Linear Discriminant Trees

Olcay Taner Yıldız YILDIZOL@CMPE.BOUN.EDU.TR
Ethem Alpaydın ALPAYDIN@BOUN.EDU.TR
Department of Computer Engineering, Boğaziçi University, 80815 İstanbul Turkey

Abstract

Univariate decision trees at each decision node consider the value of only one feature leading to axis-aligned splits. In a multivariate decision tree, each decision node divides the input space into two with an arbitrary hyperplane leading to oblique splits. In this paper, we propose linear discriminant trees where each node implements Fisher's linear discriminant analysis (LDA) to make a binary split. On twenty data sets from the UCI repository, we compare the linear discriminant trees with the univariate decision tree method ID3, multivariate decision tree methods CART, neural trees and linear machine decision trees (LMDT). Our results indicate that linear discriminant trees learn faster than other multivariate methods with binary splits, generalize well and construct small trees.

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 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

$$x_i > c$$

making a binary split where x_i is input feature i and c is a suitably chosen threshold to minimize disorder. If x_i is discrete valued with m values, then the node makes an m-ary split. *ID3 algorithm* (Quinlan, 1986) is used to construct univariate decision trees. Consider the two-dimensional instance space shown in Figure 1. To separate the two classes, the univariate decision tree uses a series of orthogonal splits, as a univariate test using a single feature can only do axis-aligned splits. This results in large trees and poor generalization.

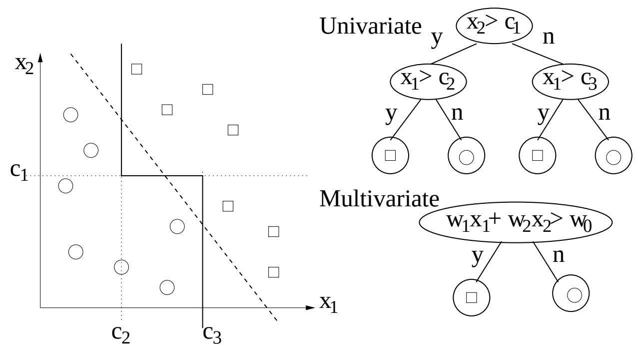

Figure 1. Comparison of univariate and multivariate (dashed) splits on the plane

In a *multivariate decision tree*, each decision node is based on more than one feature. In Figure 1, we see that with one line, we can separate the examples of two classes, reducing a tree with seven nodes to a tree with three nodes. 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

$$\sum_{i=1}^{d} w_i x_i > w_0 \qquad (1)$$

where w_i are the weights and w_0 is a threshold.

2. Multivariate Decision Tree Algorithms

2.1 Classification and Regression Trees (CART)

The CART algorithm (Breiman et al., 1984) for finding the coefficients of the available features is a stepwise procedure, where in each step one cycles through the features $x_1, x_2, \ldots, x_n$ doing a search for an improved linear combination split. At the beginning of a cycle, let the current linear combination split be $v = \sum_i w_i x_i \leq c$. For fixed γ, CART searches for the best split of the form

$$v - \delta(x_1 + \gamma) \leq c \tag{2}$$

such that

$$\begin{aligned} \delta &\geq \frac{(v-c)}{(x_1+\gamma)} \text{ where } x_1 + \gamma \geq 0 \\ \delta &\leq \frac{(v-c)}{(x_1+\gamma)} \text{ where } x_1 + \gamma \leq 0 \end{aligned} \tag{3}$$

The search for δ is carried out for $\gamma \in \{-0.25, 0.0, 0.25\}$. The resulting three splits are compared, using the chosen partition-merit criterion, and the δ and γ corresponding to the best are used to update v, producing

$$\begin{aligned} v' &= w'_1 x_1 + w'_2 x_2 + \ldots + w'_d x_d \\ & \quad \text{where } w'_1 = w_1 - \delta \text{ and } c' = c + \delta\gamma \end{aligned} \tag{4}$$

This search is repeated for other features $x_2, \ldots, x_d$ to obtain an updated split $v_1 < c_1$. The final step of the cycle is to find the best c_1, and the system searches explicitly for the split that minimizes the impurity of the resulting partition. The cycles end when the reduction of impurity is below a pre-specified constant.

