Data Mining With Decision and Regression Trees

Johannes Gehrke
Department of Computer Science
Cornell University
johannes@cs.cornell.edu
www.cs.cornell.edu/home/johannes

Wei-Yin Loh
Department of Statistics
University of Wisconsin–Madison
loh@stat.wisc.edu
www.stat.wisc.edu/˜loh

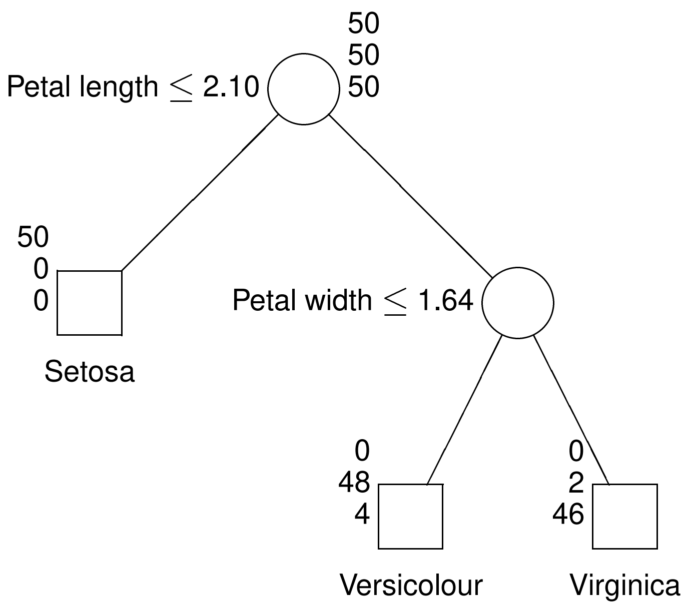
Topics

- PART I
  - Introduction. Examples of classification and regression trees. Comparison with other classification methods.
- PART II
  - Classification and regression tree split selection methods: CART, QUEST, CHAID, C4.5, GUIDE.
  - Tree pruning and missing value handling.
  - Scalable data access methods.
- PART III
  - Predictive accuracy, complexity, and training time of tree construction algorithms.
  - Selection bias in greedy search split selection methods.
  - Conclusion. Observations and recommendations.

Classification example: Fisher’s iris data

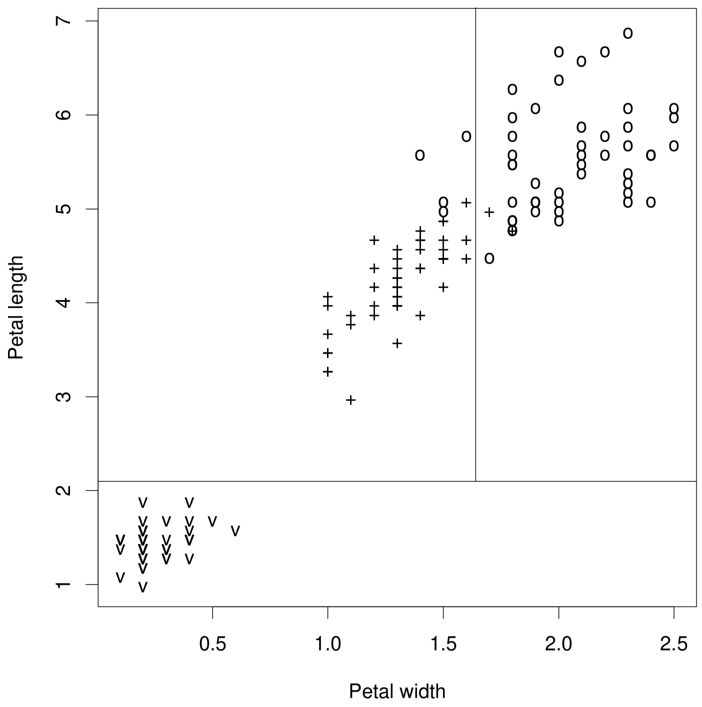
- 3 classes, 50 observations per class
- 4 predictor variables: petal length and width, sepal length and width

QUEST tree for iris data



150-fold CV pruning and 1-SE rule

QUEST method



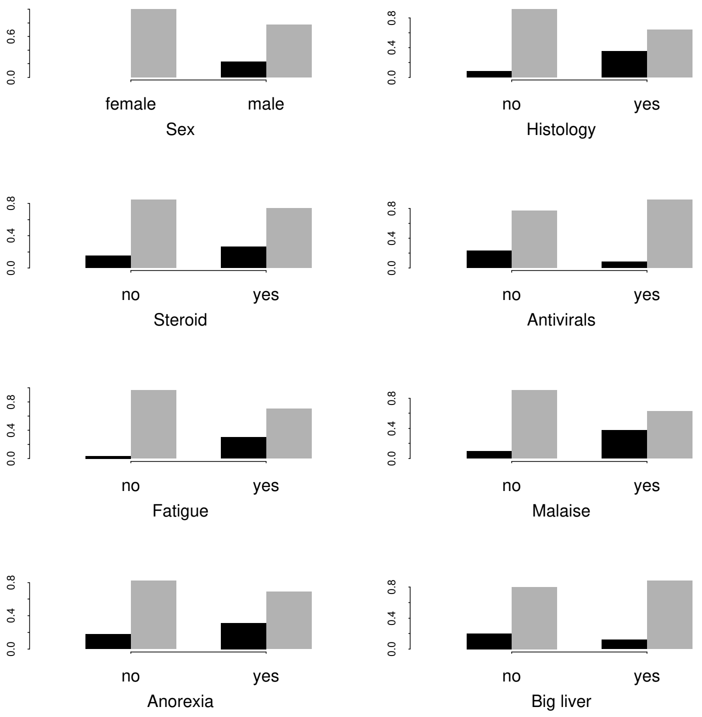
Classification example: Hepatitis data

- 155 observations
- Class variable is “die” or “live”
- 19 predictor variables (6 numerical, 13 binary)
- 15 predictor variables have missing values
- Assume 2:1 misclassification costs:

$$\text{Cost of misclassify “die”} = 2 \times \text{cost of misclassify “live”}$$

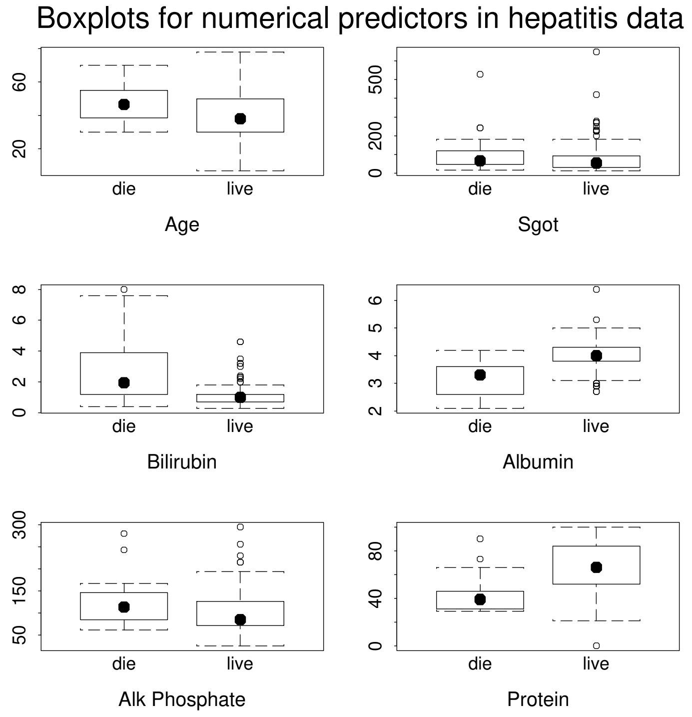
Class distributions for binary predictors

Histograms for binary predictors. Black refers to "die", gray to "live".

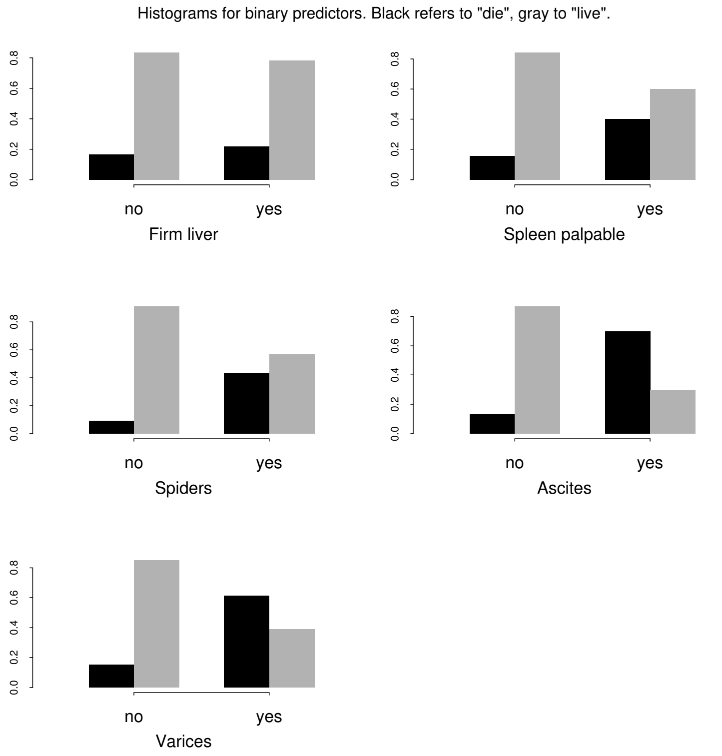


Name	Type	#miss.	Name	Type	#miss.
Person variables			Symptom variables		
Age	num.	0	Histology	bin.	0
Sex	bin.	0	Steroid	bin.	1
Medical test variables			Antivirals	bin.	0
Bilirubin	num.	6	Fatigue	bin.	1
Alk phosphate	num.	29	Malaise	bin.	1
Sgot	num.	4	Anorexia	bin.	1
Albumin	num.	16	Big liver	bin.	10
Protein	num.	67	Firm liver	bin.	11
			Spleen palpable	bin.	5
			Spiders	bin.	5
			Ascites	bin.	5
			Varices	bin.	5

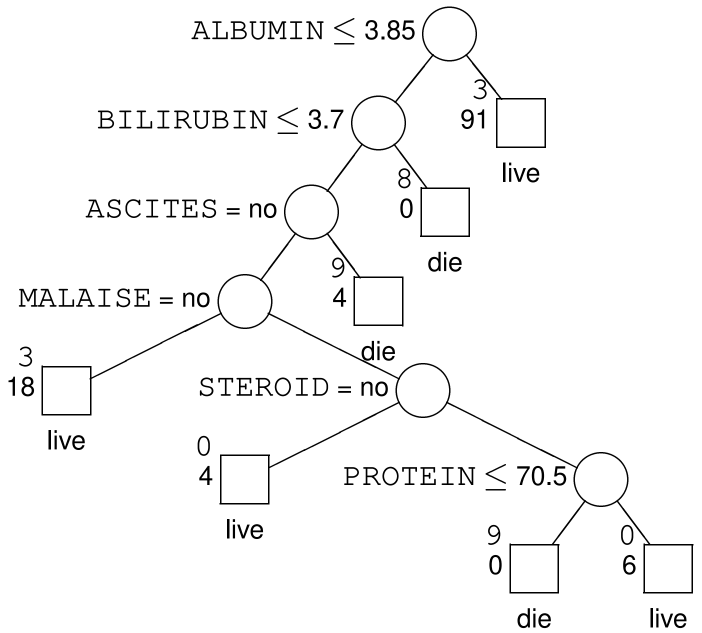
Class distributions for numerical predictors



Class dists. for binary predictors (contd.)



