also taken as one. However, if just one value is missing, the difference is taken as either the (normalized) magnitude of the available attribute or one minus that size, whichever is larger. This means that if values are missing, the difference is as large as it can possibly be.

Once we obtain the Euclidean distances, the classifiability measure can be calculated normally and is able to describe the complexity of data sets with missing values.

IV. Relationship Between Classifiability Measure and Data Complexity

To sufficiently describe the classifiability of data, the proposed measure should be able to address the possible sources of classification errors, for example, class ambiguity, decision boundary complexity, and data density [24]. In the following, we show that the classifiability measure is strongly related to Bayes error and decision boundary complexity. It also has ability to incorporate data complexity introduced by the data density.

A. Classifiability Measure Versus Bayes Error

Bayes error is the lowest achievable classification error for a given data distribution. Certain problems are known to have nonzero Bayes error, i.e., the classes are ambiguous either intrinsically or due to inadequate feature measurements. The following relationship holds between the classifiability measure and Bayes error rate:

Theorem 1: For a given data distribution, the sum of the primary classifiability measure L_0 and the Bayes error E is a constant.

The proof is attached in Appendix I. To avoid confusion, in the rest of this paper we will call $\mathcal{L}$ the classifiability measure instead of the practical classifiability measure.

B. Classifiability Measure Versus Decision Boundary Complexity

The complexity of a decision boundary is another important factor that affects the complexity of a classification problem. Several measures have been proposed in the literature to describe the decision boundary complexity. Some intend to describe the geometry of the decision boundary spanned by each class. These measures include various estimators of intrinsic dimensionality of the data set [25], [26]. Others attempt to describe the shape of the decision boundary, the existence of isolated subdecision boundary, or variation in the point densities within the decision boundary [27], [28]. With a complete sample, the class boundary can also be characterized by its Kolmogorov complexity [29], [30] or the minimum length of a computer program needed to reproduce it [31], [32]. In the following, we choose the geometrical size of the decision boundary as the boundary complexity measure and show its relation with the classifiability measure. The following theorem holds between the classifiability measure and the decision boundary length for a given data distribution:

Theorem 2: The classifiability measure $\mathcal{L}$ and the decision boundary length S satisfy the following relationship:

$$\frac{1}{\alpha}\mathcal{L} + kS = 1 \tag{14}$$

where α is the sparsity factor that we introduce in the next section and k is a constant.

The proof is attached in Appendix II.

C. Classifiability Measure Versus Data Density

In the discussions above, we assume that there are a sufficient number of instances to compute the classifiability measure. This is not always true; for example, a decision node near the leaf of a decision tree may only have few instances.

Usually a complex classifier has high risk of overfitting when applied to sparse data set. Therefore, data density should be incorporated in the data complexity descriptor. A measure of the average number of samples per dimension is proposed in [24] to represent the data complexity of a two-class problem. For multiclass problems, the number of classes also affects the sparsity of the data. For instance, consider two one-dimension classification problems $P1$ and $P2$ with ten samples each. Assume $P1$ contains two classes while $P2$ has ten classes. It is obvious that $P2$ is much sparser than $P1$. Based on all these considerations, we introduce a sparsity factor α

$$\alpha = 1 + \frac{d+c-1}{N} \tag{15}$$

where d, c, and n represent the number of attributes (inputs), the number of classes, and the number of training patterns, respectively.

The classifiability measure can describe the data complexity more accurately by introducing density factor α, especially when only a few training patterns are available. For example, when $N \gg (d+c)$, one obtains that $\alpha \simeq 1$, and $\mathcal{L} \simeq L_0$ means L_0 is sufficient to describe the complexity of the data set. On the other hand, if $N \approx (d+c-1)$ or $N \ll (d+c-1)$, the classifiability measure $\mathcal{L}$ is considerably amplified by α and the sparse data set is then considered less complicated.

V. Model Selection Based on Classifiability Measure

The decision nodes of an omnivariate tree are univariate, linear, or nonlinear classifiers. The most critical step in constructing omnivariate decision tree is selecting the split model at each node. Univariate classifiers should be chosen if the training data set is simply partitioned; linear classifiers should be chosen if the partitioning is more complex; and nonlinear classifiers for the most complex partitioning. Since the proposed classifiability measure captures the complexity of the data, it can be used for selecting the classifier to used at each decision node of a omnivariate decision trees.

Consider a decision node with enough training instances. A small $\mathcal{L}$ (rough class label surface) indicates that the data are better classified by a nonlinear multivariate model, and a large $\mathcal{L}$ indicates that a linear multivariate model or a univariate model is preferred. Since a rotation is the only difference between a

TABLE I
DESCRIPTION OF THE DATA SETS. d IS THE NUMBER OF ATTRIBUTES. d_{nu} AND d_{no} IS THE NUMBER OF NUMERIC AND NOMINAL ATTRIBUTES, RESPECTIVELY. c IS THE NUMBER OF CLASSES AND N IS THE SIZE OF DATA SET

Data set	Title	d	d_{nu}	d_{no}	c	N	Missing
ANNE	Annealing Database	38	0	38	6	798	Yes
AUTO	Automobile Database	25	15	10	5	205	Yes
BALS	Balance Scale Database	4	4	0	3	625	No
BREC	Breast cancer data [35]	9	0	9	2	286	Yes
BREW	Wisconsin Breast Cancer Databases	9	9	0	2	699	Yes
CAR	Car Evaluation Database	6	0	6	4	1728	No
CMC	Contraceptive Method Choice	9	2	7	3	1473	No
COLI	Horse Colic Database	22	7	15	2	368	Yes
CREG	German Credit Database	20	7	13	2	1000	Yes
DIAB	Diabetes Data	8	8	0	2	768	No
GLAS	Glass Identification Database	9	9	0	4	214	No
HEAC	Cleveland Heart Disease Databases	13	6	7	5	303	Yes
HEAH	Hungary Heart Disease Databases	13	6	7	5	294	Yes
HEAS	Heart Disease of Statlog Project Databases	13	13	0	2	270	No
HEPA	Hepatitis Database	19	6	13	2	155	Yes
IONO	Ionosphere Database	34	34	0	2	351	No
IRIS	Iris Plant Database	4	4	0	3	150	No
LABO	Labor relations Database	16	8	8	2	57	Yes
LYMP	Lymphography Domain [36]	18	3	15	4	148	No
PIMA	Pima Indians Diabetes Database [37]	8	8	0	2	768	No
PRIM	Primary Tumor Domain	17	0	17	22	339	Yes
TIC	Tic-Tac-Toe Endgame Database	9	0	9	2	958	No
VEHI	vehicle silhouettes [38]	18	18	0	4	946	No
VOTE	1984 United States Congressional Voting Records Database	16	0	16	2	435	Yes
WINE	Wine Recognition Database	13	13	0	3	178	No
ZOO	Zoo Database	18	1	17	7	101	No

univariate frontier and a linear frontier, and the proposed classifiability measure is rotation-invariant, an additional criterion is required to choose from a linear model and a univariate one. Given a decision node with large $\mathcal{L}$, a univariate split is suitable if there exists at least one attribute with sufficient discrimination power. Otherwise, a linear model is preferred. In this study, we choose the gain ratio [33] as such a measure to determine the discriminant ability of an attribute. Therefore, the complete model selection algorithm is described as follows:

$$f_m \quad : \quad \begin{cases} \text{nonlinear}, & \text{if } \mathcal{L} < \mathcal{L}_n, \\ \text{linear} & \text{if } \mathcal{L} \geq \mathcal{L}_n \text{ and } \mathcal{G} < \mathcal{L}_u \\ \text{univariate}, & \text{if } \mathcal{L} \geq \mathcal{L}_n \text{ and } \mathcal{G} \geq \mathcal{L}_u \end{cases} \tag{16}$$

where $\mathcal{L}_n$ and $\mathcal{L}_u$ are called nonlinear threshold and univariate threshold, respectively, and $\mathcal{G}$ is the highest gain ratio over the attributes with sufficient information gain.

An ideal algorithm for constructing omnivariate tree should select nodes that are as simple as possible and produce a tree that is as small as possible. In practice, an omnivariate splitting algorithm will attempt to find a balance between those expectations. Currently, there is no theoretical method to identify the two thresholds $\mathcal{L}_u$ and $\mathcal{L}_n$. Based on more than two dozen typical data sets, we empirically find an optimal pair of thresholds through a search procedure. This pair of thresholds could serve as a general guide for future experiments. Our approach is as follows.

