Information Sciences 181 (2011) 5214–5226

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Model selection in omnivariate decision trees using Structural Risk Minimization

Olcay Taner Yıldız

Department of Computer Engineering, Işık University, TR-34980 Şile, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 7 October 2009
Received in revised form 9 January 2011
Accepted 17 July 2011
Available online 5 August 2011

Keywords:
Classification
Machine learning
Model selection
VC-dimension
Structural Risk Minimization
Decision tree

ABSTRACT

As opposed to trees that use a single type of decision node, an omnivariate decision tree contains nodes of different types. 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. In order to apply SRM for model selection, one needs the VC-dimension of the candidate models. In this paper, we first derive the VC-dimension of the univariate model, and estimate the VC-dimension of all three models (univariate, linear multivariate or quadratic multivariate) experimentally. Second, we compare SRM with other model selection techniques including Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC) and cross-validation (CV) on standard datasets from the UCI and Delve repositories. We see that SRM induces omnivariate trees that have a small percentage of multivariate nodes close to the root and they generalize more or at least as accurately as those constructed using other model selection techniques.

1. Introduction

The aim of machine learning is to extract the "best" model from a labeled training set where goodness is measured in terms of generalization accuracy and model complexity. There is a dependency between these two: if the model is too complex, we can easily learn the training data but we risk overfitting, that is, the model does not learn a general rule but the particular training data and gives large error on previously unseen test data. If the model is too simple, even the training data may not be learned well and the error will be large on both training and test data.

Decision trees are hierarchical structures where each internal node m implements a decision function, $f_m(\boldsymbol{x})$, each branch of an internal node corresponds to one outcome of the decision, and each leaf corresponds to a class. 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. Depending on the decision function $f_m(\boldsymbol{x})$ at the nodes, one can have different tree structures:

- If the decision function $f_m(\boldsymbol{x})$ uses only one dimension of the input vector $\boldsymbol{x}$, $(f_m(\boldsymbol{x})\text{: } x_j + w_{m0} > 0)$, where w_{m0} is some constant number [29], the decision tree is a *univariate* decision tree.
- If the decision function uses a linear combination of all attributes, $(f_m(\boldsymbol{x}) : \boldsymbol{w}_m^T\boldsymbol{x} + w_{m0} = \sum_{j=1}^{d} w_{mj}x_j + w_{m0} > 0)$ the decision tree is a *linear multivariate* decision tree. To calculate the weighted sum, all the attributes should be numeric and discrete values need to be represented numerically (usually by 1-of-L encoding) beforehand [40].

E-mail address: olcaytaner@isikun.edu.tr

0020-0255/$ - see front matter © 2011 Published by Elsevier Inc.
doi:10.1016/j.ins.2011.07.028

- If the decision function is a linear combination of nonlinear basis functions, $(f_m(\boldsymbol{x}) : \sum_{j=1}^{k} w_j \phi_j(\boldsymbol{x}) > 0)$, where $\phi_j(\boldsymbol{x})$ are the nonlinear basis functions, the decision tree is a *nonlinear multivariate* decision tree. In this work, we use a polynomial basis function of degree 2 where, for example, for $\boldsymbol{x} \in \Re^2$, $\phi(\boldsymbol{x}) = \left[1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\right]$ gives us a quadratic tree.

Decision trees are widely used in many applications because they are simple, easily comprehensible, robust to noisy data and can learn disjunctive expressions [27]. A recent survey comparing different decision tree methods with other classification algorithms is given in [24].

Our approach in solving the model selection problem in decision trees is based on Structural Risk Minimization (SRM). We control the complexity of each decision node by choosing the best model having minimum generalization error. With respect to SRM, in order to calculate the generalization error of a model, one needs (i) the training error of the model and (ii) the VC-dimension (the complexity) of the model. We have three candidate models, multivariate linear model (VC-dimension is equal to $d + 1$), multivariate quadratic model and univariate model, for which we do not know the exact VC-dimension. We first derive the VC-dimension of the univariate model, then estimate the VC-dimension of all three models experimentally [35].

In the second part of our work, we compare our proposed SRM based omnivariate decision tree algorithm with previous approaches. Our simulation results indicate that our proposed algorithm generalizes at least as good as other omnivariate tree inducers using AIC, BIC or CV. SRM and BIC omnivariate trees have the smallest number of nodes with competitive complexity compared to CV.

