Model Selection in Omnivariate Decision Trees

Olcay Taner Yıldız and Ethem Alpaydın

Department of Computer Engineering,
Boğaziçi University TR-34342, Istanbul, Turkey
yildizol@cmpe.boun.edu.tr, alpaydin@boun.edu.tr

Abstract. We propose an omnivariate decision tree architecture which contains univariate, multivariate linear or nonlinear nodes, matching the complexity of the node to the complexity of the data reaching that node. We compare the use of different model selection techniques including AIC, BIC, and CV to choose between the three types of nodes on standard datasets from the UCI repository and see that such omnivariate trees with a small percentage of multivariate nodes close to the root generalize better than pure trees with the same type of node everywhere. CV produces simpler trees than AIC and BIC without sacrificing from expected error. The only disadvantage of CV is its longer training time.

1 Introduction

A decision tree is made up of internal decision nodes and terminal leaves. The input vector is composed of p attributes, $\boldsymbol{x} = [x_1, \ldots, x_p]^T$, and the aim in classification is to assign $\boldsymbol{x}$ to one of K mutually exclusive and exhaustive classes. 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. Geometrically, each $f_m(\boldsymbol{x})$ defines a discriminant in the p-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.

In a *univariate* decision tree, 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 defines a discriminant which is orthogonal to axis x_j, intersects it at $x_j = -w_{m0}$ and divides the input space into two.

A *linear multivariate* decision tree, each internal node uses a linear combination of all attributes:

$$f_m(\boldsymbol{x}) : \boldsymbol{w}_m^T \boldsymbol{x} + w_{m0} = \sum_{j=1}^{p} w_{mj} x_j + w_{m0} > 0 \tag{2}$$

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 473–484, 2005.

To be able to apply the weighted sum, all the attributes should be numeric and discrete values need be represented numerically (usually by 1-of-L encoding) beforehand. Note that the univariate numeric node is a special case of the multivariate linear node, where all but one of w_{mj} is 0 and the other, 1. 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 a *nonlinear multivariate* decision tree, the decision takes the form

$$f_m(\boldsymbol{x}) : \sum_{j=1}^{k} w_j \phi_j(\boldsymbol{x}) > 0 \tag{3}$$

where $\phi_j(\boldsymbol{x})$ are the nonlinear basis functions. In this work, we use a polynomial basis function of degree 2 where for example for $\boldsymbol{x} \in \Re^2$, $\phi(\boldsymbol{x}) = [\ 1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$ which gives us a quadratic tree.

Surveys of work on constructing and simplifying decision trees can be found in [1], [2] and [3]. A recent survey comparing different decision tree methods with other classification algorithms is given in [4].

In this paper, we compare model selection techniques AIC, BIC and CV in the context of decision tree induction. In Section 2, we show how to apply model selection techniques in tree induction and we briefly explain the model selection techniques. In Section 3 we give the related work in the literature. We give our experiments and results in Section 4 and conclude in Section 5.

2 Tuning Model Complexity

The model selection problem in decision trees can be defined as choosing the best model at each node of the tree. In our experiments, we use three candidate models, namely, univariate, linear multivariate, and nonlinear multivariate (quadratic). At each node of the tree, we train these three models to separate two class groups from each other and choose the best. While going from the univariate to more complex nodes, the idea is to check if we can have a large decrease in bias with a small increase in variance.

We have previously proposed Omnivariate Decision Tree [5], where we have used CV to decide the best model from three different models including univariate model, linear perceptron and as nonlinear model multilayer perceptron. In this work, we have also included AIC and BIC in model selection and used quadratic model as the nonlinear model because it learns faster than the multilayer perceptron model. The use of PCA to decrease model complexity is another contribution of this work.

For finding the best split at a decision node we use Linear Discriminant Analysis (LDA) [6]. Since we have binary nodes, if we have $K > 2$ classes, these classes must be divided into two class groups and LDA is used to find the best split to separate these two class groups. We use the heuristic of splitting $K > 2$ classes into two groups originally proposed by Guo and Gelfand [7].