1) For each data set, train and test (using ten-fold cross-validation) omnivariate trees with different pairs of $\mathcal{L}_u$ and $\mathcal{L}_n$. Compute the tree size and testing accuracy and normalize them over all selected pairs of thresholds.
2) Compute the average measure over all the data sets on each pair of thresholds.
3) Choose the threshold pair that produces trees with small size and high testing accuracy as the optimal thresholds.

Tree size can be represented by the number of tree nodes or the number of total free parameters. Univariate decision trees usually contain a large number of decision nodes, while nonlinear trees contain a small number of complicated decision nodes. For an omnivariate decision tree, the number of internal nodes and the number of free parameters can be adjusted by combining different split models. Typically, choosing simple classifiers as much as possible is conflicting with generating a small decision tree in terms of tree nodes. On the other hand, choosing complicated classifiers may result in a large number of free parameters. The overall performance of a decision tree on a particular pair of $\mathcal{L}_u$ and $\mathcal{L}_n$ can be computed as follows:

$$\text{performance} = \text{testingAccuracy} - \text{nodes} - \text{parameters} \tag{17}$$

where testing accuracy, nodes, and parameters are normalized to lie in [0, 1]. The optimal pair of thresholds should have maximum performance over all data sets evaluated.

VI. EXPERIMENTAL RESULTS

The performance of the proposed model selection algorithm is evaluated on 26 data sets from the UCI repository [18]. Table I summarizes these data sets.

The performance of the proposed algorithm is evaluated in terms of the training time, classification accuracy, and tree size. For comparison, the performance of pure univariate, pure linear, pure nonlinear, and 5 × 2 cv F test-based omnivariate decision trees are also reported.

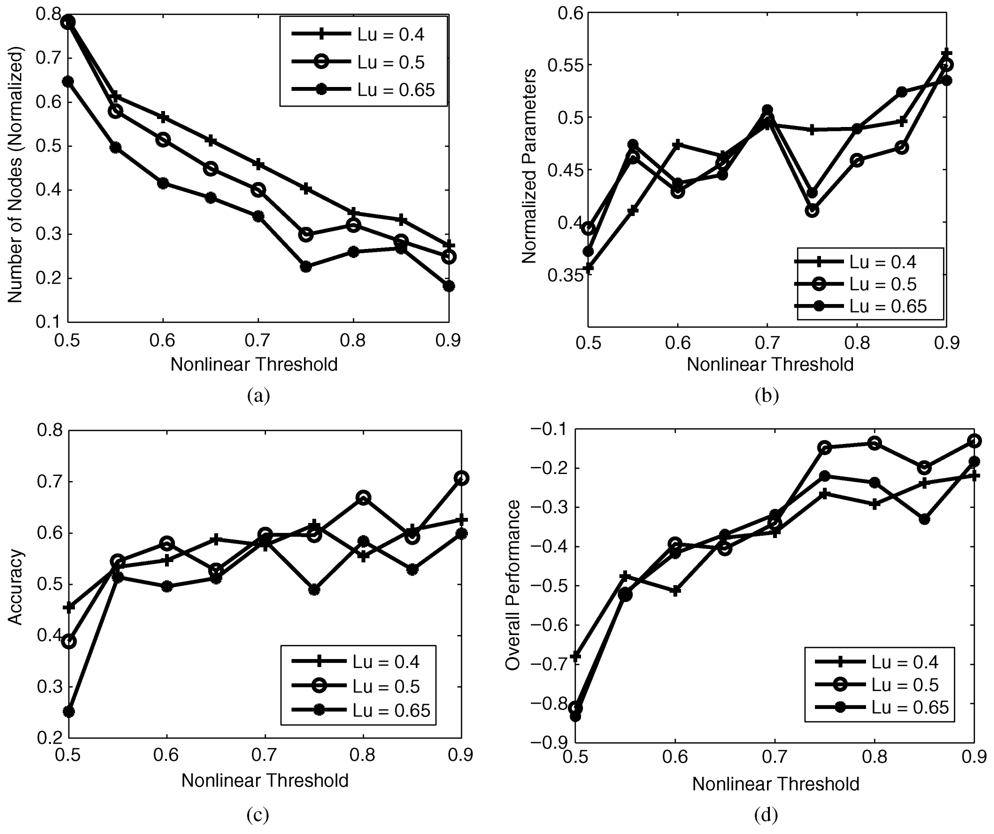

Fig. 3. Average normalized performance on different threshold pairs. (a) Tree size in terms of the number of nodes; (b) tree size in terms of the number of free parameters; (c) testing accuracy; and (d) overall performance.

We implemented and evaluated our algorithm by developing code within the framework provided by Weka [23]. Weka is a collection of machine learning algorithms for solving real-world data mining problems. The algorithms can either be applied directly to a data set or called from external Java code.

Pure univariate trees are built using the J4.8 algorithm implementation of Weka, which actually implements a later and slightly improved version of C4.5 (called C4.5 Revision 8 and the last public version of this family of algorithms before the commercial release of C5.0). Pure linear multivariate trees are constructed with a single-layer neural network with c (recall that c is the number of classes) neurons at each node. Pure nonlinear trees are built with a two-layered neural network at each node. The network has d (recall that d is the number of attributes) inputs, $(d+c)/2$ hidden neurons, and c output neurons. For both linear and nonlinear models, the classes are encoded to c-binary basis vectors and a *winner take all rule* is taken for testing [34]. In other words, if the ith output neuron produces the maximum value, the testing pattern x will take the ith branch of current node. These testing conditions are identical to the ones used with the 5×2 cv F test approach [13] and thus allow for a direct comparison.

For the proposed algorithm, the neighborhood size r is set to be $3\times$ the average Euclidean distance of each pattern from its nearest neighbor. The results are reported based on ten independent ten-fold cross-validations on each data set. All simulations are done on a Pentium IV 2.4 GHz PC with 512 M memory running Microsoft Windows XP Professional Edition.

Experiments are performed as follows. We first evaluate omnivariate decision trees on each data set with different pairs of thresholds; the results appear in Fig. 3. An optimal pair of thresholds is identified from this evaluation. The performance of omnivariate decision trees using this pair of threshold algorithms is compared with pure univariate trees, pure linear multivariate trees, nonlinear multivariate trees, and omnivariate trees constructed using 5×2 cv F test in terms of the testing accuracy (Table II), tree size (Tables III and IV), and time consumed (Table VI).

For each comparison, raw data are listed in the first table, and pairwise comparisons (by statistical t-test) are shown in the second table. The entry (i, j) of the second table represents the number of data sets on which method i is statistically significantly better than method j with 95% confidence. The row sums of the second table give the number of data sets (out of 26) where the algorithm on that row outperforms at least one of the other algorithms. The column sums give the number of data sets where the algorithm on the column is outperformed by at least one of the other algorithms.

In the following sections, we provide additional details on the threshold computation as well as on the comparitive results.

A. Optimal Thresholds

To find out a pair of optimal thresholds, a series of thresholds is preselected. Each data set listed in Table I is evaluated on those thresholds independently.

