Model Selection in Omnivariate Decision Trees using Structural Risk Minimization

Olcay Taner Yıldız [a]

[a] *Department of Computer Engineering, Işık University, TR-34980, Şile, Istanbul, Turkey*

Abstract

As opposed to trees that use a single type of decision node, an omnivariate decision tree contains nodes of different types, univariate, linear or linear multivariate. We propose to use Structural Risk Minimization (SRM) to choose between node types in omnivariate decision tree construction to match the complexity of a node to the complexity of the data reaching that node. We compare SRM with other model selection techniques including Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC) and Crossvalidation (CV) to choose between the three types of nodes on standard datasets from the UCI and Delve repositories. We see that omnivariate trees obtained via SRM with a small percentage of multivariate nodes close to the root generalize better or at least as accurately as those constructed using other model selection techniques.

Key words: Classification; machine learning; model selection; structural risk minimization; decision tree

1 Introduction

In machine learning the knowledge is extracted from a training sample for future prediction. Most machine learning methods make accurate predictions but are not interpretable, on the other hand decision trees are simple and easily comprehensible. They are robust to noisy data and can learn disjunctive expressions [26]. Surveys of work on constructing and simplifying decision trees can be found in [7], [27] and [36]. A recent survey comparing different decision tree methods with other classification algorithms is given in [23].

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 $K > 2$ 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 carries a class label. 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 at a part of the input space which contains 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 [29].

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

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

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 remaining, 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 a 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.

In multivariate linear trees, Linear Discriminant Analysis (LDA) was first used by Friedman [14] 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 the number of classes K is > 2, it converts the problem into K different subproblems, where each subproblem separates one class from the others. LTREE [15] 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. The 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 [16] 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 [6], 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 [28], 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 [25], 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 [24] 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[19] 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 [8], 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 like 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.

In this paper, we compare model selection techniques AIC, BIC, SRM and CV in the context of decision tree induction. In Section 2, we show previously applied model selection techniques in tree induction. In Section 3 we explain how to apply Structural Risk Minimization technique in decision tree induction. 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 architecture [35], where we have used cross-validation to decide on the best model from three different candidates including a univariate node, multivariate linear node (linear perceptron) and a multivariate nonlinear node (multilayer perceptron (MLP)). Continuing this work, Altınçay [4] proposed use of model ensemble-based nodes where a group of models are considered for making decisions at each decision node. The ensemble members are generated via perturbing model parameters and input parameters. In another work, Li [22] used a novel classifiability measure instead of pairwise statistical tests to perform model selection at each decision node. Their proposed classifiability measure is related to Bayes error and the boundary complexity.

In our second work [37], we have included AIC and BIC in model selection and used a quadratic model instead of the MLP as the nonlinear model because it learns faster. For finding the best split at a decision node we use Linear Discriminant Analysis (LDA) [3]. For the univariate model, we use the univariate version of LDA and the number of parameters is 2, one for the index of the used attribute and one for the threshold. For the multivariate linear model, we use the multivariate version of LDA and to avoid a singular covariance matrix, we use Principal Component Analysis (PCA) to get k new dimensions and the number of parameters 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 the multivariate LDA to find the separating hyperplane. Again to avoid a singular covariance matrix, we use PCA to get m new dimensions and the number of parameters 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 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:

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

The error at a decision node is 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 log likelihood is given as

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

Given the log likelihood, different model selection techniques use different measures, which will be explained in the next subsections.

2.1 Akaike Information Criterion

AIC [1] is calculated as

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

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

2.2 Bayesian Information Criterion

BIC [32] is calculated as

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

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

2.3 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 [13].

When we have two candidate models we choose the simpler 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 considering both the expected error and the model complexity. Q is chosen if both $H_0 : e_L \leq e_Q$ 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, 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 Structural Risk Minimization

Structural risk minimization (SRM) [33] uses the VC dimension of the estimators to select the best model by choosing the model with the smallest upper bound for the generalization error. In order to calculate the VC generalization bounds of a model, (i) the VC-dimension of the model representing its complexity, and (ii) the error of the model on the training data, are required. In a two-class classification problem, the generalization bound of a model is given as [10]

$$E_{generalization} = e + \frac{\lambda}{2}\left(1 + \sqrt{1 + \frac{4e}{\lambda}}\right) \tag{9}$$

where e is the training error and

$$\lambda = a_1 \frac{h[\log(a_2 N/h) + 1] - \log(\nu)}{N} \tag{10}$$

Here, h represents the VC dimension of the model, ν represents the confidence level (it is recommended to use $\nu = \frac{1}{\sqrt{N}}$ for large sample sizes), and a_1 and a_2 are empirically fitted constants. In our experiments we use $a_1 = 0.2$ and $a_2 = 0.8$.

VC dimension for a class of functions $f(x, \alpha)$ where α denotes the parameter vector is defined to be the largest number of points that can be shattered by members of $f(x, \alpha)$. A set of data points is *shattered* by a class of functions $f(x, \alpha)$ if for each possible class labeling of the points, one can find a member of $f(x, \alpha)$ which perfectly separates them. For example, in two dimensions, we can separate three points with a line, but we can not separate four points (if their class labelings are done like in the famous XOR problem). Therefore, the VC dimension of the linear estimator class in two dimensions is 3.

3.1 Structural Risk Minimization in Omnivariate Decision Tree Induction

In our experiments, we have three different models (univariate model, multivariate linear model, multivariate quadratic model) at each decision node to choose from. We calculate the generalization error of these three models using Equations 9, 10 and choose the model with the smallest generalization error. To calculate generalization error, we need (i) the training error (which can be estimated via Equation 5) and (ii) the VC-dimension of each model.