For the univariate model, we use univariate LDA and the model complexity is two, one for the index of the used attribute and one for the threshold. For the multivariate linear model, we use multivariate LDA and to avoid a singular covariance matrix, we use PCA with $\epsilon = 0.99$ to get k new dimensions and the model complexity is $k + 1$. For the multivariate quadratic model, we choose a polynomial kernel of degree 2 ($(x_1+x_2+\ldots+x_d+1)^2$) and use multivariate LDA to find the weights. Again to avoid a singular covariance matrix, we use PCA with $\epsilon = 0.99$ to get m new dimensions and the model complexity is $m + 1$. Then, we calculate the generalization error of each candidate model using the corresponding loglikelihood and model complexity. In the last step, we choose the optimal model having the least generalization error.

To calculate the error at a decision node, we must first assign classes to the left and right child nodes. Assume that C_L and C_R are classes assigned to the left and right nodes respectively. N_i^L and N_i^R are the number of instances of class i choosing left and right branches and N_{C_L} and N_{C_R} denote the number of instances of the classes C_L and C_R respectively.

$$N_{C_L} = \arg\max_i N_i^L, N_{C_R} = \arg\max_i N_i^R \tag{4}$$

The error at the decision node will be calculated by subtracting the number of instances of these two classes (which are correctly classified as they will label the leaves) from the total number of instances

$$e = \frac{N - N_{C_L} - N_{C_R}}{N} \tag{5}$$

When N is the total number of instances, N_i is the number of instances of class i, and N^L and N^R are the total number of instances choosing left and right branches respectively, the loglikelihood is given as

$$\mathcal{L} = \sum_{i=1}^{K} N_i^L \log \frac{N_i^L}{N^L} + \sum_{i=1}^{K} N_i^R \log \frac{N_i^R}{N^R} \tag{6}$$

Akaike Information Criterion AIC [8] is calculated as

$$AIC = 2(-\mathcal{L} + d) \tag{7}$$

where $\mathcal{L}$ represents the loglikelihood of the data and d represents the number of free parameters of the model. We choose the model with the smallest AIC over the three models we have.

Bayesian Information Criterion BIC [9] is calculated as

$$BIC = -\mathcal{L} + \frac{d}{2} \log N \tag{8}$$

where N is the number of data points. Like in AIC, we choose the model with the smallest BIC value.

Cross-validation We use 5×2 cross-validation and train all three models and test them on the validation set ten times and then apply the one-sided version of the 5×2 cv t test [10].

When we have two candidate models we choose the simple model, if it has smaller or equal error rate compared to the complex model. Only if the complex model has significantly smaller error rate then it is chosen. When we have three candidate models, univariate U, linear multivariate L, multivariate quadratic Q in increasing order of complexity with population error rates denoted by e_U, e_L, e_Q, we choose one considering both expected error and model complexity. Q is chosen if $H_0 : e_L \leq e_Q$ and $H_0 : e_U \leq e_Q$ are rejected. Otherwise, L is chosen if $H_0 : e_U \leq e_L$ is rejected. Otherwise U is chosen.

Note that AIC and BIC do not require a validation set and training is done once, whereas with CV, in each fold, half of the data is left out for validation and training is done ten times.

3 Related Work

LDA was first used in Friedman[11] for constructing decision trees. The algorithm has binary splits at each node, where a split is like in C4.5, i.e. $x_i < w_0$ but x_i can be an original variable, transgenerated, or adaptive. Linear discriminant analysis is applied to construct an adaptive variable. Kolmogorof-Smirnoff distance is used as the error measure. When there are more than $K > 2$ classes, it converts the problem into K different subproblems, where each subproblem separates one class from others. LTREE [12] is a multivariate decision tree algorithm with binary splits. LTREE uses LDA to construct new features, which are linear combinations of the original features. For all constructed features, the best split is found using C4.5's exhaustive search technique. Best of these is selected to create the two children of the current node. These new constructed features can also be used down the tree in the children of that node. Functional Trees [13] make simultenaous use of functional nodes and functional leaves in prediction problems. Bias-variance decomposition of the error showed that, the variance can be reduced using functional leaves, while bias can be reduced using functional inner nodes.