TABLE II

THE FIRST TABLE GIVES TESTING ACCURACY. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT TEN-FOLD CROSS-VALIDATION. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATA SETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j. ENTRIES IN THE Σ COLUMN CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THAT ROW OUTPERFORMS AT LEAST ONE OF THE OTHER ALGORITHMS. ENTRIES IN THE Σ ROW CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THE COLUMN IS OUTPERFORMED BY AT LEAST ONE OF THE OTHER ALGORITHMS

SET	Uni	Lin	NonLin	Omni (5x2 cv F)	Omni (Classi)
ANNE	98.7, 0.1	99.4, 0.1	99.2, 0.1	99.5, 0.1	99.5, 0.2
AUTO	82.3, 1.7	74.7, 1.4	75.0, 2.1	73.2, 2.4	75.3, 2.1
BALS	79.5, 0.9	91.8, 0.8	91.5, 0.7	89.9, 0.8	91.6, 0.6
BREC	68.5, 1.5	62.5, 2.0	66.8, 1.8	66.3, 2.2	68.1, 1.6
BREW	94.3, 0.4	96.2, 0.3	96.1, 0.3	95.2, 0.5	96.7, 0.3
CAR	92.4, 0.3	97.4, 0.3	99.4, 0.0	99.2, 0.1	99.3, 0.2
CMC	49.5, 1.1	48.4, 0.6	50.8, 1.3	48.6, 0.7	50.3, 1.0
COLI	82.6, 1.1	78.7, 1.9	79.6, 1.5	79.1, 2.1	79.4, 0.6
CREG	68.6, 1.9	68.8, 1.1	71.9, 1.1	70.3, 1.5	72.1, 1.1
DIAB	73.3, 0.8	75.7, 0.7	75.8, 0.7	74.5, 0.7	75.9, 1.1
GLAS	66.8, 2.4	64.3, 1.6	64.4, 2.5	68.0, 1.7	65.2, 1.8
HEAC	76.2, 2.2	80.2, 1.5	80.7, 1.6	79.4, 1.7	80.6, 1.3
HEAH	78.7, 1.7	79.9, 1.1	81.5, 1.3	79.6, 1.0	80.8, 1.0
HEAS	76.2, 2.5	78.8, 1.3	80.5, 1.3	76.8, 1.5	80.1, 1.0
HEPA	80.3, 2.9	80.7, 1.3	81.2, 1.8	80.3, 2.0	80.2, 1.1
IONO	90.3, 0.9	89.8, 1.2	91.3, 0.3	88.9, 1.3	91.2, 0.7
IRIS	94.4, 1.4	95.2, 0.8	96.9, 0.7	96.4, 1.1	94.9, 0.9
LABO	79.4, 2.0	90.1, 2.2	93.3, 1.1	85.0, 1.7	92.5, 2.0
LYMP	75.3, 1.8	81.8, 1.9	83.4, 0.9	80.1, 2.2	83.4, 1.4
PIMA	74.5, 0.9	75.5, 0.8	76.1, 0.5	74.7, 1.3	76.1, 0.7
PRIM	43.1, 1.4	39.3, 1.6	41.8, 1.1	39.8, 1.4	41.2, 1.5
TIC	86.3, 1.4	96.9, 0.3	97.3, 0.2	97.1, 0.3	97.4, 0.2
VEHI	72.1, 0.9	79.0, 0.8	81.0, 0.7	77.4, 1.4	82.0, 0.8
VOTE	95.6, 0.5	94.7, 0.7	94.8, 0.3	95.3, 0.3	94.7, 0.6
WINE	92.7, 0.9	98.4, 0.3	97.3, 0.6	98.2, 0.2	97.5, 0.5
ZOO	93.4, 0.5	95.2, 1.0	95.7, 0.4	95.7, 0.6	95.5, 0.7

	Uni	Lin Mul	NonLin	Omni F	Omni C	$\sum$
Uni	-	7	5	5	4	8
Lin Mul	14	-	2	6	1	14
Nonlin	19	13	-	14	1	21
Omni F	13	6	4	-	4	17
Omni C	17	11	3	13	-	20
$\sum$	19	18	10	18	7	

In the omnivariate decision trees, a univariate classifier carries one or two free parameters. If the split is performed on a nominal attribute, the only parameter is the attribute index; if the split is performed on a numeric attribute, two parameters (an attribute index and the corresponding threshold) are recorded. A linear multivariate node carries $(d+1) \times c$ parameters (recall d and c are the number of attributes and the number of classes, respectively) since each output neuron has d weights and a bias. Similarly, for nonlinear multivariate classifier, each internal node has $(d^2+(2c+1)d+c^2+3c)/2$ free parameters.[1] Tree size in terms of free parameters thus can be computed.

Fig. 3(a)–(d) shows the average performance of each pair of thresholds over 26 data sets in terms of the number of nodes, the number of parameters, the testing accuracy, and the overall performance calculated based on (17). One should be aware that the measures of tree size and testing accuracy are normalized over all pairs of thresholds.

The number of decision nodes decreases with increasing threshold [Fig. 3(a)] while the number of free parameters [Fig. 3(b)] increases with increasing value of the threshold. This indicates that a large nonlinear threshold induces a decision tree with fewer, though more complicated, decision nodes. This is true since large $\mathcal{L}_n$ usually leads to a tree with a large number of nonlinear classifiers that have significant number of free parameters. Fig. 3(a) also shows that higher univariate threshold induces decision trees with fewer nodes. At the same time, the number of free parameters varies on different univariate thresholds. This may be explained by the fact that a large number of univariate nodes may still introduce a large number of free parameters. Fig. 3(c) shows the testing accuracy increases with increasing value of the threshold, implying that a complex split can usually achieve better classification accuracy than a simple one.

The overall performance is shown in Fig. 3(d). It increases with increasing $\mathcal{L}_n$ at the left side, but changes slightly at right side. At each omnivariate decision node, we prefer a simple classifier model. A complex model is selected only when it outperforms a simple one significantly. In other words, only when the complexity of a subproblem is significant will a complex model be chosen; otherwise a simple one is preferred. Based on this consideration, a optimal pair of thresholds ($\mathcal{L}_n = 0.75$ and $\mathcal{L}_u = 0.50$) is obtained from Fig. 3(d). This optimal pair of thresholds might be finely tuned by evaluating more data sets on smaller intervals of $\mathcal{L}_n$ and $\mathcal{L}_u$. In the following discussion, the classifiability-based omnivariate trees are built based on this pair of thresholds.

B. Testing Accuracy

Table II gives the testing accuracy. Comparing univariate, linear multivariate, nonlinear, 5 × 2 cv F test-based omnivariate, and classifiability-based omnivariate trees, we see that on seven data sets out of 26, univariate trees are at least as accurate as others. On 14 data sets (ANNE, BALS, BREW, CAR, DIAB, HEAC, HEAS, LABO, LYMP, PIMA, TIC, VEHI, WINE, ZOO), the linear trees are more accurate than univariate trees. On 19 data sets (all 14 data sets above, plus CMC, GREG, HEAH, IONO, IRIS), nonlinear trees are more accurate than univariate trees. Nonlinear trees are better than linear ones on 13 data sets. Those results indicate that a univariate split is good enough sometimes; however, a linear or a nonlinear split is better on more cases. The 5 × 2 cv F test-based omnivariate trees outperform univariate trees on 13 data sets (ANNE, BALS, BREW, CAR, DIAB, HEAC, IRIS, LABO, LYMP, TIC, VEHI, WINE, ZOO), linear multivariate trees on six data sets (BREC, CAR, CREG, GLAS, IRIS, VOTE), and nonlinear trees on four data sets (ANNE, GLAS, VOTE, WINE), indicating that they can provide better accuracy in some cases. However, they are beaten by univariate trees on five data sets (AUTO, BREC, COLI, IONO, PRIM), by linear trees on six data sets (BALS, BREW, DIAB, HEAS, LABO, VEHI), and by nonlinear trees on 14 data sets (BALS, BREW, CAR, CMC, CREG, DIAB, HEAH, HEAS, IONO, LABO, LYMP, PIMA, PRIM, VEHI)

[1]$(d^2+(2c+1)d+c^2+3c)/2$ is actually the number of weights and biases for a two-layer neural network. Strictly speaking, the number of free parameters is not equal to the number of weights but also depends on the degree of nonlinearity.