This paper is organized as follows: In Section 2, we give a short survey of multivariate decision tree techniques. In Section 3, we show previously applied model selection techniques for tree induction. In Section 4 we explain how to apply Structural Risk Minimization technique in decision tree induction. We give our experiments and results in Section 5 and conclude in Section 6.

2. Survey of multivariate decision tree techniques

Fig. 1 categorizes the linear multivariate decision tree algorithms proposed so far with respect to two dimensions, which are the branching factor (two or K, number of classes) and the optimization method to find the best split vector and split point.

In multivariate linear trees, Linear Discriminant Analysis (LDA) was first used by Friedman [13] for constructing decision trees. LTREE [14] is the second work using LDA. LTREE uses Linear Discriminant Analysis to construct new features, which are linear combinations of the original features. Functional Trees [15] make simultaneous use of functional nodes and functional leaves; Gama [15] showed that the variance component of the error can be reduced using functional leaves, whereas the bias component can be reduced using decision functions in the non-leaf nodes ($f_m(\boldsymbol{x})$).

In FACT [26], each branch has its modified linear discriminant function calculated using LDA and an instance is forwarded to branch i to minimize estimated expected risk. QUEST [25] is a revised version of FACT and does not assume equal variances and uses Quadratic Discriminant Analysis (QDA) to find the best split point. CRUISE [20] makes minor improvements over the FACT algorithm. For instance, if the number of instances of two or more classes are same in a leaf, FACT selects randomly one of the classes but CRUISE selects the class which has not been assigned to any leaf node.

CART [5] is one of the best known multivariate linear decision tree construction algorithm, which fine-tunes the weights w_{mj} one by one. CART also has a preprocessing step to decrease dimensionality using subset selection. An extension of CART, OC1 [28] tries to solve the local minima problem of CART by adding a small random vector to $\boldsymbol{w}_m$.

In Neural Trees [16], each decision node uses a multilayer perceptron and this implements multivariate *nonlinear* decision trees. Backpropagation is used to learn the network parameters in each node. Logistic model trees [21] uses logistic classification to find the best split at each decision node, where a stepwise fitting process constructs logistic classification models

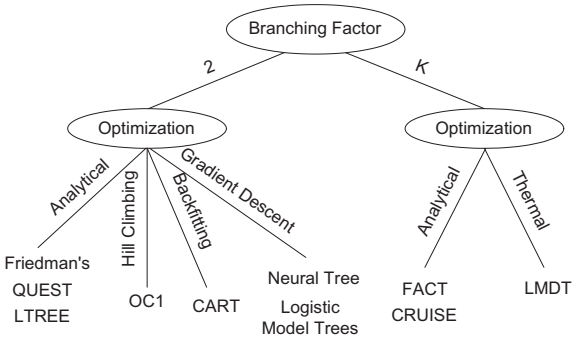

Fig. 1. Multivariate decision tree construction algorithms classified according to two dimensions, which are the branching factor and the optimization method to find the direction vector $\boldsymbol{w}$ and the split point w_0.

that can select relevant attributes in the data. In LMDT [6], 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 [2].

Fuzzy decision trees are one of the new approaches in decision tree induction [19,37]. Wang and Dong [36] propose some new refinement techniques by maximizing the information entropy and fusing the reductions based on upper–lower approximations which aim to improve the generalization error of fuzzy rules and fuzzy decision trees.

There are also other issues in decision tree construction such as efficient decision tree construction, cost-sensitive decision trees, and using new splitting measures in growing trees [7]. Yen and Chu [38] make the selection of candidate split points more flexible to improve efficiency in the search for splitting points of numerical attributes. Chen et al. [8] incorporate cost, namely test costs that must be paid to obtain the values of the decision attribute, into univariate decision tree induction.

3. Tuning model complexity in decision trees

In this paper, we are trying to solve the complexity control problem in decision trees. Controlling complexity in decision trees can be done in two ways. We can control the complexities of the decision nodes by selecting the best model for a node, and/or we can control the overall complexity of the decision tree via pruning. 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.