In CART [14], parameter adaptation is through backfitting: At each step, all the coefficients w_{mj} except one is fixed and that coefficient is tuned for possible improvement in terms of impurity. One cycles through all j until there is no further improvement. In OC1 [15], an extension to CART is made to get out of the local optima. A small random vector is added to $\boldsymbol{w}_m$ once there is convergence through backfitting. Adding a vector perturbs all coefficients together and makes a conjugate jump in the coefficient space. Another extension proposed is to run the method several (20-50) times and choose the best solution in terms of impurity.

In FACT [16], with K classes a node can have K branches. Each branch has its modified linear discriminant function calculated using LDA and an instance is channeled to the ith branch to minimize an estimated expected risk.

QUEST [17] is a revised version of FACT and uses binary splits at each decision node. It solves the problem of dividing K classes into two classes by using unsupervised 2-means clustering on the class means of the data. QUEST also differs from FACT in the way that it does not assume equal variances and uses Quadratic Discriminant Analysis (QDA) to find the two roots for the split point and uses the appropriate one. CRUISE[18] is a multivariate algorithm with K-way nodes. Like FACT, CRUISE finds $K-1$ splits using LDA. The departure from FACT occurs when the split assigns the same class to all its K children. Because such a split is not useful, the best next class is chosen. Another departure occurs while assigning a class to a leaf: When there are two or more classes which have the same number of instances in that leaf, FACT selects randomly one of them but CRUISE selects the class which has not been assigned to any leaf node.

In LMDT [19], with K classes, as in FACT, a node is allowed to have K branches. For each class, i.e., branch, there is a vector of parameters, and the node implements a K-way split. There is an iterative algorithm that adjusts the parameters of classes to minimize the number of misclassifications, rather than an impurity measure as entropy or Gini.

In Logistic model trees [20], logistic classification is used to find the best split at each decision node. They use a stagewise fitting process to construct logistic classification models that can select relevant attributes in the data.

4 Experiments

The proposed omnivariate trees are compared on twenty-two datasets from the UCI machine learning repository [21]. We compare pure univariate, linear and quadratic trees with omnivariate trees based on AIC, BIC, and CV. Our comparison criteria are generalization error (on the validation folds of 5×2 cross-validation), complexity (as measured by the total number of free parameters in the tree) and learning time (seconds on a Pentium Xeon 2.7). The error, model complexity and learning time figures contain boxplots, where the box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers, shown with '+', are data with values beyond the ends of the whiskers.

In Section 4.1, we give our results in two datasets, *pendigits* and *segment* in detail and in Section 4.2, we give our results on all twenty-two datasets.

4.1 Results on *Pendigits* and *Segment*

Figures 1 and 2 show the expected error, model complexity and learning time plots for *pendigits* (*pen*) and *segment* (*seg*) respectively. On *pendigits*, the pure quadratic tree is more accurate than the pure linear tree which in turn is more accurate than the pure univariate tree. On *segment*, the pure univariate tree is more accurate than the pure linear tree which in turn is more accurate than the pure quadratic tree. On *pendigits*, omnivariate trees have expected error close

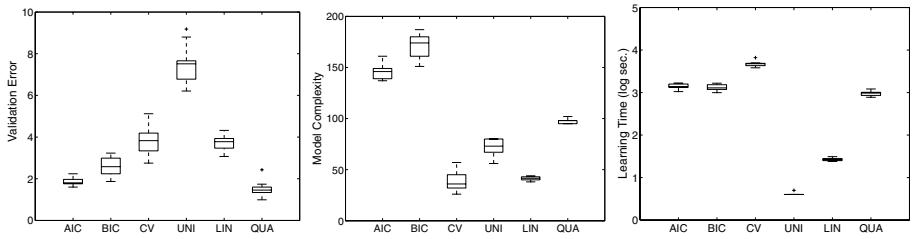

Fig. 1. The expected error, model complexity and learning time plots for *pendigits*

