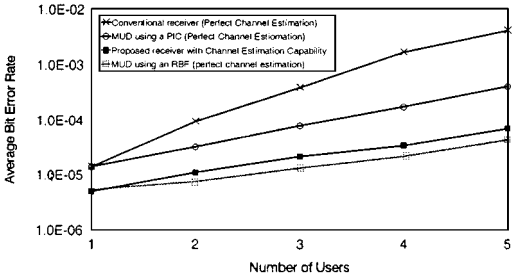

Fig. 6. Performance comparison with respect to the number of users. ($Eb/No = 18$ dB, $L = 5$).

Therefore, we can conclude from results presented in Fig. 5 that the proposed receiver effectively utilizes the frequency diversity. BER comparisons between the proposed receiver and the MUD using an RBF network under conditions of perfect channel estimation, confirm that the channel estimation of the proposed receiver can be properly executed at 15 dB Eb/No.

Fig. 6 shows the BERs for the three receivers plotted against the number of users, where Eb/No is fixed at 18dB and the number of multipaths is fixed at five. This figure illustrates that the performance improvement of the receiver using an RBF network turns out to be greater than that of the MUD with PIC. The reason for this is that the proposed receiver effectively utilizes the signal components of other users, which are considered interference and are canceled in the case of PIC.

VI. Conclusion

In this paper, we proposed a multiuser receiver with channel estimation capability using an RBF network in an MC-CDMA system. Simulations were performed over frequency-selective and multipath Rayleigh fading channels. The RBF network structure showed itself capable not only of permitting the effective utilization of the frequency diversity but also of executing channel estimation under the conditions of a frequency-selective multipath fading channel. Computer simulations demonstrated that the proposed receiver outperforms a MUD with PIC as well as conventional receivers. Furthermore, simulation results showed that a multiuser receiver using an RBF network has the potential to be used for the purpose of increasing the number of active users.

References

[1] S. Chen, B. Mulgrew, and P. M. Grant, "Clusting technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 570–579, July 1993.

[2] S. Chen, S. Mclaughlin, B. Mulgrew, and P. M. Grant, "Adaptive Bayesian decision feedback equalizer for dispersive mobile radio channels," *IEEE Trans. Commun.*, vol. 43, pp. 1937–1946, May 1995.

[3] U. Mitra and H. V. Poor, "Neural network techniques for adaptive multiuser demodulation," *IEEE Trans. Commun.*, vol. 12, pp. 1460–1470, Dec. 1994.

[4] D. G. M. Cruickshank, "Radial basis function receivers for DS-CDMA," *Electron. Lett.*, vol. 32, no. 3, pp. 188–190, 1996.

[5] S. Verdu, *Multiuser Detection*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[6] B. Aazhang, B. Paris, and G. Orsak, "Neural networks for multiuser detection in code-division multiple-access communications," *IEEE Trans. Commun.*, vol. 40, pp. 1212–1222, July 1992.

[7] U. Mitra and H. V. Poor, "Adaptive receiver algorithms for near–far resistant CDMA," *IEEE Trans. Comm.*, vol. 43, pp. 1713–1724, Feb./Mar./Apr. 1995.

[8] K. Ko, S. Choi, C. Kang, and D. Hong, "Simplified multiuser receiver of DS-CDMA system," in *Proc. IJCNN'01*, vol. 3, 2001, pp. 1977–1982.

[9] J. G. Proakis, *Digital Communication*. New York: McGraw-Hill, 1995.

[10] T. Kim, Y. Kim, J. Park, K. Ko, S. Choi, C. Kang, and D. Hong, "Performance of an MC-CDMA system with frequency offsets in correlated fading," in *Proc. IEEE ICC 2000*, vol. 2, 2000, pp. 1095–1099.

[11] S. Hara and R. Prasad, "DS-CDMA, MC-CDMA and MT-CDMA for mobile multimedia communications," in *Proc. IEEE VTC'96*, 1996, pp. 1106–1110.

[12] K. Ko, S. Choi, and D. Hong, "Multistage interference cancellation for an MC-CDMA systems with carrier frequency offset," *Proc. IEEE ICOIN-13*, pp. 4C3.1–4C3.6, 1999.

[13] ——, "Multiuser detector with an ability of channel estimation using an RBF network in an MC-CDMA system," in *Proc. IJCNN 2000*, vol. 5, 2000, pp. 348–353.

[14] W. C. Jakes, *Microwave Mobile Communications*. New York: Wiley, 1974.

Omnivariate Decision Trees

Olcay Taner Yıldız and Ethem Alpaydın

***Abstract*—Univariate decision trees at each decision node consider the value of only one feature leading to axis-aligned splits. In a linear multivariate decision tree, each decision node divides the input space into two with a hyperplane. In a nonlinear multivariate tree, a multilayer perceptron at each node divides the input space arbitrarily, at the expense of increased complexity and higher risk of overfitting. We propose omnivariate trees where the decision node may be univariate, linear, or nonlinear depending on the outcome of comparative statistical tests on accuracy thus matching automatically the complexity of the node with the subproblem defined by the data reaching that node. Such an architecture frees the designer from choosing the appropriate node type, doing model selection automatically at each node. Our simulation results indicate that such a decision tree induction method generalizes better than trees with the same types of nodes everywhere and induces small trees.**

***Index Terms*—Univariate decision trees, multivariate decision trees, neural trees, statistical tests.**

I. Introduction

A decision tree is made up of internal decision nodes and terminal leaves. The input vector is composed of d attributes, $\boldsymbol{x} = [x_1, \ldots, x_d]^T$, and the aim in classification is to assign $\boldsymbol{x}$ to one of K mutually exclusive and exhaustive classes, $\mathcal{C}_1, \ldots, \mathcal{C}_K$. Each internal node m implements a decision function, $f_m(\boldsymbol{x})$, where each branch of the node corresponds to one outcome of the decision. Each leaf of the tree carries a class label. In this work, we use binary decision nodes even when $K > 2$, but the approach we advocate in this paper holds also for trees with K-way splitting nodes.

Starting from the root, at each internal node, $f_m(\boldsymbol{x})$ is calculated and depending on the outcome, the corresponding branch is taken and the

Manuscript received July 1, 2001. This work was supported by Grant 00A101D from Boğaziçi University Research Funds. The work of E. Alpaydın, a Distinguished Young Scientist of the Turkish Academy of Sciences, was supported under the TÜBA/GEBIP program.

The authors are with the Department of Computer Engineering, Boğaziçi University, TR-80815, Istanbul, Turkey.

Publisher Item Identifier S 1045-9227(01)10191-8.

1045–9227/01$10.00 © 2001 IEEE

process is continued recursively until a leaf node is met, at which point the label of the leaf defines the class.

