Univariate and Multivariate Decision Trees

Olcay Taner Yıldız and Ethem Alpaydın

Department of Computer Engineering
Boğaziçi University
İstanbul 80815 Turkey

Abstract. Univariate decision trees at each decision node consider the value of only one feature leading to axis-aligned splits. In a linear multivariate decision tree, each decision node divides the input space into two with an arbitrary hyperplane leading to oblique splits. In a nonlinear one, a multilayer perceptron at each node divides the input space arbitrarily, at the expense of increased complexity. In this paper, we detail and compare using a set of simulations linear and non-linear neural based decision tree methods with the univariate decision tree induction method ID3 and the linear multivariate decision tree induction method CART. We also propose hybrid trees where the decision node may be linear or nonlinear depending on the outcome of a statistical test on accuracy.

1 Introduction

Decision tree construction algorithms are greedy in that at each step, we decide on a decision node that best splits the data for classification. Different decision tree learning methods differ in the way they make decisions at a node. In a *univariate decision tree*, at each node, only one feature is used. If the feature is continuous, the decision is of the form

$$x_i > c_k$$

making a binary split where x_i is the ith input feature and c_k is a suitably chosen threshold. If x_i is discrete valued with m values, then the node makes an m-ary split. *ID3 algorithm* [9] is used to construct univariate decision trees. Consider the two-dimensional instance space shown in Figure 1. To approximate the hyperplane boundary, the corresponding univariate decision tree uses a series of orthogonal splits. This example shows that a univariate test using feature x_i can only split a space with a boundary that is orthogonal to x_i axis. This results in large trees and poor generalization.

In a *multivariate decision tree*, each decision node is based on more than one feature. In Figure 1 we see that with one line (or with one non-linear function), we can separate the examples of two classes, reducing a tree with three nodes to a tree with one node. The *linear* multivariate tree-constructing algorithms select not the best attribute but the best linear combination of the attributes. The decision at each node is of the form

$$\sum_{i=1}^{d} w_i x_i > w_0 \tag{1}$$

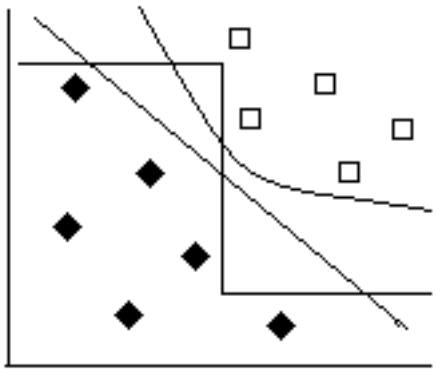

Figure 1. Comparison of univariate and multivariate splits on the plane (Diamond class 1, rectangle class 2).

where w_i are the weights, and w_0 is a threshold. CART algorithm is a stepwise procedure to learn the w_i and w_0 from a data set, where at each step, one cycles through the features $x_1, x_2 \ldots x_n$ doing a search for an improved linear combination split [3].

2 Issues in Tree Construction

A decision tree algorithm must handle both ordered and unordered features. In multivariate decision tree algorithms, we must convert each unordered feature i into a numeric representation to be able to take a linear combination of the features. In our implementation where x_i is the unordered feature we convert it into multiple feature set as $i_1, i_2, \ldots$ in where i_j=1 if x_i takes value a_j and 0 otherwise. This mapping avoids imposing any order on the unordered values of the feature [10].

In cases where a feature has a missing value, if the missing value is for an ordered feature, we fill it with the sample mean value of that feature as it appears in the training set. If the feature is unordered, we fill it with the most probable value of that feature in the data set. These statistics are stored so that the same filling can be done for the test set.

At the leaves of the tree, the leaf node has the label having the most instances. In our implementation we have used pre-pruning to avoid overfitting. At a node, we stop and do not split further if the number of instances is less than, e.g., 5%, of the training set. [3] [4] [5].