TABLE III
THE FIRST TABLE GIVES TREE SIZE IN TERMS OF NUMBER OF NODES. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT TEN-FOLD CROSS-VALIDATIONS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATA SETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j. ENTRIES IN THE Σ COLUMN CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THAT ROW OUTPERFORMS AT LEAST ONE OF THE OTHER ALGORITHMS. ENTRIES IN THE Σ ROW CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THE COLUMN IS OUTPERFORMED BY AT LEAST ONE OF THE OTHER ALGORITHMS

SET	Uni	Lin	NonLin	Omni (5x2 cv F)	Omni (Classi)
ANNE	72.0, 0.0	14.0, 4.2	9.0, 3.5	19.5, 4.3	31.2, 4.0
AUTO	88.0, 0.0	46.6, 4.2	27.4, 9.0	63.7, 16.9	33.4, 5.2
BALS	119.0, 0.0	32.2, 9.9	7.0, 3.5	63.1, 12.5	7.9, 4.5
BREC	168.0, 0.0	27.4, 3.0	5.0, 1.6	55.5, 36.5	9.6, 4.2
BREW	45.0, 0.0	5.2, 0.6	3.8, 1.4	31.4, 10.6	9.4, 0.8
CAR	186.0, 0.0	68.2, 14.1	6.6, 2.1	15.4, 9.6	7.4, 4.3
CMC	617.0, 0.0	412.0, 36.2	85.9, 19.9	433.4, 79.5	114.1, 10.9
COLI	125.0, 0.0	12.4, 1.3	9.4, 1.3	59.4, 20.7	10.2, 1.4
CREG	435.0, 0.0	64.4, 7.2	7.2, 2.0	138.2, 63.2	12.0, 2.4
DIAB	43.0, 0.0	43.0, 12.5	20.6, 7.2	69.6, 13.5	22.2, 8.2
GLAS	59.0, 0.0	111.6, 26.4	54.9, 21.7	77.4, 6.4	58.8, 10.6
HEAC	77.0, 0.0	46.5, 7.6	16.0, 2.4	56.3, 15.9	25.0, 3.9
HEAH	47.0, 0.0	50.5, 9.3	17.5, 8.2	53.6, 12.4	32.5, 14.2
HEAS	61.0, 0.0	22.4, 3.8	3.8, 1.0	46.8, 13.3	9.4, 1.8
HEPA	31.0, 0.0	8.4, 1.3	3.6, 1.0	25.8, 6.3	4.6, 0.8
IONO	35.0, 0.0	7.2, 1.8	3.0, 0.0	17.6, 10.3	3.0, 0.0
IRIS	9.0, 0.0	9.4, 1.3	4.3, 0.9	7.6, 0.8	8.0, 0.0
LABO	22.0, 0.0	3.0, 0.0	3.0, 0.0	8.6, 7.7	3.0, 0.0
LYMP	38.0, 0.0	15.4, 2.1	9.0, 0.0	13.0, 6.3	12.2, 1.7
PIMA	43.0, 0.0	39.4, 16.0	16.4, 7.0	51.8, 19.9	22.8, 11.8
PRIM	123.0, 0.0	454.2, 29.7	397.0, 40.2	291.2, 58.5	383.4, 50.6
TIC	208.0, 0.0	14.4, 1.9	3.0, 0.0	29.4, 9.8	3.0, 0.0
VEHI	207.0, 0.0	101.0, 23.8	30.2, 9.6	103.0, 30.7	40.0, 16.3
VOTE	37.0, 0.0	9.0, 0.0	5.0, 1.3	32.4, 6.7	6.0, 1.9
WINE	9.0, 0.0	4.0, 0.0	4.0, 0.0	4.0, 0.0	4.0, 0.0
ZOO	17.0, 0.0	8.0, 0.0	8.0, 0.0	8.0, 0.0	8.0, 0.0

	Uni	Lin Mul	NonLin	Omni F	Omni C	$\sum$
Uni	-	2	1	3	1	3
Lin Mul	16	-	0	14	2	19
Nonlin	18	23	-	22	11	24
Omni F	18	4	1	-	2	20
Omni C	18	21	0	20	-	24
$\sum$	22	23	1	23	12	

PRIM), which shows that this algorithm cannot always generate the most accurate decision trees.

Comparing 5×2 cv F test-based omnivariate trees with the proposed approach, we see that the classifiability-based omnivariate trees outperform univariate trees on 17 data sets (ANNE, BALS, BREW, CAR, CREG, DIAB, HEAC, HEAH, HEAS, IONO, LABO, LYMP, PIMA, TIC, VEHI, WINE), outperform linear trees on 11 data sets (BREC, BREW, CAR, CMC, CREG, HEAS, IONO, LABO, PRIM, TIC, VEHI), and nonlinear trees on three data sets (ANNE, BREW, VEHI). Furthermore, the proposed algorithm gives more accurate results than a 5×2 cv F test-based algorithm on 13 data sets (BALS, BREW, CMC, CREG, DIAB, HEAH, HEAS, IONO, LABO, LYMP, PIMA, TIC, VEHI). It does worse than a 5×2 cv F test-based algorithm on only four data sets (GLAS, IRIS, VOTE, WINE). It is only outperformed by univariate trees on four data sets (AUTO, COLI, PRIM, VOTE), by linear trees on one data set (WINE), and by nonlinear trees on one data set (IRIS).

The above discussion shows that among these five algorithms, the proposed one produces decision trees with the highest accuracy.

C. Tree Size

The total number of nodes generated by the five methods is reported in Table III. The number of free parameters is given in Table IV since different split models have different complexities. We also report the number of univariate, linear, and nonlinear nodes in the omnivariate trees in Table V.

As expected, the order of the tree size in terms of the number of tree nodes is Uni $>$ Lin $>$ Nonlin. The order of tree size in terms of the number of free parameters is exactly the opposite. An omnivariate tree using a 5×2 cv F test has a fairly large number of nodes but less than univariate trees; an omnivariate tree induces by classifiability measure has relatively small number of tree nodes. Insofar as the the number of free parameters is concerned, we found both omnivariate trees outperform linear trees on ten data sets (five of them are the same); 5×2 cv F test-based omnivariate trees outperform nonlinear trees on 13 data sets, and classifiability-based omnivariate trees outperform nonlinear trees on nine data sets (eight of them are the same). Classifiability-based omnivariate trees produce trees with fewer parameters than a 5×2 cv F test-based one on four data sets while the latter one outperforms the previous one

TABLE IV

THE FIRST TABLE GIVES TREE SIZE IN TERMS OF FREE PARAMETERS. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT TEN-FOLD CROSS-VALIDATIONS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATA SETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j. ENTRIES IN THE Σ COLUMN CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THAT ROW OUTPERFORMS AT LEAST ONE OF THE OTHER ALGORITHMS. ENTRIES IN THE Σ ROW CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THE COLUMN IS OUTPERFORMED BY AT LEAST ONE OF THE OTHER ALGORITHMS

SET	Uni	Lin	NonLin	Omni (5x2 cv F)	Omni (Classi)
ANNE	19.0, 0.0	608.4, 197.3	1593.6, 696.4	523.6, 344.8	869.8, 157.9
AUTO	36.8, 0.0	988.0, 90.9	2068.0, 707.6	506.5, 265.8	1017.0, 133.4
BALS	118.0, 0.0	156.0, 49.6	62.0, 35.8	206.5, 56.0	52.1, 25.9
BREC	25.0, 0.0	264.0, 29.5	136.0, 55.5	142.7, 75.9	143.6, 58.6
BREW	44.0, 0.0	42.0, 6.3	95.2, 47.5	121.0, 46.3	90.0, 22.3
CAR	52.0, 0.0	470.4, 98.6	82.6, 30.5	81.7, 32.4	78.9, 38.3
CMC	284.8, 0.0	4110.0, 362.2	2292.3, 536.0	2726.8, 338.7	2300.8, 238.5
COLI	43.5, 0.0	262.2, 31.0	1268.4, 191.0	506.9, 294.6	467.6, 32.2
CREG	139.1, 0.0	1331.4, 150.8	790.5, 253.6	1343.3, 411.2	444.0, 49.5
DIAB	42.0, 0.0	378.0, 112.2	558.6, 204.3	273.2, 67.1	418.4, 146.7
GLAS	58.0, 0.0	632.0, 150.6	731.5, 293.8	240.3, 93.3	536.3, 119.6
HEAC	45.3, 0.0	637.0, 106.7	528.0, 83.0	398.8, 104.7	442.0, 55.2
HEAH	26.3, 0.0	693.0, 129.7	580.8, 288.0	500.1, 329.0	568.2, 196.5
HEAS	60.0, 0.0	299.6, 52.9	170.8, 63.0	288.0, 107.4	211.6, 25.7
HEPA	19.7, 0.0	148.0, 27.0	302.9, 112.5	178.6, 140.6	265.0, 16.9
IONO	34.0, 0.0	217.0, 61.3	668.0, 0.0	204.2, 286.2	668.0, 0.0
IRIS	8.0, 0.0	42.0, 6.3	34.1, 9.8	18.6, 0.8	19.0, 0.0
LABO	13.5, 0.0	34.0, 0.0	173.0, 0.0	93.2, 79.6	173.0, 0.0
LYMP	17.5, 0.0	273.6, 39.2	514.0, 0.0	134.3, 88.6	393.8, 32.0
PIMA	42.0, 0.0	345.6, 143.9	438.9, 199.4	222.0, 114.3	369.9, 172.5
PRIM	56.0, 0.0	8157.6, 534.6	14436.0, 1464.2	5854.3, 1362.0	7832.0,1218.9
TIC	69.0, 0.0	134.0, 19.0	68.0, 0.0	109.8, 36.5	68.0, 0.0
VEHI	206.0, 0.0	1900.0, 451.8	1876.1, 618.3	1462.5, 431.6	1385.5, 492.7
VOTE	18.0, 0.0	136.0, 0.0	346.0, 115.3	172.4, 91.8	224.0, 33.0
WINE	8.0, 0.0	42.0, 0.0	139.0, 0.0	42.0, 0.0	139.0, 0.0
ZOO	8.4, 0.0	133.0, 0.0	332.0, 0.0	133.0, 0.0	332.0, 0.0