Geometrically, each $f_m(\boldsymbol{x})$ defines a discriminant in the d-dimensional input space dividing it into as many subspaces as there are branches. As one takes a path from the root to a leaf, these subspaces are further subdivided until we end up with a part of the input space which contains the instances of one class only. Different decision tree methods assume different models for the discriminant f_m and the model class defines the shape of the discriminant.

In *univariate* decision trees, the decision at internal node m uses only one attribute, i.e., one dimension of $\boldsymbol{x}$, x_j. If that attribute is numeric, the decision is of the form

$$f_m(\boldsymbol{x}) : x_j + w_{m0} > 0 \tag{1}$$

where w_{m0} is some constant number. This test has two outcomes, true and false, labeling the two branches, left and right, and thus the node is binary. This defines a discriminant which is orthogonal to axis x_j, intersects it at $x_j = -w_{m0}$ and divides the input space into two.

If the attribute is discrete valued with L possible values, $\{a_1, a_2, \ldots, a_L\}$, the decision is of the form

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

This has L outcomes, one for each possible value, and thus there are L branches and the node is L-ary, dividing the input space into L. In a univariate tree, successive decision nodes on a path from the root to a leaf further divide these into two, or L, with splits orthogonal to each other and the leaf nodes define hyperrectangles in the input space.

When the inputs are correlated, looking at one feature may be too restrictive. A *linear multivariate tree*, at each internal node, uses a linear combination of all attributes.

$$f_m(\boldsymbol{x}) : \boldsymbol{w}_m^T \boldsymbol{x} + w_{m0} = \sum_{j=1}^{d} w_{mj} x_j + w_{m0} > 0. \tag{3}$$

To be able to apply the weighted sum, all the attributes should be numeric and discrete values need be represented numerically (usually by one-of-L encoding) before. The weighted sum returns a number and the node is binary. Note that the univariate numeric node is a special case of the multivariate linear node, where all but one of w_{mj} is zero and the other, one. In this linear case, each decision node divides the input space into two with a hyperplane of arbitrary orientation and position where successive decision nodes on a path from the root to a leaf further divide these into two and the leaf nodes define polyhedra in the input space.

In the more general case, one can use a quadratic model as

$$f_m(\boldsymbol{x}) : \boldsymbol{x}^T \mathbf{W}_m \boldsymbol{x} + \boldsymbol{w}_m^T \boldsymbol{x} + w_{m0} = \sum_i \sum_j W_{mij} x_i x_j + \sum_j w_{mj} x_j + w_{m0} > 0. \tag{4}$$

The linear model is a special case where $W_{ij} = 0, \forall i, j = 1, \ldots, d$. Another possibility to get a nonlinear split at a node is to write the decision as a weighted sum of H nonlinear basis functions

$$f_m(\boldsymbol{x}) : \sum_{h=0}^{H} w_{mh} g_{mh}(\boldsymbol{x}) + w_{m0} > 0 \tag{5}$$

where $g_{mh}(\boldsymbol{x})$ are the nonlinear basis functions. The multilayer perceptron (MLP) is such a model where the basis function is the soft-thresholded weighted sum

$$g_{mh}(\boldsymbol{x}) = \frac{1}{1 + \exp\left[-(\boldsymbol{v}_{mh}^T \boldsymbol{x} + v_{mh0})\right]}. \tag{6}$$

It is this model that we are going to take as the *nonlinear multivariate decision tree*. Because MLP is a universal approximator and can approximate any function given sufficiently big H, a decision tree need not have any node more complex than such a node. Note, however, that any other method can be used to train a linear or nonlinear node and our approach is not limited to having neural networks in the decision nodes. The difference between the univariate, linear multivariate and nonlinear multivariate splits is shown on an example in Fig. 1.

II. Training Decision Trees

Training corresponds to constructing the tree given a training set. Finding the smallest decision tree that classifies a training set correctly is NP-hard [14]. For large training sets and input dimensions, even for the univariate case, one cannot exhaustively search through the complete space of possible decision trees. Decision tree algorithms are thus greedy in that at each step, we decide on one decision node. Assuming a model for f_m (univariate, or linear, or nonlinear multivariate), we look for the parameters (w_m coefficients) that best split the data hitting node m, starting with the complete dataset in deciding on the root node. Once we decide on a split, tree construction continues recursively for each child with training instances taking that branch. Surveys about constructing and simplifying decision trees can be found in [6] and [15]. A recent survey comparing different decision tree methods with other classification algorithms is given in [11].

The best split is when all the instances from a class lie on the same side of the decision boundary, i.e., return the same truth value for f_m. There are various measures proposed for measuring the "impurity" of a split; examples are entropy [17] and the Gini index [4]. Murthy *et al.* [14] describe some other impurity indexes. Our results and those of previous researchers indicate that there is no significant difference between these impurity measures.

For constructing univariate decision trees with discrete attributes, Quinlan proposed the *ID3 algorithm* [17] and later generalized it for numeric attributes with the *C4.5 algorithm* [18]. In this univariate case, at each decision node, one can check for all possible splits for all attributes and choose the best as measured by the purity index. For a discrete attribute, there is only one possible split. For a numeric attribute, there are $N_m - 1$ possible splits, where N_m is the number of training instances reaching node m.

In the case of a linear multivariate tree, even the problem of finding the optimal split at a node when optimality is measured in terms of misclassification errors is NP-hard [14]. The problem of finding the best split is then an optimization problem to find the best coefficients, $w_{mj}, j = 0, \ldots, d$, that minimize impurity as defined by the entropy or Gini index. An iterative local search algorithm is used for optimization which does not guarantee optimality and may get stuck in local optima.

The classification and regression trees (CART) algorithm of Breiman *et al.* [4] uses an iterative backfitting algorithm where cycling over coefficients, at each iteration, one coefficient is optimized keeping the other coefficients fixed. In the OC1 algorithm(Oblique trees) proposed by Murthy *et al.* [14], an extension is made to CART to get out of local optima: A small random vector is added to $\boldsymbol{w}_m$ after convergence through backfitting. This perturbs all coefficients together and causes a conjugate jump in the coefficient space. In the linear machine decision trees (LMDT) algorithm, proposed by Brodley and Utgoff [7], with K classes, a node is allowed to have K children. For each child, a separate coefficient vector is used to separate the instances of that class from the other classes. There is an iterative algorithm that adjusts the coefficients to minimize the number of misclassifications, rather than an impurity measure as the entropy or the Gini index. The fast algorithm for classification trees (FACT) of Loh and Vanichsetakul [13] uses parametric discriminant analysis

(Gaussian classes with shared covariance matrix) to implement K-way splits. Loh and Shih's quick unbiased efficient statistical tree (Quest) algorithm [12] also uses parametric discriminant analysis but with binary splits and uses 2-means algorithm to partition K classes to two at each node. Linear discriminant trees, proposed by Yıldız and Alpaydın [23], uses Fisher's linear discriminant to find the coefficients in a binary tree and uses Guo and Gelfand's [10] exchange heuristic to group K classes into two.