2.2 Neural Trees

The decision at a multivariate node making a binary split can also be seen as the output of any decision maker with binary output. Guo and Gelfand (1992) propose to use a multilayer perceptron with a hidden layer trained to make a good split. We use a single layer perceptron which has the same form as Eq. (1) whereas the multilayer perceptron implements a *non-linear* boundary in the input space. The w_i and w_0 of the perceptron can be determined by training a perceptron with 0/1 output making a binary split.

Finding the best binary split with neural network algorithms is done as a 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. 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 solve the outer optimization problem which is selecting an appropriate partition for the K classes we have into two (Section 4.2).

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. Let $\mathcal{X} = \{x^t, d^t\}$ 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 the left class group and 0 for the right class group. y^t is the real output found by the formula

$$y^t = \text{sigmoid}\left(\sum_{i=1}^{d} w_i x_i^t + w_0\right) \tag{5}$$

A stochastic gradient algorithm for minimizing the mean square error criterion on $\mathcal{X}$ is used

$$E(w_1, \ldots, w_d, w_0|\mathcal{X}) = \sum_t (d^t - y^t)^2 \tag{6}$$

2.3 Oblique Classifier (OC1)

OC1 (Oblique Classifier) (Murthy, Kasif & Salzberg, 1994) is similar to CART in that it iterates over the weights to minimize an impurity measure. A small random vector is added to weights once there is convergence through backfitting, to get out of local minima. 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.

2.4 Linear Machine Decision Trees (LMDT)

In TLM (Thermal Linear Machine) (Brodley & Utgoff, 1995), with K classes, 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 linear split. There is an iterative algorithm that adjusts the parameters of classes to minimize the number of misclassifications, rather than an impurity measure as entropy or Gini. Note that this is drastically different from CART, Neural Trees, and the setting we give above that restricts nodes to binary splits even when $K > 2$.

3. 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 x_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 with possible values $a_1, \ldots, a_j$, we convert it into a feature set as $i_1, \ldots, i_j$ 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 but increases dimensionality (Utgoff & Brodley, 1991).

In cases where a feature has a missing value, if it is an ordered feature, we fill it with the sample mean 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 (Breiman et al., 1984; Breslow & Aha, 1994; Brodley & Utgoff, 1995).

At the leaves of the tree, the leaf node has the label having the most instances. In our implementation we use prepruning to avoid overfitting. At a node, we stop and do not split further if the number of instances is less than a proportion, 5%, of the training set. There are two reasons why we favor prepruning over postpruning. First, the multivariate models have many parameters (especially the nonlinear nodes) and thus have large variance and are prune to overfitting on small samples. Second, the multivariate tree induction methods are time consuming and we do not want to run them frequently, especially if they have a high chance of being useless. The number of nodes increase exponentially with depth and pruning the tree early in training saves from learning time.

In the multivariate decision tree construction method CART, using all of the features may be costly and may lead to overfitting on a small training set. In order to get rid of these unnecessary features, we use feature selection (Breiman et al., 1984).

4. LDA Trees

Linear discriminant trees use a statistical approach to determine the set of coefficients in a linear decision node. Finding the best split with Fisher's Linear Discriminant Analysis (LDA) (Duda & Hart, 1973) is done as a nested optimization problem. In the inner optimization problem, Fisher's linear discriminant finds 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.

Linear discriminant analysis aims 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. If there are only two classes present in that node one class belongs to C_L and the other to C_R. Otherwise we must select an appropriate partition for the K classes we have (Section 4.2).

4.1 Linear Discriminant Analysis

The inner optimization problem is solved by using LDA at each node of the decision tree. The crite-