In multivariate decision tree construction method CART, we take linear combination of features. Our main purpose is to find the correct coefficients wi for these features. But using all of the features may be costly and may lead to overfitting on a small training set. So in order to get rid of these unnecessary features, we also used feature selection [3].

3 Neural Network Based Decision Tree Construction Methods

The decision at a multivariate node can also be seen as a single layer perceptron and thus w_i (and w_0) can be determined by training a perceptron with 0/1 output making a binary split.

Finding the best split with neural network algorithms is done as a two-nested optimization problem. In the inner optimization problem, the gradient-descent algorithm is used to find w_i (and w_0) that minimize the mean-square error and so find a good split for the given two distinct groups of classes. In the outer optimization problem, we use a method to find the best split of K classes into two groups.

In order to train the linear perceptron, gradient-descent is used. This algorithm is used to train a single-output linear perceptron to differentiate between two disjoint groups of classes, which are the classes in the left branch C_L and the classes in the right branch C_R. Desired output for each instance is 1 when its class belongs to the group C_L and 0 when its class belongs to the group C_R. If there are only two classes present in that node one class belongs to C_L and other to C_R. Otherwise we must select an appropriate partition for the K classes we have (Section 3.2).

3.1 Training neural networks

The inner optimization problem will be solved by using a separate neural network at each node of the decision tree. For this purpose we have used three different kinds of neural network models. These are linear perceptrons (linear multivariate), multilayer perceptrons (nonlinear multivariate), and a hybrid combination.

Linear Perceptron Model x_i's correspond to the normalized values of the attributes for each instance. w_i's correspond to the coefficients of the attributes and -w_0 is an input-independent offset. Training of this network is done by gradient-descent algorithm. Let (x^t, d^t) denote the training data, where x^t is the t'th instance and d^t is the desired output for that instance, which is 1 for left child and 0 for right child. y^t is the real output found by the formula:

$$y^t = sig(\sum_{i=1}^{d} w_i x_i^t + w_0) \tag{2}$$

A stochastic gradient algorithm for minimizing the least-squares error criterion is used (*sig* is the sigmoid or logistic function).

Multilayer Perceptron Model In a multilayer perceptron, the output y^t is found as:

$$y^t = sig(\sum_{h=0}^{H} T_h sig(\sum_{i=0}^{d} w_{hi} x_i^t + w_{h0}) + T_0) \tag{3}$$

The parameters are both w_{hi} and T_h which respectively are the first and second layer weights. H is the number of hidden units and is taken as $d/2$, half of the number of inputs[2].

A Hybrid Model If we compare the complexity of the decision at a node, we see that the univariate methods are the simplest methods as they test only one feature. Linear methods, which take a linear combination of features, are more complex and non-linear methods such as multilayer perceptrons, are the most complex. But the aim of the decision trees is to find a way of representation of the decision that is as simple as possible and as accurate as possible. So we should not use always non-linear methods, which have too many parameters, are prone to overfitting, and are not interpretable.

So it seems better to find a way of combining linear and non-linear methods in a hybrid model. In this model, at each node we train both a linear and non-linear perceptron and use a statistical test to check if there is a significant performance difference between the two. If the performance of the multilayer perceptron is better then the performance level of the linear perceptron with a confidence level of %95, the multilayer perceptron is chosen as the training network, else linear perceptron is chosen. 95% is a good value as we prefer simpler trees and rarely would we choose the multilayer perceptron over the linear perceptron. We used two different tests; the combined 5x2 cv F test[1] and the paired t test [6]. Our results indicate that the performances of these two tests are almost equal and we prefer the F test, as it requires 10-fold validation whereas the t test requires 30-fold validation.

3.2 Class Separation by Exchange Method

One detail we seem to have skipped above is the following: The sigmoidal neural network makes a binary decision and we may have $K > 2$ classes and thus at each node, we need to find the best way of partitioning K classes into two subsets. If there are K classes available at a node then there are $2K$-1 distinct partitions available. So because of the number of available partitions grows exponentially we can not test for all possible partitions.