The VC-dimension of the multivariate linear model with d dimensions is $d + 1$. The multivariate quadratic model can be written as a linear model with $\frac{d^2 + 3d + 2}{2}$ (d parameters for x_i^2, $\frac{d^2 - d}{2}$ parameters for $x_i x_j$, d parameters for x_i and 1 parameter for bias) parameters, which we take as its VC-dimension (We will also show in section 4.2 that these numbers are in accordance with our VC-dimension estimations). Since the VC-dimension of the univariate model whose best split is found by LDA is not known, we used a technique proposed in [34] to estimate the VC-dimension experimentally. This uses the fact that for two-class classification problems, the deviation of the expected training error of the classifier from 0.5 (which is the worst a two-class classifier can do) is correlated with the VC-dimension of that classifier and the data size, and a theoretical formula for this correlation is derived. A set of experiments on artificial sets are done and based on the frequency of the errors on these sets, a best fit for the theoretical formula is calculated. The details of this technique and experimental results on three models are given in sections 3.2 and 4.2 respectively.

In AIC, BIC, and CV the same criterion is used in both growing the tree and post-pruning it. Postpruning, where we decide to remove or not to remove a subtree, has originally been proposed using cross-validation; we just compare the validation error of a subtree and the leaf to replace it. For AIC and BIC, the number of parameters of a subtree can be taken as the sum of parameters in all the nodes. For SRM however, we need to calculate the VC dimension of a subtree possibly containing nodes of different types, and this is not straightforward. This is because (i) we do not know the VC-dimension of any pure subtree (subtree containing nodes of the same type) with n nodes, (ii) the VC-dimension may change according to the structure of the decision tree, (iii) even if we know the VC-dimension of pure trees, our trees are not pure and we can not simply add VC-dimensions of pure trees to get the VC-dimension of omnivariate trees, where the univariate, linear or quadratic nodes can be anywhere in the decision tree. Therefore we use pre-pruning, where at each decision node we have now four choices to select from. The first three original choices are univariate, linear and quadratic models, the fourth choice is the simplest model, the leaf which we take as a constant model with VC-dimension 1. If the leaf has the smallest generalization error, we simply make the node leaf node.

3.2 Estimation of VC-dimension: Uniform Design

As pointed out in the previous section, we need a methodology to estimate the VC-dimension of the univariate model whose VC-dimension can not be found theoretically. In this section, we briefly review the methodology that we have used in estimating the VC-dimension [34], namely uniform design and how we apply it.

With respect to the VC-theory, the maximum difference $\epsilon(N)$ between the error rates of two independently labeled data sets of size N is bounded by $\Phi(N/h)$ where

$$\Phi(\rho) = a\frac{ln(2\rho)+1}{\rho-k}\left(\sqrt{1+\frac{b(\rho-k)}{ln(2\rho)+1}}+1\right) \tag{11}$$

when $\rho \geq 0.5$ with $\rho = N/h$ and the constants $a = 0.16$, $b = 1.2$, $k = 0.14928$ are determined empirically [34]. If $\rho < 0.5$, $\Phi(\rho) = 1$. According to Vapnik et al. (1994), this bound is tight; so we can consider $\epsilon(N)$ to be approximately equal to $\Phi(N/h)$.

An experiment design consists of design points and a number of experiments for each point. In our case, an experiment is performed to get a single epsilon estimate $\epsilon(N)$ for a specific number of data points N. If the number of ex-

```
1   VcEstimate UniformDesign($d$; $D$)
2       for $i = 1$ to $D$
3           $N_i$ = Sample size where $0.5 \leq N_i/h \leq 30$
4           for $j = 1$ to $m_i$
5               Generate a random uniform sample $S$
                of size $2N_i$
6               Split $S$ into two samples
                of equal size ($S_1$, $S_2$)
7               Flip class labels of $S_2$
8               $C$ = TrainClassifier($S$)
9               Flip class labels of $S_2$ back again
10              $E(S_1)$ = Error rate of $C$ on $S_1$
11              $E(S_2)$ = Error rate of $C$ on $S_2$
12              $\epsilon(N_{i,j}) = |E(S_1) - E(S_2)|$
13          $\overline{\epsilon}(N_i) = \frac{\sum_{j=1}^{m_i} \epsilon(N_{i,j})}{m_i}$
14      $h$ = Best fit between $\Phi(N_i/h)$ and $\overline{\epsilon}(N_i)$
15      return $h$
```

Fig. 1. Pseudocode of Uniform Design: d: Number of input features, D: Number of design points

periments at the design points are equal, the design is called uniform design [21].

The pseudocode of uniform design for estimating the VC-dimension is given in Figure 1. In a single experiment, to get the maximum difference of error rates, the classifier (in our case univariate model) is trained on the whole sample S (Line 8), is tested on the first part of the dataset S_1 to minimize the error rate (Line 10), and then tested on the second part of the dataset S_2 with class labels complemented to maximize the error rate (Lines 9, 11).

In uniform design, at each design point i, m_i experiments are performed (Line 4) and the average of epsilon estimates $\overline{\epsilon}(N_i)$ is calculated (Line 13). In order to reduce the variability of the estimates, the sample size for each design point, N_i, is different and selected in the range $0.5 \leq N_i/h \leq 30$ (Line 3). The VC-dimension of the univariate model is then h (Line 14) which minimizes

$$\sum_{i=1}^{D} [\overline{\epsilon}(N_i) - \Phi(N_i/h)]^2 \tag{12}$$

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

We use a total of 17 two-class datasets where 14 of them (*artificial, breast, bupa, german, haberman, heart, hepatitis, mammographic, monks, parkinsons, pima, tictactoe, transfusion, vote*) are from UCI [5] and 3 (*titanic, ringnorm, and twonorm*) are from Delve [30] repositories.

4.1.2 Learning algorithms

We compared our proposed SRM based omnivariate decision tree algorithm with six different algorithms:

- *AIC*: Omnivariate decision tree algorithm where model selection is done using Akaike Information Criterion.
- *BIC*: Omnivariate decision tree algorithm where model selection is done using Bayesian Information Criterion.
- *CV*: Omnivariate decision tree algorithm where model selection is done using cross-validation.
- *LDA*: Linear discriminant analysis classifier [3].
- *QDA*: Quadratic discriminant analysis classifier [3].
- *SVM*: Support vector machine with a linear kernel. We use the LIBSVM 2.82 library [9].