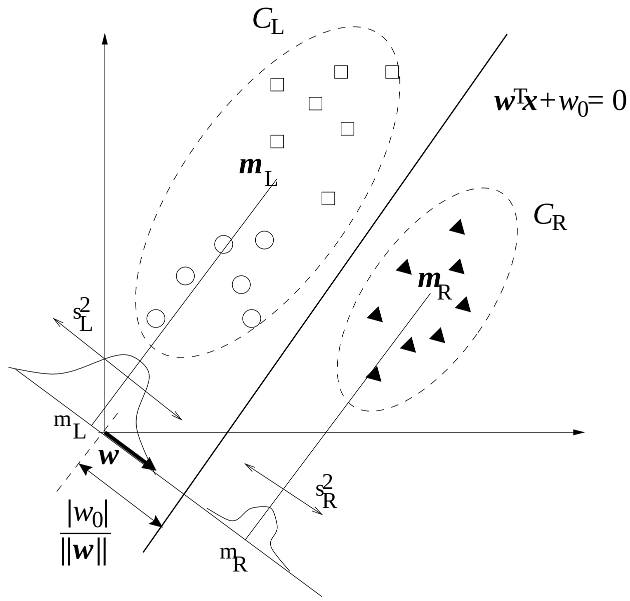

Figure 2. Class separation by LDA.

rion proposed by Fisher is the ratio of between-class to within-class variances, which we want to maximize. We must find a direction $\boldsymbol{w}$ to maximize (Fig. 2)

$$J_F = \frac{|\boldsymbol{w}^T(\boldsymbol{m}_L - \boldsymbol{m}_R)|^2}{|\boldsymbol{w}^T \mathbf{S}_w \boldsymbol{w}|} \tag{7}$$

where $\boldsymbol{m}_L$ and $\boldsymbol{m}_R$ are the two left and right group means

$$\boldsymbol{m}_L = \frac{1}{n_L} \sum_{\boldsymbol{x} \in C_L} \boldsymbol{x} \tag{8}$$

and $\mathbf{S}_w$ is the within-class covariance matrix, which is bias corrected as

$$\frac{1}{n_L + n_R}(n_L \boldsymbol{\Sigma}_L + n_R \boldsymbol{\Sigma}_R) \tag{9}$$

where $\boldsymbol{\Sigma}_L$ and $\boldsymbol{\Sigma}_R$ are the covariance matrices of class groups C_L and C_R respectively

$$\boldsymbol{\Sigma}_L = \sum_{\boldsymbol{x} \in C_L} (\boldsymbol{x} - \boldsymbol{m}_L)(\boldsymbol{x} - \boldsymbol{m}_L)^T \tag{10}$$

There are n_L samples in the left class group and n_R samples in the right class group. The solution for $\boldsymbol{w}$ that maximizes J_F can be obtained by differentiating it with respect to $\boldsymbol{w}$ and equating it to zero. So we define $\boldsymbol{w}$, namely the set of coefficients of the linear combination as

$$\boldsymbol{w} = \mathbf{S}_w^{-1}(\boldsymbol{m}_L - \boldsymbol{m}_R) \tag{11}$$

But the coefficients of the features is only for the direction, we must also specify a threshold w_0, which is calculated as

$$w_0 = -\frac{1}{2}(\boldsymbol{m}_L + \boldsymbol{m}_R)^T \mathbf{S}_w^{-1}(\boldsymbol{m}_L - \boldsymbol{m}_R) - \log \frac{n_L}{n_R} \quad (12)$$

Note however that though the direction given in the Eq. (11) has been derived without any assumptions of normality, normal assumptions are used in Eq. (12) to set the threshold for discrimination.

So in our case, we first divide $K > 2$ classes into two groups as C_L and C_R by an appropriate class selection procedure (as defined in Section 4.2). The inner optimization procedure is carried out by LDA to find the linear split. There is no iterative training (as with CART, neural trees, OC1, or LMDT) but only simple calculations with matrices. So we expect less training time in LDA trees, than the other multivariate tree construction methods.

4.2 Class Separation by Exchange Method

One detail we seem to have skipped above is the following: The linear discriminant 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 $2^{K-1} - 1$ distinct partitions available. So because the number of available partitions grows exponentially, we can not test for all possible partitions.

We use exchange method in class separation as proposed in Guo & Gelfand (1992), to handle the same problem in neural trees with single output, learning binary splits. 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 in step 1, instead of starting randomly. The two classes C_i and C_j with the maximum intermean distance are found and placed into C_L and C_R. For each of the classes $k \in \{C_1, \ldots, C_m\}$, we find the one with the minimum intermean 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.