Previous model selection approaches used in decision tree induction are penalization techniques and cross-validation. Penalization techniques (Akaike's Information Criterion (AIC) or Bayesian Information Criterion (BIC)), where prior information, proportional to the complexity of the model, is introduced in the form of a penalty term. The performance of the model is the sum of the error of the model on the data and the complexity of the model. Cross-validation (CV) technique does not make any assumptions about model complexity or the prior model probabilities, and choose the best model according to the performance of the candidate models on the unseen validation data. Although cross-validation is an unbiased technique for model selection, large-scale cross-validation can be computationally expensive.

We have previously proposed the omnivariate decision tree architecture [39], using cross-validation to decide on the best model from three different candidates of a univariate node, multivariate linear node (linear perceptron) and a multivariate nonlinear node (implemented by a multilayer perceptron (MLP)). Continuing this work, Altınçay [3] 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 their work, Li et al. [23] used a novel classifiability measure related to Bayes error and the boundary complexity instead of pairwise statistical tests to perform model selection at each decision node.

In our later work [41], 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) [2]. 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, $(x_1 + x_2 + \cdots + x_d + 1)^2$, 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 log-likelihood and model complexity. In the last step, we choose the optimal model having the least generalization error.

In order to apply model selection techniques in tree induction, we need the log likelihood and error at each decision node. We apply the decision function and label the left and right branches with the class (C_L for the left branch, C_R for the right branch) having the most number of instances. N_{C_L} and N_{C_R}, the number of instances of the classes C_L and C_R respectively, will be calculated as:

$$N_{C_L} = \max_i N_i^L, \quad N_{C_R} = \max_i N_i^R \tag{1}$$

where N_i^L and N_i^R are the number of instances of class i choosing the left and right branches respectively.

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:

$$e = \frac{N - N_{C_L} - N_{C_R}}{N} \tag{2}$$

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

$$\mathcal{L} = \sum_{i=1}^{K} N_i^L \log \frac{N_i^L}{N^L} + \sum_{i=1}^{K} N_i^R \log \frac{N_i^R}{N^R} \tag{3}$$

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

3.1. Akaike's Information Criterion

AIC [1] is calculated as

$$AIC = 2(-\mathcal{L} + p) \tag{4}$$

where $\mathcal{L}$ represents the log-likelihood 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.

3.2. Bayesian Information Criterion

BIC [32] is calculated as

$$BIC = -\mathcal{L} + \frac{p}{2}\log N \tag{5}$$

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

3.3. Cross-validation

We use 5×2 cross-validation and train all three models and test them on the validation set 10 times and then apply the one-sided version of the 5×2 cv t test [12].

When we have two candidate models we choose the simpler model, if it has smaller or equal error rate compared to the complex model. The complex model is chosen only if it has significantly smaller error rate. 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 $H_0 : e_L \leqslant e_Q$ and $H_0 : e_U \leqslant e_Q$ are rejected. Otherwise, L is chosen if $H_0 : e_U \leqslant e_L$ is rejected. Otherwise U is chosen.

Note that AIC and BIC do not require a validation set and training is done once, whereas with 5×2 CV, in each fold, half of the data is left out for validation and training is done 10 times.

4. Structural Risk Minimization

Structural Risk Minimization (SRM) [34] 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 [9]

$$E_{generalization} = e + \frac{\lambda}{2}\left(1 + \sqrt{1 + \frac{4e}{\lambda}}\right) \tag{6}$$

where e is the training error and

$$\lambda = a_1 \frac{h[\log(a_2 N/h) + 1] - \log(v)}{N} \tag{7}$$

Here, h represents the VC dimension of the model, v represents the confidence level (it is recommended to use $v = \frac{1}{\sqrt{N}}$ for large sample sizes), and a_1 and a_2 are empirically fitted constants. In our experiments we did a grid-search on a_1 and a_2 using cross-validation and used $a_1 = 0.2$ and $a_2 = 0.8$.

For a class of functions $f(x,\alpha)$, where α denotes the parameter vector, the VC-dimension 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.

4.1. Structural Risk Minimization in omnivariate decision tree induction

In our experiments, we have three different models (univariate model, multivariate linear model, multivariate quadratic model) to choose from at each decision node. We calculate the generalization error of these three models using Eqs. (6) and (7), and choose the model with the smallest generalization error. To calculate the generalization error, we need (i) the training error (which can be estimated via Eq. (2)), 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 5.2 that these numbers are in accordance with our VC-dimension estimations).