4.1.3 Comparison Criteria

Our comparison criteria are generalization error (on the validation folds of 5×2 cross-validation), model complexity (as measured by the total number of free parameters) and training time of the learning algorithm (in seconds).

4.1.4 Statistical Test used in Comparisons

To compare two learning algorithms, statistical tests have been proposed [13,12]. In choosing between two, one can use a pairwise test to compare their performance and select the one that has better performance. Typically, cross-validation is used to generate a set of training, validation folds, and we compare their performance on the validation folds. Examples of such tests are parametric tests, such as k-fold paired t test, 5×2 cv t test [13], 5×2 cv

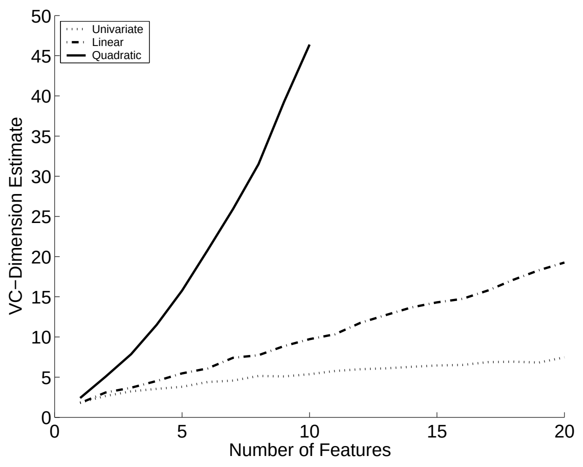

Fig. 2. VC-dimension estimate of of univariate, linear and quadratic models with respect to number of features in the dataset.

F test [2], nonparametric tests, such as the sign test and Friedman's test, or range tests, such as Wilcoxon signed rank test.

Although these tests are good for comparing the means of two populations (that is, the performances of two algorithms), they can not be used to compare multiple populations (algorithms). A further need is to be able to compare algorithms over not a single data set but over multiple data sets. Demsar [12] examines various methods, such as the sign test and Friedman's test together with its post-hoc Nemenyi's test, for comparing multiple algorithms over multiple data sets. Following the same idea, we used sign test in comparing our proposed SRM based omnivariate decision tree algorithm with other learning algorithms.

4.2 VC-Dimension Estimation

In the following experiment, we estimate the VC-dimension of three models namely (a) univariate, (b) linear multivariate, and (c) quadratic models using uniform design. To cover the state space well, we estimate the VC-dimension on d dimensional data where d varies from 1 to 20. Finding the eigenvectors of a k dimensional matrix (as required by PCA) takes $O(k^3)$ time. In the multivariate quadratic model, we have d^2 parameters and therefore we need approximately $O(d^6)$ time to calculate the eigenvectors. Because of this time burden, in quadratic model d varies from 1 to 10.

With d input features the multivariate linear and quadratic models have $d+1$ and $\frac{d^2+3d+2}{2}$ number of free parameters respectively. Figure 2 shows VC-dimension estimates for this case. We see that for both multivariate linear and quadratic models, the estimated VC-dimension is nearly equal to the number of free parameters. For univariate model, the VC-dimension estimate is nearly equal to $\log d + 1$, which we can safely use as the VC-dimension in our generalization error calculations.

4.3 Performance of SRM Compared with Other Omnivariate Decision Tree Algorithms

In this subsection, we try to answer the following question that motivates us: Is our proposed model selection technique SRM is at least as good as other model selection techniques in terms of error, model complexity and learning time? In order to answer this question, we compared omnivariate trees based on AIC, BIC, and CV with omnivariate trees based on SRM on 17 datasets. The average and standard deviations of expected error, model complexity of decision trees produced by different model selection techniques and time required to construct the omnivariate decision trees for 17 datasets are given in Tables 1, 2 and 3 respectively. Since there are more than two omnivariate decision tree algorithms to compare, we give two tables where in the first table the raw results are shown. The second table contains pairwise comparisons; the entry (i, j) in this second table gives the number of datasets (out of 17) on which method i is better than method j (without any check for significance). The number of wins that are statistically significantly different using the sign test over 17 runs are shown in bold.

The average and standard deviations of expected error of the omnivariate decision trees for 17 datasets are given in Table 1. In terms of expected error rate, SRM is significantly better than AIC (12 wins against 4 losses) and BIC (13 wins against 3 losses). Although the difference is not statistically significant, SRM is also better than CV in 12 datasets. Other comparisons do not yield a significant difference, but in general we can say that CV is better than AIC and BIC, where they have nearly equal performance.

Table 2 shows total decision tree complexity in terms of number of free parameters for different omnivariate decision tree techniques. As explained in Section 2, total complexity of an omnivariate decision tree is the sum of complexities of its decision nodes. The complexity of univariate, multivariate linear and multivariate quadratic decision nodes are 2, $k + 1$, and $m + 1$ respectively, where k and m are the reduced number of dimensions after PCA for multivariate linear and multivariate quadratic models respectively. The model complexity table shows that SRM, BIC and CV construct simpler trees than AIC, where BIC generates significantly simpler trees when compared with them. The comparison result between BIC with AIC is in accordance with the literature stating that BIC has a tendency to choose simpler models [17].

Table 1
Expected error rates and number of wins of omnivariate decision tree algorithms using average error rates. The bold face entries show statistically significant difference using the sign test.