QUEST tree for hepatitis data



2:1 costs, 155-fold CV, 1-SE rule

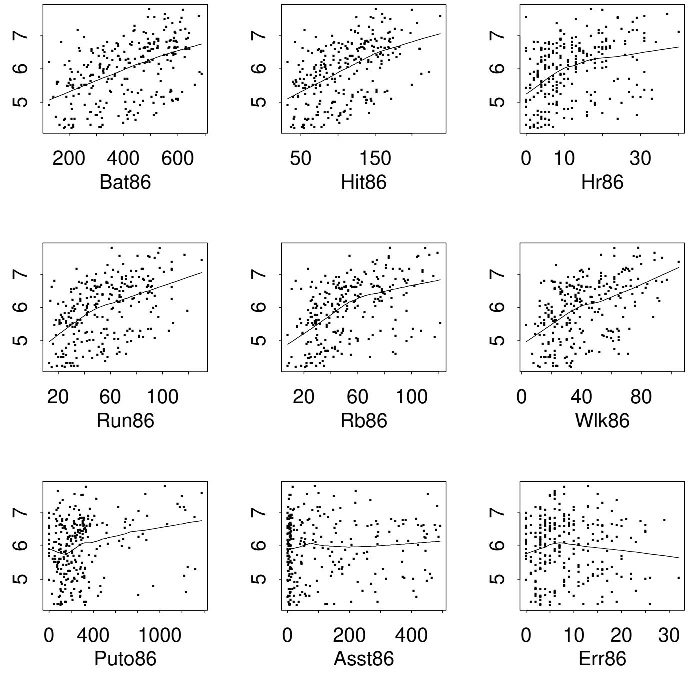
Regression tree example: Baseball data

- Response is log(1987 salary on opening day in thousands of dollars)
- 22 predictor variables based on 1986 and career attributes
- 16 numerical and 6 categorical predictor variables
- 263 cases
- No missing values

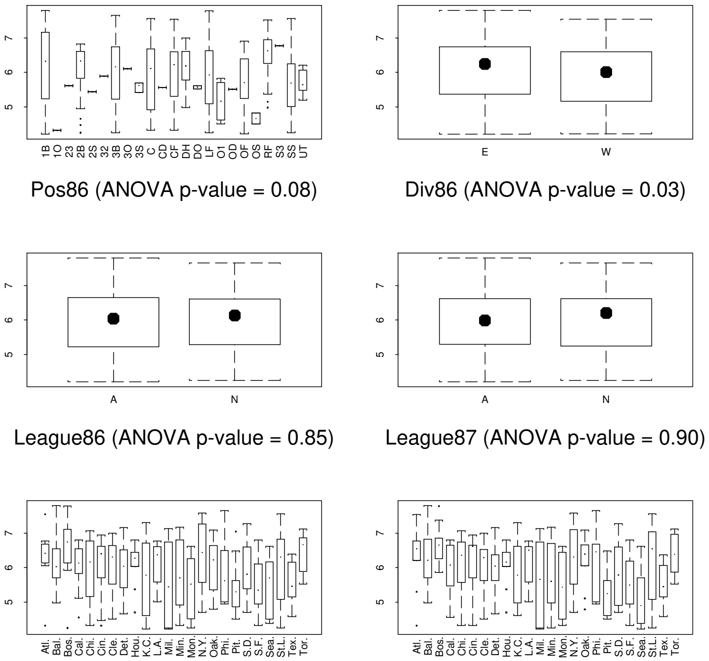
22 predictor variables

Bat86	#times at bat, 1986	Rbcr	#runs batted in, career
Hit86	#hits, 1986	Wlkcr	#walks, career
Hr86	#home runs, 1986	Leag86	league, end 1986 (2 cat.)
Run86	#runs, 1986	Div86	division, end 1986 (2 cat.)
Rb86	#runs batted in, 1986	Team86	team, end 1986 (24 cat.)
Wlk86	#walks, 1986	Pos86	position(s), 1986 (23 cat.)
Yrs	#years in major leagues	Puto86	#put outs, 1986
Batcr	#times at bat, career	Asst86	#assists, 1986
Hitcr	#hits, career	Err86	#errors, 1986
Hrcr	#home runs, career	Leag87	league, begin 1987 (2 cat.)
Runcr	#runs, career	Team87	team, begin 1987 (24 cat.)

Log-salary versus ordered predictor variables



Boxplots of log-salary vs. categorical predictors



Pos86 (ANOVA p-value = 0.08)

Div86 (ANOVA p-value = 0.03)

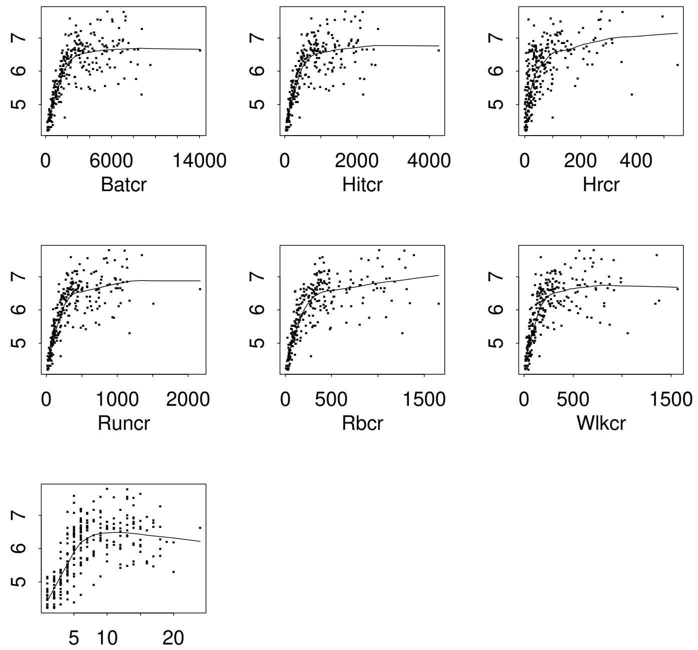
League86 (ANOVA p-value = 0.85)

League87 (ANOVA p-value = 0.90)

Team86 (ANOVA p-value = 0.15)

Team87 (ANOVA p-value = 0.003)

Log-salary vs. ordered predictor variables (cont'd)

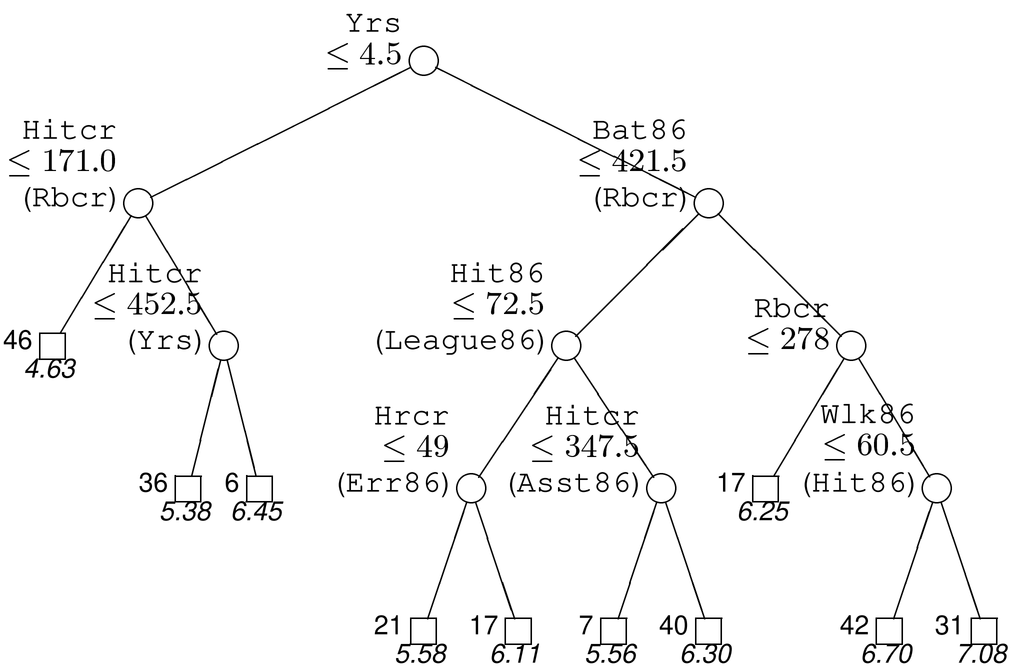


Hoaglin-Velleman model for baseball data

$$\begin{aligned}\texttt{Logsalary} &= \beta_0 + \beta_1 \texttt{Runcr}/\texttt{Yrs} \\ &\quad + \beta_2 \sqrt{\texttt{Run86}} \\ &\quad + \beta_3 \min[(\texttt{Yrs} - 2)_+, 5] \\ &\quad + \beta_4 \max[(\texttt{Yrs} - 7)_+, 0]\end{aligned}$$

Reference: Hoaglin and Velleman (1995), *Amer. Statist.*, 49, 277–285

GUIDE piecewise-constant regression tree for baseball data



263-fold CV pruning and 1-SE rule

Standard classification methods

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA)
- Density estimation methods
- Nearest-neighbor methods
- Logistic regression
- Neural networks
- Fuzzy set theory

These may produce accurate classifiers, but they do not provide much insight into the predictive structure of the problem.

Difficulties with LDA and QDA

- Multivariate normal assumption often not true
- Not designed for categorical variables
- Form of classifier in terms of linear or quadratic discriminant functions is hard to interpret

Difficulties with histogram density estimation methods

- Curse of dimensionality: Number of cells increases rapidly with number of dimensions. Thus a very large sample size is needed to prevent cells from having zero observations.
- Each cell boundary is a discontinuity. Beyond boundary cells, estimate falls abruptly to zero.

Difficulties with kernel density estimation methods

1. How to choose bandwidth h for finite sample size?
   (a) Optimal h depends on criterion (e.g., mean squared error or integrated mean squared error).
   (b) Optimal h for one class may not be optimal for another.
   (c) Optimal h depends on form of kernel.
   (d) What is optimal in one part of $\mathcal{X}$ may not be optimal in another.
2. How to choose form of kernel?

Difficulties with k-nearest-neighbor methods

1. Data must be stored and recalled each time a new case is classified. The classifier cannot be constructed beforehand.
2. Computation of nearest-neighbor distances expensive in high dimensions.
3. Choice of k is unknown for finite n.

Difficulties with logistic regression

- Hard to determine parametric form of model; few goodness of fit tests and model selection techniques.
- Usually assumes class probabilities are polynomial functions of predictor variables.
- Categorical predictors are transformed to dummy vectors. Thus a disproportionately large number of degrees of freedom may be wasted.

Difficulties with neural networks and fuzzy set theory

- Classifiers are like black boxes—hard to interpret
- Difficult to choose the network topology and initial weights
- Categorical predictors are treated via dummy vectors as in logistic regression

Desirable properties of classification models

High accuracy: Classify unseen cases with low error

Intuitive, comprehensible structure: Provide insight into the roles and relative importance of the predictor variables

Unbiased inference: Permit reliable and unbiased inferences about predictor variables

Fast training: Construct classifiers quickly

Notations

Y**:** response variable

J**:** number of classes

$\mathcal{C} = \{1, \ldots, J\}$**:** set of classes

K**:** number of predictor variables

$\mathbf{X} = (X_1, \ldots, X_K)$**:** vector of predictor variables

$\mathcal{X}$**:** Space of predictor variables

Definitions

Definition 1 *A classifier or classification rule is a function* $d(\mathbf{x})$ *defined on* $\mathcal{X}$ *such that for every* $\mathbf{x}$*,* $d(\mathbf{x})$ *is equal to one of the numbers* $1, 2, \ldots, J$*.*

Let

$$\begin{aligned} A_j &= \{\mathbf{x} : d(\mathbf{x}) = j\}, \\ \mathcal{X} &= \cup_j A_j. \end{aligned}$$

Definition 2 *A classifier is a partition of* $\mathcal{X}$ *into* J *disjoint subsets* $A_1, \ldots, A_J$*,* $\mathcal{X} = \cup_j A_j$ *such that for every* $\mathbf{x} \in A_j$*, the predicted class is* j*.*

Definition 3 *A* ***learning sample*** $\mathcal{L}$ *consists of data* $(\mathbf{x}_1, j_1), \ldots, (\mathbf{x}_N, j_N)$ *on* N *cases where* $\mathbf{x}_n \in \mathcal{X}$ *and* $j_n \in \mathcal{C}$*,* $n = 1, \ldots, N$*, i.e.,*

$$\mathcal{L} = \{(\mathbf{x}_1, j_1), \ldots, (\mathbf{x}_N, j_N)\}.$$

Types of predictor variables

- A variable is called **numerically ordered** if its values are ordered and they can be represented on the real line (e.g., age, annual gross income, years of education).
- A variable is called **nominal** or **categorical** if it takes values in a finite set not having any natural ordering (e.g., hair color, occupation, marital status).
- A variable is called **ordinal** if it takes values in a finite set whose values possess a clear ordering but the absolute distances among them is unknown (e.g., preference scale, severity of an injury, social class).