III. Neural Trees

Neural trees were introduced to combine neural networks and decision trees. In literature, neural trees can be classified into two groups.

The first group uses decision trees to form the structure of the neural network. The key idea is to construct a decision tree and then convert the tree into a neural network. Sethi [21] produces a three-layered neural network from decision trees, extracting the hidden nodes of the neural network from the decision tree. Brent [5] gives the details, complexity analysis and some practical refinements on this subject. An extension to this idea, which uses linear decision trees as building blocks, can be found in [16]. Cios [8] proposes the CID3 algorithm as a modification of the ID3 algorithm. CID3 creates a hidden layer in a manner similar to the ID3 generation of a decision tree. In the learning process, new hidden layers are added by the CID3 algorithm to the network until a learning task becomes linearly separable at the output layer. Cauchy training is used to train the resulting hybrid structure. Golea and Marchand [9] propose a linearly separable neural-network decision tree architecture to learn a given but arbitrary Boolean function.

The second group uses neural networks as building blocks in decision trees. The nonlinear multivariate decision tree with multilayer perceptrons at the internal nodes was proposed by Guo and Gelfand [10]. They also proposed a heuristic to group $K > 2$ classes into two, which is necessary as the nodes in the tree are binary. Thus they use a nested optimization problem where in the inner optimization, gradient-descent is used to find the weights that minimize the mean-square error as usual in training neural networks and so find a good split for the given two distinct groups of classes. In the outer optimization problem, exchange heuristic is used to find the best split of K classes into two groups through a local search with backtracking, with time complexity $\mathcal{O}(K^2)$. This same algorithm can also be used with single layer perceptrons instead of multilayer perceptrons thereby generating *linear* multivariate decision nodes. This was first introduced as neural tree networks by Sankar and Mammone [20]. They also introduced a new pruning algorithm which uses a Langrangian cost function [19]. The comparison of this latter method with other linear multivariate decision tree construction methods is given in [23]. Behnke and Karayiannis use competitive learning to form a competitive decision tree architecture named CNET [2]. A hybrid form which contains neural networks at the leaves of the tree and univariate nodes in the nonleaf nodes of the tree was proposed by Utgoff [22].

IV. Pruning

A greedy algorithm is a local search method where at each step, one tries to make the best decision and proceeds to the next decision, never backtracking and reevaluating a decision after it has been made. Similarly in decision tree induction, once a decision node is fixed, it cannot be changed after its children have been created. This may cause suboptimal trees where for example subtrees are replicated. The only exception is the *pruning* of the tree.

In pruning, we consider replacing a subtree with a leaf node labeled with the class most heavily represented among the instances that are covered by the subtree. If there is overfitting, we expect the more complex subtree to learn the noise and perform worse than the simple leaf. If this is indeed the case on a validation set different from the training set, then the subtree is replaced by the leaf. Otherwise it is kept. It makes sense to start with the smaller subtrees closer to leaves and proceed up toward the root.

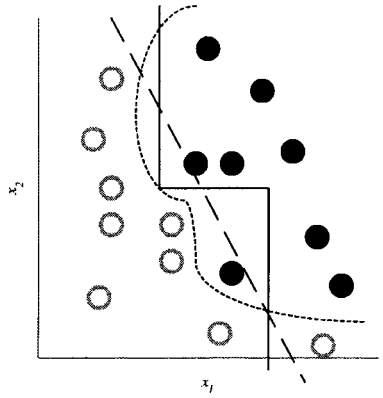

Fig. 1. Example univariate (continuous line), linear multivariate (dashed line), and nonlinear multivariate (dotted line) splits that separate instances of two classes.

This process is called *postpruning* to differentiate it from *prepruning*. In postpruning, the tree is constructed until there is no misclassification error and then pruned simpler. In prepruning, the tree is not fully constructed until zero training error but is kept simple by early termination. At any node, if the dataset reaching that node is small, even if it is not pure, it is not further split and a leaf node is created instead of growing a subtree. Prepruning is faster. Postpruning may be more accurate but is slower and requires a separate validation set.

V. Tuning the Node Complexity Automatically: Omnivariate Decision Trees

In approximating the real (unknown) discriminant, with univariate nodes we are limited to a piecewise approximation using axis-aligned hyperplanes. With multivariate linear nodes, we can use arbitrary hyperplanes and thus approximate the discriminant better. In Fig. 1 for example, one linear multivariate node is used instead of three univariate nodes. It is clear that if the underlying discriminant is curved, a nonlinear approximation through a nonlinear multivariate node allows a better approximation using a smaller number of nodes and leaves. Thus there is a dependency between the complexity of a node and the size of the tree. With complex nodes the tree may be quite small; with simple nodes one may grow large trees.

However we should keep in mind that an MLP node has $\mathcal{O}(d \times H)$ parameters, compared to linear's $\mathcal{O}(d)$ and univariate's $\mathcal{O}(1)$. A complex model with a larger number of parameters requires a larger training dataset and risks overfitting on small amount of data. For example in Fig. 1, the nonlinear split has less error than the linear split but is too wiggly. Thus one should be careful in tuning the complexity of a node with the properties of the data reaching that node.

Each node type has a certain bias; using multivariate linear nodes for example, we are assuming that the input space can be divided using hyperplanes into localized regions (volumes) where classes, or groups of classes are linearly separable. Using a decision tree with the same type of nodes everywhere, we assume that the same bias is appropriate at all levels.

This paper advocates the view that this assumption is not always correct and that at each node of the tree, which corresponds to a different subproblem defined by the subset of the training data reaching that node, a different model may be appropriate, and that the right model

TABLE I
DETERMINATION OF THE WINNER NODE TYPE AS A RESULT OF THE STATISTICAL TEST RESULTS OF THE COMPARISON OF THREE NODE TYPES. IF LIN>UNI IS TRUE, THEN THE TEST DETECTS THAT THE LINEAR MULTIVARIATE SPLIT IS BETTER THAN THE UNIVARIATE SPLIT. NONLIN IS THE NONLINEAR MULTIVARIATE SPLIT

Lin>Uni	Nonlin>Uni	Nonlin>Lin	Winner
False	False	False	Uni
False	False	True	Uni
False	True	False	Nonlin
False	True	True	Nonlin
True	False	False	Lin
True	False	True	Nonlin
True	True	False	Lin
True	True	True	Nonlin

should be found and used. For example, we expect that though closer to the root a nonlinear model may be used, as we get closer to the leaves, we have easier problems in effectively smaller dimensional subspaces and at the same time, we have smaller training data and simple, e.g., univariate, splits may suffice and generalize better. Our results given in the next section support our intuition. We name this hybrid structure an *omnivariate decision tree* as this type of decision tree embraces all variants.