Dataset	AIC	BIC	CV	SRM
artificial	0.00±0.00	0.56±1.78	2.00±3.41	0.00±0.00
breast	4.81±0.94	4.61±0.79	4.35±0.57	4.29±0.65
bupa	36.81±2.23	42.03±0.18	36.52±5.34	32.35±2.06
german	31.56±1.30	30.00±0.00	28.84±2.10	28.26±1.71
haberman	28.10±2.86	26.47±0.16	27.72±2.00	27.78±2.27
heart	22.37±3.44	41.63±8.90	18.96±3.15	25.70±1.98
hepatitis	22.46±4.54	23.74±4.52	20.26±2.17	18.33±3.04
m.graphic	19.33±1.82	46.31±0.06	19.31±1.75	18.69±1.00
monks	10.79±5.38	31.94±17.69	17.59±8.35	25.28±2.71
parkinsons	13.96±4.18	14.89±4.75	18.16±6.48	13.86±3.80
pima	30.44±2.38	34.90±0.00	30.36±5.86	23.72±1.21
ringnorm	3.26±0.12	5.18±0.37	3.75±1.17	2.62±0.25
tictactoe	20.52±3.15	13.72±11.15	14.38±4.29	24.70±1.85
titanic	21.19±0.87	21.66±0.77	21.66±0.97	22.04±0.76
vote	5.70±1.65	4.50±1.11	4.69±0.85	4.23±1.09
transfusion	23.80±0.00	23.80±0.00	23.69±0.34	23.40±0.66
twonorm	4.25±0.28	2.25±0.19	2.30±0.28	2.25±0.19

	AIC	BIC	CV	SRM
AIC	0	10	5	4
BIC	6	0	6	3
CV	12	10	0	5
SRM	**12**	**13**	12	0

Table 3 shows time required to generate the decision trees for each omnivariate algorithm. Since CV tries all possible models ten times (because of 5×2 cross-validation), whereas AIC, BIC and SRM tries all possible models once, the former's time complexity is much higher than the others. We also see that, SRM is better than AIC and BIC. This is due to the fact that, SRM uses prepruning for pruning the decision tree and therefore is able to cut down the search early. On the other hand, AIC and BIC use postpruning, where they generate whole tree (which takes time) and prune the decision subtrees whose

Table 2
Tree complexities in terms of number of free parameters and number of wins of omnivariate decision tree algorithms using average tree complexities. The bold face entries show statistically significant difference using the sign test.

Dataset	AIC	BIC	CV	SRM
artificial	12.60±2.84	11.60±2.07	11.90±4.18	12.60±2.84
breast	63.20±11.77	26.80±6.20	20.80±12.93	15.10±6.87
bupa	133.20±16.17	1.00±0.00	13.30±12.53	24.60±15.37
german	342.50±24.39	1.00±0.00	19.50±32.54	241.70±53.39
haberman	12.70±14.57	1.00±0.00	4.50±6.72	9.90±12.85
heart	78.30±10.66	2.50±4.74	21.60±9.71	108.10±35.26
hepatitis	37.60±8.10	13.90±16.68	5.80±8.75	83.70±24.11
m.graphic	11.00±7.77	1.00±0.00	18.30±16.22	13.70±9.87
monks	64.90±35.41	6.70±8.99	37.60±15.46	13.40±29.73
parkinsons	34.50±5.54	30.70±5.38	10.60±10.57	23.50±7.63
pima	239.80±18.41	1.00±0.00	5.30±5.62	20.40±15.14
ringnorm	626.10±31.07	227.90±0.32	366.00±115.17	520.10±71.16
tictactoe	301.30±39.25	80.10±45.50	121.90±30.15	267.40±39.35
titanic	24.70±4.52	10.30±3.59	12.60±4.30	6.10±1.45
vote	34.90±9.54	9.40±7.32	7.60±4.65	27.10±18.06
transfusion	1.00±0.00	1.00±0.00	2.60±5.06	4.90±6.49
twonorm	437.10±27.30	23.00±0.00	26.90±12.33	23.00±0.00

	AIC	BIC	CV	SRM
AIC	0	0	2	4
BIC	**16**	0	**13**	**13**
CV	**15**	4	0	12
SRM	**12**	3	5	0

removal do not decrease the generalization error calculated via AIC and BIC.

Table 3
Time complexities in terms of seconds to generate the decision tree and number of wins of omnivariate decision tree algorithms using average time complexities. The bold face entries show statistically significant difference using the sign test.

Dataset	AIC	BIC	CV	SRM
artificial	0.15±0.00	0.15±0.02	1.34±0.16	0.14±0.00
breast	0.34±0.04	0.39±0.06	1.81±0.74	0.18±0.03
bupa	0.16±0.02	0.16±0.01	0.78±0.05	0.03±0.01
german	680.86±74.67	700.99±67.98	3416.58±242.67	39.45±9.54
haberman	0.01±0.00	0.01±0.01	0.05±0.01	0.00±0.00
heart	3.49±0.64	3.47±0.52	16.40±3.22	0.85±0.10
hepatitis	16.43±5.99	18.09±5.08	57.16±25.43	5.45±1.48
m.graphic	33.84±1.63	33.19±1.28	164.53±23.02	3.15±1.05
monks	22.30±5.85	22.59±5.84	145.14±19.86	3.12±1.17
parkinsons	43.87±7.26	46.49±6.27	237.60±28.20	17.39±2.80
pima	1.16±0.16	1.18±0.14	5.70±0.57	0.12±0.01
ringnorm	121.65±27.07	359.93±21.04	442.94±301.72	28.25±2.67
tictactoe	699.23±94.47	634.94±79.65	3046.30±599.68	91.86±13.63
titanic	0.70±0.03	0.68±0.02	4.09±0.26	0.23±0.02
vote	407.29±98.26	465.57±94.78	2086.34±861.60	184.78±36.62
transfusion	0.06±0.00	0.06±0.00	0.28±0.02	0.01±0.01
twonorm	261.56±16.81	286.42±20.79	1216.78±97.66	20.73±0.09

	AIC	BIC	CV	SRM
AIC	0	9	**17**	0
BIC	4	0	**17**	0
CV	0	0	0	0
SRM	**17**	**17**	**17**	0

4.4 Model Selection in SRM Compared with Other Omnivariate Decision Tree Algorithms

Table 4 shows the average and standard deviations of node counts of omnivariate decision trees. We see that SRM and BIC choose smaller trees but with more complex nodes. BIC and SRM usually generate smallest trees whereas other omnivariate decision tree algorithms use more than one tree node; espe-

Table 4
The average and standard deviations of node counts of omnivariate decision trees.