Binary Splits

- Recursive (binary) partitioning.

  Univariate split on numerically ordered or ordinal X:

  $$X \leq c$$

  Univariate split on categorical X:

  $$X \in A$$

  Linear combination split on numerically ordered variables:

  $$\sum_i a_i X_i \leq c$$

- Constants c, $\{a_i\}$ and subset A are chosen to maximize separation of the sample.

Tree notation

t denotes a node

T denotes a tree

$\tilde{T}$ is the set of terminal nodes of T

$|\tilde{T}|$ is number of terminal nodes of T

T_t is a subtree of T with root node t

$\{t\}$ is a subtree of T_t containing only the root node t

Top-Down Decision Tree Induction Schema

Input: node n, partition D, split selection method $\mathcal{CL}$
Output: decision tree for D rooted at node n

BuildTree(node t, datapartition D, split selection method $\mathcal{CL}$)
(1) Apply $\mathcal{CL}$ to D to find the splitting criterion
(2) Let k be the number of subnodes of t
(3) **if** (k $>$ 0)
(4) Create k subnodes $t_1, \ldots, t_k$ of n
(5) Use best split to partition D into $D_1, \ldots, D_k$
(6) **for** ($i = 1; i \leq k$; i++)
(7) BuildTree(t_i, D_i, $\mathcal{CL}$)
(8) **endfor**
(9) **endif**

Algorithmic components of tree construction

- Split selection method (C4.5, CART, CHAID, GUIDE, QUEST)
- Pruning method (direct stopping rule, cross-validation pruning, statistical tests, bootstrapping)
- Data access methods (necessary if training dataset is larger than main memory)

Split selection in CART [BFOS84]

Let $p(j|t)$ be the proportion of class j learning samples in node t. Define the **node impurity measure**

$$i(t) = \phi(p(\cdot|t)) \geq 0$$

where ϕ is a symmetric function with maximum value $\phi(J^{-1}, J^{-1}, \ldots, J^{-1})$ and $\phi(1, 0, \ldots, 0) = \phi(0, 1, \ldots, 0) = \ldots = \phi(0, 0, \ldots, 0, 1) = 0$.

Examples:

Entropy. $i(t) = -\sum_{j=1}^{J} p(j|t) \log p(j|t)$.

Gini index. $i(t) = 1 - \sum_j p^2(j|t)$. (Equals twice binomial variance if $J = 2$.)

Split selection in CART (Contd.)

1. Define the goodness of a split s as

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

   where t_L and t_R are the left and right subnodes of t and p_L and p_R are the probabilities of being in those subnodes.

2. Define a set $\mathcal{S}$ of binary splits generated by questions of the form:

   Is $\mathbf{x} \in A$, for $A \subset \mathcal{X}$?

3. Find s^* such that $\Delta i(s^*, t) = \max_{s \in \mathcal{S}} \Delta i(s,t)$.

Nonnegative decrease in impurity

Theorem 1 *Let $\phi(p_1, \ldots, p_J)$ be a strictly concave function on $0 \leq p_j \leq 1$, $j = 1, \ldots, J$, $\sum_j p_j = 1$. Then for*

$$i(t) = \phi(p(1|t), \ldots, p(J|t))$$

and any split s,

$$\Delta i(s,t) \geq 0$$

with equality if, and only if,

$$p(j|t_L) = p(j|t_R) = p(j|t), \quad j = 1, \ldots, J.$$

Note: The Gini index and entropy possess this property

CART linear combination splits

1. For each $\mathbf{a} = (a_1, \ldots, a_K)$, find split at c of the form

$$\sum_k a_k x_k \leq c, \quad \|\mathbf{a}\|^2 = \sum_k a_k^2 = 1.$$

   Denote the split by $s^*(\mathbf{a})$ and its decrease in impurity by $\Delta i(s^*(\mathbf{a}), t)$.

2. Suppose $\mathbf{a}^*$ satisfies

$$\Delta i(s^*(\mathbf{a}^*), t) = \max_{\mathbf{a}} \Delta i(s^*(\mathbf{a}), t).$$

3. CART selects the split: $\sum_k a_k^* x_k \leq c^*$.

4. Search algorithm can be trapped in local maxima; not guaranteed to find $\mathbf{a}^*$.

Other key steps

- Final tree obtained by
  1. Direct stopping rules, or
  2. Cross-validation pruning.
- Terminal node assignment: Let $C(i|j)$ be the cost of misclassifying a class j case as class i. Assign terminal node t to class j^* if

$$\sum_j C(j^*|j) p(j|t) = \min_i \sum_j C(i|j) p(j|t).$$

Cost complexity pruning

1. Given α and tree T, define the cost-complexity function

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|.$$

2. For each α, there is a tree T that minimizes the cost-complexity. As α increases, the tree gets shorter.

3. Let t be any node and T_t be the branch of T with root node t. Then

$$R_\alpha(\{t\}) = R(t) + \alpha, R_\alpha(T_t) = R(T_t) + \alpha|\tilde{T}_t|.$$

4. The critical value of α for which $R_\alpha(T_t) = R_\alpha(\{t\})$ is $\alpha = g(t)$, where

$$g(t) = [R(t) - R(T_t)]/[|\tilde{T}_t| - 1].$$

5. Prune away branches at nodes t_1 for which

$$g(t_1) = \min\{g(t) \,:\, t \in T - \tilde{T}\}.$$

6. Iterate to obtain a nested sequence of trees.

Subtree selection by test-sample estimation

- Estimate the misclassification cost for each subtree with the test sample
- Select the subtree with the smallest estimated cost

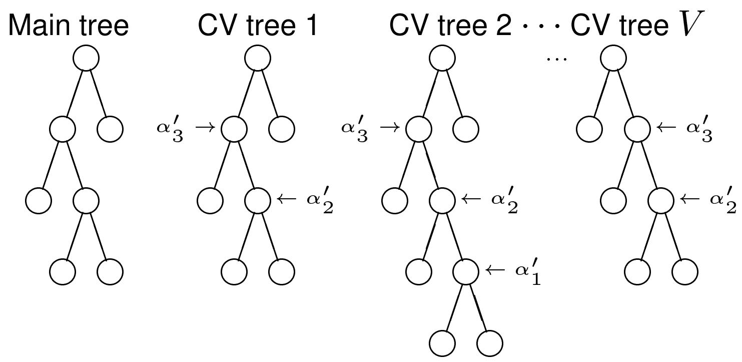
Subtree selection by V-fold cross-validation

1. Let $\alpha_1 < \alpha_2 < \ldots$ be the α-values associated with the pruned sequence of subtrees $T_1 \succ T_2 \succ \ldots$. Define $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$.
2. Divide $\mathcal{L}$ into V subsets $\mathcal{L}_1, \ldots, \mathcal{L}_V$.
3. Let $T^{(v)}(\alpha'_k)$ be the minimal cost-complexity tree grown from $\mathcal{L} - \mathcal{L}_v$, $v = 1, \ldots, V$.
4. Let $R'(T^{(v)}(\alpha'_k))$ be the estimate of the misclassification cost of $T^{(v)}(\alpha'_k)$ based on the test sample $\mathcal{L}_v$.
5. The V-fold CV estimate for subtree T_k is

$$R^{cv}(T_k) = V^{-1} \sum_{v=1}^{V} R'(T^{(v)}(\alpha'_k)).$$

6. Select the subtree with the smallest CV cost.

V-fold cross-validation

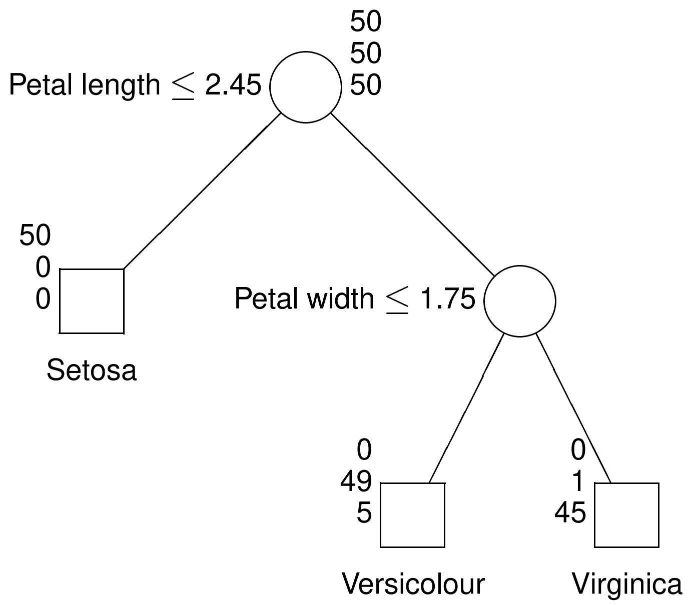


- Main tree T_{max} is grown using all data points
- Each CV tree is grown using $(V-1)$ subsets

k-SE rule

1. Let $\hat{R}(T)$ be the estimated misclassification cost for T and let $\widehat{\mathrm{SE}}(\hat{R}(T))$ be its estimated standard error.
2. Suppose subtree T^* minimizes $\hat{R}(T_k)$.
3. The k-SE tree T^{**} is the smallest subtree such that

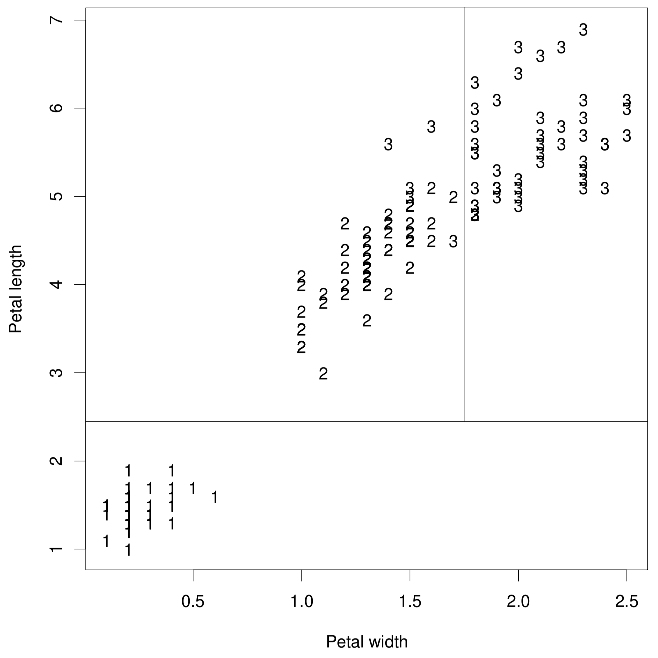
$$\hat{R}(T^{**}) \leq \hat{R}(T^*) + k\widehat{\mathrm{SE}}(\hat{R}(T^*)).$$



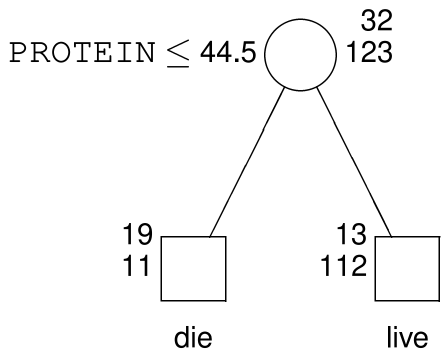
CART tree for iris data with 150-fold CV and 1-SE rule

CART partitions for Iris data

Exhaustive search method



CART tree for hepatitis data



Gini, surrogate splits, estimated priors, 2:1 costs,

N-fold CV, 0-SE and 1-SE trees same

Problems with CART

- Biased towards variables with more splits—an M-category variable has $(2^{M-1} - 1)$ splits, an M-valued ordered variable has $(M - 1)$ splits.
- Missing value method ignores quantity of missing values—e.g., a variable taking a unique value for exactly one case in each class and missing on all other cases yields the largest decrease in impurity.
- Extremely computationally expensive with linear combination splits and categorical variables with many categories.
- Univariate splits typically not much more accurate than LDA.
- No guarantee of global optimization with linear combination splits.