4.3 Feature Extraction in LDA trees

In implementing the idea, we see that if there is a linear dependency between two or more features then $\mathbf{S}_w$ becomes singular. In this case the determinant is zero and we can not find $\mathbf{S}_w^{-1}$. A possible answer to the problem is *principal component analysis* (PCA) (Rencher, 1995), where we find the eigenvectors and eigenvalues of the matrix $\mathbf{S}_w$, sort the eigenvalues and get rid of the eigenvectors with small eigenvalues. Let us say that the eigenvalues of the matrix $\mathbf{S}_w$ are $\lambda_1, \ldots, \lambda_d$, sorted in decreasing order. We find the k eigenvectors that explain more than ϵ of the variance

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_k + \cdots + \lambda_d} > \epsilon \quad (13)$$

The instances are then projected from the original d-dimensional space into the new k-dimensional space defined by the leading k eigenvectors. After this we do LDA in this new space and find a new $\mathbf{S}_w$, means, and calculate the split. Because the eigenvectors are orthogonal, now the new $\mathbf{S}_w$ has an inverse. Note that because PCA is also a linear mapping, the eigenvectors can be incorporated into the w_i so that there is no increase in memory need.

Using PCA is a dimensionality reducing feature extraction process and may be used even if there is no singularity. Using PCA differs from subset selection used in tree construction methods like CART in that PCA finds new k dimensions that are linear combinations of the original d dimensions whereas subset selection keeps k of the original d dimensions, discarding the $d - k$.

5. Evaluation of Decision Tree Construction Methods

To evaluate the five decision tree construction methods; univariate ID3, linear multivariate CART, linear perceptron neural trees ID-LP, linear machine decision trees LMDT, and linear discriminant trees ID-LDA,

Table 1. Description of the data sets

Set	Classes	Instances	Features
Breast	2	699	10
Bupa	2	345	7
Car	4	1728	7
Cylinder	2	541	36
Dermatology	6	366	35
Ecoli	8	336	8
Flare	3	323	11
Glass	7	214	10
Hepatitis	2	155	20
Horse	2	368	27
Iris	3	150	5
Ironosphere	2	351	35
Monks	2	432	7
Mushroom	2	8124	23
OcrDigits	10	3823	64
PenDigits	10	7494	16
Segment	7	2310	19
Vote	2	435	17
Wine	3	178	14
Zoo	7	101	17

we performed experiments on 20 data sets from the UCI repository (Merz & Murphy, 1998). 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 and standard deviations of each method, as well as comparing their accuracies using the combined 5×2 cv F test (Alpaydın, 1999). The three criteria are accuracy on the test set, tree size measured in terms of the number of nodes in the tree and learning time in seconds on a Pentium III-450. Table 1 describes the properties of the data sets. *Breast*, *Cylinder*, *Hepatitis*, *Horse*, *Vote* data sets have missing values.

5.1 Results with LDA trees

The results with LDA trees are shown in Table 2 for accuracy, in Table 3 for tree sizes, and in Table 4 for learning time. On *Breast, Bupa, Iris, Wine* data sets, PCA is never needed. Results show that increasing ϵ, proportion of variance explained, in PCA, from 0.90 to 0.99 improves accuracy but does not significantly increase tree size or learning time. Thus we propose to use a large ϵ as possible, just enough to get rid of the eigenvectors with zero eigenvalues, but not any eigenvector contributing to variance.

5.2 Comparison with Other Decision Tree Methods

In this section the five decision tree construction methods are compared. For each comparison type two

Table 2. Accuracy results with LDA trees. Bold indicates %95 statistically significant difference.

Set	No PCA	PCA with $\epsilon = 0.90$	PCA with $\epsilon = 0.99$
BRE	95.85±0.72		
BUP	67.42±2.97		
CAR		70.02±1.75	**92.09±1.07**
CYL		67.39±2.39	69.80±3.01
DER		94.75±1.91	96.17±1.59
ECO		83.69±3.58	83.75±2.53
FLA		88.05±2.39	88.17±2.83
GLA		55.51±4.43	57.29±4.16
HEP		83.61±2.12	82.06±5.60
HOR		72.39±2.62	**81.09±1.64**
IRI	97.20±1.47		
IRO		86.38±2.68	91.11±2.22
MON		74.31±2.26	74.86±1.54
MUS		94.15±0.83	**98.25±0.57**
OCR		89.19±0.92	**94.59±0.49**
PEN		91.99±0.94	**95.52±0.44**
SEG		82.19±2.35	**90.31±1.20**
VOT		90.85±2.35	94.85±2.17
WIN	96.07±2.66		
ZOO		82.56±5.62	81.41±7.25