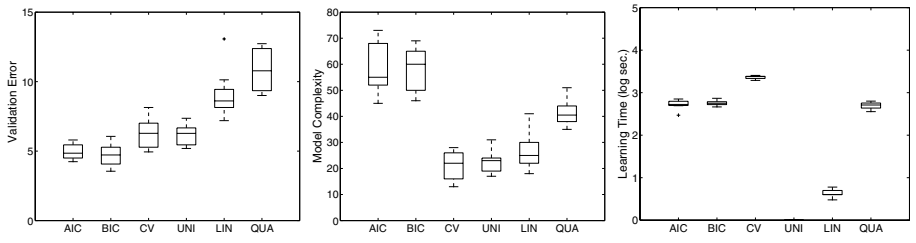

Fig. 2. The expected error, model complexity and learning time plots for *segment*

to the pure quadratic tree and on *segment*, they have expected error close to that of the pure univariate tree showing that they can automatically adapt to the problem. Omnivariate tree with CV produces smaller trees (in terms of the total number of parameters in the tree) than AIC and BIC. AIC and BIC have similar performances with respect to tree complexity even though BIC penalizes the complex models more. Due to this reason, on *pendigits*, AIC selects quadratic nodes more, which makes it better in terms of expected error compared to BIC and CV. Since CV uses 5×2 cross-validation, its learning time is higher than that of AIC and BIC. AIC and BIC trees also have their postpruning stages where the AIC (or BIC) of a subtree is compared to that of a leaf to possibly be replaced by it.

Figures 3 and 4 show the number of times of univariate, multivariate linear and multivariate quadratic nodes are selected at different levels of tree for *pendigits* and *segment* respectively. We see that more complex nodes are selected early in the tree, closer to the root where the problem is more complex and where there is more data. As we go down the tree, we have less data and complex nodes overfit and get rejected. Since pure quadratic tree is the best on *pendigits* and pure univariate tree is the best on *segment*, and since the omnivariate tree follows the best model, it selects the quadratic node on *pendigits* and the univariate node on *segment* more in the upper levels of the tree. AIC selects more complex nodes than BIC, which selects more complex nodes than CV, though in terms of the overall number of nodes, CV has the least because CV-based postpruning prunes more than AIC-based or BIC-based postpruning.

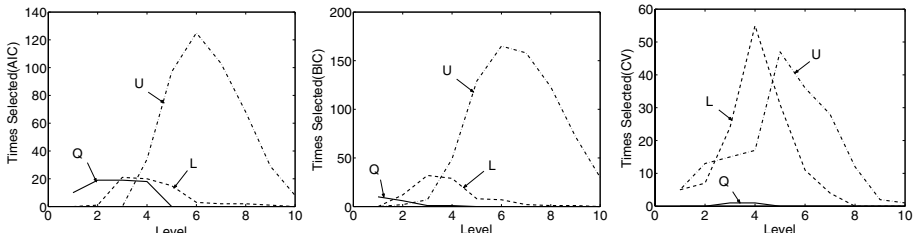

Fig. 3. The number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for *pendigits*

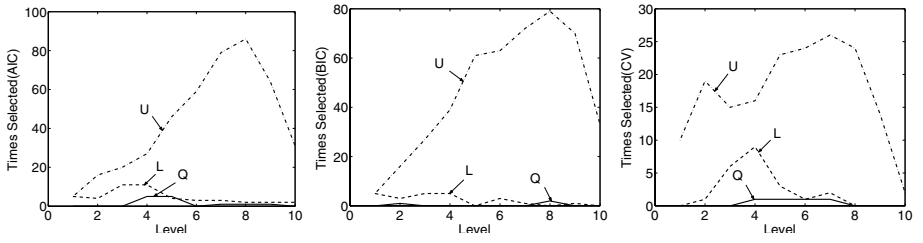

Fig. 4. The number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for *segment*

4.2 Results on all Datasets

The average and standard deviations of expected error and model complexity of decision trees produced by different model selection techniques and univariate, multivariate linear and multivariate quadratic decision trees for twenty-two datasets are given in Tables 1, 2, and 3 respectively. Since there are more than two decision tree algorithms to compare, we give two tables where in the first table the raw results are shown. The third table contains pairwise comparisons; the entry (i, j) in this second table gives the number of datasets (out of 22) on which method i is statistically significantly better than method j with at least 95% confidence. In the third table, row and column sums are also given. The row sum gives the number of datasets out of 22 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.