Dataset	AIC	BIC	CV	SRM
artificial	2.00±0.00	2.20±0.42	2.70±1.16	2.00±0.00
breast	12.40±3.34	5.70±2.26	1.20±0.63	1.30±0.67
bupa	38.30±4.42	0.00±0.00	3.10±3.48	2.60±1.65
german	101.00±10.26	0.00±0.00	3.80±7.27	1.50±0.85
haberman	3.50±4.33	0.00±0.00	1.10±2.13	1.50±2.01
heart	21.10±3.57	0.10±0.32	3.40±2.67	2.30±1.49
hepatitis	11.00±2.49	4.30±5.56	0.50±0.85	2.20±1.14
m.graphic	1.50±0.97	0.00±0.00	4.70±5.85	1.70±0.95
monks	20.00±10.90	1.90±3.00	11.60±4.99	1.20±0.63
parkinsons	9.60±2.37	9.70±2.00	2.50±2.84	1.80±0.63
pima	70.00±6.88	0.00±0.00	0.50±0.71	1.20±0.42
ringnorm	58.50±10.75	1.00±0.00	2.50±1.90	3.00±1.25
tictactoe	51.70±13.47	26.00±15.28	15.50±6.42	2.70±0.67
titanic	5.00±1.49	1.90±0.88	2.60±1.26	1.70±0.48
vote	8.70±2.45	2.80±2.44	2.20±1.55	2.20±1.03
transfusion	0.00±0.00	0.00±0.00	0.40±1.26	0.70±1.25
twonorm	122.70±5.66	1.00±0.00	2.30±4.11	1.00±0.00

cially, AIC generates most populated decision trees.

It may be the case that a linear model is appropriate on a dataset and a univariate may be sufficient on another. The best model even changes as we go down the same tree; simpler models suffice as we get closer to the leaves. The best model is not known a priori and the omnivariate tree does the model selection itself, freeing the user from an explicit trial of possible models.

The number of times the univariate, multivariate linear and multivariate quadratic nodes are selected in omnivariate decision trees produced by AIC, BIC, CV and SRM are given in Table 5. We see that, as expected, quadratic nodes are selected the least and the univariate nodes are selected the most, except with SRM. Although SRM has the smallest number of nodes, it has the highest percentage of multivariate nodes (linear 31.05 percent, nonlinear 41.17 percent). AIC, BIC and CV do not prune as much as SRM and therefore their node counts are higher than the SRM tree. CV has the second highest percentage of multivariate nodes (linear 12.71 percent, quadratic 5.12 percent).

Table 5
The number of times univariate, multivariate linear and multivariate quadratic nodes selected in decision trees produced by AIC, BIC, CV, and SRM.

	AIC			BIC			CV			SRM		
	U	L	Q	U	L	Q	U	L	Q	U	L	Q
artificial	8	9	3	8	14	0	18	9	0	8	9	3
breast	103	18	3	46	11	0	2	7	3	0	13	0
bupa	345	32	6	0	0	0	25	6	0	3	14	9
german	988	22	0	0	0	0	35	2	1	0	2	13
haberman	32	2	1	0	0	0	10	1	0	4	5	6
heart	199	12	0	0	1	0	25	9	0	6	1	16
hepatitis	108	2	0	43	0	0	3	2	0	3	6	13
m.graphic	10	5	0	0	0	0	44	3	0	9	6	2
monks	193	7	0	19	0	0	113	3	0	9	1	2
parkinsons	92	3	1	96	1	0	23	2	0	6	6	6
pima	659	39	2	0	0	0	1	4	0	0	9	3
ringnorm	563	2	20	0	0	10	8	1	16	3	2	25
tictactoe	497	10	10	259	1	0	138	9	8	1	4	22
titanic	38	2	10	15	0	4	16	7	3	17	0	0
vote	81	6	0	28	0	0	22	0	0	14	5	3
transfusion	0	0	0	0	0	0	2	2	0	2	2	3
twonorm	1189	38	0	0	10	0	13	10	0	0	10	0
Σ	5105	209	56	514	38	14	498	77	31	85	95	126
%	95.1	3.9	1.0	90.8	6.7	2.5	82.2	12.7	5.1	27.8	31.0	41.2

Where other model selection criteria seem to generate large trees with simple nodes, SRM seems to favour smaller trees with more complex nodes.

The number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for AIC, BIC, CV and SRM are given in Figure 3. We see that AIC selects quadratic nodes more than BIC. All omnivariate techniques except BIC select more complex models in the first level of the tree (linear for AIC and CV, quadratic for SRM). After the first level, univariate model is the first choice in all omnivariate techniques except SRM, where univariate model remains as the third choice until the fourth level. Again with the exception of SRM, all omnivariate techniques select linear model more than quadratic model in every level of the tree, where the