In our proposed omnivariate decision tree, at each node, we train and compare all three possible nodes; univariate, linear multivariate, and nonlinear multivariate, and using a statistical test, we choose the best and continue tree induction recursively. Each node implements a binary split to induce simple and interpretable trees. To group $K > 2$ classes into two, we use Guo and Gelfand's [10] exchange heuristic which uses class information and thus is better than the unsupervised 2-means algorithm used by Loh and Shih in Quest [12]. Postpruning is used to simplify the tree after induction for better generalization.

In the OC1 algorithm [14], at each decision node, both a univariate and a multivariate linear split is found and the latter is used if its impurity is less than that of the former. However, this compares the two on the training set whereas one should make the decision using cross-validation through a statistical test, as we propose here.

VI. CHOOSING THE BEST OF THREE MODELS

Statistical tests in the literature, and the 5×2cv F test we use [1], compares two models and to be able to choose the best of more than two models, we need a methodology. Given three models M_i, M_j, and M_k, we may not have a full order and a clear winner as M_i is better than both M_j and M_k, and M_j better than M_k : $M_i > M_j, M_i > M_k, M_j > M_k$. In the absence of a clear winner among the three due to no statistically significant difference between the methods, we use our prior information and prefer the simpler model.

Table I shows the eight possible results of the statistical tests between the three node types; univariate, linear multivariate, and nonlinear multivariate. The winner node type is chosen to satisfy two criteria: 1) results of the tests and 2) Our prior preferrence to choose the simpler model unless the test, based on the data, chooses the more complex model. We apply three tests to see at each test if the more complex model is better than the simpler model or not. If the test returns false, this may be either because 1) there is no difference between the models or 2) it may be because the simpler model is better. In both cases, we choose the simpler model; in 1) due to its simplicity and in 2) due to its being more accurate.

We choose the best after three such pairwise tests are made. For example, if the linear multivariate model is better than the univariate model and if the nonlinear multivariate model is better than the univariate model, but if the nonlinear model is not better than the linear model (case 7 of Table I), we select the linear model in that node.

TABLE II
DESCRIPTION OF THE DATASETS. K IS THE NUMBER OF CLASSES, N IS THE DATASET SIZE, AND d IS THE NUMBER OF INPUTS

Set	K	N	d	Missing	Attr Type
BAL	3	625	5	No	symbolic
BRE	2	699	10	Yes	numeric
BUP	2	345	7	No	numeric
CAR	4	1728	7	No	symbolic
CMC	3	1473	10	No	mixed
CRE	2	690	16	No	mixed
CYL	2	541	36	Yes	mixed
DER	6	366	35	Yes	numeric
ECO	8	336	8	No	numeric
FLA	3	323	11	No	mixed
GLA	7	214	10	No	numeric
HAB	2	306	4	No	numeric
HEP	2	155	20	Yes	numeric
HOR	2	368	27	Yes	mixed
IRI	3	150	5	No	numeric
IRO	2	351	35	No	numeric
MON	2	432	7	No	numeric
MUS	2	8124	23	Yes	symbolic
NUR	5	12960	9	No	symbolic
OCR	10	3823	64	No	numeric
PEN	10	7494	16	No	numeric
PIM	2	768	9	No	numeric
SEG	7	2310	19	No	numeric
SPA	2	4601	58	No	numeric
TIC	2	958	10	No	symbolic
VOT	2	435	17	Yes	symbolic
WAV	3	5000	22	No	numeric
WIN	3	178	14	No	numeric
YEA	10	1484	9	No	numeric
ZOO	7	101	17	No	numeric

VII. EVALUATION

To compare our proposed omnivariate decision tree architecture with pure univariate, linear multivariate, and nonlinear multivariate trees, we tested all four methods on 30 datasets from the UCI repository [3]. Table II describes the properties of the data sets.

The pure univariate tree is constructed using C4.5 algorithm. The linear multivariate tree is constructed with a single-layer perceptron at each decision node. The nonlinear multivariate tree is constructed with a multilayer perceptron at each node having d inputs, one output, and $(d+1)/2$ hidden units. (Number of inputs + number of outputs)$/2$ is a common heuristic in determining the number of hidden units in an MLP. We have also made tests with $d+1$ and $3(d+1)/2$ hidden units and noticed no significant difference in accuracy and tree size.

Both of the multivariate trees use binary nodes and use Guo and Gelfand's [10] exchange heuristic to group $K > 2$ classes into two groups. Discrete attributes are 1-of-L encoded numerically before using these two methods. The omnivariate tree, if it chooses a univariate node, uses a discrete attribute as it is. Postpruning is used for tree simplification in all four architectures.

The test we use is the combined 5×2cv F Test [1] which performs five two-fold cross-validation runs on each data set. The results of the ten runs are then averaged and we report the mean and standard deviation of accuracy (Table III), tree size in terms of the number of nodes

TABLE III
THE FIRST TABLE GIVES THE ACCURACY RESULTS. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT RUNS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATASETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j