tables are given. In the first table the raw results are shown for ID3, CART, ID-LP (neural trees) and LMDT. ID-LDA raw results (with $\epsilon = 0.99$) are given in Tables 2, 3, and 4. The second table indicates pairwise test of statistically significant difference (%95 or more) with the 5×2cv F test among all five methods.

Comparing Table 2 and Table 5, we see that the accuracies of ID-LP, ID-LDA, LMDT are nearly equal and they seem to outperform CART and ID3. There is not a significant difference between ID3 and CART in case of accuracy. So there is an order of

$$\text{ID-LP} = \text{ID-LDA} = \text{LMDT} > \text{CART} \ ?= \text{ID3}$$

In Tables 3 and 6, we see that ID-LP constructs the smallest trees among the four linear decision tree construction methods. ID-LDA performs better than LMDT and CART, which performs better than ID3. This result can be expected due to the univariate/multivariate difference. So there is an order of

$$\text{ID-LP} < \text{ID-LDA} < \text{LMDT} < \text{CART} < \text{ID3}$$

Note however that the number of nodes is not a direct measure of complexity. A univariate numeric node has two parameters: the index of the used variable and the threshold. In multivariate linear methods like CART, ID-LDA and neural trees, with d inputs, there are $d+1$ parameters. In LMDT, with K classes, each decision node has $K \times (d+1)$ parameters; for fairness, LMDT nodes count as K regular nodes in our results.

Table 3. Tree size (number of nodes) with LDA trees.

SET	NO PCA	PCA WITH $\epsilon = 0.90$	PCA WITH $\epsilon = 0.99$
BRE	7.20±0.63		
BUP	8.20±1.93		
CAR		**1.00±0.00**	12.00± 2.54
CYL		10.80±3.33	16.40± 8.95
DER		17.00±2.11	12.80± 1.48
ECO		17.60±4.81	20.00± 2.71
FLA		5.20±3.71	5.60± 3.13
GLA		25.60±3.78	26.20± 4.44
HEP		4.60±3.10	8.60± 3.10
HOR		10.20±3.29	16.80± 4.05
IRI	5.40±0.84		
IRO		5.40±1.84	11.60± 2.67
MON		7.20±2.39	7.80± 4.13
MUS		17.40±3.63	19.20± 4.85
OCR		87.40±8.37	**59.40± 2.07**
PEN		80.80±3.71	89.00± 6.25
SEG		40.40±5.66	39.80±11.08
VOT		9.20±3.05	9.80± 2.53
WIN	5.40±0.84		
ZOO		12.60±2.07	11.80± 1.93

Table 4. Learning time (seconds) with LDA trees.

SET	NO PCA	PCA WITH $\epsilon = 0.90$	PCA WITH $\epsilon = 0.99$
BRE	2±0		
BUP	1±0		
CAR		**6± 2**	28± 6
CYL		34± 14	74± 57
DER		18± 2	16± 1
ECO		6± 2	6± 1
FLA		3± 2	4± 3
GLA		7± 1	7± 1
HEP		1± 1	2± 1
HOR		38± 16	94± 40
IRI	0± 0		
IRO		3± 1	11± 4
MON		1± 0	1± 1
MUS		1846±776	2533±1533
OCR		3189±270	**2288± 108**
PEN		**996± 86**	1211± 91
SEG		154± 15	148± 40
VOT		8± 4	9± 3
WIN	1± 0		
ZOO		2± 0	2± 0

Table 7 (and Table 4) compares the learning time of the methods. CART takes the most time in learning, whereas LMDT and ID3 have the smallest learning time. Note however that LMDT at each node learns a K-way split, separating one class from all other classes, whereas ID-LDA and ID-LP, at each node, learns a binary split, thus requiring $\log_2 K$ training steps. Among the two, ID-LDA outperforms ID-LP in case of learning time because there is no iterative learning. So there is an order of:

$$\text{LMDT} < \text{ID3} < \text{ID-LDA} < \text{ID-LP} < \text{CART}.$$

6. Conclusions

In this paper, different univariate and linear decision tree construction methods are compared and a new multivariate decision tree method based on Fisher's Linear Discriminant Analysis is proposed.