We use exchange method in class separation[7]. This is a local search with backtracking. Let t be a decision node and $C = \{C_1, \ldots, C_K\}$ be the set of K classes at node t.

1. Select an initial partition of C into C_L and C_R.
2. Train the network to separate C_L and C_R. Compute the entropy E_0 with the selected entropy formula.
3. For each of the classes $k \in \{C_1, \ldots, C_K\}$ form the partitions $C_L(k)$ and $C_R(k)$ by changing the assignment of the class C_k in the partitions C_L and C_R.
4. Train the neural network with the partitions $C_L(k)$ and $C_R(k)$. Compute the entropy E_k and the decrease in the entropy $\Delta E_k = E_k - E_0$.
5. Let ΔE^* be the maximum of the impurity decreases for all classes. If this impurity decrease is less than zero then exit else set $C_L = C_L(k), C_R = C_R(k)$, and goto step 2.

In order to get a good solution in reasonable time we use a heuristic technique to start, instead of starting randomly. The two classes C_i and C_j with the maximum distance are found and placed into C_L and C_R. For each of the classes k $= C_1, \ldots, C_m$ we find the one with the minimum distance to C_L or C_R and then we put it into that group. We repeat this second step until no more classes are left.

4 Evaluation of Decision Tree Construction Methods

To evaluate the four decision tree construction methods: univariate ID3, linear multivariate CART, linear multivariate neural network (ID-LP), nonlinear multivariate neural network (ID-MLP), hybrid using the F test (ID-hybrid), we performed experiments on 20 data sets from the UCI repository [8]. For each decision tree method, we performed five two-fold cross-validation runs on each data set. The results of the ten runs are then averaged and we report the mean accuracy of each method, as well as comparing their accuracies using the combined 5x2 cv F Test [1]. Table 1 describes the properties of the data sets.

Table 1. Description of the data sets

Set	Classes	Instances	Features	Missing values	Feature Values
Breast	2	699	10	Yes	numeric
Bupa	2	345	7	No	numeric
Car	4	1728	7	No	symbolic
Cylinder	2	541	36	Yes	mixed
Dermatology	6	366	35	Yes	numeric
Ecoli	8	336	8	No	numeric
Flare	3	323	11	No	mixed
Glass	7	214	10	No	numeric
Hepatitis	2	155	20	Yes	numeric
Horse	2	368	27	Yes	mixed
Iris	3	150	5	No	numeric
Ironosphere	2	351	35	No	numeric
Monks	2	432	7	No	numeric
Mushroom	2	8124	23	Yes	symbolic
Ocrdigits	10	3823	64	No	numeric
Pendigits	10	7494	16	No	numeric
Segment	7	2310	19	No	numeric
Vote	2	435	17	Yes	symbolic
Wine	3	178	14	No	numeric
Zoo	7	101	17	No	numeric

4.1 Accuracy Comparison

Accuracy comparison of different data sets is shown in Figure 2 and in Table 2.