Set	Uni	Lin Mul	Nonlin	Omni
BAL	65.5,3.0	91.6, 2.0	91.7, 2.0	92.7, 2.3
BRE	94.6,1.8	96.2, 0.7	96.2, 0.8	95.5, 1.5
BUP	62.8,3.3	64.1, 3.8	57.9, 1.3	64.1, 3.9
CAR	86.0,1.6	92.7, 1.7	97.0, 1.4	94.3, 2.2
CMC	52.6,3.1	44.6, 2.2	44.0, 2.6	51.7, 2.0
CRE	84.7,1.0	82.5, 2.2	83.8, 1.6	82.5, 2.1
CYL	67.2,4.6	69.2, 2.7	69.5, 5.0	68.1, 3.2
DER	92.5,2.4	97.2, 1.0	96.6, 0.8	92.4, 3.1
ECO	78.2,4.0	80.0, 4.1	80.3, 4.1	78.4, 4.3
FLA	88.3,2.5	88.0, 2.6	88.8, 2.4	88.8, 2.5
GLA	60.0,5.5	57.3, 7.4	56.7, 4.5	58.8, 7.0
HAB	71.9,3.6	73.3, 3.2	73.4, 3.1	72.4, 3.8
HEP	78.9,4.4	81.5, 4.4	81.2, 4.0	79.9, 3.3
HOR	73.8,6.7	79.8, 6.6	79.6, 6.5	78.6, 6.2
IRI	92.9,3.3	95.4, 2.3	85.8,14.2	93.1, 3.6
IRO	86.1,3.7	87.8, 2.8	89.1, 1.7	87.7, 4.0
MON	89.8,7.7	72.3, 5.6	67.2,10.7	88.2, 8.0
MUS	99.8,0.1	99.9, 0.0	99.9, 0.0	100.0, 0.0
NUR	94.4,0.4	93.4, 0.6	95.6, 9.5	99.6, 0.2
OCR	84.8,0.8	92.4, 0.8	94.6, 0.8	92.4, 0.9
PEN	92.5,0.6	94.9, 2.7	94.8, 4.6	95.6, 3.8
PIM	70.7,2.9	73.9, 1.3	75.2, 1.5	72.1, 2.9
SEG	92.0,0.9	87.5,10.6	86.8,12.5	92.8, 1.2
SPA	91.4,0.7	91.1, 0.6	93.0, 0.7	92.5, 0.9
TIC	78.7,1.8	97.9, 0.7	97.8, 0.6	97.5, 0.8
VOT	95.6,0.6	95.4, 0.9	95.3, 0.7	95.6, 1.1
WAV	75.9,0.6	82.8, 0.9	82.4, 8.0	83.0, 1.8
WIN	86.6,1.9	96.0, 1.9	95.0, 2.1	89.6, 3.2
YEA	54.4,3.0	47.7, 4.4	51.9, 3.6	52.6, 1.9
ZOO	82.9,7.3	79.2, 9.5	79.6, 8.2	83.1, 6.4

Method	Uni	Lin Mul	Nonlin	Omni	$\sum$
Uni	-	0	1	0	1
Lin Mul	6	-	0	0	6
Nonlin	6	3	-	0	7
Omni	6	1	1	-	7
$\sum$	8	4	2	0	

TABLE IV
THE FIRST TABLE GIVES THE TREE SIZE IN TERMS OF THE NUMBER OF NODES IN THE TREE. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT RUNS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATASETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j

Set	Uni	Lin Mul	Nonlin	Omni
BAL	38.0,11.5	6.0, 4.0	7.0, 4.1	5.5, 2.4
BRE	13.0, 5.0	3.2, 0.6	3.0, 0.0	4.6, 3.5
BUP	17.4,12.5	4.0, 3.0	1.0, 0.0	21.2,13.6
CAR	89.4,10.4	15.4, 6.6	7.0, 0.0	22.6, 9.3
CMC	120.0,26.6	38.2,14.4	15.0,11.7	79.6,15.7
CRE	21.2,13.7	7.8, 5.3	3.2, 0.6	11.0, 8.3
CYL	20.8, 8.7	9.4, 4.0	3.4, 1.3	23.3, 8.9
DER	12.4, 1.3	11.0, 0.0	11.0, 0.0	12.8, 2.6
ECO	14.2, 4.6	9.6, 2.1	8.6, 1.3	12.2, 5.5
FLA	4.7, 6.0	2.6, 2.6	1.0, 0.0	1.0, 0.0
GLA	14.2, 4.0	8.6, 3.1	6.6, 1.8	14.0, 6.3
HAB	8.8,10.5	2.4, 2.3	1.4, 0.8	7.0,10.1
HEP	2.8, 2.4	2.0, 1.1	1.8, 1.0	3.4, 3.6
HOR	37.5,19.5	3.6, 1.3	3.2, 1.1	4.6, 5.8
IRI	5.4, 0.8	5.0, 0.0	4.6, 0.8	5.6, 1.0
IRO	7.6, 2.7	4.4, 1.3	3.2, 0.6	6.2, 3.2
MON	26.8,12.3	8.4, 5.2	3.2, 1.5	26.0, 6.9
MUS	26.8, 2.0	3.0, 0.0	3.0, 0.0	3.0, 0.0
NUR	351.4,41.7	88.2,14.9	7.0, 3.0	16.4, 1.3
OCR	107.4, 9.9	26.8, 4.4	19.0, 0.0	36.2, 0.9
PEN	134.4,13.5	46.6,10.2	19.6, 2.1	37.0, 8.0
PIM	27.0,14.5	9.4, 6.9	3.4, 0.8	20.8, 9.9
SEG	42.8, 7.0	20.6, 6.9	12.8, 2.6	37.2, 6.6
SPA	75.4,20.7	27.4, 9.3	3.2, 0.6	40.8,30.7
TIC	68.8, 9.3	3.4, 1.3	3.0, 0.0	5.9, 4.7
VOT	4.0, 2.2	3.2, 0.6	3.0, 0.0	4.2, 2.7
WAV	143.8,21.7	39.8,15.1	12.0, 5.4	65.8,32.5
WIN	6.8, 2.6	5.0, 0.0	5.0, 0.0	6.4, 1.6
YEA	75.6,21.2	34.4,13.9	18.4, 4.8	72.0,19.3
ZOO	9.2, 2.4	7.8, 2.3	7.6, 1.3	9.0, 2.3

Method	Uni	Lin Mul	Nonlin	Omni	$\sum$
Uni	-	0	0	1	1
Lin Mul	14	-	0	4	16
Nonlin	16	5	-	9	18
Omni	10	1	0	-	10
$\sum$	17	5	0	10	

(Table IV) and the number of parameters (Table V), and learning time in seconds on a Pentium III-600 (Table VI).

For each comparison, there are two tables where in the first table the raw results are shown and the second table contains pairwise comparisons; the entry (i, j) in this second table gives the number of datasets on which method i is statistically significantly better than method j with 95% confidence. In the second table, row and column sums are also given. The row sum gives the number of datasets out of 30 where the algorithm on the row outperforms at least one of the other algorithms. The column sum gives the number of datasets where the algorithm on the column is outperformed by at least one of the other algorithms.

A. Accuracy

Comparing univariate, linear, and nonlinear multivariate trees among themselves, we see that on 22 datasets out of 30, the univariate tree is as accurate as the others (Table III), showing that a univariate split is good enough in most cases. On six datasets (CAR, OCR, WIN, BAL, TIC, WAV), the linear multivariate tree is more accurate than the univariate tree and on six datasets (CAR, OCR, PEN, WIN, BAL, TIC), the nonlinear multivariate tree is more accurate than the univariate tree. The nonlinear one is better than the linear only on three datasets (CAR, OCR, PIM). These indicate that a multivariate split is sometimes better. But on WAV, although the linear tree is more accurate than the univariate tree, the nonlinear tree is not more accurate than the univariate tree; as can be seen by the high standard deviation value of 8.0, the nonlinear multivariate tree overfits here.