We give the following theorem for the discrete univariate decision tree with a single decision node. Although the theorem is for discrete univariate decision trees, for single node case, since there is only one split to choose, discrete and continuous are the same.

Theorem 1. *The VC-dimension of discrete univariate decision tree with a single decision node that classifies d dimensional data is* $\lfloor log_2(d + 1) \rfloor + 1$.

Proof. To show that the VC-dimension of the discrete univariate decision tree with a single decision node is at least h, we need to find a sample S of size h such that for each possible class labelings of these h points, there is an instantiation of our single node decision tree hypothesis class H that classifies it correctly. We construct the sample S such that each feature x_i corresponds to a distinct possible class labeling of h points, implying a one-to-one mapping between class labelings and features (namely, the identity function, since both features and class labelings come from the same set). So for each possible class labeling, we will choose the decision tree hypothesis which has the corresponding feature as the split feature. A sample with h examples can be divided into two classes in $2^{h-1} - 1$ different ways. If we set the number of features to that number:

$$\begin{aligned} d &= 2^{h-1} - 1 \\ d + 1 &= 2^{h-1} \\ \log_2(d + 1) &= h - 1 \\ h &= \log_2(d + 1) + 1 \end{aligned} \tag{8}$$

To show that the VC-dimension of the discrete univariate decision tree with a single decision node is at most $\lfloor \log_2(d + 1) \rfloor + 1$, we go in reverse direction. If the VC-dimension of a single node univariate decision tree is h, for each possible class combination of h examples, we must be able to separate them into two classes. In a single node discrete decision tree, we can have at most d possible orthogonal splits. A sample with h examples can be divided into two classes in $2^{h-1} - 1$ different ways. In order to be able to separate h instances for each possible class combination, the total number of splits must be at least as large as the number of possible class divisions. So,

$$\begin{aligned} d &\geqslant 2^{h-1} - 1 \\ h &\leqslant \log_2(d + 1) + 1 \qquad \square \end{aligned} \tag{9}$$

We also estimate the VC-dimension of all three models by using a technique proposed in [35] 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. Hence 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 4.2 and 5.2 respectively.

4.2. Estimation of VC-dimension: uniform design

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

$$\Phi(\rho) = a\frac{ln(2\rho) + 1}{\rho - k}\left(\sqrt{1 + \frac{b(\rho - k)}{ln(2\rho) + 1}} + 1\right) \tag{10}$$

when $\rho \geqslant 0.5$ with $\rho = N/h$ and the constants $a = 0.16$, $b = 1.2$, $k = 0.14928$ are determined empirically [35]. If $\rho < 0.5$, $\Phi(\rho) = 1$. According to Vapnik et al. [35], 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 experiments at the design points are equal, the design is called uniform design [22].

The pseudocode of uniform design for estimating the VC-dimension is given in Fig. 2. 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 $\bar{\epsilon}(N_i)$ 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 $0.5 \leqslant N_i/h \leqslant 30$ (Line 3). The VC-dimension of the univariate model is then h (Line 14) which minimizes

```
1  VcEstimate UniformDesign(d; D)
2     for i = 1 to D
3        N_i = Sample size where 0.5 ≤ N_i/h ≤ 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          ε(N_{i,j}) = |E(S_1) − E(S_2)|
13       ε̄(N_i) = (Σ_{j=1}^{m_i} ε(N_{i,j})) / m_i
14    h = Best fit between Φ(N_i/h) and ε̄(N_i)
15    return h
```

Fig. 2. Pseudocode of uniform design: *d*: number of input features and *D*: number of design points.

$$\sum_{i=1}^{D}[\bar{\epsilon}(N_i) - \Phi(N_i/h)]^2 \tag{11}$$

4.3. Pruning in omnivariate decision tree induction

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 simply 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.

5. Experiments

5.1. Experimental setup

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 [4] and 3 (*titanic, ringnorm, and twonorm*) are from Delve [30] repositories. We also use 2 (*acceptors, polyadenylation*) bioinformatics datasets [33] in our experiments, which are high dimensional sparse datasets.