In general, omnivariate trees are more accurate than pure trees in terms of expected error. The number of wins of omnivariate trees against pure trees is more than the number of wins of pure trees against omnivariate ones. We also see that if there is a significant difference between the pure trees, the omnivariate tree follows the better tree. For example, on *balance*, the pure linear tree is the most accurate and omnivariate trees have similar accuracy to the linear tree by including many linear nodes. On *car*, pure univariate and linear trees are the best

Table 1. The average and standard deviations of expected errors of omnivariate decision trees and pure trees

	Pure Decision Trees			Omnivariate Decision Trees		
Set	UNI	LIN	QUA	AIC	BIC	CV
bal	25.89± 4.92	12.00± 1.92	24.10± 2.96	13.22± 1.56	12.70± 2.03	10.62± 1.13
bre	5.69± 1.65	4.41± 0.54	3.72± 0.87	5.06± 1.12	4.92± 0.90	4.63± 1.33
bup	40.17± 6.31	33.86± 4.17	41.22± 1.44	37.80± 4.06	37.44± 4.00	35.94± 4.80
car	7.38± 1.67	8.52± 1.64	21.32± 1.93	5.83± 1.07	6.08± 0.85	7.29± 1.77
der	6.99± 1.88	3.33± 1.01	8.75± 1.99	5.74± 1.69	5.69± 1.48	7.70± 1.75
eco	22.42± 3.78	18.17± 2.37	19.45± 2.62	21.41± 3.13	19.23± 2.30	19.74± 3.21
fla	11.15± 0.54	11.39± 0.75	11.15± 0.54	14.79± 3.28	12.14± 2.49	11.21± 0.63
gla	40.48± 9.06	42.71± 3.02	43.59± 6.21	36.80± 3.72	34.30± 3.12	38.40± 6.20
hab	26.54± 0.21	26.47± 0.16	26.41± 0.97	33.58± 4.08	26.47± 0.16	26.60± 0.39
hep	21.67± 1.95	20.39± 0.87	18.33± 2.88	22.57± 3.85	21.93± 4.26	21.93± 3.22
iri	5.47± 4.10	3.07± 1.41	4.80± 3.51	4.14± 1.93	4.27± 2.07	5.73± 1.99
iro	14.14± 2.80	11.56± 2.25	9.51± 2.62	11.51± 2.24	13.79± 1.65	12.71± 2.10
mon	20.79± 7.39	27.50±10.65	16.90± 2.21	10.42± 3.58	9.21± 3.30	20.65± 6.61
pen	7.50± 0.93	3.70± 0.38	1.52± 0.37	1.86± 0.20	2.57± 0.44	3.83± 0.79
pim	29.77± 3.90	23.10± 1.18	26.69± 3.04	30.70± 1.89	29.61± 2.90	30.78± 5.52
seg	6.19± 0.75	9.05± 1.66	10.92± 1.47	4.96± 0.53	4.72± 0.77	6.27± 1.07
tic	13.69± 4.47	29.42± 2.12	27.94± 4.83	18.89± 3.63	8.39± 1.67	16.41± 4.20
vot	4.83± 1.32	5.34± 2.04	8.69± 1.99	5.98± 1.73	5.98± 1.75	4.32± 0.54
wav	25.86± 1.09	14.77± 0.57	15.52± 0.46	20.47± 0.39	20.64± 1.04	16.25± 1.49
win	15.95± 4.16	3.15± 1.91	7.42± 4.00	5.40± 1.68	7.16± 4.59	6.15± 3.76
yea	48.99± 4.04	45.09± 2.50	47.72± 3.64	51.07± 2.03	50.89± 1.63	49.96± 4.95
zoo	10.70± 4.29	22.61± 5.07	21.50± 5.85	5.32± 2.73	5.32± 2.73	11.22± 4.69

and the omnivariate trees select univariate and linear nodes more. On *wave* and *wine*, pure linear and quadratic trees are more accurate than the pure univariate tree and the omnivariate trees have expected error close to the expected error of those trees. The model complexity table shows that CV constructs simpler trees than AIC and BIC. It has more wins (22 against 10 and 15) and less losses (6 against 20 and 22). We see that, CV chooses smaller trees but with more complex nodes and the trees constructed by CV are as accurate as trees constructed by AIC and BIC. Since omnivariate trees try all possible models, they have learning time more than those of trees and because CV tries all possible models ten times (because of 5×2 cross-validation), it has the longest learning time.