The omnivariate model outperforms the univariate tree on six datasets (CAR, OCR, BAL, NUR, TIC, WAV) which is almost the union of the datasets on which the linear and nonlinear multivariate trees outperform the univariate tree, indicating that the omnivariate tree includes as its special cases all three possibilities, using whichever is more accurate at each node. The omnivariate tree outperforms the linear multivariate on one dataset (NUR), and the nonlinear multivariate one on one dataset (CMC); this latter is also the dataset where the univariate is better than the nonlinear multivariate.

TABLE V
THE FIRST TABLE GIVES THE TREE SIZE IN TERMS OF THE NUMBER OF FREE PARAMETERS IN THE TREE. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT RUNS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATASETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j

Set	Uni	Lin Mul	Nonlin	Omni
BAL	46,14	56, 44	700, 479	44, 8
BRE	19, 7	13, 3	58, 0	22, 19
BUP	26,19	13, 12	1, 0	33, 21
CAR	134,16	167, 76	766, 0	291, 158
CMC	168,38	485,187	2290,1919	270, 106
CRE	26,16	161,124	1192, 342	226, 368
CYL	31,13	299,141	2985,1573	119, 99
DER	18, 2	181, 0	3161, 0	29, 17
ECO	21, 7	40, 10	145, 24	29, 14
FLA	7, 9	21, 33	1, 0	1, 0
GLA	21, 6	43, 17	161, 52	21, 10
HAB	13,16	5, 6	3, 5	10, 15
HEP	4, 4	12, 11	86, 109	5, 5
HOR	56,29	130, 67	5339,2755	92, 33
IRI	8, 1	13, 0	32, 7	8, 2
IRO	11, 4	62, 24	696, 200	103, 193
MON	40,19	31, 21	34, 22	42, 12
MUS	40, 3	69, 0	2281, 0	69, 0
NUR	459,54	1265,217	1225, 608	792, 10
OCR	161,15	852,144	19324, 0	4613,2553
PEN	201,20	411, 92	1443, 164	548, 106
PIM	40,22	43, 34	57, 20	53, 35
SEG	64,10	197, 69	1134, 247	134, 28
SPA	113,31	780,275	1885, 542	4380,5589
TIC	92,12	36, 18	409, 0	47, 33
VOT	6, 3	38, 11	564, 0	37, 4
WAV	215,33	447,174	1404, 683	822, 416
WIN	10, 4	31, 0	215, 0	32, 44
YEA	113,32	168, 69	410, 113	247, 77
ZOO	13, 4	62, 21	513, 105	15, 7

Method	Uni	Lin Mul	Nonlin	Omni	$\sum$
Uni	-	11	19	11	22
Lin Mul	0	-	16	2	17
Nonlin	0	0	-	1	1
Omni	0	4	14	-	15
$\sum$	0	11	19	12	

TABLE VI
THE FIRST TABLE GIVES THE LEARNING TIME IN SECONDS ON A PENTIUM III-600. VALUES ARE AVERAGE AND STANDARD DEVIATIONS OF TEN INDEPENDENT RUNS. THE SECOND TABLE CONTAINS PAIRWISE COMPARISONS WHERE (i, j) VALUES ARE THE NUMBER OF DATASETS ON WHICH MODEL i IS STATISTICALLY SIGNIFICANTLY BETTER THAN MODEL j

Set	Uni	Lin Mul	Nonlin	Omni
BAL	2, 0	2, 1	7, 5	48, 6
BRE	1, 1	1, 0	0, 0	9, 2
BUP	3, 0	1, 0	0, 0	12, 3
CAR	11, 1	17, 6	22, 5	326, 34
CMC	24, 2	45, 16	54, 36	451, 57
CRE	5, 0	6, 1	6, 1	191, 19
CYL	12, 1	9, 2	10, 3	243, 41
DER	3, 0	7, 1	17, 5	184, 16
ECO	3, 0	4, 1	3, 0	36, 9
FLA	3, 1	1, 1	1, 0	16, 7
GLA	4, 1	2, 1	2, 0	26, 6
HAB	1, 0	0, 0	0, 0	15, 4
HEP	2, 0	0, 0	0, 0	7, 1
HOR	6, 1	4, 1	12, 5	141, 40
IRI	1, 0	0, 0	0, 0	4, 1
IRO	18, 4	1, 0	1, 0	65, 14
MON	12, 6	1, 0	0, 0	7, 2
MUS	12, 2	136, 72	272, 85	2077, 32
NUR	245, 11	664,111	404,151	4305, 398
OCR	5303,1493	662,153	1233,393	26184,4240
PEN	617, 32	823,167	481,178	7445,1133
PIM	9, 1	4, 1	1, 0	63, 5
SEG	352, 15	85, 15	69, 9	1837, 184
SPA	789, 53	167, 41	40, 11	4743, 930
TIC	5, 0	2, 1	2, 0	57, 16
VOT	8, 2	2, 1	1, 0	22, 7
WAV	1272, 34	142, 37	64, 25	6496, 449
WIN	4, 1	1, 0	1, 0	10, 2
YEA	28, 1	47, 9	36, 9	636, 51
ZOO	4, 1	3, 0	2, 0	25, 11

Method	Uni	Lin Mul	Nonlin	Omni	$\sum$
Uni	-	3	2	27	27
Lin Mul	14	-	1	30	30
Nonlin	15	6	-	30	30
Omni	1	0	0	-	1
$\sum$	16	9	2	30	

The omnivariate tree is never outperformed by any of the pure trees on any dataset, showing that when it is not more accurate, it is as accurate. This justifies our claim that it is not good to assume the same bias on all levels of the tree and it is best to match the model complexity at each level with each subproblem, i.e., the data arriving to that node. Thus the model takes the best of the three models; the simplicity of univariate nodes when additional complexity is not justified and the power of linear or nonlinear multivariate nodes when it is.

B. Tree Size

In Table IV, we compare the sizes of the trees induced by the four methods in terms of the number of nodes in the tree. We also give the tree sizes in terms of free parameters because the nodes have different complexities (Table V). A univariate node has one or two parameters (the attribute index and the threshold for a numeric attribute), a linear multivariate node has $d + 1$ parameters and a nonlinear multivariate node has $(d^2+3d+4)/2$ parameters. We see that in terms of the number of nodes, the ordering is Uni>Omni>Lin>Nonlin but in terms of the number of parameters, the ordering is exactly the opposite: The univariate tree has a large number of simple nodes and in the other extreme, the nonlinear multivariate tree has a small number of very complex nodes. The omnivariate tree uses univariate nodes unless the additional complexity is justified and heavily uses simple univariate nodes.