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). Demsar [11] 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 both sign test and Friedman's test together with its post hoc Nemenyi's test in comparing our proposed SRM based omnivariate decision tree algorithm with other learning algorithms. The details of the sign test and Friedman's test with its post hoc Nemenyi's test are given in the Appendix.

5.2. 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 $\mathcal{O}(k^3)$ time. In the multivariate quadratic model, we have d^2 parameters and therefore we need approximately $\mathcal{O}(d^6)$ 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. Fig. 3 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.

5.3. 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 1–3 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) according to sign test. 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.

If we compare algorithms using Friedman's test and its post hoc Nemenyi test, on average with respect to expected error, SRM ranked first (average rank: 1.76), CV ranked second (2.38), and AIC (2.82) and BIC (3.02) ranked third and fourth respectively. Friedman's test rejects the null hypothesis that all classifiers have the same expected error. Further analysis show that, at the p-value of 0.10, AIC and BIC are significantly worse than SRM, and the difference between CV and other algorithms are not significant.

Different model selection criteria rely on different parameters. For example, AIC uses log-likelihood and number of parameters, BIC uses log-likelihood and number of instances, etc. Two of these parameters, namely number of dimensions and number of instances, appear nearly in each approach. We therefore look how the relative performance of model selection criteria changes with respect to these parameters. In terms of number of dimensions, the relative performance of algorithms do not change, the average ranks for low-dimensional datasets ($d < 10$) are 2.86, 3, 2.36, 1.79; the average ranks for high dimensional datasets ($d > 10$) are 2.8, 3.05, 2.4, 1.75 for AIC, BIC, CV, and SRM respectively. In terms of number of instances, as we go from small-size datasets ($N < 500$) to large-size datasets ($N > 500$), the relative performance of AIC decreases (which do not use N in its calculations), BIC, SRM, and CV (which use N in their calculations) increases.

Table 2 shows total decision tree complexity in terms of number of free parameters for different omnivariate decision tree techniques. As explained in Section 3, 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 qua-

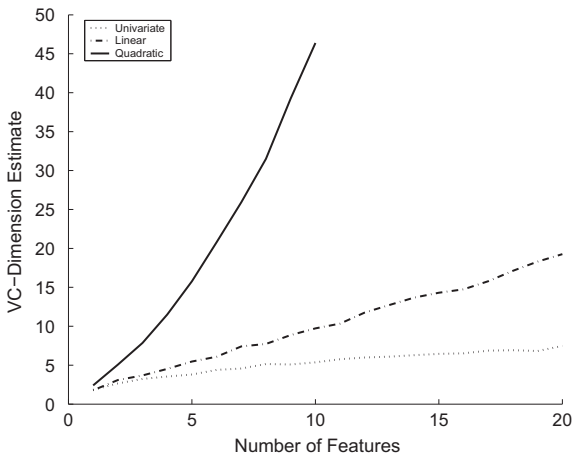

Fig. 3. VC-dimension estimate of univariate, linear and quadratic models with respect to number of features in the dataset.

Table 1
The first table shows expected error rates of omnivariate decision tree algorithms. For each dataset the best result is shown in bold face. The second table contains pairwise comparisons where (i,j) values are the number of datasets on which algorithm i is better than algorithm j. The bold face entries in the second table 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 2
The first table shows tree complexities of omnivariate decision tree algorithms in terms of number of free parameters. For each dataset the best result is shown in bold face. The second table contains pairwise comparisons where (i,j) values are the number of datasets on which algorithm i is better than algorithm j. The bold face entries in the second table 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

dratic 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].

According to the Friedman's test and its post hoc Nemenyi test, with respect to tree complexity, BIC ranked first (average rank: 1.47), CV ranked second (2.17), and SRM (2.76) and AIC (3.59) ranked third and fourth respectively. Friedman's test rejects the null hypothesis that all classifiers have the same tree complexity. At the p-value of 0.10, BIC is significantly better than AIC and SRM, where CV is also significantly better than AIC.