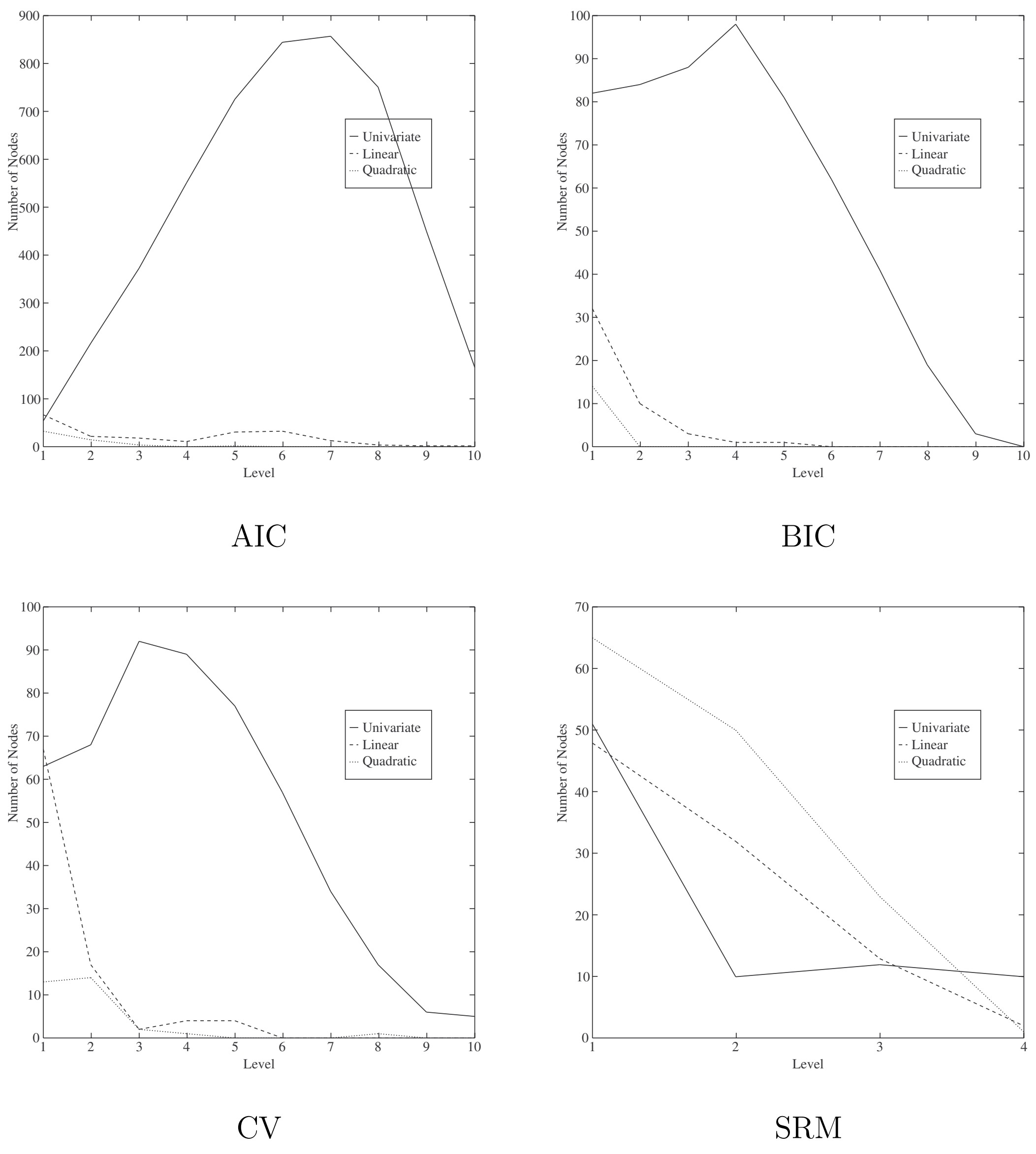

Fig. 3. Number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for AIC, BIC, CV and SRM.

former does the reverse.

We also see that, there are a lot of nodes in the deeper levels of the trees of AIC, BIC and CV, whereas the maximum depth of an SRM tree is 4 with a small number of nodes at each level. With respect to the performance of SRM in accuracy compared with other model selection techniques in omnivariate decision tree induction, we can easily say that there is usually no need to generate large trees, small trees can perform as well as large trees, and sometimes they do better. This may be due to two factors. First, SRM and BIC usually choose simpler models than other two model selection algorithms AIC and CV [17]. Second, our SRM based omnivariate decision tree algorithm uses prepruning, which usually prunes trees more than post-pruning.

There are also other related works that deal with simplicity in either decision tree induction or rule-learning. Holte [18] showed that single node univariate decision trees (decision stumps) perform quite well on 17 datasets taken from UCI repository. In another work Rückert [31] showed that with much smaller rule sets (which composed of fewer literals) one can obtain similar level of accuracy as other state-of-the-art rule-learners.

4.5 Comparison of SRM with Nontree Algorithms

In this subsection, we try to answer the following question that motivates us: Is our proposed model selection technique SRM is at least as good as state-of-the-art nontree algorithms in terms of error, model complexity and learning time? In order to answer this question, we compared omnivariate trees based on SRM with LDA, QDA and SVM on 17 datasets. The average and standard deviations of expected error, model complexity of different learning algorithms and time required to train those algorithms for 17 datasets are given in Tables 6, 7 and 8 respectively.

The average and standard deviations of expected error of the SRM based omnivariate decision tree algorithm and other state-of-the-art algorithms for 17 datasets are given in Table 6. There seems no statistical significant difference between four algorithms, though SRM wins more than any other three algorithms if compared (10 wins - 6 losses against LDA, 10 wins - 7 losses against QDA and 11 wins - 6 losses against SVM). If in a dataset nonlinear models get the best performance, such as on *ringnorm*, SRM will select nonlinear models as the root nodes because their performance is better than the linear models. Most of the datasets in our experiments are not so complex and they can be solved with linear models. Therefore the total win and loss numbers are usually dominated by those results.

Table 7 shows total model complexity in terms of number of free parameters for LDA, QDA, SVM and SRM. The model complexity of LDA, QDA are $d + 1$, $\frac{d^2+3d+2}{2}$ respectively. The model complexity of SVM is $d \times sv$, where d is number of features in the dataset and sv is total number of support vectors found for that problem. We see a clear ordering of SRM $>$ LDA $>$ QDA $>$ SVM from the table.