QUEST [LS97]

(Quick, Unbiased, Efficient, Statistical Tree)

`www.stat.wisc.edu/~loh/quest.html`

Main features:

- Fast construction by linear statistical methods
- Unbiased variable selection by statistical tests
- Pruning by test sample or cross-validation

Stopping and terminal node assignment rules

A node t is not split if any of these conditions hold for the learning samples in t:

- The cases belong to the same class
- The X values of the cases are identical
- All values are missing
- At most one class has number of cases greater than a user-specified number
- All the cases go to the same subnode if t is split

The terminal node assignment rule is the same as CART.

QUEST variable selection

Let α be a selected significance level (default is 0.05). Suppose $X_1, \ldots, X_{K_1}$ are numerically ordered and $X_{K_1+1}, \ldots, X_K$ are categorical variables.

1. Find P-value from ANOVA F-test for each ordered variable. Suppose X_{k_1} has the smallest P-value $\hat{\alpha}_1$.
2. Find P-value from χ^2-test for each categorical variable. Suppose X_{k_2} has the smallest P-value $\hat{\alpha}_2$.
3. Define $k' = \begin{cases} k_1, & \hat{\alpha}_1 \le \hat{\alpha}_2 \\ k_2, & \hat{\alpha}_1 > \hat{\alpha}_2 \end{cases}$
4. Choose $X_{k'}$ as split variable if $\min(\hat{\alpha}_1, \hat{\alpha}_2) < \alpha/K$ (Bonferroni correction).
5. Otherwise, find P-value for Levene's F-test on absolute deviations about class mean for each ordered variable. Suppose $X_{k''}$ has smallest P-value $\tilde{\alpha}$.
   (a) Split on $X_{k''}$ if $\tilde{\alpha} < \alpha/(K + K_1)$ (2nd Bonferroni correction).
   (b) Otherwise, split on $X_{k'}$.

CRIMCOORD transformation for categorical variables

1. Suppose X takes categorical values $a_1, \ldots, a_c$
2. Define $V = (v_1, \ldots, v_c)$ where $v_i = I(X = a_i)$
3. Transform the V-vectors to principal component axes and drop components with near zero variance
4. Project the reduced-dimensional V-data onto the largest discriminant coord ξ
5. Replace X with the real-valued ξ in the rest of the algorithm

Advantages of CRIMCOORD transformation

- Each ξ has the form $\xi = \sum_{i=1}^{c} b_i I(X = a_i)$ for some $b_1, \ldots, b_c$
- There is a 1-1 correspondence between ξ and X:

$$\xi = b_i \longleftrightarrow X = a_i$$

- Hence
  1. Any split of form "$\xi \leq a$" is naturally mapped to one of form "$X \in A$"—the CRIMCOORD transformation is hidden from the user
  2. Imputation of missing values is straightforward—map the mode of X to the corresponding value of ξ

QUEST split point selection

1. Let X be the selected variable (transformed to CRIMCOORDS if necessary)
2. If $J > 2$, apply a 2-means clustering algorithm to the sample class means (along the X-axis) to form two superclasses
3. Find the split point by applying to the one-dimensional two-class problem either
   (a) exhaustive search (ES) with Gini impurity function, or
   (b) quadratic discriminant analysis (QDA)

Split point selection by QDA for two classes

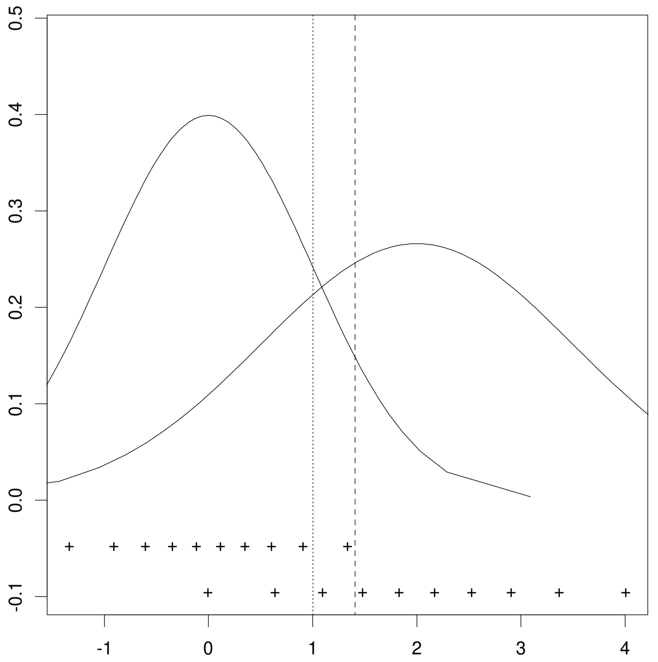
- Suppose the selected variable is X
- Let $\bar{x}_j$ and s_j^2 be the sample mean and variance for the jth class ($j = 1, 2$)
- Let $\phi(x) = (2\pi)^{-1/2} \exp(-x^2/2)$ be the standard normal density
- QDA splits the X-axis into three intervals $(-\infty, d_1)$, (d_1, d_2), and (d_2, ∞), where d_1 and d_2 are the roots of the equation

$$p(1|t)s_1^{-1}\phi\{(x-\bar{x}_1)/s_1\} = p(2|t)s_2^{-1}\phi\{(x-\bar{x}_2)/s_2\}$$

- Select as split point the root that is closer to the sample class means

QDA (dotted) vs ES (dashed); data are 10 pop. order statistics

(a) N(0,1) vs N(2,2.25)



QUEST algorithm for linear combination splits

1. Transform all categorical variables to CRIMCOORDS
2. Apply principal component analysis (PCA) to the correlation matrix of the combined (ordered and transformed categorical) data
3. Drop the small principal components and project the remaining components onto the largest CRIMCOORD
4. Apply 2-means clustering on the sample class means along the largest CRIMCOORD to obtain two superclasses
5. Find split on largest CRIMCOORD using QDA or exhaustive search

Missing values in learning sample: QUEST univariate splits

1. Each F or χ^2 test is computed from the cases with non-missing values in the respective variable.
2. After the split variable is selected, the split value is chosen using only the cases with non-missing values in that variable.
3. Cases with missing values in the split variable are imputed by the sample class mean or mode at the node to enable them to pass through the split.

Missing values in learning sample: QUEST linear combination splits

1. Each missing value in a case is estimated by the sample mean or mode for its class at the node.
2. After all the missing values are estimated, each categorical variable is transformed to a CRIMCOORD variable.
3. Split selection is then carried out as if there were no missing values.

QUEST missing value imputation for cases to be classified

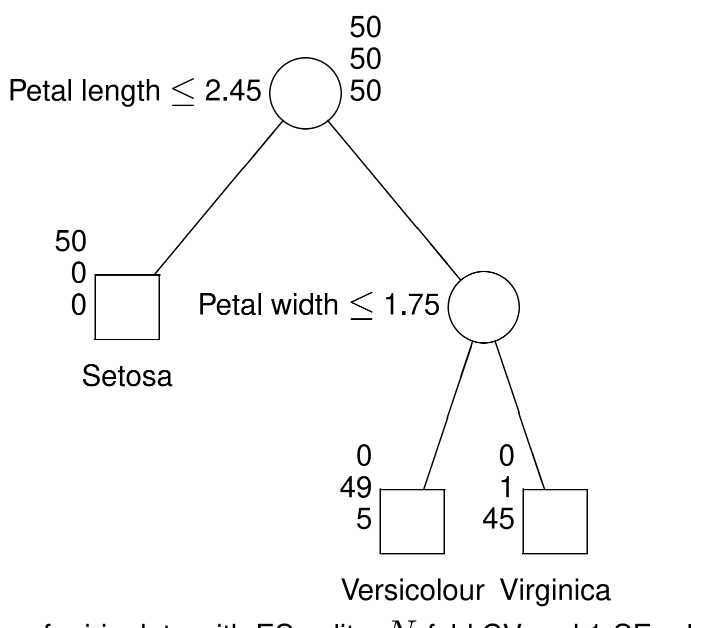
- Suppose a new case is missing a value required by a split.
- First estimate its class membership using maximum likelihood theory, assuming the variables are locally independent and numerically ordered variables are normally distributed.
- Once its class membership is estimated, its missing values are imputed by the learning sample node mean or mode for that class.

Key differences between QUEST and CART

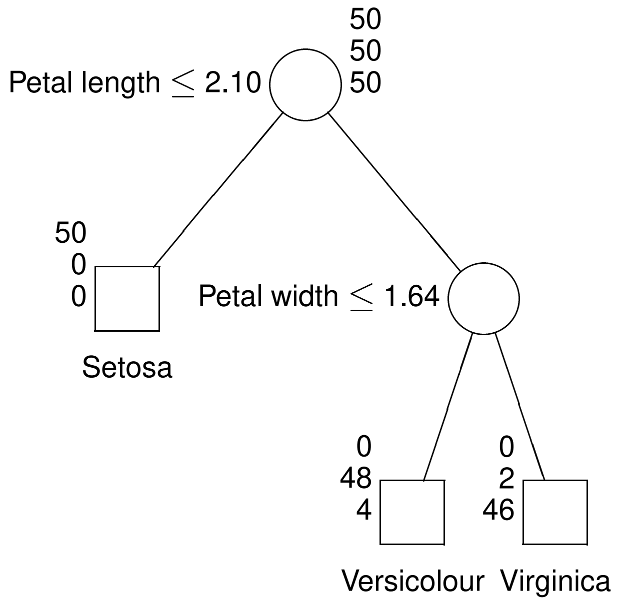
Feature	QUEST	CART
Unbiased variable selection	Yes	No
Variable selection for univar. splits	F and χ^2 tests	Exhaustive search
Split point selection for univar. splits	QDA or ES	Exhaustive search
Categorical variables treated by	PCA, LDA	Exhaustive search
Linear combination split method	PCA, LDA, QDA	Exhaustive search
Missing values handled by	Imputation	Surrogate splits
Unequal priors handled by	QDA	Impurity function
Monotone transf. in ordered vars	Not invariant	Invariant
Ordinal variables allowed	No	Yes
Tree construction speed	Excellent	Can be very slow

Similarities between QUEST and CART

- Binary trees
- Unequal misclassification costs handled via altered priors
- Pruning via test-sample or cross-validation