Table 2. Accuracy Results

Set	ID3	CART	ID-LP	ID-MLP	ID-Hybrid
Breast	94.1± 1.2	94.9±1.4	96.6± 0.6	96.8± 0.9	96.6± 0.6
Bupa	62.3± 5.3	61.7±3.4	63.5± 2.8	63.2± 4.3	63.4± 2.6
Car	81.0± 1.3	83.8±2.0	89.5± 4.0	96.9± 2.3	94.5± 1.2
Cylinder	68.5± 2.2	59.5±4.1	70.2± 4.5	70.4± 9.6	71.3± 1.7
Dermatology	92.8± 2.4	80.9±4.6	85.7± 7.1	87.8±13.6	94.5± 4.7
Ecoli	78.1± 3.6	74.7±3.8	82.6± 4.1	80.1± 5.1	83.1± 4.2
Flare	85.3± 2.0	81.6±3.6	88.4± 2.4	87.7± 2.6	88.1± 2.4
Glass	60.7± 6.0	53.9±4.2	55.0± 7.8	58.0±13.3	55.1± 9.7
Hepatitis	78.4± 3.7	79.0±4.0	84.1± 2.9	83.7± 2.4	83.7± 3.4
Horse	87.6± 2.0	77.0±3.0	82.1± 3.5	84.7± 2.6	82.7± 2.6
Iris	93.9± 2.8	89.3±4.4	77.6±15.7	92.7± 3.6	92.7± 3.3
Ironosphere	87.6± 3.2	86.8±4.0	87.8± 2.2	87.5± 2.1	87.8± 2.2
Monks	92.3±10.2	91.2±6.9	66.3± 1.9	67.0± 2.2	66.4± 1.9
Mushroom	99.7± 0.1	93.5±1.8	100.0± 0.0	100.0± 0.0	100.0± 0.0
Ocrdigits	78.4± 1.5	81.4±2.1	93.9± 0.9	83.9±10.2	92.8± 2.2
Pendigits	85.7± 1.0	87.1±2.9	91.9± 4.2	91.4± 6.6	90.8± 9.6
Segment	91.1± 1.2	88.1±1.7	79.8±11.6	80.4±12.4	81.8±13.0
Vote	94.9± 1.1	90.3±3.2	94.7± 1.1	95.6± 1.7	94.7± 1.1
Wine	88.7± 3.7	87.3±4.4	87.8±12.6	96.0± 2.1	96.1± 2.1
Zoo	92.1± 4.8	69.9±9.7	79.4± 8.1	85.3±11.9	86.9± 5.4

Generally multivariate method CART is not significantly better than the univariate ID3. In linear multivariate techniques, neural network based training is statistically significantly better than CART in eight data sets whereas CART is better only in two. ID-MLP is better then ID-LP in only three data sets, which has a confidence level between 90% and 95%. This may also be the reason why the hybrid method rarely selects nonlinear nodes over linear nodes.

4.2 Nodesize Comparison

Node size comparisons are shown in Figure 3 and Table 3. Neural network based methods construct significantly smaller trees than ID3 and CART. There is no significant difference in size between the three neural network based methods.

4.3 Learning Time Comparison

Learning time comparisons are shown in Figure 4 and Table 4. CART has significantly larger time complexity than all other methods.

5 Conclusions

In this paper, different univariate, linear and nonlinear decision tree construction methods are compared and a hybrid method combining linear and nonlinear decisions is