Table 3 shows time required to generate the decision trees for each omnivariate algorithm. Since CV tries all possible models 10 times (because of 5×2 cross-validation), whereas AIC, BIC and SRM tries all possible models once, the former's

Table 3
The first table shows time complexities of omnivariate decision tree algorithms in terms of number of seconds to generate the decision tree. For each dataset the best result is shown in bold face. The second table contains pairwise comparisons where (i,j) values are the number of datasets on which algorithm i is better than algorithm j. The bold face entries in the second table 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

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

Comparison using Friedman's test and its post hoc Nemenyi test according to time complexity shows that there are three groups of algorithms: SRM is the best classifier as a group of its own (average rank: 1.00), AIC (2.35) and BIC (2.65) are the members of the second group, which is significantly worse than the best group and significantly better than the third group, and CV (4.00) is the only member of the worst group.

5.3.1. Comparison on sparse datasets

One of the main contributions of Structural Risk Minimization is the generalization ability in sparse high-dimensional data, where the the number of features d is larger than the number of observations N. Such problems get more and more interest currently, especially in the field of genomics and other areas of computational biology. Since the number of dimensions is high, overfitting and therefore model selection is a major concern for these problems. We selected two bioinformatics datasets for this setting, namely *acceptors* and *polyadenylation*, and compare our proposed SRM based omnivariate decision tree algorithm with previous approaches.

On *acceptors* dataset, SRM is the best model selection technique with minimum error (8.02) and least complexity (260.00). The second best algorithm is based on cross-validation and has nearly same complexity as SRM (260.20). AIC and BIC based omnivariate decision tree algorithms do not prune the decision tree well, and therefore have both significantly large error rate and tree complexity.

On *polyadenylation* dataset, the results again show that AIC and BIC do not perform model selection as good as SRM. They either select too simple models (BIC prunes decision tree heavily and returns a leaf node with high error rate (48.06)), or too complex models (AIC does not prune the decision tree and returns a significantly large tree (tree complexity 1230) with high error rate (29.28)). On the other hand, SRM and CV select simple and accurate models with a tree complexity of 99.40 and an error rate of 21.40.

5.4. 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; especially, 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.

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

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%, nonlinear 41.17%). 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%, quadratic 5.12%). Where other model selection criteria seem to generate large trees with simple nodes, SRM seems to favor 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 Fig. 4. 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 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].