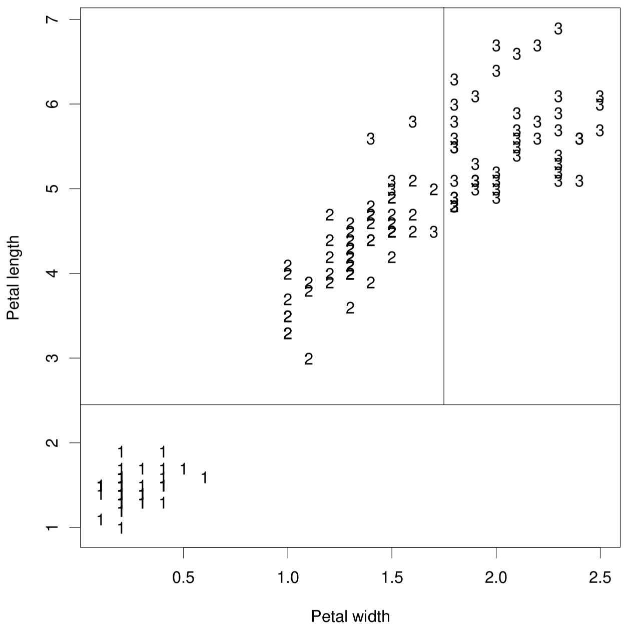
QUEST tree for iris data with ES splits, N-fold CV and 1-SE rule



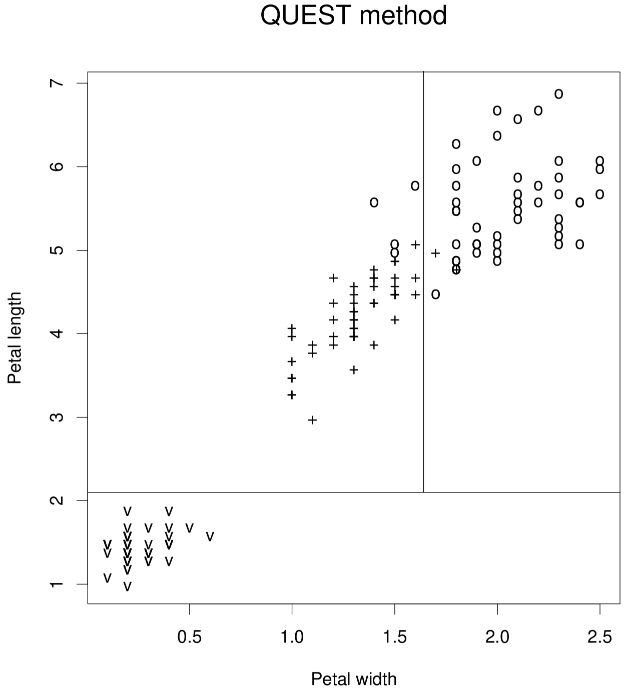
QUEST tree for iris data with QDA splits, N-fold CV and 1-SE rule

QUEST with ES splits for iris data (result same as CART)

Exhaustive search method



QUEST with QDA splits for iris data



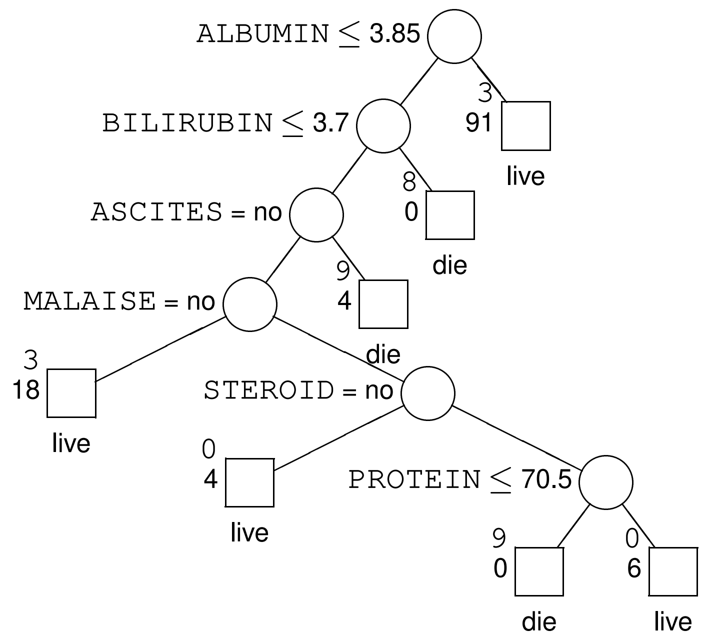
Comparison of methods for iris data

Split type	Univariate			Linear		
Variable selection	CARTⓒ	QUEST		QUEST		Stepwise
Split point selection	ES	ES	QDA	ES	QDA	LDA
No. terminal nodes	3	3	3	3	3	3
CV est. error $\pm$ .03	.07	.05	.04	.03	.03	.02

All trees use N-fold CV, equal priors and costs, and 1-SE rule.

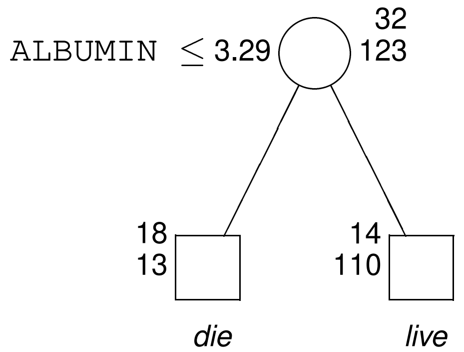
Result for stepwise LDA is from example in BMDP manual.

QUEST tree with ES split points for hepatitis data



Mean/mode imputation, estimated priors, 2:1 costs, N-fold CV and 1-SE rule

QUEST tree with QDA split points for hepatitis data



Mean/mode imputation, estimated priors, 2:1 costs, N-fold CV and 1-SE rule

Comparison of trees for hepatitis data

Split type	Univariate			Linear	
Variable selection	CARTⓒ	QUEST		QUEST	
Split point selection	ES	ES	QDA	ES	QDA
No. terminal nodes	2	7	2	2	2
CV est. cost $\pm$ .05	.25	.28	.28	.18	.19

All trees use N-fold CV, estimated priors, 2:1 costs, and 1-SE rule.

CHAID [Kas80]

- An extension of AID to categorical dependent variable
- Uses significance tests to select split variables and split points
- Uses Bonferroni method to control for multiple testing
- Values of continuous predictors must be grouped into 10 intervals first
- Uses a direct stopping rule; no pruning
- Each node can be split into many subnodes
- SPSS AnswerTree automatically categorizes continuous predictors

CHAID predictor types

Monotonic: Ordinal categorical

Free: Nominal categorical

Floating: Ordinal categorical with exception of a single category that either does not belong to the rest or whose position on the ordinal scale is unknown, e.g., “missing” category

CHAID algorithm

Let α_1, α_2 ($\alpha_1 > \alpha_2$) and α_3 be given significance levels.

1. For each predictor X, cross-tabulate categories of X with categories of Y.
2. Find the allowable (according to type of predictor) pair of categories of X whose $2 \times J$ subtable is least significant.

   If the p-value $> \alpha_1$, merge the two categories, consider the merger a single category and repeat this step.
3. For each compound category containing three or more of the original categories, find the most significant binary split (constrained by type of predictor).

   If the p-value $< \alpha_2$, split the compound category and return to Step 2.
4. Compute the Bonferroni-corrected p-value of the χ^2 statistic for each merged predictor.
5. If the smallest corrected p-value $< \alpha_3$, split the node according to the merged categories of the chosen predictor. Otherwise make the node terminal.

CHAID Bonferroni multipliers

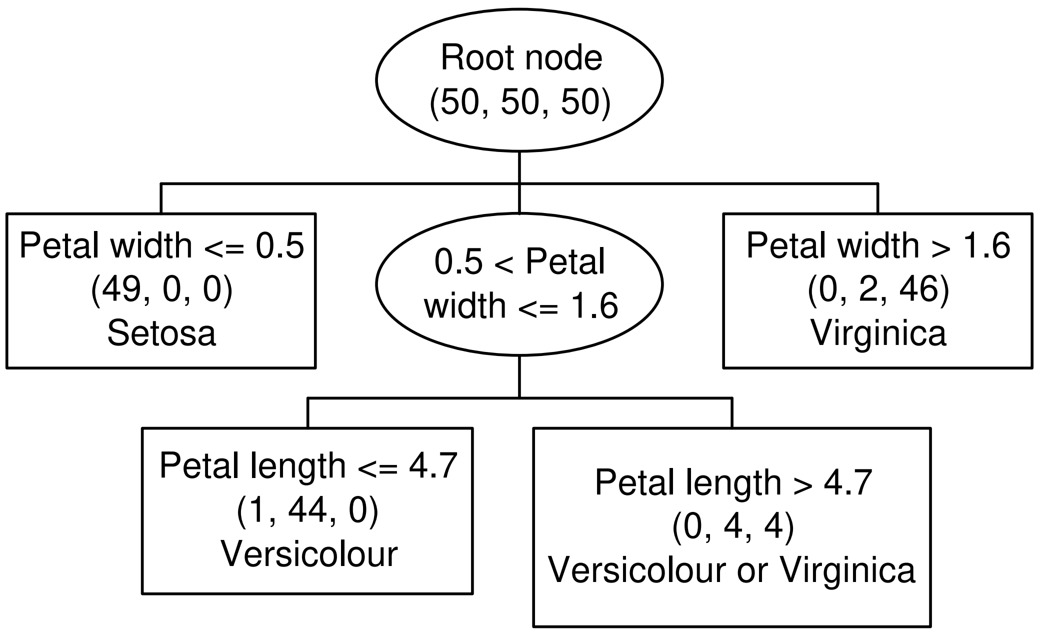
Suppose a predictor with original c categories is merged into r categories

$$\text{Monotonic:} \quad B = \binom{c-1}{r-1}$$

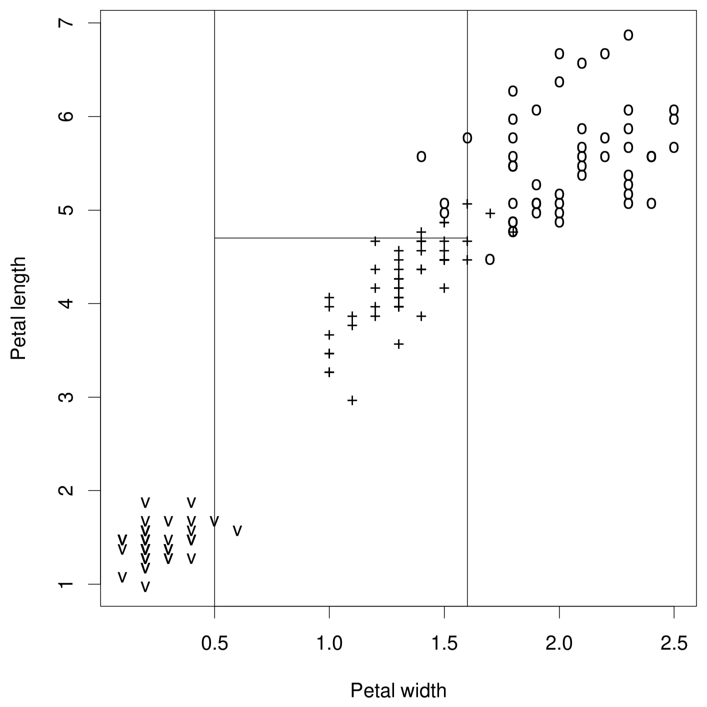
$$\text{Free:} \quad B = \sum_{i=0}^{r-1} (-1)^i \frac{(r-i)^c}{i!\,(r-i)!}$$

$$\text{Floating:} \quad B = \binom{c-2}{r-2} + r \binom{c-2}{r-1}$$

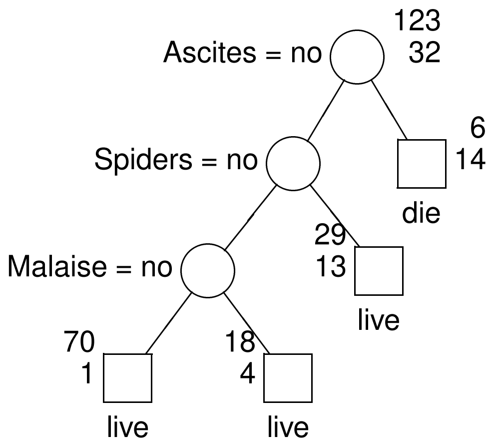
CHAID tree for iris data



CHAID method



CHAID tree for hepatitis data



2:1 costs, estimated priors

C4.5 [Qui93]