In Figure 3, we plot all five methods on all data sets in terms of accuracy vs tree size. We see that ID-LP generates the smallest and most accurate trees, its nearest competitor is ID-LDA, and the others are ordered as LMDT and ID3 and then CART. Note that a univariate ID3 node is much simpler than a multivariate node of CART, ID-LP, or ID-LDA. LMDT nodes are counted as K nodes where K is the number of classes.

Looking at the accuracy vs learning time plot, we see that ID3 and LMDT are the fastest learners, then we have ID-LDA, ID-LP and CART. LMDT learns faster then any other multivariate algorithm, because it updates each class's separate linear combination at each iteration, rather than grouping them into two as do ID3, CART, ID-LP and ID-LDA.

In terms of tree size vs learning time, we see clearly that ID3 and LMDT learn fast but generate large trees and ID-LP and ID-LDA learn slower but generate small trees; CART learns slowly and generates large trees.

If the data set is small and has few classes, then a univariate technique does not overfit and can be sufficient and the univariate ID3 has better performance than multivariate linear methods. ID3 learns fast, learns simple and interpretable rules.

If the variables are highly correlated, then the univariate method is not sufficient and we may resort to multivariate methods. We have shown that ID-LDA has better performance than CART in terms of accuracy, node size and very significantly in learning time. It has also smaller learning time than ID-LP and the same accuracy. ID-LDA generates smaller trees than ID3, CART and LMDT. This shows that to generate a linear multivariate tree, using ID-LDA is preferable over CART, may be preferable over ID-LP if learning time is critical, and may be preferable over LMDT if tree size is critical.

Table 5. Comparison of accuracies. The first table shows the raw results. The second table gives pairwise comparison of accuracies; values are the number of data sets out of twenty on which the method on the row is significantly better than the method on the column.

Set	ID3	CART	ID-LP	LMDT
BRE	94.1± 1.2	94.9±1.4	96.6± 0.6	96.2±0.7
BUP	62.3± 5.3	61.7±3.4	63.5± 2.8	60.1±6.9
CAR	81.0± 1.3	83.8±2.0	89.5± 4.0	86.4±1.2
CYL	68.7± 2.0	59.5±4.0	70.2± 4.5	68.6±2.9
DER	92.8± 2.4	80.9±4.6	85.7± 7.1	96.4±1.3
ECO	78.1± 3.6	74.6±3.8	82.6± 4.1	81.2±3.5
FLA	85.3± 2.0	81.5±3.6	88.4± 2.4	86.7±3.1
GLA	60.7± 6.2	53.9±4.2	55.0± 7.8	59.3±4.7
HEP	78.4± 3.7	79.0±4.0	84.1± 2.9	81.3±3.7
HOR	79.4± 3.9	77.0±3.0	82.1± 3.5	81.1±3.0
IRI	93.9± 2.8	89.3±4.4	77.6±15.7	90.4±3.5
IRO	87.6± 3.2	86.8±4.0	87.8± 2.2	85.6±3.3
MON	92.2±10.2	91.2±6.9	66.3± 1.9	71.8±2.8
MUS	99.7± 0.1	93.4±1.8	99.9± 0.0	99.9±0.1
OCR	78.4± 1.5	81.4±2.1	93.9± 0.9	93.8±0.5
PEN	85.7± 1.0	87.1±2.9	91.9± 4.2	95.5±0.8
SEG	91.1± 1.2	88.1±1.7	79.8±11.6	90.8±1.5
VOT	94.9± 1.1	90.3±3.2	94.7± 1.0	94.2±1.6
WIN	88.7± 3.7	87.3±4.4	87.8±12.6	95.3±2.0
ZOO	91.3± 6.3	69.9±9.7	79.4± 8.1	89.3±5.2

Method	ID3	CART	ID-LP	LMDT	ID-LDA
ID3		4	4	1	3
CART	0		2	1	1
ID-LP	4	6		0	1
LMDT	4	6	1		1
ID-LDA	5	8	2	2	

Table 6. Comparison of tree sizes.