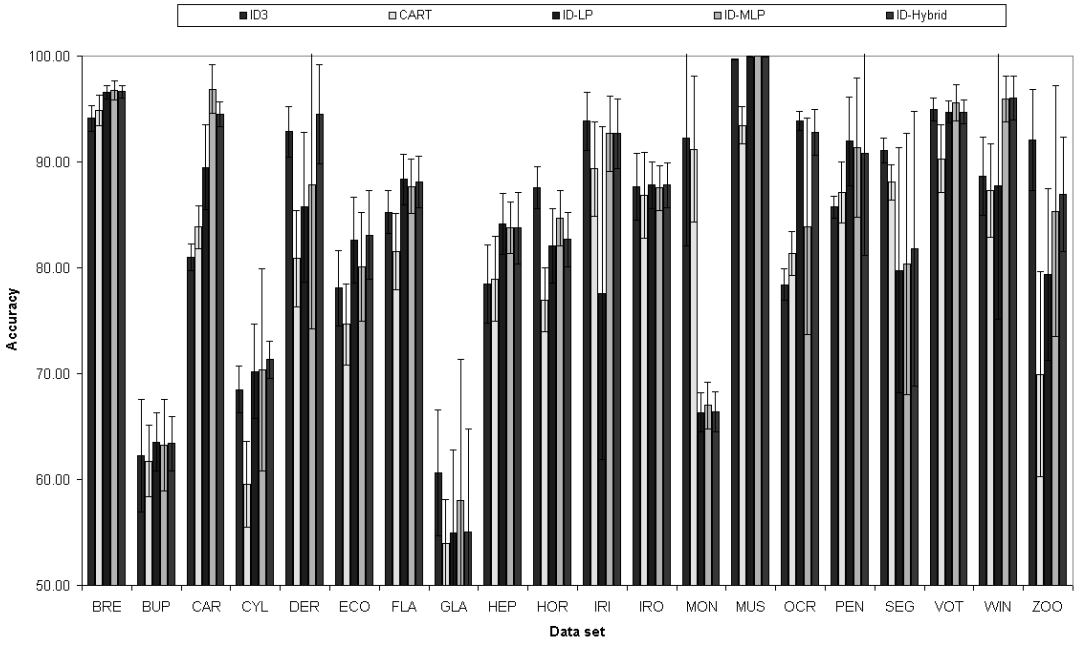

Figure 2. Comparison of accuracy

Table 3. Tree Size Results

Set	ID3	CART	ID-LP	ID-MLP	ID-Hybrid
Breast	17.0± 2.1	11.6± 2.7	3.0±0.0	3.0±0.0	3.0±0.0
Bupa	53.4± 5.5	43.2± 3.8	4.6±1.8	3.8±1.0	4.4±1.9
Car	25.4± 0.7	29.0± 3.4	7.4±0.8	6.6±1.3	7.6±1.0
Cylinder	54.1± 5.9	45.0± 4.9	8.4±1.9	6.4±1.7	8.8±1.8
Dermatology	20.4± 2.7	28.0± 4.7	8.8±1.5	9.6±2.3	11.2±1.1
Ecoli	33.8± 2.7	34.0± 5.0	10.8±2.9	8.8±2.2	10.6±2.3
Flare	37.9± 4.5	33.8± 6.2	3.2±2.2	2.2±1.0	3.0±1.3
Glass	38.2± 5.9	42.4± 4.1	10.2±4.6	7.2±3.7	11.0±5.5
Hepatitis	19.6± 3.8	14.0± 3.4	3.0±0.0	3.0±0.0	3.0±0.0
Horse	55.8± 5.9	28.0± 5.2	5.0±1.6	4.0±1.4	5.6±2.8
Iris	8.4± 1.4	10.2± 2.4	4.0±1.1	5.0±0.0	5.0±0.0
Ironosphere	19.2± 3.1	16.4± 3.8	3.8±1.0	3.8±1.0	4.0±1.1
Monks	25.4±13.5	17.8±10.2	3.0±0.0	3.0±0.0	3.0±0.0
Mushroom	23.0± 0.0	43.0± 6.5	3.0±0.0	3.0±0.0	3.0±0.0
Ocrdigits	74.4± 4.0	70.8± 4.0	34.8±4.9	18.4±3.5	25.4±3.8
Pendigits	81.8± 5.5	77.8±10.1	30.4±6.4	17.6±1.4	23.4±5.8
Segment	41.8± 3.8	45.2± 9.0	16.6±6.7	11.6±2.1	14.4±2.8
Vote	18.2± 3.2	17.2± 5.3	4.2±1.9	3.0±0.0	4.2±1.9
Wine	10.4± 1.4	9.4± 2.3	4.4±1.0	5.0±0.0	5.0±0.0
Zoo	15.0± 1.9	25.2± 4.9	8.8±1.8	10.6±2.8	12.4±1.9