- Univariate splits only
- Binary split on ordered predictor via exhaustive search; splits at data values
- Multiway split on categorical predictor; one subnode for each categorical value (has option to merge categories)
- Pruning based on statistical heuristics; no cross-validation
- Missing values handled by case weights
- Priors and misclassification costs cannot be specified
- Cross-validation error estimate available

C4.5: Gain ratio split criterion

- Define the "info" at node t as the entropy

$$\text{info}(t) = -\sum_j p(j|t) \log_2\{p(j|t)\}$$

- Suppose t is split into subnodes $t_1, \ldots, t_n$ by predictor X. Define

$$\begin{aligned}
\text{info}_X &= \sum_i \text{info}(t_i) N(t_i)/N(t) \\
\text{gain}(X) &= \text{info}(t) - \text{info}_X(t) \\
\text{split info}(X) &= -\sum_i \{N(t_i)/N(t)\} \log_2\{N(t_i)/N(t)\} \\
\text{gain ratio}(X) &= \frac{\text{gain}(X)}{\text{split info}(X)}
\end{aligned}$$

- Split that yields the highest gain ratio is selected

C4.5: Generalization where there are missing values

- Suppose $N_m(t)$ cases are missing X at t
- Let $f = \{N(t) - N_m(t)\}/N(t)$ and define

$$\text{gain}(X) = f \times \{\text{info}(t) - \text{info}_X(t)\}$$

- Let X split the non-missing cases into subnodes $t_1, \ldots, t_n$. Define

$$\text{split info}(X) = -\sum_{i=1}^{n} \frac{|t_i|}{N(t)} \log_2 \frac{|t_i|}{N(t)} - \frac{N_m(t)}{N(t)} \log_2 \frac{N_m(t)}{N(t)}$$

- Define gain ratio$(X) = \frac{\text{gain}(X)}{\text{split info}(X)}$

C4.5: Case weights for missing values

- Initialize the weights for each case in learning sample to be 1
- Suppose t is split by X into subnodes $t_1, \ldots, t_n$
- If a case in learning sample with weight w is missing X, send it down each subnode with weight in t_i equal to

$$w_i = w \frac{|t_i|}{\sum_l |t_l|}$$

  The total sample size in each terminal node is the sum of the weights
- Do the same for each test case. If a test case ends up in more than 1 terminal node, assign it the class with largest total weight

C4.5: Pruning

- Suppose $N_E(t)$ learning cases are misclassified in node t
- C4.5 estimates the true misclassification probability with the upper 75% confidence bound p where

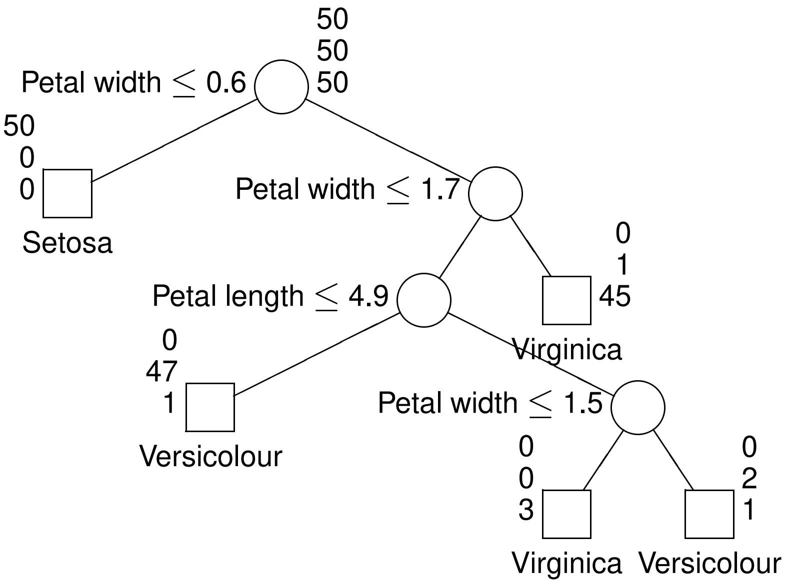
$$\sum_{i=0}^{N_E(t)} \frac{N(t)!}{i!\,(N(t)-i)!} p^i (1-p)^{N(t)-i} = 0.25$$

- Let $\nu_1 = 2(N(t) - N_E(t) + 1)$, $\nu_2 = 2N_E(t)$ and $F_{\nu_1,\nu_2;0.75}$ be the 75% percentile of the F_{ν_1,ν_2} dist. Then (Owen 1962, p. 273)

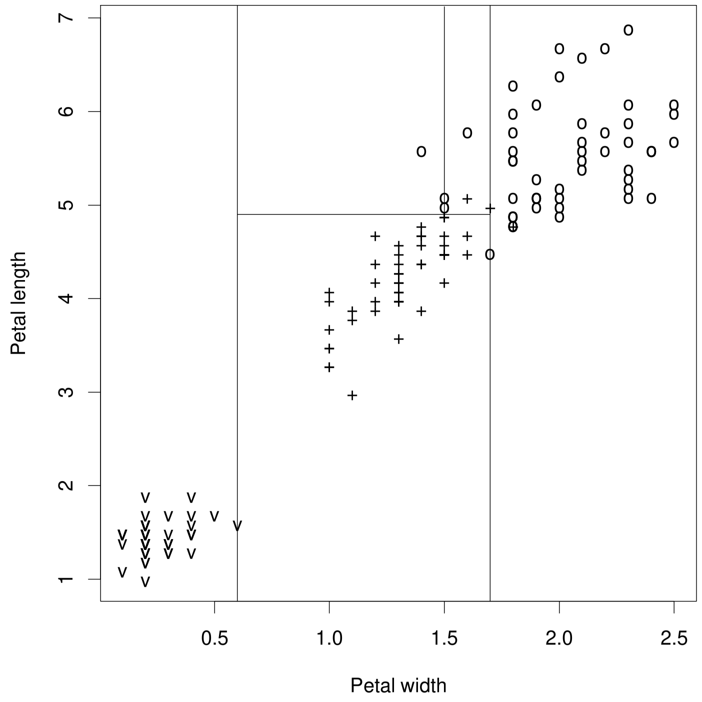
$$p = 1 - \frac{N_E(t)}{N_E(t) + (N(t) - N_E(t) + 1) F_{\nu_1,\nu_2;0.75}}$$

- The misclassification cost at t is estimated by $N(t)p$
- A branch is pruned if its estimated cost is larger than its root node

C4.5 tree for iris data



C4.5 method



CART regression trees

- Fit a constant regression model to each node
- Replace impurity function with sample Y-variance
- For node t, define $R(t) = N^{-1} \sum (y_i - \bar{y}(t))^2$, where the sum is over the cases in t and $\bar{y}(t)$ is node sample Y-mean, and $R(T) = \sum_{t \in \tilde{T}} R(t)$
- For any split s of t into t_L and t_R, let

$$\Delta R(s,t) = R(t) - R(t_L) - R(t_R)$$

- The best split s^* satisfies

$$\Delta R(s^*,t) = \max_s \Delta R(s,t)$$

- Tree is pruned using same method as in classification

Remarks

- Tree sequence tends to be larger than in classification, because pruning usually removes only two terminal nodes at a time
- There tends to be more subtrees within 1 SE of the tree with lowest CV error
- Stratification of CV samples according to values of Y appears to be beneficial
- Short-cut algorithm for categorical predictors applies to regression too.
  - Let $X \in \{b_1, \ldots, b_L\}$ be a categorical predictor
  - Let $\bar{y}(b_l)$ = mean of Y in node with $X = b_l$
  - Let $\bar{y}(b_{l_1}) \leq \bar{y}(b_{l_2}) \leq \ldots \leq \bar{y}(b_{l_L})$
  - The best split on X is one of the $L-1$ splits $X \in \{b_{l_1}, \ldots, b_{l_h}\}$, $h = 1, \ldots, L-1$
- A promising alternative to improving the fit is to grow a small tree using only a few of the top splits and then do multiple linear regression in each terminal node (CART p. 248)

Problems with exhaustive search approach

- Computationally intensive—expensive to extend beyond piecewise-constant models
- Variable selection bias—predictors with more splits are more likely to be selected

GUIDE regression tree [Loh97]

Generalized Unbiased Interaction Detection and Estimation

- Piecewise constant or piecewise linear models
- Least squares or quantile regression
- Numerically ordered or categorical predictors allowed
- Numerically ordered predictors can be used in model fitting only, splitting only, or both
- Categorical predictors used only in splitting
- Split selection sensitive to curvature and pairwise interactions
- Selection bias corrected by bootstrap
- Final tree determined by CART pruning

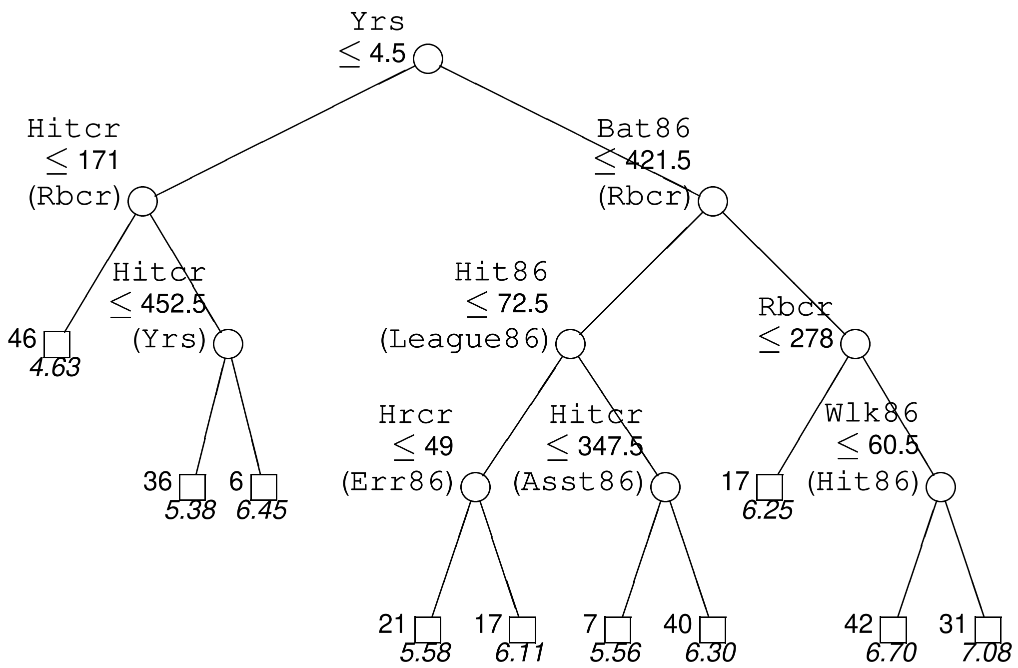
χ^2-tests for piecewise constant model

1. Fit a constant model to the data in the node and obtain residuals
2. For each numerically ordered predictor X:
   (a) Divide cases into 4 groups at the X quartiles
   (b) Cross-tab data with signs of residuals as rows and groups as columns
   (c) Compute p-value from χ^2-test (call this a ”curvature” test)
3. Do the same for each categorical predictor, using categories to form groups
4. For each pair (X_i, X_j) of numerically ordered predictors:
   (a) Divide the (X_i, X_j)-space into 4 quadrants at the sample medians
   (b) Cross-tab data with signs of residuals as rows and quadrants as columns
   (c) Compute p-value from χ^2-test (call this an ”interaction” test)