Set	ID3	CART	ID-LP	LMDT
BRE	17.0± 2.1	11.0± 2.7	3.0±0.0	6.6± 2.6
BUP	53.4± 5.5	43.2± 3.8	4.6±1.8	20.2± 9.9
CAR	25.4± 0.7	29.0± 3.4	7.4±0.8	35.4± 9.7
CYL	54.6± 6.4	45.0± 4.9	8.4±1.9	23.8± 2.5
DER	20.0± 3.0	28.0± 5.0	9.0±1.0	7.0± 0.0
ECO	34.0± 3.0	34.0± 5.0	11.0±3.0	31.0± 5.0
FLA	38.0± 4.7	33.8± 6.2	3.2±2.2	15.0± 9.3
GLA	38.4± 5.8	42.4± 4.1	10.2±4.6	38.3±10.9
HEP	19.6± 3.8	14.0± 3.4	3.0±0.0	10.6± 3.0
HOR	55.8± 5.9	28.0± 5.2	5.0±1.6	7.8± 1.9
IRI	8.4± 1.3	10.2± 2.3	4.0±1.1	9.4± 3.1
IRO	19.2± 3.0	16.4± 3.8	3.8±1.0	9.8± 3.0
MON	25.2±13.2	17.8±10.2	3.0±0.0	19.2± 6.0
MUS	23.0± 0.0	43.0± 6.5	3.0±0.0	3.0± 0.0
OCR	74.4± 4.0	70.6± 4.0	34.8±4.9	68.0± 4.8
PEN	81.8± 5.5	77.8±10.1	30.4±6.4	125.0±14.0
SEG	41.8± 3.8	45.2± 9.0	16.6±6.7	48.2±11.8
VOT	18.2± 3.2	17.2± 5.3	4.2±1.9	7.8± 1.4
WIN	10.4± 1.3	9.4± 2.3	4.4±1.0	5.0± 1.4
ZOO	15.0± 1.9	25.2± 4.9	8.8±1.8	8.1± 0.7

Method	ID3	CART	ID-LP	LMDT	ID-LDA
ID3		2	0	1	0
CART	4		0	1	0
ID-LP	20	19		11	9
LMDT	14	9	0		4
ID-LDA	13	10	0	5	

Acknowledgements

We would like to thank the area chair and the referees for their constructive comments. This work is supported by grants 98HA0101 and 00A101D from Boğaziçi University Research Funds.

References

Alpaydın, E. (1999). Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, *11*, 1975–1982.

Breiman, L., Friedman, J. H., Olshen, R.A., & Stone, C. J. (1984). *Classification and regression trees*. Belmonth, CA: Wadsworth International Group.

Breslow, L. A., & Aha, D. W. (1997). *Simplfying decision trees: A survey* (Technical Report AIC-96-014), Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington DC.

Brodley, C. E, & Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning*, *19*, 45–77.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley-Interscience Publication.

Guo, H., & Gelfand, S. B. (1992). Classification trees with neural network feature extraction. *IEEE Transactions on Neural Networks*, *3*, 923–933.

Merz, C. J., & Murphy, P. M. (1998). *UCI repository of machine learning databases*. http://www.ics.uci.edu/~mlearn/MLRepository.html

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

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Rencher, A. C. (1995). *Methods of multivariate anal-*

Table 7. Comparison of learning times.

SET	ID3	CART	ID-LP	LMDT
BRE	1± 0	107± 17	5± 0	0± 0
BUP	3± 1	252± 23	3± 1	0± 0
CAR	1± 0	1178± 148	152± 16	9± 3
CYL	12± 2	4589± 343	19± 2	8± 1
DER	6± 1	858± 170	42± 9	1± 0
ECO	6± 1	221± 25	57± 15	1± 0
FLA	5± 1	1032± 203	9± 4	1± 1
GLA	7± 2	320± 25	33± 9	1± 0
HEP	4± 1	209± 47	1± 0	0± 0
HOR	9± 3	3481±1101	14± 2	4± 1
IRI	0± 0	31± 11	3± 0	0± 0
IRO	28± 4	544± 94	4± 1	1± 0
MON	9± 3	126± 61	3± 0	0± 0
MUS	18± 3	33613±2942	628± 204	22± 3
OCR	207± 9	9148± 713	8035± 757	231±19
PEN	476±22	3311± 350	18340±3319	150±20
SEG	345±10	1212± 170	937± 103	17± 5
VOT	0± 0	805± 167	6± 1	1± 0
WIN	2± 0	84± 26	4± 1	0± 0
ZOO	1± 0	453± 61	10± 2	0± 0