Table 8 shows time required to train each learning algorithm. Since LDA only does linear discriminant analysis, it has the smallest training time. QDA is the second winner, because this time only the number of parameters increases. There is not significant difference between SRM and SVM although SRM has

Table 6
Expected error rates and number of wins of all algorithms using average error rates. The bold face entries show statistically significant difference using the sign test.

Dataset	LDA	QDA	SVM	SRM
artificial	5.44±5.21	3.94±4.72	2.38±3.47	0.00±0.00
breast	4.32±0.59	5.04±1.74	3.78±0.88	4.29±0.65
bupa	32.30±2.54	39.19±2.02	34.03±2.71	32.35±2.06
german	23.72±1.85	31.94±8.01	26.00±1.31	28.26±1.71
haberman	24.90±1.29	25.17±1.23	26.73±0.48	27.78±2.27
heart	15.85±1.92	19.93±2.41	17.11±2.43	25.70±1.98
hepatitis	22.05±4.33	19.09±1.96	18.71±3.13	18.33±3.04
m.graphic	21.21±2.01	21.83±2.09	20.17±2.16	18.69±1.00
monks	54.91±4.64	8.56±6.72	25.00±2.93	25.28±2.71
parkinsons	16.52±3.92	13.14±2.64	15.80±2.69	13.86±3.80
pima	23.10±1.33	26.30±1.81	24.66±2.67	23.72±1.21
ringnorm	23.09±0.42	1.45±0.16	23.75±0.61	2.62±0.25
tictactoe	32.65±2.16	5.34±1.20	1.67±0.50	24.70±1.85
titanic	24.68±2.14	25.44±1.23	22.37±0.74	22.04±0.76
vote	15.68±1.81	5.56±1.30	4.55±1.32	4.23±1.09
transfusion	22.97±0.73	22.27±0.92	23.72±0.18	23.40±0.66
twonorm	2.25±0.19	2.29±0.20	2.48±0.36	2.25±0.19

	LDA	QDA	SVM	SRM
LDA	0	9	8	6
QDA	8	0	6	7
SVM	9	11	0	6
SRM	10	10	11	0

12 wins against 5 losses. Omnivariate tree trains all three models (univariate, multivariate linear and multivariate quadratic) at each node, where multivariate quadratic model requires most time to obtain (O(d^6) time to find the eigenvectors of the covariance matrix). LIBSVM uses sequential minimal optimization (SMO) to train Support Vector Machines with a time complexity of O(N^3) approximately.

Table 7
Model complexities in terms of number of free parameters and number of wins of all algorithms using average model complexities. The bold face entries show statistically significant difference using the sign test.

Dataset	LDA	QDA	SVM	SRM
artificial	62.00±0.00	132.00±0.00	289.00±294.94	12.60±2.84
breast	101.00±0.00	110.00±0.00	1178.10±715.13	15.10±6.87
bupa	50.00±0.00	56.00±0.00	663.00±32.53	24.60±15.37
german	600.00±0.00	650.00±0.00	6316.80±115.82	241.70±53.39
haberman	17.00±0.00	20.00±0.00	211.50±14.58	9.90±12.85
heart	197.00±0.00	210.00±0.00	963.30±279.25	108.10±35.26
hepatitis	392.60±10.84	420.00±0.00	556.70±41.09	83.70±24.11
m.graphic	206.00±0.00	272.00±0.00	4156.50±1155.69	13.70±9.87
monks	211.00±0.00	342.00±0.00	2551.70±114.31	13.40±29.73
parkinsons	292.40±7.59	552.00±0.00	833.80±77.23	23.50±7.63
pima	82.00±0.00	90.00±0.00	1564.80±173.12	20.40±15.14
ringnorm	442.00±0.00	462.00±0.00	39854.00±2334.71	520.10±71.16
tictactoe	495.00±0.00	812.00±0.00	8653.50±621.36	267.40±39.35
titanic	52.00±0.00	90.00±0.00	4556.00±5.66	6.10±1.45
vote	525.60±17.56	1122.00±0.00	1606.40±266.79	27.10±18.06
transfusion	20.00±0.00	30.00±0.00	589.20±6.55	4.90±6.49
twonorm	442.00±0.00	462.00±0.00	12880.00±5784.69	23.00±0.00

	LDA	QDA	SVM	SRM
LDA	0	**17**	**17**	1
QDA	0	0	**17**	1
SVM	0	0	0	0
SRM	**16**	**16**	**17**	0

5 Conclusion

We propose a novel omnivariate decision tree architecture based on Structural Risk Minimization. The ideal node type is determined via model selection method SRM. Such an omnivariate tree, instead of assuming the same bias at each node, matches the complexity of a node with the data reaching that node. Our previous works and also other works indicate that omnivariate

Table 8
Time complexities in terms of seconds to train the classifier and number of wins of all algorithms using average time complexities. The bold face entries show statistically significant difference using the sign test.

Dataset	LDA	QDA	SVM	SRM
artificial	0.00±0.00	0.03±0.00	0.04±0.01	0.14±0.00
breast	0.01±0.01	0.00±0.01	1.30±1.08	0.18±0.03
bupa	0.00±0.00	0.00±0.00	2.83±0.66	0.03±0.01
german	0.03±0.00	0.03±0.00	87.19±5.14	39.45±9.54
haberman	0.00±0.00	0.00±0.00	1.68±0.78	0.00±0.00
heart	0.00±0.00	0.00±0.00	3.43±2.59	0.85±0.10
hepatitis	0.01±0.01	0.14±0.03	0.06±0.06	5.45±1.48
m.graphic	0.01±0.00	0.01±0.00	53.98±37.62	3.15±1.05
monks	0.01±0.01	0.16±0.00	38.44±4.75	3.12±1.17
parkinsons	0.01±0.00	2.76±0.10	0.89±0.50	17.39±2.80
pima	0.00±0.00	0.01±0.01	11.37±1.67	0.12±0.01
ringnorm	0.14±0.01	0.14±0.00	1119.56±44.08	28.25±2.67
tictactoe	0.03±0.00	1.63±0.02	15.49±13.04	91.86±13.63
titanic	0.01±0.00	0.01±0.00	23.73±12.12	0.23±0.02
vote	0.02±0.00	10.62±0.38	0.41±0.62	184.78±36.62
transfusion	0.00±0.00	0.00±0.00	3.83±2.02	0.01±0.01
twonorm	0.14±0.00	0.14±0.01	120.17±20.22	20.73±0.09

	LDA	QDA	SVM	SRM
LDA	0	**7**	**17**	**16**
QDA	1	0	**14**	**16**
SVM	0	3	0	5
SRM	0	0	12	0

architecture generalizes better than pure trees with the same type of node everywhere.