5. Do the same for each pair (X_i, X_j) of categorical predictors, using the (X_i, X_j) value-pairs to form the columns
6. For each pair (X_i, X_j), where X_i is numerically ordered and X_j is categorical:
   (a) Let the median of X_i in the node be m_i and the values taken by X_j be $c_1, \ldots, c_j$
   (b) Divide the (X_i, X_j) space into the sets $\{X_i \leq m_i, X_j = c_1\}, \ldots, \{X_i \leq m_i, X_j = c_j\}$, $\{X_i > m_i, X_j = c_1\}, \ldots, \{X_i > m_i, X_j = c_j\}$
   (c) Cross-tab the data with the signs of the residuals as rows and the $2c_j$ sets as columns
   (d) Compute the p-value from the χ^2-test

Variable selection for constant model

- If the smallest p-value is from a curvature test, select the associated predictor
- If the smallest p-value is from an interaction test, then:
  - If both predictors are numerically ordered:
    - temporarily split the node into two along the sample mean of each predictor
    - For each split, find the residual SS for a constant fit
    - Select the predictor with the smaller SS
- Otherwise, select the one with the smaller curvature p-value



GUIDE piecewise-constant regression tree for baseball data; N-fold CV and 1-SE rule

Traditional approach for categorical predictors in linear models

- Transform each categorical predictor X into a 0-1 dummy vector $(U_1, \ldots, U_c)$
- Use $U_1, \ldots, U_c$ as predictors in model
- Resulting ANCOVA model
  - uses up many degrees of freedom if c is large
  - forces same slope coefficients for numerically ordered predictors at all values of X
- Split on $U_i \leq a$ takes the form $X \in A$ versus $X \notin A$ with A an atom

GUIDE solution for categorical predictors

- Categorical predictors not used in model fitting
- ANCOVA model avoided with savings in degrees of freedom
- Categorical predictors enter tree model only via splits
- Splits on categorical variable of the form $X \in A$
- A is not limited to an atom or complement of an atom

Predictor roles in GUIDE

n: Numerically ordered variable used for fitting and splitting

f: Numerically ordered variable used for fitting only

s: Numerically ordered variable used for splitting only

c: Categorical variable used for splitting only

w: Optional numerically ordered variable used as a weight variable

χ^2-tests for piecewise linear model

1. Fit a linear model to the `n` and `f`-variables in the node and obtain residuals
2. For each `s` and `n`-variable X:
   (a) Divide cases into 4 groups at the X quartiles
   (b) Cross-tab data with signs of residuals as rows and groups as columns
   (c) Compute p-value from χ^2-test
3. Do the same for each `c`-variable, using categories to form columns of table
4. For each pair of `n`-variables (X_i, X_j):
   (a) Divide the (X_i, X_j)-space into 4 quadrants at the sample medians
   (b) Cross-tab data with signs of residuals as rows and quadrants as columns
   (c) Compute p-value from χ^2-test
5. Do the same for each pair of `s`-variables

6. Also do the same for each pair of `c`-variables (X_i, X_j), using the (X_i, X_j) value-pairs to form the columns
7. For each pair (X_i, X_j), where X_i is `n` and X_j is `c`:
   (a) Let the median of X_i in the node be m_i and the values taken by X_j be $c_1, \ldots, c_j$
   (b) Divide the (X_i, X_j) space into the sets
   $\{X_i \leq m_i, X_j = c_1\}, \ldots, \{X_i \leq m_i, X_j = c_j\}$,
   $\{X_i > m_i, X_j = c_1\}, \ldots, \{X_i > m_i, X_j = c_j\}$
   (c) Cross-tab the data with the signs of the residuals as rows and the $2c_j$ sets as columns
   (d) Compute the p-value from the χ^2-test
8. Do the same for each `s`−`c` pair and each `s`−`n` pair

Variable selection for linear model

- Above procedure gives 9 sets of χ^2-tests:
  - 3 sets for curvature in the `n`, `s`, and `c`-variables
  - 3 sets for interaction between variable pairs of same type
  - 3 sets for interaction between variable pairs of different type
- If the smallest p-value is from a curvature test, select the associated predictor
- If the smallest p-value is from an interaction test, then:
  - If neither variable is `n`-type, choose the one with smaller curvature p-value
  - If both are `n`-type, split in turn at sample mean of each variable and find total SSE of subnodes. Choose variable that yields smaller total SSE.
  - If one variable is `n` and the other `c`, choose `c`-variable
  - If one variable is `n` and the other `s`, choose `s`-variable

GUIDE missing value strategy

- Regression models fitted with cases non-missing in `n` and `f` variables
- Variable and split point selection based on non-missing cases in that variable
- If necessary, node mean/mode imputation used to direct cases with missing values through splits

Scalability to large databases

- Problem: Training database is larger than main memory
- Goal: Tree construction, using the complete training database
- Problem: Disk access 10msecs, main memory access: 50 nanoseconds
- Use of virtual memory: trashing

Example training database

Rid	Sal	Age	Emp	Label
1	30K	30	Self	Yes
2	40K	35	Ind	Yes
3	70K	50	Acad	Yes
4	60K	45	Self	Yes
5	70K	30	Acad	No
6	60K	35	Ind	No
7	60K	35	Self	No
8	70K	30	Self	No
9	40K	45	Ind	Yes

SPRINT [SAM96]

- Scalable data access method for CART split selection method
- Review: CART
- Review: Types of Splits.
  - Univariate split on numerically ordered or ordinal X: $X \leq c$; each $c \in \mathtt{dom}(X)$ induces values for $p(j|t)$
  - Univariate split on categorical X: $X \in A$, $A \subset \mathtt{dom}(X)$; need $p(j|t, x)$ for each $x \in \mathtt{dom}(X)$
  - To find best split, need to sort database for each ordered attribute

Training database sorted on salary and age

Rid	Sal	Age	Emp	Label
1	30K	30	Self	Yes
5	70K	30	Acad	No
8	70K	30	Self	No
2	40K	35	Ind	Yes
6	60K	35	Ind	No
7	60K	35	Self	No
9	40K	45	Ind	Yes
4	60K	45	Self	Yes
3	70K	50	Acad	Yes

Rid	Sal	Age	Emp	Label
1	30K	30	Self	Yes
2	40K	35	Ind	Yes
9	40K	45	Ind	Yes
4	60K	45	Self	Yes
6	60K	35	Ind	No
7	60K	35	Self	No
5	70K	30	Acad	No
8	70K	30	Self	No
3	70K	50	Acad	Yes

SPRINT algorithm

Initialization: Create attribute lists

SPRINT(Attribute lists $L_{X_1}, \ldots L_{X_K}$)

(1) **if** (not all tuples have the same class)

(2) **foreach** attribute X_i **do**

(3) evaluate split on attribute X_i using the attribute list L_{X_i}

(4) **endforeach**

(5) **foreach** attribute X_i **do**

(6) Use best split to partition L_{X_i} into $L^1_{X_i}$ and $L^2_{X_i}$

(7) **endforeach**

(8) SPRINT($L^1_{X_1}, \ldots L^1_{X_K}$)

(9) SPRINT($L^2_{X_1}, \ldots L^2_{X_K}$)

(10) **endif**

SPRINT attribute lists

Rid	Age	Label
1	30	Yes
5	30	No
8	30	No
2	35	Yes
6	35	No
7	35	No
9	45	Yes
4	45	Yes
3	50	Yes

Rid	Sal	Label
1	30K	Yes
2	40K	Yes
9	40K	Yes
4	60K	Yes
6	60K	No
7	60K	No
5	70K	No
8	70K	No
3	70K	Yes

Rid	Emp	Label
1	Self	Yes
2	Ind	Yes
3	Acad	Yes
4	Self	Yes
5	Acad	No
6	Ind	No
7	Self	No
8	Self	No
9	Ind	Yes

Partitioning the attribute lists

Rid	Age	Label
1	30	Yes
5	30	No
8	30	No
2	35	Yes
6	35	No
7	35	No
9	45	Yes
4	45	Yes
3	50	Yes

Rid	Where
1	Left
2	Left
3	Right
4	Right
5	Left
6	Left
7	Left
8	Left
9	Right

Rid	Sal	Label
1	30K	Yes
2	40K	Yes
9	40K	Yes
4	60K	Yes
6	60K	No
7	60K	No
5	70K	No
8	70K	No
3	70K	Yes

SPRINT (Contd.)

- Advantages:
  - Completely scalable data access method
  - Well-known split selection method (CART)
  - Can be implemented using SQL and user-defined functions
- Disadvantages:
  - Scales only certain class of split selection methods
  - Creation of attribute lists (worst case: three times the size of the database)

MDL pruning [MRA95]

- Pruning without looking at the data
- Pruning method based on the Minimum Description Length (MDL) Principle
- Tree "communicates" class labels
- MDL Principle: Best tree encodes labels using fewest bits
- How to encode the tree? Consider node t:
  - Type of node (1 bit)
  - Leaf node: cost of encoding the records in t
    $|t| \cdot \texttt{ent}(t)$ (entropy of the records)
  - Internal node: cost of encoding splitting criterion
- Prune a node if cost of encoding subtree is larger than cost of encoding records at the node

PUBLIC [RS98]

- Idea: Instead of growing the tree completely and then pruning it back, stop tree growth by estimating lower bound on MDL cost of a subtree
- Theorem: For a node t, the MDL cost $c(T_t)$ of any subtree with s splits is at least:

$$c(T_t) \geq 2 \cdot s + 1 + s \cdot \log(K) + |t| \cdot \sum_{i=s+2}^{J} p(i|t).$$

- Check periodically if subtree needs to be expanded further
- Early stopping cuts tree construction time by a factor of two to three

RainForest [GRG98]

- Construct *sufficient statistics* called *AVC-sets* at each node
- AVC-sets are compact (reason: univariate splits)

Sal	**Yes**	**No**
30K	1	0
40K	2	0
60K	1	2
70K	1	2

Age	**Yes**	**No**
30	1	2
35	1	2
45	2	0
50	1	0

- Allows plug-and-play:
  - Use any split selection method for univariate splits
  - Use any univariate discretization method
  - Extends to piecewise constant regression trees

RainForest: RF-Write

RF-Write(Database D)
(1) **if** (not all tuples have the same class)
(2) Make one scan over D and construct AVC-sets in-memory
(3) Find best split using in-memory AVC-sets
(4) Use best split to partition D into D_1 and D_2
(5) RainForest(D_1)
(6) RainForest(D_2)
(7) **endif**

- Assumes AVC-sets at root fit in-memory
- Two database scans per level of the tree
- Usually more main memory available

RainForest: RF-Read

RF-Read(Database D)

Q = $\{$rootnode$\}$
while($Q \neq \emptyset$) **do**
    $Q' = \emptyset$
    Scan D and construct AVC-sets for all nodes in Q in-memory
    foreach ($t \in Q$)
        Find best split for t using in-memory AVC-sets
        Create subnodes t_1 and t_2
        if(t_1 not is pure) $Q' = Q' \cup \{t_1\}$
        if(t_2 not is pure) $Q' = Q' \cup \{t_2\}$
    endforeach
    $Q = Q'$
endwhile

RainForest: RF-Hybrid

RF-Hybrid(Database D, node t)

Q = $\{t\}$
while($Q \neq \emptyset$) **do**
    $Q' = \emptyset$
    Scan D and construct AVC-sets for all nodes in Q in-memory
    foreach ($t \in Q$)
        Find best split for t using in-memory AVC-sets
        Create subnodes t_1 and t_2
        if(t_1 not is pure) $Q' = Q' \cup \{t_1\}$
        if(t_2 not is pure) $Q' = Q' \cup \{t_2\}$
    endforeach
    $Q = Q'$
endwhile

Integration with relational databases [GFC98]

- AVC-sets contain sufficient statistics for univariate splits:

```
SELECT      "Sal" AS AttName, Sal AS AttVal, Label, COUNT(*)
FROM        Relation
GROUP BY    Att, AttVal, Label
UNION ALL
SELECT      "Age" AS AttName, Age AS AttVal, Label, COUNT(*)
...
```

- Query evaluation: several separate scans over the relation
- Approach: new relational operator **unpivot**
- Result is new table with schema (Rid, AttName, AttVal, Label, Count)
- Compare to attribute lists in SPRINT

BOAT — Optimistic Tree Construction [GGRL99]

- Main idea: Build an approximate from a sample and then "adjust" the approximate tree in a full database scan
- Contributions:
  - In most cases only two full scans over the database required
  - Always constructs exactly the same tree as if complete database would fit in-memory
  - Approximate tree is immediately available to the user (interactivity)
  - Allows to update the tree with new data incrementally
  - Can use nearly any split selection method
  - Extends to piecewise constant regression trees

Empirical comparison of classifiers
(How to choose a classification algorithm?)