The number of times the univariate, multivariate linear and multivariate quadratic nodes are selected in omnivariate decision trees produced by AIC, BIC and CV are given in Table 4. We see that, as expected, quadratic nodes are selected the least and the univariate nodes are selected the most. Although CV has the smallest tree complexity, it has the highest percentage of multivariate nodes (linear 22 percent, nonlinear 1.74 percent). AIC and BIC trees do not prune as much as the CV tree and therefore their node counts are higher than the CV tree.

Table 2. The average and standard deviations of model complexities of omnivariate decision trees and pure trees

	Pure Decision Trees			Omnivariate Decision Trees		
Set	UNI	LIN	QUA	AIC	BIC	CV
bal	11.7± 4.7	17.9± 1.3	106.9± 0.9	45.5± 5.9	43.2± 10.0	17.0± 0.0
bre	6.0± 3.2	10.2± 0.4	37.1± 1.0	28.0± 10.0	22.4± 5.0	17.1± 12.0
bup	5.3± 4.9	7.3± 2.8	6.1± 9.8	52.3± 5.3	45.1± 16.8	7.9± 4.6
car	28.6± 4.7	19.9± 2.5	110.1± 1.4	67.9± 9.8	63.0± 10.3	27.3± 6.5
der	7.2± 1.2	33.7± 0.5	112.2± 1.5	25.7± 14.7	13.3± 2.1	6.4± 0.7
eco	5.5± 2.9	10.7± 1.3	21.0± 1.4	30.5± 7.8	26.8± 2.8	4.8± 1.2
fla	0.3± 0.9	3.9± 8.2	0.0± 0.0	17.8± 3.4	2.4± 5.1	0.4± 1.3
gla	6.9± 3.7	11.3± 2.4	21.9± 2.6	29.7± 3.7	30.2± 3.9	7.6± 2.3
hab	1.0± 3.2	0.0± 0.0	3.3± 5.3	36.1± 3.7	0.0± 0.0	0.7± 2.2
hep	2.1± 3.2	5.8± 9.3	28.8± 24.9	12.8± 3.2	13.3± 2.2	2.6± 5.9
iri	3.6± 1.0	5.7± 0.5	10.5± 0.5	4.5± 1.4	4.1± 1.7	3.0± 0.0
iro	4.2± 1.9	29.6± 0.7	57.4± 2.9	36.8± 7.9	18.2± 2.5	17.2± 14.5
mon	9.1± 3.6	8.4± 3.6	26.2± 0.4	37.5± 5.4	29.8± 2.2	17.1± 11.8
pen	72.0± 8.2	41.4± 2.0	97.4± 2.2	146.0± 7.7	171.5± 11.5	38.5± 10.0
pim	7.4± 7.6	9.5± 0.8	36.0± 12.7	101.1± 15.6	92.2± 10.5	4.0± 5.3
seg	22.8± 4.1	26.3± 6.7	41.7± 4.9	58.0± 9.8	57.9± 8.0	21.0± 5.4
tic	21.4± 3.5	21.5± 3.0	96.6± 66.7	185.0± 58.1	52.8± 9.6	20.3± 9.2
vot	2.9± 1.5	16.9± 0.7	82.8± 2.4	13.6± 3.0	10.0± 3.9	2.4± 1.3
wav	35.9± 14.3	26.6± 1.7	221.0± 1.2	406.2± 96.4	331.2± 10.4	23.8± 0.9
win	4.0± 0.7	14.0± 0.0	48.0± 1.2	14.3± 0.7	11.4± 4.2	13.1± 3.6
yea	20.0± 10.3	20.2± 4.7	37.6± 3.9	287.9± 13.8	281.3± 13.4	6.9± 3.6
zoo	6.7± 0.8	15.8± 0.9	25.7± 1.6	8.1± 0.9	8.1± 0.9	6.2± 1.1

Table 3. Pairwise comparisons of expected error and model complexities of omnivariate decision trees and pure trees

Expected Error							
	AIC	BIC	CV	UNI	LIN	QUA	∑
AIC	0	2	5	7	7	10	14
BIC	4	0	5	9	7	10	15
CV	5	3	0	5	4	8	15
UNI	4	1	1	0	3	6	8
LIN	9	6	5	10	0	9	17
QUA	7	6	3	6	3	0	11
∑	14	10	12	17	8	13	