	Uni	Lin Mul	NonLin	Omni F	Omni C	$\sum$
Uni	-	21	19	23	18	24
Lin Mul	0	-	10	3	7	11
Nonlin	1	8	-	4	0	8
Omni F	0	10	13	-	8	16
Omni C	1	10	9	4	-	16
$\sum$	1	21	22	23	20	

on eight data sets. Generally speaking, these two algorithms generate decision trees with a similar number of free parameters. However, for 5×2 cv F test-based omnivariate trees, 11 data sets (ANNE, AUTO, BREC, COLI, HEAH, HEPA, IONO, LABO, LYMP, PIMA, VOTE) out of 26 have large deviations (more than 50%). At the same time, no data set has deviations larger than 50% for classifiability based omnivariate trees. The reason is that only half of the instances reaching a particular node are tested for model selection in the 5×2 cv F test (the other half of the instances are used for classifier training), which leads to less generality. For classifiability-based induction, the whole subproblem is evaluated to produce a data complexity measure with more generality, which leads to a relatively stable architecture.

Table V shows the nodes distribution for both kinds of omnivariate trees on univariate splits, linear multivariate splits, and nonlinear splits. From Table V, we see that 5×2 cv F test-based omnivariate trees contain 68.84% of univariate nodes in average, and only 7.55% are nonlinear nodes. This indicates that the induction mainly choose the univariate model to construct a decision hyperplane. Our results are consistent with [13]. For the proposed algorithm, the portion of univariate nodes is 43.61% and the nonlinear portion is 37.77%, indicating that the proposed method of decision tree induction does not necessarily choose simple nodes, as is the case with 5×2 cv F. Our experiments also show the node distribution is greatly improved by the introduction of the sparsity factor α. Without the sparsity factor, the classifiability measure may have a low value near the leaf nodes where the amount of data is substantially reduced. Consequently a decision tree with a large number of nonlinear nodes might be produced and may cause overfitting given the sparsity of data. Such a problem is solved by sparsity factor since it significantly amplifies the classifiability measure in such cases.

D. Training Time

It takes $\mathcal{O}(n \cdot d)$ time for training a univariate node, $\mathcal{O}(n \cdot d \cdot e \cdot c^2)$ for training a linear multivariate node, and $\mathcal{O}(n \cdot d^2 \cdot e \cdot c^2)$ for training a nonlinear node, where n is the number of instances at that node, e is the number of epochs, c is the number of classes, and d is the number of attributes. Thus, constructing a nonlinear tree, which contains very complex internal nodes, is greatly time consuming, especially when the data set has a

TABLE V
NODE DISTRIBUTION. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN-FOLD CROSS-VALIDATION. THE TOTAL NUMBER AND PERCENTAGE OF NODES OF EACH TYPE ARE SUMMARIZED IN THE LAST TWO ROWS

SET	Omni (5x2 cv F)			Omni (Classi)		
	Uni	Lin	NonLin	Uni	Lin	NonLin
ANNE	2.80, 1.23	1.80, 0.42	0.10, 0.32	4.0, 0.0	3.7, 0.7	0.0, 0.0
AUTO	10.30, 4.03	1.60, 1.07	0.60, 0.52	0.6, 0.8	4.2, 1.0	1.0, 0.0
BALS	19.80, 4.34	4.10, 1.85	3.40, 1.35	0.0, 0.0	1.2, 1.3	1.1, 0.3
BREC	7.50, 5.19	2.00, 2.54	1.40, 0.84	0.0, 0.0	3.1, 2.0	1.2, 0.6
BREW	11.50, 4.43	3.20, 1.69	0.50, 0.53	1.0, 0.0	2.7, 0.8	0.5, 0.5
CAR	2.50, 2.12	0.30, 0.48	1.20, 0.42	0.0, 0.0	0.5, 0.8	1.1, 0.3
CMC	85.50, 20.64	51.50, 7.89	13.30, 2.00	1.8, 1.3	12.9, 2.2	23.6, 2.8
COLI	13.40, 5.40	2.10, 0.99	1.30, 1.06	0.0, 0.0	3.6, 0.7	1.0, 0.0
CREG	27.90, 13.09	6.80, 2.39	4.00, 1.63	0.0, 0.0	4.5, 1.2	1.0, 0.0
DIAB	25.90, 5.24	6.60, 2.27	1.80, 0.92	0.4, 0.5	4.2, 2.9	6.0, 2.0
GLAS	24.90, 4.28	3.10, 1.29	0.70, 0.95	0.9, 0.6	4.1, 0.9	3.9, 1.2
HEAC	11.60, 4.97	4.70, 0.82	0.30, 0.48	0.0, 0.0	3.8, 0.8	1.0, 0.0
HEAH	12.80, 3.82	2.10, 1.79	1.90, 1.66	0.0, 0.0	5.1, 2.9	1.2, 0.4
HEAS	17.20, 6.36	4.70, 1.89	1.00, 0.94	0.0, 0.0	3.2, 0.9	1.0, 0.0
HEPA	10.20, 3.01	1.80, 1.23	0.40, 0.70	0.0, 0.0	0.8, 0.4	1.0, 0.0
IONO	7.30, 5.64	0.80, 0.42	0.20, 0.42	0.0, 0.0	0.0, 0.0	1.0, 0.0
IRIS	1.80, 0.42	1.00, 0.00	0.00, 0.00	2.0, 0.0	1.0, 0.0	0.0, 0.0
LABO	2.40, 3.44	0.60, 0.52	0.40, 0.52	0.0, 0.0	0.0, 0.0	1.0, 0.0
LYMP	1.90, 1.45	1.40, 0.70	0.10, 0.32	0.0, 0.0	1.8, 0.4	1.0, 0.0
PIMA	18.60, 7.93	5.20, 3.61	1.60, 1.17	0.6, 0.7	5.6, 4.1	4.7, 1.9
PRIM	9.10, 4.82	9.90, 1.91	2.40, 1.51	3.8, 2.0	14.3, 2.3	2.7, 1.3
TIC	7.00, 2.91	3.10, 1.45	0.60, 0.52	0.0, 0.0	0.0, 0.0	1.0, 0.0
VEHI	29.80, 8.32	7.30, 3.09	3.30, 1.06	0.7, 0.9	5.7, 3.2	3.7, 1.3
VOTE	13.90, 3.28	1.10, 0.99	0.70, 0.67	0.0, 0.0	1.5, 1.0	1.0, 0.0
WINE	0.00, 0.00	1.00, 0.00	0.00, 0.00	0.0, 0.0	0.0, 0.0	1.0, 0.0
ZOO	0.00, 0.00	1.00, 0.00	0.00, 0.00	0.0, 0.0	0.0, 0.0	1.0, 0.0
$\sum$	375.6	128.8	41.2	15.8	87.5	62.7
%	68.84	23.61	7.55	9.5	52.7	37.77

large number of attributes and classes. A univariate tree contains simple internal nodes with only one (index of split attribute) or two (split attribute with the threshold) parameters; it takes the least learning time though such a model intends to produce large trees in terms of tree nodes.