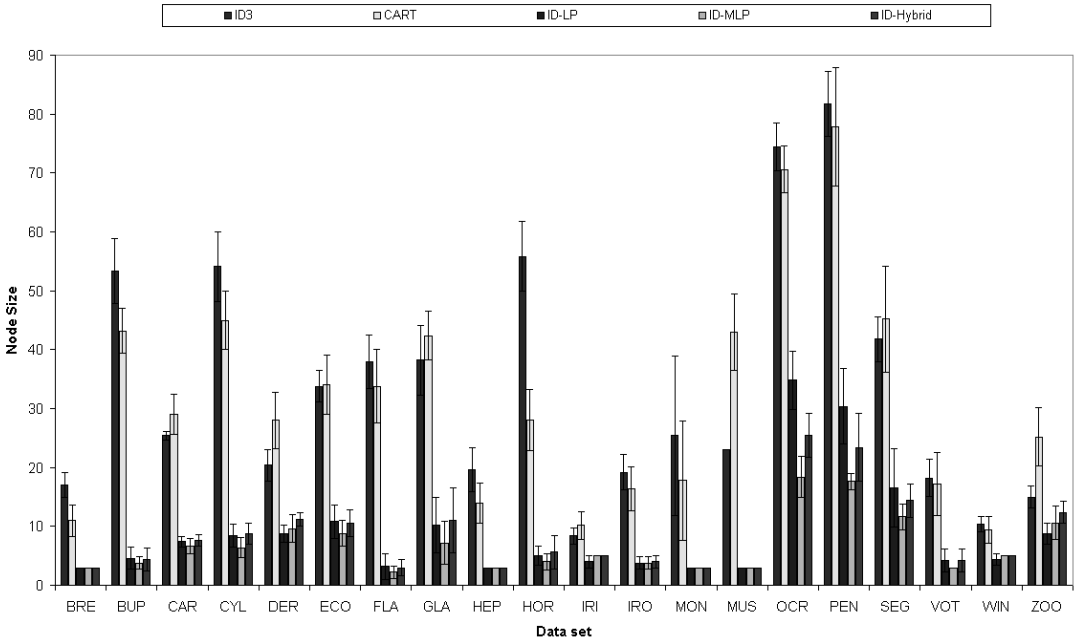

Figure 3. Comparison of tree size

Table 4. Learning Time Results

Set	ID3	CART	ID-LP	ID-MLP	ID-Hybrid
Breast	2± 0	107± 17	5± 0	7± 0	30± 1
Bupa	3± 1	252± 23	3± 1	3± 1	15± 3
Car	5± 0	1178± 148	152± 16	216± 18	911± 151
Cylinder	10± 2	4589± 343	19± 2	102± 15	366± 45
Dermatology	3± 0	858± 170	42± 9	122± 19	399± 25
Ecoli	3± 0	221± 25	57± 15	50± 13	219± 38
Flare	2± 0	1032± 203	9± 4	18± 7	67± 26
Glass	3± 0	320± 25	33± 9	27± 8	137± 28
Hepatitis	1± 0	209± 47	1± 0	3± 0	10± 0
Horse	4± 0	3481±1101	14± 2	97± 20	327± 89
Iris	0± 0	31± 11	3± 0	3± 0	14± 1
Ironosphere	39± 7	544± 94	4± 1	14± 2	53± 9
Monks	2± 1	126± 61	3± 0	3± 0	16± 0
Mushroom	113±33	33613±2942	628± 204	1858± 270	5529± 414
Ocrdigits	207± 9	9148± 713	8035± 757	10993±1402	14791±3204
Pendigits	476±22	3311± 350	18340±3319	8473±1742	9942±1566
Segment	345±10	1212± 170	937± 103	927± 126	4756± 453
Vote	1± 0	805± 167	6± 1	13± 2	53± 11
Wine	1± 0	84± 26	4± 1	4± 1	19± 2
Zoo	1± 0	453± 61	10± 2	18± 4	66± 10