Looking at Table V, the 19 datasets on which the univariate trees are smaller than the nonlinear trees include the 16 datasets on which the linear trees are smaller than the nonlinear trees. Similarly, the four datasets (DER, IRI, ZOO, NUR) on which the omnivariate trees are smaller than the linear trees are among the 11 datasets on which the univariate trees are smaller than the linear trees and the 14 datasets on which the omnivariate trees are smaller than the nonlinear trees are among the 19 datasets on which the univariate trees are smaller than the nonlinear trees. These 14 datasets are the union of the datasets on which either the univariate tree or linear tree is smaller than the nonlinear tree. This fact indicates again that the omnivariate tree sticks to simple nodes, mostly univariate and sometimes linear, rarely resorting to a complex nonlinear node.

C. Learning Time

When τ is the number of nodes and N is the number of instances, training a univariate tree takes $\mathcal{O}(\tau \cdot N \cdot d)$ time. When training takes e

TABLE VII
NUMBER OF UNIVARIATE, LINEAR AND NONLINEAR NODES IN THE OMNIVARIATE TREES

Set	Uni	Linear	Nonlinear
BAL	0.2, 0.6	1.9, 0.3	0.0, 0.0
BRE	1.0, 2.2	0.6, 0.5	0.2, 0.4
BUP	9.8, 6.7	0.3, 0.5	0.0, 0.0
CAR	3.6, 2.3	5.4, 2.1	0.6, 0.7
CMC	31.6, 7.0	5.2, 2.0	0.1, 0.3
CRE	1.1, 1.5	2.3, 0.9	0.1, 0.3
CYL	5.8, 2.7	1.3, 1.3	0.0, 0.0
DER	5.6, 1.3	0.3, 0.5	0.0, 0.0
ECO	4.3, 2.7	1.2, 1.1	0.1, 0.3
FLA	0.0, 0.0	0.0, 0.0	0.0, 0.0
GLA	6.5, 3.2	0.0, 0.0	0.0, 0.0
HAB	2.9, 5.1	0.1, 0.3	0.0, 0.0
HEP	1.2, 1.8	0.0, 0.0	0.0, 0.0
HOR	0.2, 0.6	0.9, 0.3	0.0, 0.0
IRI	2.2, 0.6	0.1, 0.3	0.0, 0.0
IRO	1.6, 1.6	0.9, 0.3	0.1, 0.3
MON	11.9, 3.5	0.6, 1.1	0.0, 0.0
MUS	0.0, 0.0	1.0, 0.0	0.0, 0.0
NUR	4.0, 0.0	1.1, 0.3	1.0, 0.0
OCR	8.6, 4.2	7.1, 1.5	1.9, 1.2
PEN	8.0, 4.3	7.6, 1.3	2.4, 0.7
PIM	9.3, 4.8	0.3, 0.5	0.3, 0.5
SEG	13.5, 3.0	4.6, 1.3	0.0, 0.0
SPA	17.5,14.2	1.1, 0.9	1.3, 1.7
TIC	0.7, 1.3	1.5, 1.1	0.0, 0.0
VOT	0.6, 1.3	1.0, 0.0	0.0, 0.0
WAV	28.4,16.4	2.5, 1.1	1.5, 1.0
WIN	2.4, 1.2	0.1, 0.3	0.2, 0.4
YEA	31.8, 9.1	3.2, 1.9	0.5, 0.5
ZOO	3.9, 1.1	0.1, 0.3	0.0, 0.0
$\sum$	218.2	52.3	10.3
%	77.7	18.6	3.7

epochs, training a linear multivariate tree is $\mathcal{O}(\tau \cdot N \cdot d \cdot e \cdot K^2)$ time and training a nonlinear multivariate tree is $\mathcal{O}(\tau \cdot N \cdot d^2 \cdot e \cdot K^2)$ time. However, because the univariate tree has more nodes, the overall training time may be less with multivariate trees. Postprunining also has more effect on univariate trees; the multivariate trees are almost not affected by pruning. The nonlinear multivariate tree thus are induced fastest and then it is the linear multivariate and then the univariate tree (Table VI).

Because, to be able to choose the best, all three nodes should be trained and because the test uses 5×2 cross-validation, the only drawback of omnivariate tree induction is learning time, with complexity $\mathcal{O}(10 \cdot \tau \cdot N/2 \cdot (d + deK^2 + d^2eK^2))$. Note, however, that the 5×2 validations, as well as the training of univariate, linear, and nonlinear nodes, are independent and can effectively be parallelized on a multiprocessor system.

The number of univariate, linear, and nonlinear nodes in the omnivariate trees are shown in Table VII. We see that a large majority (77.7%) of the nodes are univariate, which, our analyzes of the induced trees indicate, are closer to the leaves. 18.6% of the nodes are linear and only 3.7% are nonlinear. The linear and nonlinear nodes are closer to the root, indicating that an early complex discriminant is helpful (Fig. 2). But as we go down the tree and work on subproblems confined to a small subspace of the input space, simple univariate splits generalize the best.

This can also be seen in Table VIII where we see that in 77.4% of the cases, linear or nonlinear multivariate splits are not statistically significantly superior to the univariate split (row 1 of Table I) and thus the univariate split is chosen due to its simplicity. In 17.3% of the

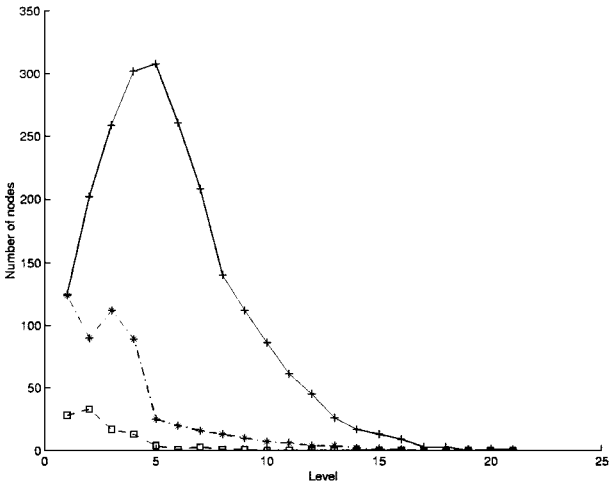

Fig. 2. The number of nonlinear, linear, and univariate nodes in all trees as a function of level in the tree. $+$: univariate, *: linear, $\square$: nonlinear. We see that complex nodes are used early in the tree.