Table VI shows the learning time of constructing univariate, linear multivariate, nonlinear multivariate, 5×2 cv F test-based omnivariate, and classifiability-based omnivariate trees, respectively. Table VI shows that the fastest algorithm among those five is the univariate model. It outperforms all the others. Nonlinear tree outperforms linear tree only on one data set (CAR). From Table VI, we also see that 5×2 cv F test-based induction is the slowest algorithm. It beats none of the other algorithms on any data set. More precisely, it takes about 10 to 20 times (12.4 times in average) longer than the corresponding nonlinear algorithm. The proposed algorithm, on the other hand, is much faster than the 5×2 cv F test. It not only outperforms nonlinear algorithm on most data sets (14 out of 26) but also outperforms linear algorithm on two data sets (CAR, IRIS). Actually, it only takes 4.7% in average of the time taken by the 5×2 cv F test algorithm in constructing an omnivariate decision tree.

The reason is obvious. For the 5×2 cv F test, in order to make a model selection at a particular decision node, each one of the three models, univariate, linear, and nonlinear, has to be tested ten times with five runs of two-fold cross-validation on the instances at this node. The entire process is very slow. Moreover, such an algorithm usually generates big decision trees with large percentage of univariate nodes. The ratio of nonlinear nodes in such trees is small, usually less than 10% (shown in Table III), but testing a nonlinear model at every internal node (including univariate ones) is computationally very intensive and makes a major contribution to the total time spent for training. Therefore, statistical-based model selection is not an efficient approach for omnivariate tree constructing.

Comparatively, classifiability-based algorithm only needs a single run to capture the complexity of the data at a decision node. Since we need to compute $(n-1)$ distances to count the points that are within the neighborhood r of pattern $x^{(i)}$, each run will take $\mathcal{O}(n^2 \cdot d)$ time. In addition, when the distance is computed using some efficient data structures, the actual complexity can be quite modest. In other words, the proposed algorithm is substantially superior than the 5×2 cv F test in terms of learning speed for model selection.

VII. Conclusion and Discussion

In this paper, we proposed a classifiability measure for model selection in constructing omnivariate decision trees. The classifiability measure is strongly realted to the Bayes error and the boundary complexity. A sparsity factor allows the classifiability measure to perform superior model selection particularly near the leaf nodes where the number of examples are greatly reduced. Experiment results over 26 data sets show that, on average, our algorithm can significantly improve the learning speed when compared with the conventional approach using combined 5×2 cv F statistical test while achieving better classification accuracy.

TABLE VI

THE FIRST TABLE GIVES LEARNING TIME IN SECONDS ON A PENTIUM IV 2.4 G PC. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT TEN-FOLD CROSS-VALIDATIONS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATA SETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j. ENTRIES IN THE Σ COLUMN CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THAT ROW OUTPERFORMS AT LEAST ONE OF THE OTHER ALGORITHMS. ENTRIES IN THE Σ ROW CORRESPOND TO THE NUMBER OF DATA SETS (OUT OF 26) WHERE THE ALGORITHM ON THE COLUMN IS OUTPERFORMED BY AT LEAST ONE OF THE OTHER ALGORITHMS

SET	Uni	Lin	NonLin	Omni (5x2 cv F)	Omni (Classi)
ANNE	0.10, 0.03	31.28, 8.21	170.65, 28.31	1647.53, 414.13	41.61, 9.80
AUTO	0.03, 0.00	5.43, 0.56	29.41, 4.47	300.55, 60.84	16.60, 0.96
BALS	0.02, 0.00	1.39, 0.19	1.49, 0.20	23.58, 2.19	1.67, 0.32
BREC	0.01, 0.01	6.29, 0.77	22.12, 3.56	204.15, 40.37	11.51, 1.32
BREW	0.03, 0.00	1.34, 0.06	2.88, 0.44	31.66, 3.06	2.30, 0.20
CAR	0.02, 0.00	22.30, 4.27	14.96, 1.30	133.95, 36.37	15.86, 1.51
CMC	0.09, 0.01	23.96, 2.20	58.65, 5.11	584.42, 35.29	53.50, 6.45
COLI	0.04, 0.01	5.90, 0.98	52.15, 6.25	696.51, 121.87	20.55, 0.97
CREG	0.10, 0.01	53.72, 13.74	138.02, 25.19	1506.56, 210.93	64.76, 6.28
DIAB	0.05, 0.00	2.82, 0.81	5.25, 0.83	59.93, 11.24	6.28, 1.64
GLAS	0.03, 0.00	2.01, 0.32	2.97, 0.51	37.49, 2.23	2.72, 0.33
HEAC	0.02, 0.00	4.90, 0.41	8.04, 0.72	110.16, 10.27	5.66, 0.53
HEAH	0.03, 0.01	4.67, 0.63	7.63, 1.18	152.40, 27.46	5.92, 0.94
HEAS	0.03, 0.01	1.42, 0.22	1.92, 0.22	35.13, 7.09	1.78, 0.13
HEPA	0.02, 0.01	0.66, 0.12	1.62, 0.06	27.82, 2.80	1.13, 0.11
IONO	0.21, 0.01	2.34, 0.25	8.53, 0.32	140.54, 43.21	6.44, 0.44
IRIS	0.00, 0.01	0.22, 0.01	0.28, 0.02	3.06, 0.22	0.13, 0.01
LABO	0.00, 0.01	0.09, 0.01	0.56, 0.01	7.20, 4.43	0.59, 0.04
LYMP	0.01, 0.01	1.39, 0.15	6.26, 0.06	47.77, 9.16	3.89, 0.32
PIMA	0.06, 0.01	2.39, 0.60	5.11, 1.48	51.02, 5.17	6.75, 2.20
PRIM	0.02, 0.00	14.75, 1.08	27.41, 2.15	287.28, 13.10	20.79, 2.05
TIC	0.02, 0.01	7.91, 0.87	10.03, 0.18	178.08, 27.26	10.63, 0.41
VEHI	0.20, 0.01	8.71, 1.11	15.88, 1.07	252.68, 27.55	15.08, 1.80
VOTE	0.02, 0.01	1.52, 0.02	4.27, 0.84	71.03, 8.35	2.58, 0.30
WINE	0.03, 0.02	0.26, 0.03	0.74, 0.03	4.92, 0.02	0.80, 0.18
ZOO	0.00, 0.01	2.13, 0.03	17.92, 0.23	105.83, 4.20	18.42, 1.19

	Uni	Lin Mul	NonLin	Omni F	Omni C	$\sum$
Uni	-	26	26	26	26	26
Lin Mul	0	-	24	26	24	26
Nonlin	0	1	-	26	2	26
Omni F	0	0	0	-	0	0
Omni C	0	2	14	26	-	26
$\sum$	0	26	26	26	26	

APPENDIX I

In this section, we will prove Theorem 1, i.e.: *For a given data distribution, the sum of the primary classifiability measure L_0 and the Bayes error E is a constant,* with a simplified example. In the proof, we use an uppercase $P(\cdot)$ to denote a probability mass function and use a lowercase $p(\cdot)$ to denote a probability density function.

Consider a two-class classification problem with the classes characterized by the fixed but unknown density functions $p(x|\omega_1)$ and $p(x|\omega_2)$ defined on the input x. Fig. 4 shows such a problem in one dimension. By Bayes rule, we have

$$P(\omega_1|x) = \frac{p(x|\omega_1)P(\omega_1)}{p(x)} \tag{18}$$

and

$$P(\omega_2|x) = \frac{p(x|\omega_2)P(\omega_2)}{p(x).} \tag{19}$$

The Bayes decision boundary is then given by

$$p(x|\omega_1)P(\omega_1) = p(x|\omega_2)P(\omega_2). \tag{20}$$

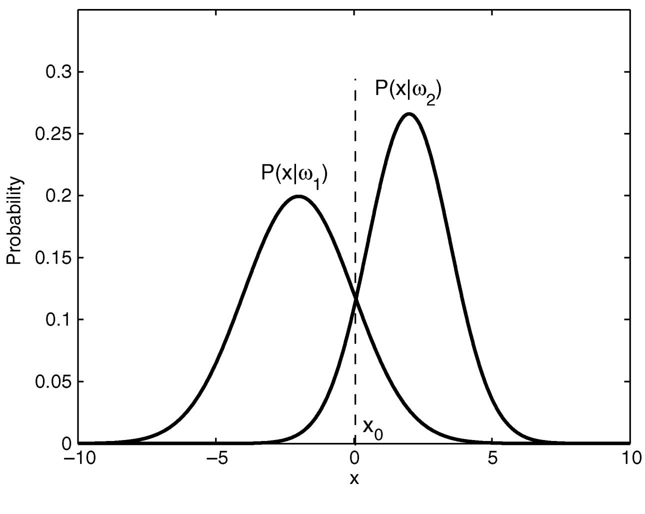

Fig. 4. A simple classification problem and the associated Bayes decision boundary (broken line).