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

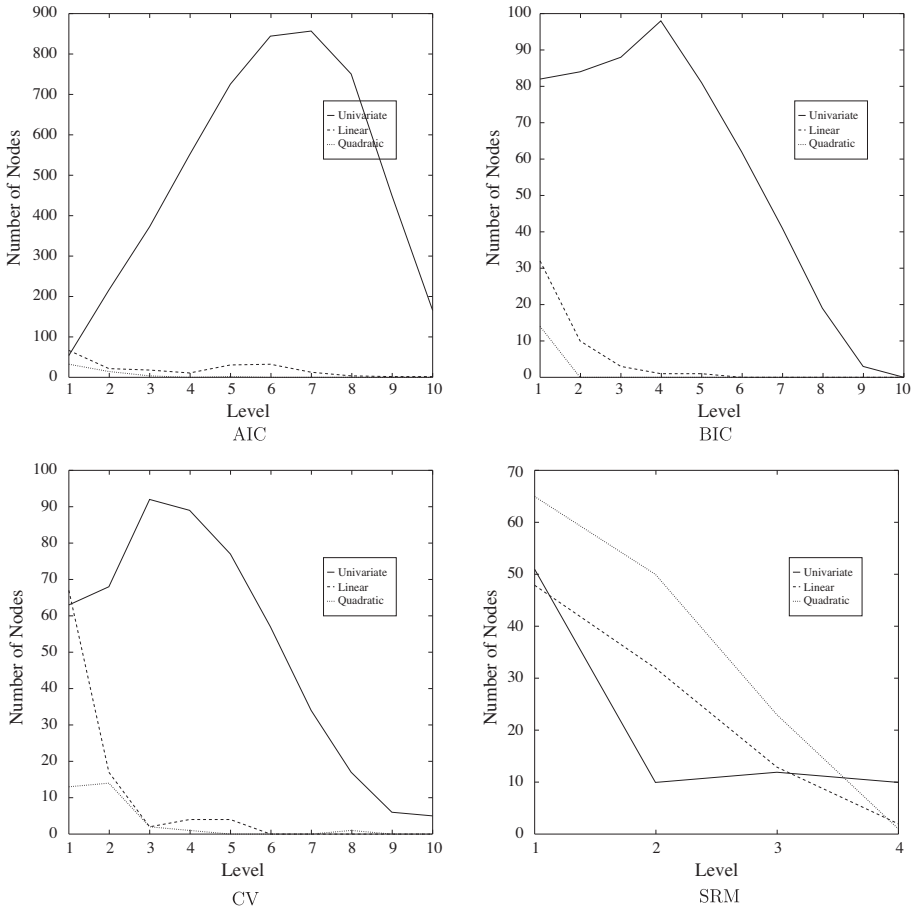

Fig. 4. Number of times univariate, multivariate linear and multivariate quadratic nodes selected at different levels of tree for AIC, BIC, CV and SRM.

Second, our SRM based omnivariate decision tree algorithm uses prepruning, which usually prunes trees more than postpruning.

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.

6. 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 to that node. Our previous work [39] and work by others indicate that omnivariate architecture generalizes better than pure trees with the same type of node everywhere.

For implementing SRM, we first give and prove the VC-dimension of the univariate model (which is $1 + \log d$), then estimate the VC-dimension of all three models experimentally. The experimental results show that the VC-dimension of the quadratic model is $\frac{d^2+3d+2}{2}$.

Our simulation results comparing our proposed omnivariate architecture with previous approaches show that SRM-based omnivariate trees generalize at least as well as other omnivariate tree inducers using AIC, BIC, or CV. SRM and BIC omnivariate trees have the smallest number of nodes with competitive complexity compared to CV.

Contrary to other omnivariate techniques with SRM, 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.

The main advantage of SRM over CV is that, SRM can use whole training set to induce the omnivariate tree, whereas CV needs a separate validation set, which is usually taken from the training set by reducing its size. For small datasets, this reduction will degrade the performance of the classifier. SRM selects complex models more than AIC, BIC, and CV resulting in (i) higher accuracy (complex models are better than simple models if overfitting does not occur), (ii) higher complexity and (iii) lower training time (small number of omnivariate nodes requires less time to train).

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) is used to prune the tree; the same is also true for rule learners such as Ripper [10]. 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.

Acknowledgments

The authors thank the three anonymous referees and the editor for their constructive comments, pointers to related literature, and pertinent questions which allowed us to better situate our work as well as organize the ms and improve the presentation. This work has been supported by the Turkish Scientific Technical Research Council TÜBİTAK EEEAG 107E127.

Appendix A. Sign test

Given M data sets, let the number of wins of one algorithm over the other be w, and let the number of losses be l where $M' = w + l$ (we ignore the ties). The sign test assumes that the wins/losses are binomially distributed and tests the null hypothesis that $w = l$. We calculate $p = B(w, M')$ of the binomial distribution and if $p > \alpha$, we fail to reject the hypothesis that the two have equal error with significance α. Otherwise, we say that the first one is more accurate if $w > l$, and the second one is more accurate otherwise.

Appendix B. Friedman's test and Nemenyi's test

Friedman's test is the nonparametric equivalent of ANOVA. First, all algorithms are ranked on each data set using the average error on the validation folds, giving the best one rank 1. If the algorithms have no difference between their expected errors, then their ranks should not be different either, which is what is tested by Friedman's test. Let r_{ij} be the rank of algorithm $j = 1,\ldots,L$ on data set $i = 1,\ldots,M$ and $R_j = \frac{1}{M}\sum_i r_{ij}$ be the average rank of algorithm j. The Friedman test statistic is:

$$X^2 = \frac{12M}{L(L+1)}\left[\sum_j R_j^2 - \frac{L(L+1)^2}{4}\right]$$

which is chi-square distributed with $L-1$ degrees of freedom. If the test fails to reject, we will say that we cannot find any difference between the means of the L algorithms and we do no further processing; if the test rejects, one uses the post hoc Nemenyi's test.

According to Nemenyi's test, two algorithms have different error rates if their average ranks differ by at least a critical difference, $CD = q_\alpha\sqrt{\frac{L(L+1)}{6M}}$, where values for q_α are based on the Studentized range statistic divided by $\sqrt{2}$.