For implementing SRM, we estimate the VC-dimension of univariate model whose VC-dimension can not be found theoretically. Experimental results showed that the VC-dimension estimate is equal to $1 + \log d$.

Our simulation results indicate that such an omnivariate architecture based on SRM generalizes at least as better as other omnivariate tree inducers using AIC, BIC or CV. SRM and BIC omnivariate trees have the smallest number of nodes with competitive complexity as CV.

Contrary to other omnivariate techniques, the univariate node type, is selected the least, followed by the linear node and the quadratic node. On the other hand, similar to other omnivariate techniques, 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. This shows that having a small percentage of multivariate (linear or quadratic) nodes is effective.

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 [11]. Because omnivariate 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.

References

[1] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: Second International Symposium on Information Theory, 1973, pp. 267–281.
URL `citeseer.ist.psu.edu/31739.html`

[2] E. Alpaydın, Combined 5×2 cv f test for comparing supervised classification learning classifiers, Neural Computation 11 (1999) 1975–1982.

[3] E. Alpaydın, Introduction to Machine Learning, The MIT Press, 2004.

[4] H. Altınçay, Decision trees using model ensemble-based nodes, Pattern Recognition 40 (2007) 3540–3551.

[5] C. Blake, C. Merz, UCI repository of machine learning databases (2000).
URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[6] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, John Wiley and Sons, 1984.

[7] L. A. Breslow, D. W. Aha, Simplifying decision trees: A survey, Tech. Rep. AIC-96-014, Navy Center for Applied Research in AI, Naval Research Laboratory, Washington DC, USA (1997).

[8] C. E. Brodley, P. E. Utgoff, Multivariate decision trees, Machine Learning 19 (1995) 45–77.

[9] C. C. Chang, C. J. Lin, LIBSVM: a library for support vector machines (2001). URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[10] V. Cherkassky, F. Mulier, Learning From Data, John Wiley and Sons, 1998.

[11] W. W. Cohen, Fast effective rule induction, in: The Twelfth International Conference on Machine Learning, 1995, pp. 115–123.

[12] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[13] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning classifiers, Neural Computation 10 (1998) 1895–1923.

[14] J. H. Friedman, A recursive partitioning decision rule for non-parametric classification, IEEE Transactions on Computers (1977) 404–408.

[15] J. Gama, Discriminant trees, in: 16th International Conference on Machine Learning, Morgan Kaufmann, New Brunswick, New Jersey, 1999, pp. 134–142.

[16] J. Gama, Functional trees, Machine Learning 55 (2004) 219–250.

[17] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Verlag, New York, 2001.

[18] R. C. Holte, Very simple classification rules perform well on most commonly used datasets, Machine Learning 11 (1993) 63–90.

[19] H. Kim, W. Loh, Classification trees with unbiased multiway splits, Journal of the American Statistical Association (2001) 589–604.

[20] N. Landwehr, M. Hall, E. Frank, Logistic model trees, in: Proceedings of the European Conference in Machine Learning, 2003, pp. 241–252.

[21] W. Li, C. F. J. Wu, Columnwise-pairwise algorithms with applications to the construction of supersaturated designs, Technometrics 39 (1997) 171–179.

[22] Y. Li, M. Dong, R. Kothari, Classifiability-based omnivariate decision trees, IEEE Transactions on Neural Networks 16 (6) (2005) 1547–1560.

[23] T. S. Lim, W. Y. Loh, Y. S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning 40 (2000) 203–228.

[24] W. Y. Loh, Y. S. Shih, Split selection methods for classification trees, Statistica Sinica 7 (1997) 815–840.

[25] W. Y. Loh, N. Vanichsetakul, Tree-structured classification via generalized discriminant analysis, Journal of the American Statistical Association 83 (1988) 715–725.

[26] T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[27] S. K. Murthy, Automatic construction of decision trees from data: A multi-disciplinary survey, Data Mininig and Knowledge Discovery 2 (4) (1998) 345–389.

[28] S. K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, Journal of Artificial Intelligence Research 2 (1994) 1–32.

[29] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Meteo, CA, 1993.

[30] C. E. Rasmussen, R. M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, R. Tibshirani, Delve data for evaluating learning in valid experiments (1995-1996).
URL `http://www.cs.toronto.edu/~delve/`

[31] U. Rückert, L. D. Raedt, An experimental evaluation of simplicity in rule learning, Artificial Intelligence 172 (2008) 19–28.

[32] G. Schwarz, Estimating the dimension of a model, Annals of Statistics 6 (1978) 461–464.

[33] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, New York, 1995.

[34] V. Vapnik, E. Levin, Y. L. Cun, Measuring the vc-dimension of a learning machine, Neural Computation 6 (1994) 851–876.

[35] O. T. Yıldız, E. Alpaydın, Omnivariate decision trees, IEEE Transactions on Neural Networks 12 (6) (2001) 1539–1546.

[36] O. T. Yıldız, E. Alpaydın, Linear discriminant trees, International Journal of Pattern Recognition and Artificial Intelligence 19 (3) (2005) 323–353.

[37] O. T. Yıldız, E. Alpaydın, Model selection in omnivariate decision trees, in: Proceedings of the 18th European Conference on Machine Learning, 2005.