Let the Bayes decision boundary for the one-dimensional problem shown in Fig. 2 be given by $x = x_0$. The Bayes error can then be calculated as

$$E = P(\omega_1)\int_{x_0}^{\infty} p(x|\omega_1)\mathrm{d}x + P(\omega_2)\int_{-\infty}^{x_0} p(x|\omega_2)\mathrm{d}x. \quad (21)$$

For arbitrary densities, (21) can be generalized to

$$E = P(\omega_1)\sum_{A_i}\int_{A_i} p(x|\omega_1)\mathrm{d}x + P(\omega_2)\sum_{B_i}\int_{B_i} p(x|\omega_2)\mathrm{d}x \quad (22)$$

where

$$p(x|\omega_1)P(\omega_1) < p(x|\omega_2)P(\omega_2) \quad (23)$$

in areas A_i and

$$p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2) \quad (24)$$

in areas B_i.

Equation (22) can be extended to the discrete distribution of x by replacing integrals with corresponding sums and probability density functions with corresponding probability mass functions

$$E = P(\omega_1)\sum_{A_i}\sum_{x\in A_i} P(x|\omega_1) + P(\omega_2)\sum_{B_i}\sum_{x\in B_i} P(x|\omega_2). \quad (25)$$

For primary classifiability, from (7) and (8), we have

$$\begin{aligned} L_0 &= \sum_z C(z) \\ &= \sum_z\sum_y P(y)\left[P(\omega_1|y)P(\omega_1|z) + P(\omega_2|y)P(\omega_2|z)\right] \\ &= \sum_z\sum_y P(y)\left[\frac{P(y|\omega_1)P(\omega_1)}{P(y)}P(\omega_1|z)\right. \\ &\qquad\qquad \left. + \frac{P(y|\omega_2)P(\omega_2)}{P(y)}P(\omega_2|z)\right] \\ &= \sum_z\sum_y \left[P(y|\omega_1)P(\omega_1)P(\omega_1|z)\right. \\ &\qquad\qquad \left. + P(y|\omega_2)P(\omega_2)P(\omega_2|z)\right]. \end{aligned} \quad (26)$$

Note that $P(y|\omega_1)$ and $P(y|\omega_2)$ in (26) are identical to $P(x|\omega_1)$ and $P(x|\omega_2)$ in (25), and a given pattern z belongs to either class ω_1 or to class ω_2. Equation (26) can be rewritten as

$$L_0 = \sum_{B_i}\sum_x P(x|\omega_1)P(\omega_1) + \sum_{A_i}\sum_x P(x|\omega_2)P(\omega_2) \quad (27)$$

where x is a pattern in the neighborhood of pattern z and we have

$$P(\omega_2|z) = 1, \quad P(\omega_1|z) = 0 \quad (28)$$

in areas A_i and

$$P(\omega_1|z) = 1, \quad P(\omega_2|z) = 0 \quad (29)$$

in areas B_i.

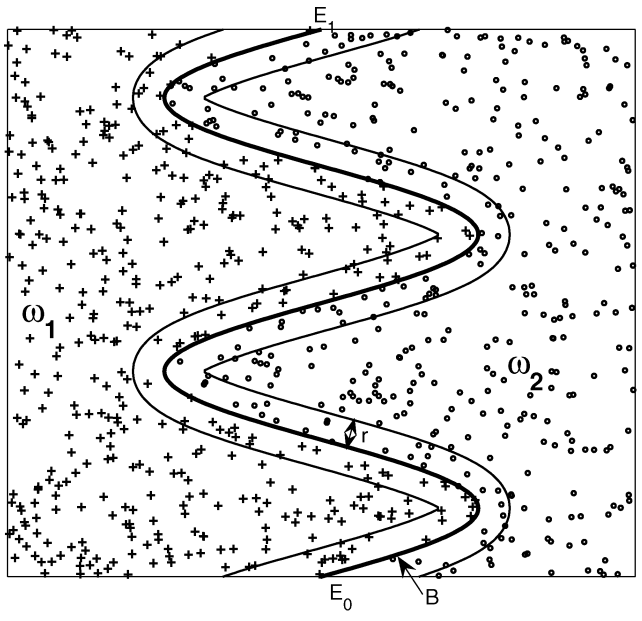

Fig. 5. A two-class (ω_1 and ω_2) classification problem with zero Bayes error and uniform distribution. B: decision boundary. r: size of neighborhood. E_0 and E_1 are two ends of the boundary.

Note that A_i and B_i in (27) are the same as A_i and B_i in (25) and corresponding to each other. Thus, using (27) and (25), we get

$$\begin{aligned} L_0 + E &= \sum_{A_i}\sum_x \left[P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)\right] \\ &\quad + \sum_{B_i}\sum_x \left[P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)\right] \\ &= \sum_x \left[P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)\right]. \end{aligned} \quad (30)$$

For a given distribution, it is obvious that $\sum_x[P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)]$ is a constant.

Equation (30) shows that the sum of the Bayes error and the primary classifiability is a constant. In other words, an increase in the Bayes error implies a lower primary classifiability and a decrease in Bayes error corresponds to an increase in the primary classifiability.

Appendix II

In this section, we will prove Theorem 2.

Consider a two-class (ω_1 and ω_2) classification problem with two features, as shown in Fig. 5. To examine the relation between the classifiability measure and the decision boundary complexity without the effect of other factors, we choose a problem with zero Bayes error, that is, there is a classifier that can perfectly separate the two classes. Suppose the ends of the boundary are fixed: the boundary complexity thus can be represented by its length. In the following, we show that the classifiability measure $\mathcal{L}$ is inversely related to the boundary length.

As shown in Fig. 5, only patterns within the neighborhood (within a distance r) of the decision boundary contribute to the off-diagonal elements of the co-occurrence matrix A in (10). Let y denote a pattern within the neighborhood of another pattern z, $P(y)$ the probability of occurrence of y, and $P(\omega_i|y)$ the posterior probability. The sum of the off-diagonal elements of A before normalization can be computed as follows:

$$\begin{aligned}\sum_{\substack{j,k=1\\ j\neq k}}^{c} A_{jk} &= \sum_{i=1}^{n}\sum_{\substack{j,k=1\\ j\neq k}}^{c} w_{jk}\left(x^{(i)}\right)\\ &= \sum_{z}\sum_{y} P(y)\left[P(\omega_1|y)P(\omega_2|z)\right.\\ &\qquad\left.+P(\omega_2|y)P(\omega_1|z)\right]\\ &= \sum_{z\in A_N}\sum_{y} P(y)\left[P(\omega_1|y)P(\omega_2|z)\right.\\ &\qquad\left.+P(\omega_2|y)P(\omega_1|z)\right] \end{aligned} \quad (31)$$

where A_N is the neighborhood area of the decision boundary. Without loss of generality, we assume that the decision boundary length S satisfies $S \gg r$. The sum of z can be approximated with integral over A_N, such that (31) can be rewritten as

$$\begin{aligned}\sum_{\substack{j,k=1\\ j\neq k}}^{c} A_{jk} = \iint\limits_{A_N} P(z)\sum_{y} P(y)\left[P(\omega_1|y)P(\omega_2|z)\right.\\ \left.+P(\omega_2|y)P(\omega_1|z)\right]\mathrm{d}A_N \end{aligned} \quad (32)$$

where $P(z)$ is the probability of occurrence of z and is independent to the boundary itself. Thus, (32) can be rewritten as

$$\begin{aligned}\sum_{\substack{j,k=1\\ j\neq k}}^{c} A_{jk} &= \int\limits_{S} r\cdot P(z)\sum_{y} P(y)\\ &\quad\times\left[P(\omega_1|y)P(\omega_2|z)+P(\omega_2|y)P(\omega_1|z)\right]\mathrm{d}S\\ &= S\cdot r\cdot P(z)\sum_{y} P(y)\\ &\quad\times\left[P(\omega_1|y)P(\omega_2|z)+P(\omega_2|y)P(\omega_1|z)\right]\\ &= \beta S \end{aligned} \quad (33)$$

where

$$\beta = r\cdot P(z)\sum_{y} P(y)\left[P(\omega_1|y)P(\omega_2|z)+P(\omega_2|y)P(\omega_1|z)\right].$$

On the other hand, classifiability measure $\mathcal{L}$ is of the form

$$\begin{aligned}\mathcal{L} &= \alpha L_0\\ &= \alpha\left(1-\frac{\sum_{\substack{j,k=1\\ j\neq k}}^{c} A_{jk}}{\sum A_{jk}}\right)\\ &= \alpha(1-kS) \end{aligned} \quad (34)$$

where $k = \beta/\sum A_{jk}$.