TABLE VIII
NUMBER OF TIMES STATISTICAL TEST CASES OF TABLE I ARE SEEN IN THE DATASETS DURING THE INDUCTION OF OMNIVARIATE TREES. 1–8 CORRESPOND TO ROWS OF TABLE I

Set	1	2	3	4	5	6	7	8
BAL	2	0	0	0	0	0	19	0
BRE	10	0	2	0	1	0	5	0
BUP	98	0	0	0	2	0	1	0
CAR	33	3	5	0	0	0	54	1
CMC	316	0	0	1	1	0	51	0
CRE	11	0	0	0	0	0	23	1
CYL	58	0	0	0	0	0	13	0
DER	55	1	0	0	3	0	0	0
ECO	43	0	1	0	3	0	9	0
FLA	0	0	0	0	0	0	0	0
GLA	65	0	0	0	0	0	0	0
HAB	29	0	0	0	0	0	1	0
HEP	12	0	0	0	0	0	0	0
HOR	2	0	0	0	0	0	9	0
IRI	22	0	0	0	0	0	1	0
IRO	16	0	1	0	3	0	6	0
MON	119	0	0	0	2	0	4	0
MUS	0	0	0	0	0	0	10	0
NUR	40	0	0	0	0	0	11	10
OCR	84	2	0	0	0	0	71	19
PEN	79	1	5	2	5	0	71	17
PIM	92	1	3	0	0	0	3	0
SEG	135	0	0	0	2	0	44	0
SPA	174	1	4	1	2	0	11	6
TIC	7	0	0	0	0	0	15	0
VOT	6	0	0	0	0	0	10	0
WAV	283	1	14	1	5	0	20	0
WIN	24	0	2	0	0	0	1	0
YEA	318	0	5	0	9	0	23	0
ZOO	39	0	0	0	1	0	0	0
$\sum$	2172	10	42	5	39	0	486	54
%	77.4	0.3	1.5	0.2	1.4	0.0	17.3	1.9

cases, we have both the linear and nonlinear splits better than the univariate and no significant difference between the linear and nonlinear and we choose the linear split (row 7 of Table I). We also see that case 6 (Lin$>$Uni, Nonlin$\not>$Uni, Nonlin$>$Lin) never occurs; this indicates that we have a good test. Case 3, which is another problematic case (Lin$\not>$Uni, Nonlin$>$Uni, Nonlin$\not>$ Lin), is met in 1.5% of the cases,

which shows our prior belief of Uni>Lin>Nonlin is true for 98.4% of the cases. Though in Table I in four cases out of eight, the nonlinear split is chosen, these four cases happen very rarely in data and thus the nonlinear split is chosen in only 3.7% of the cases.

In terms of computation during test, a univariate node is $\mathcal{O}(1)$, a linear multivariate node is $\mathcal{O}(d)$, and a nonlinear multivariate node (when implemented as an MLP with $\mathcal{O}(d)$ hidden units) is $\mathcal{O}(d^2)$. Thus an omnivariate tree with a large percentage of univariate nodes takes much less time to give an output than pure linear or nonlinear multivariate trees.

VIII. Conclusion

We propose a novel decision tree architecture, the omnivariate decision tree, which is a hybrid tree that contains both univariate, linear multivariate, and nonlinear multivariate nodes. Though we have used neural networks in the decision nodes for the linear and nonlinear nodes, some other method can also be used to train a linear or nonlinear node and our approach is not limited to having neural networks in the decision nodes. The ideal node type is determined by taking into account our preference due to simplicity and the results of statistical tests comparing accuracy. Such a tree, instead of assuming the same bias at each node, matches the complexity of a node with the data reaching that node. Our simulation results indicate that such an architecture generalizes better than trees with the same types of nodes everywhere and generates smaller trees. The only handicap is the longer training time, but the test runs can be parallelized very easily on a parallel processor system. We believe that with processors getting faster and cheaper, omnivariate architectures with built-in automatic model selection will become more popular as they free the designer from choosing the appropriate structure.

Acknowledgment

The authors would like to thank the editor and reviewers for their constructive comments on the previous version of the manuscript.

References

[1] E. Alpaydın, "Combined 5×2cv F test for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 11, pp. 1975–1982, 1999.

[2] S. Behnke and B. Karayiannis, "Competitive neural trees for pattern classification," *IEEE Trans. Neural Networks*, vol. 9, pp. 1352–1369, 1998.

[3] C. Blake and C. J. Merz. (1998) UCI Repository of Machine Learning Databases. [Online]. Available: http://www.ics.uci.edu/~mlearn/ML-Repository.html

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

[5] R. P. Brent, "Fast training algorithms for multilayer neural nets," *IEEE Trans. Neural Networks*, vol. 2, pp. 346–354, 1991.

[6] L. A. Breslow and D. W. Aha, "Simplifying Decision Trees: A Survey," Navy Center for Applied Research in AI, Naval Research Laboratory, Washington, DC, NCARAI Tech. Rep. AIC-96-014, 1997.

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

[8] K. J. Cios, "A machine learning method for generation of a neural-network architecture: A continuous ID3 algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 280–291, 1992.

[9] M. Golea and M. Marchand, "A growth algorithm for neural-network decision trees," *Europhys. Lett.*, vol. 12, pp. 205–210, 1990.

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

[11] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of 33 old and new classification algorithms," *Machine Learning*, vol. 40, pp. 203–228, 2000.

[12] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica Sinica*, vol. 7, pp. 815–840, 1997.

[13] W.-Y. Loh and N. Vanichsetakul, "Tree-structured classification via generalized discriminant analysis," *J. Amer. Statist. Assoc.*, vol. 83, no. 403, pp. 715–725, 1988.

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

[15] S. K. Murthy, "Automatic construction of decision trees from data: A multidisciplinary survey.," *Data Mining and Knowledge Discovery*, vol. 4, pp. 345–389, 1998.

[16] Y. Park, "A comparison of neural-net classifiers and linear tree classifiers: Their similarities and differences," *Pattern Recognition*, vol. 27, pp. 1493–1503, 1994.

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

[18] ——, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[19] A. Sankar and R. J. Mammone, "Optimal pruning of neural tree networks for improved generalization," in *Proc. Int. Joint Conf. Neural Networks*, Seattle, WA, July 8–12, 1991, pp. 219–224.

[20] ——, "Growing and pruning neural tree networks," *IEEE Trans. Comput.*, vol. 42, pp. 291–299, 1993.

[21] I. K. Sethi, "Entropy nets: From decision trees to neural networks," *Proc. IEEE*, vol. 78, pp. 1605–1613, 1990.

[22] P. E. Utgoff, "Perceptron trees: A case study in hybrid concept representations," *Connection Sci.*, vol. 1, pp. 377–391, 1989.

[23] O. T. Yıldız and E. Alpaydın, "Linear discriminant trees," in *Proc. 17th Int. Conf. Machine Learning*, P. Langley, Ed. San Mateo, CA: Morgan Kaufmann, 2000, pp. 1175–1182.