Model Complexity							
	AIC	BIC	CV	UNI	LIN	QUA	∑
AIC	0	1	0	0	3	9	10
BIC	10	0	0	0	5	11	15
CV	21	18	0	3	11	20	22
UNI	21	19	6	0	12	19	21
LIN	17	14	2	3	0	19	21
QUA	12	9	0	0	0	0	12
∑	22	20	7	4	15	20	

The number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for AIC, BIC and CV is given in Figure 5. We see that AIC selects quadratic nodes more than BIC. This is in accordance with the literature stating that BIC has a tendency to choose simple models [22]. BIC selects more quadratic nodes than CV in the early levels of the tree. For linear nodes, their percentages are close to each other.

Table 4. The number of times univariate, multivariate linear and multivariate quadratic nodes selected in decision trees produced by AIC, BIC, and CV

Set	AIC			BIC			CV		
	UNI	LIN	QUA	UNI	LIN	QUA	UNI	LIN	QUA
bal	261	34	0	251	21	0	0	10	0
bre	143	18	1	119	14	1	18	8	2
bup	416	44	8	400	26	1	21	9	0
car	452	77	0	433	47	0	101	22	0
der	64	14	0	123	0	0	51	3	0
eco	232	30	4	230	18	0	32	6	0
fla	164	3	1	22	0	0	3	0	0
gla	273	9	5	283	7	2	65	1	0
hab	309	21	13	0	0	0	6	0	0
hep	116	2	0	123	0	0	4	1	0
iri	26	8	1	22	9	0	20	0	0
iro	93	11	6	172	0	0	20	11	0
mon	107	5	13	35	3	10	28	1	7
pen	465	65	66	742	92	18	176	137	2
pim	799	40	9	853	23	1	4	4	0
seg	454	48	13	487	24	3	174	22	4
tic	596	8	11	516	2	0	160	10	7
vot	122	4	0	89	1	0	14	0	0
wav	2792	59	5	3072	30	0	1	27	0
win	8	14	0	31	7	0	10	12	0
yea	2401	157	60	2662	102	18	50	8	1
zoo	71	0	0	71	0	0	52	0	0
$\sum$	10364	671	216	10736	426	54	1010	292	23
%	92.12	5.96	1.92	95.72	3.80	0.48	76.22	22.04	1.74

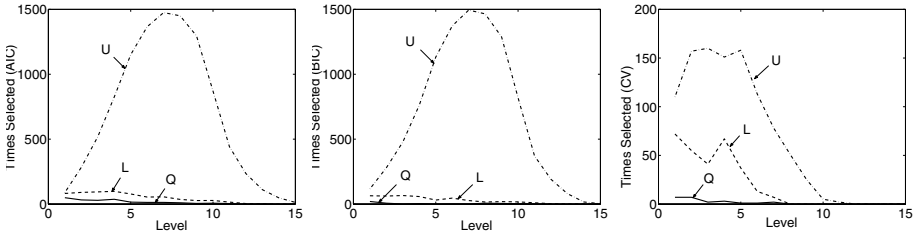

Fig. 5. Number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for AIC, BIC and CV

5 Conclusion

We propose a novel decision tree architecture, the omnivariate decision tree, which contains both univariate, multivariate linear, and multivariate quadratic nodes. The ideal node type is determined via model selection using AIC, BIC or CV. 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 omnivariate architecture generalizes better than pure trees with the same type of node everywhere. As expected, the quadratic node, is selected the least, followed by the linear node and the univariate node. More complex nodes are selected early in the tree closer to the root. Since there are more nodes in the lower levels, the percentage of univariate nodes is much higher ($> 90\%$ for AIC and BIC, $> 75\%$ for CV). This shows that having a small percentage of multivariate (linear or quadratic) nodes is effective.

CV finds simpler trees than BIC which in turn finds simpler trees than AIC. Although omnivariate CV trees are simple, they contain complex nodes (multivariate linear and nonlinear) more than AIC and BIC. All three methods have nearly the same error rate. CV produces simpler models without sacrificing from the expected error. But it has the disadvantage that its training time is longer due to multiple cross-validation runs.