Finally, we have

$$\frac{1}{\alpha}\mathcal{L}+kS = 1. \quad (35)$$

For a given distribution, α and k are constants. Equation (35) shows that classifiability measure is strongly and inversely related to boundary complexity in terms of the boundary size, which is the boundary length in this particular case. In a real-world situation, a classification problem might be multidimensional and multiclass, implying that the decision boundary could be one or more multidimension surface with ambiguous shape. Correspondingly, the size of decision boundary is represented by the area of the surface. It is easy to see that the same relationship exists between the classifiability measure and the boundary size in those cases.

REFERENCES

[1] J. R. Quinlan, "Induction of decision trees," *Machine Learn.*, vol. 1, pp. 81–106, 1986.

[2] L. Breiman, J. H. Friedman, J. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[3] H. Guo and S. B. Gelfand, "Classification trees with neural network feature extraction," *IEEE Trans. Neural Netw.*, vol. 3, pp. 923–933, 1992.

[4] G. P. J. Schmitz, C. Aldrich, and F. S. Gouws, "ANN-DT: an algorithm for extraction of decision trees from artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1392–1401, 1999.

[5] M. A. Sánchez-Montañés and F. J. Corbacho, "A new information processing measure for adaptive complex systems," *IEEE Trans. Neural Netw.*, vol. 15, no. 4, pp. 917–927, 2004.

[6] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[7] S. K. Murthy and S. Salzberg, "Lookahead and pathology in decision tree induction," in *Proc. 14th Int. Conf. Artificial Intelligence*, San Mateo, California, 1995, pp. 1025–1031.

[8] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees," *Machine Learn.*, vol. 19, pp. 45–77, 1995.

[9] W.-Y. Loh and N. Vanichsetakul, "Tree-structured classification via generalized discriminant analysis," *J. Amer. Statist. Assoc.*, vol. 83, pp. 715–728, 1988.

[10] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 3rd ed, E. Cliks, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[11] A. Bermak and D. Martinez, "A compact 3-d VLSI classifier using bagging treshold network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1097–1109, 2003.

[12] I. Sethi, "Neural implementation of tree classifiers," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 8, pp. 1243–1249, 1995.

[13] O. T. Yıldız and E. Alpaydın, "Omnivariate decision tree," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1539–1546, 2001.

[14] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Intell. Res.*, vol. 2, pp. 1–32, 1994.

[15] E. Alpaydın, "Combined 5 × 2 cv F test for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 11, pp. 1885–1892, 1999.

[16] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer, 1995.

[17] B. Karacali and H. Krim, "Fast minimization of structural risk by nearest neighbor rule," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 127–137, 2003.

[18] C. Blake and C. Merz. (1998) UCI repository of machine learning databases. [Online]http://www.ics.uci.edu/~mlearn/MLRepository.html

[19] M. Dong and R. Kothari, "Look-ahead based fuzzy decision tree induction," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 3, pp. 461–468, 2001.

[20] ——, "Texture based look-ahead for decision-tree induction," in *Proc. Int. Conf. Advances Pattern Recognition*, S. Singh, Ed., 2001.

[21] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, pp. 786–804, 1980.

[22] A. R. Rao, *A Taxonomy for Texture Description and Identification*. New York: Springer-Verlag, 1990.

[23] I. H. Witten and E. Fank, *Data Mining: Practical Machine Learning Tools With Java Implementations*, D. D. Cerra, Ed. San Mateo, CA: Morgan Kaufmann, 1999.

[24] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 3, pp. 289–300, 2002.

[25] P. J. Verveer and R. P. W. Duin, "An evaluation of intrinsic dimensionality estimators," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, no. 1, pp. 81–86, 1995.
[26] N. Wyse, R. Dubes, and A. K. Jain, "A critical evaluation of intrinsic dimensionality algorithms," in *Pattern Recognition in Practice*, E. Gelsema and L. N. Kanal, Eds. Amsterdam, the Netherlands: North-Holland, 1980, pp. 415–425.
[27] S. P. Smith and A. K. Jain, "A test to determine the multivariate normality of a data set," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 5, pp. 757–761, 1988.
[28] D. S. Broomhead, R. Jones, and G. P. King, "Topological dimension and local coordinates," *J. Phys. A Math. Gen.*, vol. 20, no. 6, pp. 563–569, 1987.
[29] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," *Prob. Inform. Transmission*, vol. 1, pp. 4–7, 1965.
[30] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd ed. Berlin, Germany: Springer-Verlag, 1997.
[31] G. J. Chaitin, "A theory of program size formally identical to information theory," *J. ACM*, vol. 22, pp. 329–340, 1975.
[32] M. Gell-Mann, "What is complexity?," *Complexity*, vol. 1, no. 1, pp. 16–19, 1995.
[33] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[34] A. Sankar and R. J. Mammone, "Growing and pruning neural tree networks," *IEEE Trans. Comput.*, vol. 42, no. 3, p. 291, 1993.
[35] M. Zwitter and M. Soklic. (1988) Breast cancer data. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRestricted.html
[36] ——, (1988) Lymphography domain. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRestricted.html
[37] ——, (1988) Primary tumor domain. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRestricted.html
[38] D. Mowforth and B. Shepherd. (1988) Vehicle silhouettes. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRestricted.html

Yuanhong Li (S'03) received the B.S. and Ph.D. degrees in condensed state physics from the University of Science and Technology of China, Hefei, China, in 1994 and 1999, respectively. He is now pursuing the Ph.D. degree in the Machine Vision and Pattern Recognition Laboratory, Department of Computer Science, Wayne State University, Detroit, MI.

His research interests include pattern recognition, machine learning, and data clustering.

Ming Dong (S'00–M'02) received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 1995 and the Ph.D. degree from the University of Cincinnati, Cincinnati, Ohio, in 2001, all in electrical engineering.

He joined the Faculty of Wayne State University, Detroit, MI, in 2002 and is currently an Assistant Professor in the Department of Computer Science. He is also Director of the Machine Vision and Pattern Recognition Laboratory. His research interests include pattern recognition, computer vision, multimedia, and financial engineering. He is a member of the Editorial Board of the *International Journal on Semantic Web and Information Systems* and has been a Program Committee Member of various conferences. He is a Board Member of the Association for Information Systems SIG on Semantic Web and Information Systems.

Ravi Kothari (S'89–M'91–SM'99) received the B.E. degree (with distinction) from Birla Institute of Technology, India, the M.S. degree from Louisiana State University, Baton Rouge, and the Ph.D. degree from West Virginia University, Morgantown, all in electrical engineering.

He joined the Department of Electrical and Computer Engineering and Computer Science (ECECS), University of Cincinnati (UC), Cincinnati, OH, in 1992 as an Assistant Professor, where he later became a tenured Associate Professor and Director of the Artificial Neural Systems Laboratory. Since 2002, he has been with IBM-India Research Laboratory, New Delhi, India. His areas of interest include pattern recognition, machine learning, and data mining. He has been an Editorial Board Member of *Pattern Analysis and Applications* and a Program Committee Member of numerous conferences.

Dr. Kothari is a member of Sigma Xi, Eta Kappa Nu, and Phi Kappa Phi. He is an IEEE Distinguished Visitor (2003–2005). He received the William E. Restemeyer Teaching Excellence Award from the Department of ECECS at UC in 1994 and the Eta Kappa Nu Outstanding Professor of the Year Award from the Department of ECECS at UC in 1995. He serves or has served as an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON NEURAL NETWORKS.