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.
[2] E. Alpaydın, Introduction to Machine Learning, The MIT Press, 2004.
[3] H. Altınçay, Decision trees using model ensemble-based nodes, Pattern Recognition 40 (2007) 3540–3551.
[4] C. Blake, C. Merz, UCI Repository of Machine Learning Databases, 2000. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
[5] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, John Wiley and Sons, 1984.
[6] C.E. Brodley, P.E. Utgoff, Multivariate decision trees, Machine Learning 19 (1995) 45–77.
[7] B. Chandra, P.P. Varghese, Moving towards efficient decision tree construction, Information Sciences 179 (2009) 1059–1069.
[8] Y.-L. Chen, C.-C. Wu, K. Tang, Building a cost-constrained decision tree with multiple condition attributes, Information Sciences 179 (2009) 967–979.
[9] V. Cherkassky, F. Mulier, Learning From Data, John Wiley and Sons, 1998.
[10] W.W. Cohen, Fast effective rule induction, in: The Twelfth International Conference on Machine Learning, 1995, pp. 115–123.
[11] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[12] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning classifiers, Neural Computation 10 (1998) 1895–1923.
[13] J.H. Friedman, A recursive partitioning decision rule for non-parametric classification, IEEE Transactions on Computers (1977) 404–408.
[14] J. Gama, Discriminant trees, in: Sixteenth International Conference on Machine Learning, Morgan Kaufmann, New Brunswick, New Jersey, 1999, pp. 134–142.
[15] J. Gama, Functional trees, Machine Learning 55 (2004) 219–250.

[16] H. Guo, S.B. Gelfand, Classification trees with neural network feature extraction, IEEE Transactions on Neural Networks 3 (1992) 923–933.
[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] C.Z. Janikow, Fuzzy decision trees: issues and methods, IEEE Transactions on Systems, Man and Cybernetics 28 (1998) 1–14.
[20] H. Kim, W. Loh, Classification trees with unbiased multiway splits, Journal of the American Statistical Association (2001) 589–604.
[21] N. Landwehr, M. Hall, E. Frank, Logistic model trees, in: Proceedings of the European Conference in Machine Learning, 2003, pp. 241–252.
[22] W. Li, C.F.J. Wu, Columnwise–pairwise algorithms with applications to the construction of supersaturated designs, Technometrics 39 (1997) 171–179.
[23] Y. Li, M. Dong, R. Kothari, Classifiability-based omnivariate decision trees, IEEE Transactions on Neural Networks 16 (6) (2005) 1547–1560.
[24] 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.
[25] W.Y. Loh, Y.S. Shih, Split selection methods for classification trees, Statistica Sinica 7 (1997) 815–840.
[26] W.Y. Loh, N. Vanichsetakul, Tree-structured classification via generalized discriminant analysis, Journal of the American Statistical Association 83 (1988) 715–725.
[27] S.K. Murthy, Automatic construction of decision trees from data: a multi-disciplinary survey, Data Mining 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 Kaufman, 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. <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] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, S. Levy, A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis, Bioinformatics 21 (2005) 631–643.
[34] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, New York, 1995.
[35] V. Vapnik, E. Levin, Y.L. Cun, Measuring the VC-dimension of a learning machine, Neural Computation 6 (1994) 851–876.
[36] X.Z. Wang, C.R. Dong, Improving generalization of fuzzy if-then rules by maximizing fuzzy entropy, IEEE Transactions on Fuzzy Systems 17 (2009) 556–567.
[37] X.Z. Wang, J.H. Zhai, S.X. Lu, Induction of multiple fuzzy decision trees based on rough set technique, Information Sciences 178 (2008) 3188–3202.
[38] E. Yen, I.-W.M. Chu, Relaxing instance boundaries for the search of splitting points of numerical attributes in classification trees, Information Sciences 177 (2007) 1276–1289.
[39] O.T. Yıldız, E. Alpaydın, Omnivariate decision trees, IEEE Transactions on Neural Networks 12 (6) (2001) 1539–1546.
[40] O.T. Yıldız, E. Alpaydın, Linear discriminant trees, International Journal of Pattern Recognition and Artificial Intelligence 19 (3) (2005) 323–353.
[41] O.T. Yıldız, E. Alpaydın, Model selection in omnivariate decision trees, in: Proceedings of the 18th European Conference on Machine Learning, 2005, pp. 473–484.