In the literature, in choosing between nodes or choosing node parameters, accuracy (information gain or some other measure calculated from fit to data) is used in growing the tree and some other measure (cross-validation error on a pruning set or MDL) has been used to prune the tree; the same is also true for rule learners such as Ripper [23]. Our work proposes to use one criterion (AIC, BIC, CV) in both growing the tree *and* pruning it. We believe this makes more sense than many approaches where separate criteria are used to grow the tree and then to prune it to alleviate overfitting. Because our approach allows choosing among multiple models, it is also possible to have different algorithms for the same node type and choose between them. That is, one can have a palette of univariate nodes (one by LDA, one by information gain, etc) and the best one will then be chosen. The same also holds for linear or nonlinear (quadratic, other kernels, etc) nodes. Our emphasis is on the idea of an omnivariate decision tree and the sound use of model selection in inducing it, rather than the particular type of nodes we use in our example decision tree. The nice thing about LDA is that the same criterion can be used in training univariate, linear and quadratic nodes and we know that any difference is due to node complexity.

The same model selection idea can also be applied in regression trees by creating an omnivariate regression tree where at each node there are three candidate models, namely, univariate model, linear multivariate model and linear quadratic model. Then the candidate models at each node try to minimize the sum of mean square errors of child nodes. For CV based model selection, the same 5×2 paired t test can be used for comparing the mean square errors of the candidate models. One can also include different type of models in the leaf nodes such as linear models and can make model selection for the leaf nodes for both classification and regression trees.

Acknowledgements

This work has been supported by the Turkish Academy of Sciences, in the framework of the Young Scientist Award Program (EA-TÜBA-GEBİP/2001-1-1) and Boğaziçi University Scientific Research Projects 02A104D and 03K120250.

References

1. Breslow, L.A., Aha, D.W.: Simplifying decision trees: A survey. Technical Report AIC-96-014, Navy Center for Applied Research in AI, Naval Research Laboratory, Washington DC, USA (1997)
2. Murthy, S.K.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mininig and Knowledge Discovery **2** (1998) 345–389
3. Yıldız, O.T., Alpaydın, E.: Linear discriminant trees. International Journal of Pattern Recognition and Artificial Intelligence **19** (2005) 323–353
4. Lim, T.S., Loh, W.Y., Shih, Y.S.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning **40** (2000) 203–228
5. Yıldız, O.T., Alpaydın, E.: Omnivariate decision trees. IEEE Transactions on Neural Networks **12** (2001) 1539–1546
6. Alpaydın, E.: Introduction to Machine Learning. The MIT Press (2004)
7. Guo, H., Gelfand, S.B.: Classification trees with neural network feature extraction. IEEE Transactions on Neural Networks **3** (1992) 923–933
8. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Second International Symposium on Information Theory. (1973) 267–281
9. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics **6** (1978) 461–464
10. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning classifiers. Neural Computation **10** (1998) 1895–1923
11. Friedman, J.H.: A recursive partitioning decision rule for non-parametric classification. IEEE Transactions on Computers (1977) 404–408
12. Gama, J.: Discriminant trees. In: 16th International Conference on Machine Learning, New Brunswick, New Jersey, Morgan Kaufmann (1999) 134–142
13. Gama, J.: Functional trees. Machine Learning **55** (2004) 219–250
14. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. John Wiley and Sons (1984)
15. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research **2** (1994) 1–32
16. Loh, W.Y., Vanichsetakul, N.: Tree-structured classification via generalized discriminant analysis. Journal of the American Statistical Association **83** (1988) 715–725
17. Loh, W.Y., Shih, Y.S.: Split selection methods for classification trees. Statistica Sinica **7** (1997) 815–840
18. Kim, H., Loh, W.: Classification trees with unbiased multiway splits. Journal of the American Statistical Association (2001) 589–604
19. Brodley, C.E., Utgoff, P.E.: Multivariate decision trees. Machine Learning **19** (1995) 45–77
20. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. In: Proceedings of the European Conference in Machine Learning. (2003) 241–252
21. Blake, C., Merz, C.: UCI repository of machine learning databases (2000)
22. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Verlag, New York (2001)
23. Cohen, W.W.: Fast effective rule induction. In: The Twelfth International Conference on Machine Learning. (1995) 115–123