METHOD	ID3	CART	ID-LP	LMDT	ID-LDA
ID3		20	17	4	10
CART	0		1	0	0
ID-LP	0	17		0	1
LMDT	15	20	19		15
ID-LDA	3	20	13	0	

ysis. New York: Wiley.

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

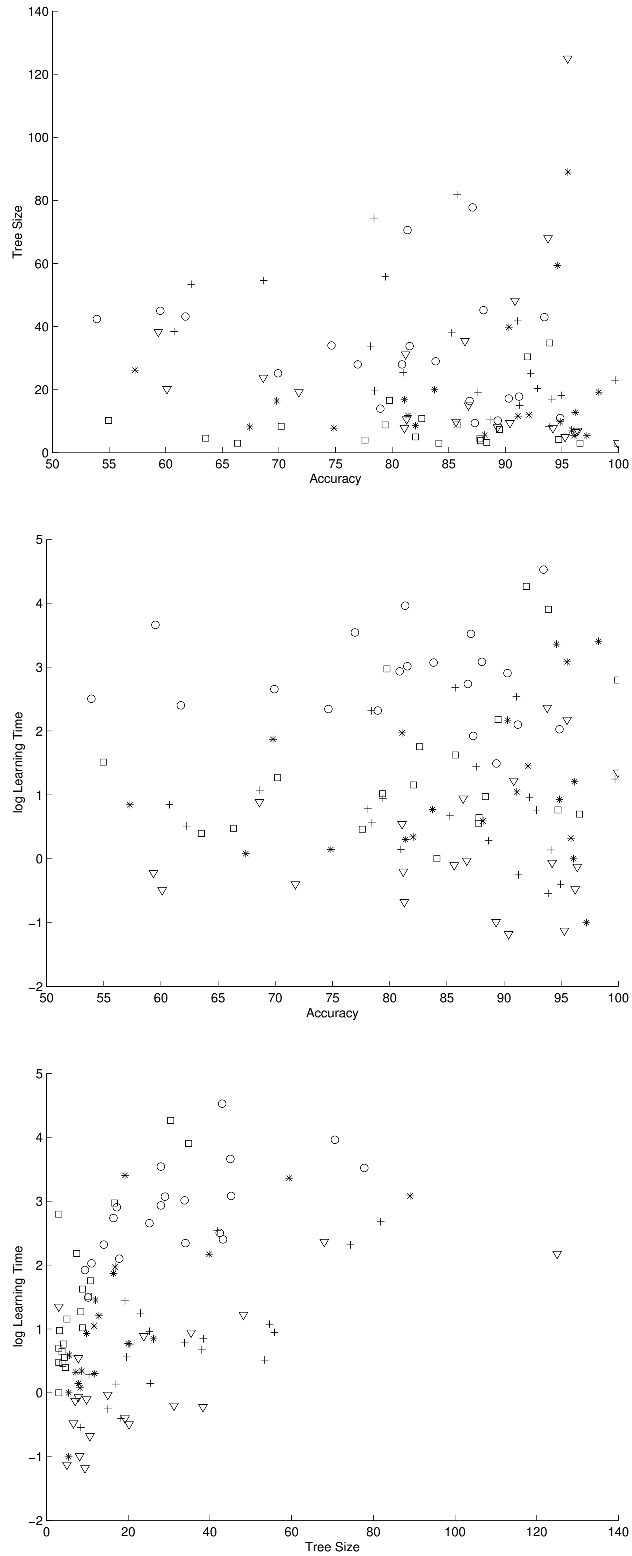

Figure 3. Comparison of decision tree methods. '+' : ID3, '∘' : CART, '□' : ID-LP, '*' : ID-LDA, '▽' : LMDT