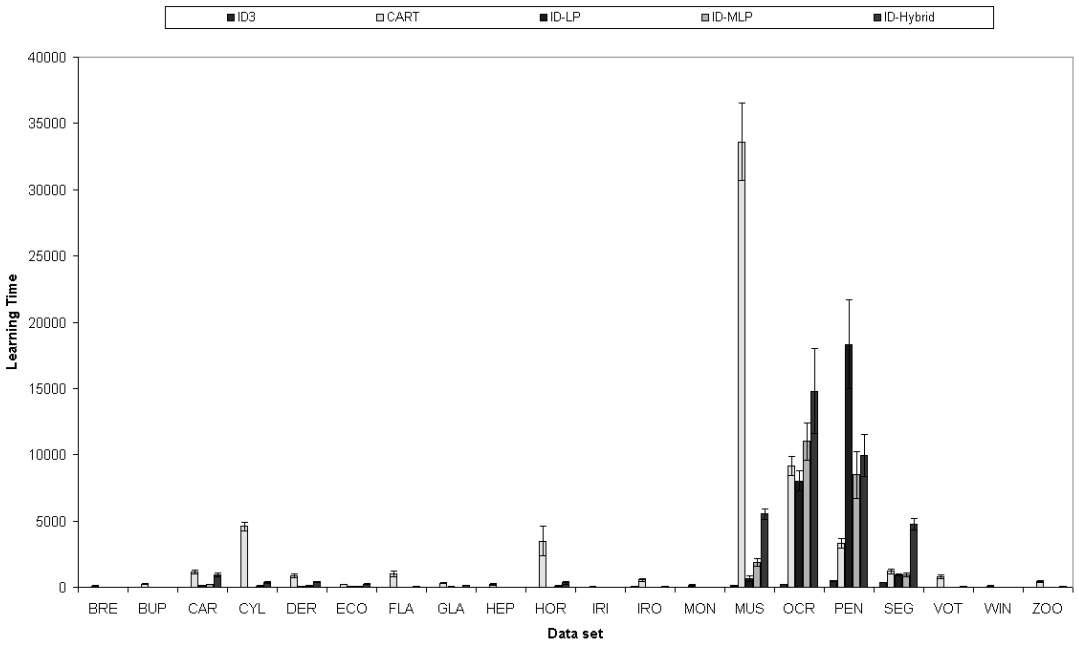

Figure 4. Comparison of learning time

proposed.

If the data set is small or if one class has a significantly larger number of instances compared to other classes, then a univariate technique does not overfit and can be sufficient and univariate ID3 has better performance than multivariate linear methods. Also ID3 learns fast, learns simple and interpretable rules.

If the variables are highly correlated, then a univariate method is not sufficient and we may resort to multivariate methods. We have shown that the ID-LP has better performance than CART in terms of accuracy, node size and very significantly in learning time. The neural network based methods generate much smaller trees than ID3 and CART. This shows that to generate a linear multivariate tree, using ID-LP is preferable over CART.

If nonlinear methods will be used, we suggest not using nonlinear models in all decision nodes but ID-hybrid as the decision at some nodes may be handled in a simpler fashion, e.g., linearly (or even univariate).

References

1. E. Alpaydin Combined 5x2 cv F Test for Comparing Supervised Classification Learning Algorithms *Neural Computation* Vol. 11, pp. 1975-1982. (1999)

2. C. Bishop *Neural Networks for Pattern Recognition* Oxford Univ Press. (1996)

3. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone *Classification and Regression Trees* Wadsworth (1984)

4. L. A. Breslow and D. W. Aha Simplification Decision Trees: A Survey *NCARAI Technical Report No. AIC-96-014* (1997)

5. C. E. Brodley and P. E. Utgoff Multivariate Decision Trees *Machine Learning* Vol. 19, pp. 45-77. (1995)

6. T. Dietterich Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms *Neural Computation* Vol. 10, pp. 1895-1923. (1998)

7. H. Guo and S. B. Gelfond Classification Trees with Neural Network Feature Extraction *IEEE Transactions on Neural Networks* Vol. 3, pp. 923-933. (1992)

8. C. J. Merz and P. M. Murphy UCI Repository of Machine Learning Databases http://www.ics.uci.edu/ mlearn/MLRepository.html. (1998)

9. J. R. Quinlan Induction of Decision Trees *Machine Learning* Vol. 1, pp. 81-106. (1986)

10. P. E. Utgoff and C. E. Brodley Linear Machine decision trees (*COINS Technical Report 91-100*), Amberst, MA: University of Massachusetts, Department of Computer and Information Science. (1991)