- 33 classification algorithms
- 32 data sets mostly from UC Irvine Machine Learning Repository
- Each algorithm applied to each data set
- Classification accuracy, computational speed, and classifier complexity recorded
- Lim, Loh and Shih (1999), *Machine Learning*, in press

33 classification algorithms

22 trees: IND, S-Plus tree, C4.5, FACT, QUEST, OC1, LMDT, CAL5, T1

9 statistical classifiers: LDA, QDA, NN, LOG, FDA, PDA, MDA, POL

2 neural networks: LVQ, RBF

16 primary data sets

Name	Code	N	Class	Num	Cat
Breast cancer	`bcw`	683	2	9	0
Contraceptive meth.	`cmc`	1473	3	2	7
DNA	`dna`	2000	3	0	60
Heart disease	`hea`	270	2	7	6
Boston housing	`bos`	506	3	12	1
LED	`led`	2000	10	0	7
BUPA liver disorders	`bld`	345	2	6	0
PIMA Indian diabetes	`pid`	532	2	7	0
Satellite	`sat`	4435	6	36	0
Image segmentation	`seg`	2310	7	19	0
Smoking	`smo`	1855	3	3	5
Thyroid	`thy`	3772	3	6	15
Vehicle silhouette	`veh`	846	4	18	0
Voting records	`vot`	435	2	0	16
Waveform	`wav`	600	3	21	0
Teaching evaluation	`tae`	151	3	1	4

Experimental details

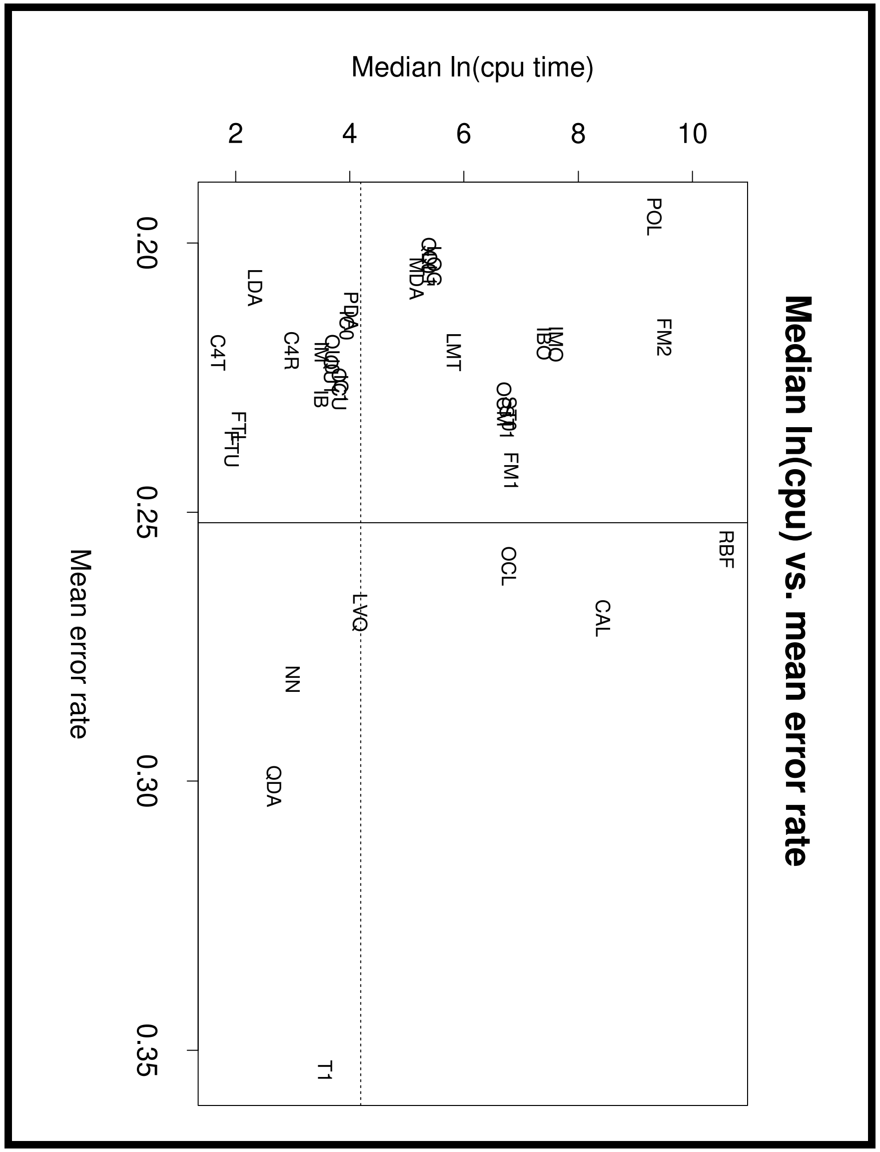
- 16 data sets created by adding noise variables to each primary data set
- Categorical variables converted to 0-1 vectors for classifiers that accept only numerical variables
- Error rates for 6 primary data sets (`dna`, `led`, `sat`, `smo`, `thy`, `wav`) estimated from test sets
- 10-fold cross-validation used for other data sets

Ranking by mean error rate (cont'd)

Rank	Code	Description	Mean	Sig.
18	OCU	OC1 univ. tree	.227	no
19	IC1	IND CART	.227	no
20	IB	IND Bayes tree	.229	no
21	OCM	OC1 mixed tree	.230	no
22	ST0	Splus tree	.232	no
23	ST1	Splus tree	.233	no
24	FTL	FACT linear	.234	no
25	FTU	FACT univ.	.238	no
26	FM1	FDA MARS	.242	no
27	RBF	Radial basis function net	.257	yes
28	OCL	OC1 linear tree	.260	yes
29	LVQ	Learning vector quant. net	.269	yes
30	CAL	CAL5 tree	.270	yes
31	NN	Nearest neighbor	.281	yes
32	QDA	Quadratic discrim. anal.	.301	yes
33	T1	T1 tree	.354	yes

Ranking by mean error rate

Rank	Code	Description	Mean	Sig.
1	POL	POLYCLASS	.195	no
2	QL0	QUEST linear	.202	no
3	LOG	Logistic regression	.204	no
4	QL1	QUEST linear	.205	no
5	MDA	Mixture discrim. anal.	.207	no
6	LDA	Linear discrim. anal.	.208	no
7	PDA	Penalized discrim. anal.	.213	no
8	IC0	IND CART	.215	no
9	FM2	FDA MARS	.217	no
10	IBO	IND Bayes tree	.219	no
11	IMO	IND Bayes tree	.219	no
12	C4R	C4.5 rules	.220	no
13	LMT	LMDT tree	.220	no
14	IM	IND Bayes tree	.220	no
15	C4T	C4.5 tree	.220	no
16	QU0	QUEST univ.	.221	no
17	QU1	QUEST univ.	.224	no



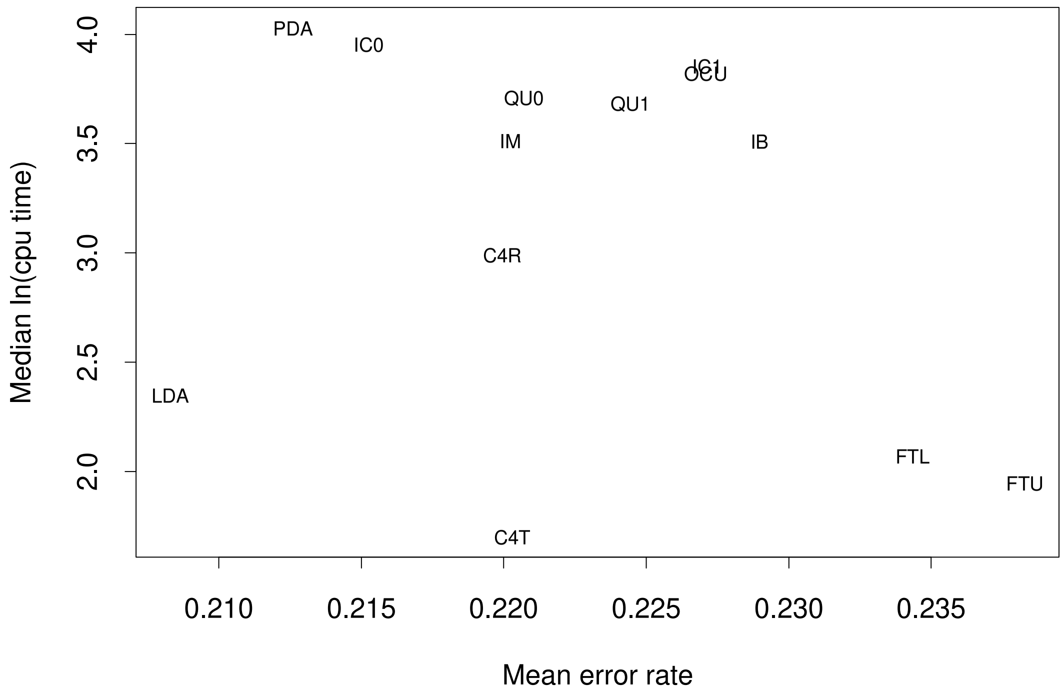
KDD99 T1-122

Ranking by median cpu time

Algorithm	CPU	Algorithm	CPU
C4T C4.5 tree	5s	MDA Mixture disc.	3m
FTU FACT univ.	7s	QL0 QUEST lin.	4m
FTL FACT lin.	8s	QL1 QUEST lin.	4m
LDA Lin. disc.	10s	LOG Logistic reg.	4m
QDA Quad. disc.	15s	LMT LMDT	6m
C4R C4.5 rules	20s	OCM OC1 mixed	14m
NN Nearest neigh.	20s	ST1 Splus tree	14m
IB IND Bayes	34s	OCL OC1 lin.	15m
IM IND Bayes	34s	ST0 Splus tree	15m
T1 One-split	36s	FM1 FDA MARS	16m
QU1 QUEST univ.	40s	IBO IND Bayes	28m
QU0 QUEST univ.	41s	IMO IND Byes	34m
OCU OC1 univ.	46s	CAL CAL5	1h
IC1 IND CART	47s	POL POLYCLASS	3h
IC0 IND CART	52s	FM2 FDA MARS	4h
PDA Penal. disc.	56s	RBF Radial basis	11h
LVQ Lin. vector	1m		

KDD99 T1-121

Median ln(cpu) vs. mean error for fast and accurate classifiers



Ranking by median number of leaves

Rank	Algorithm	Median
1	`FTL` (FACT linear)	3.0
2	`T1` (One-split)	5.0
3	`QL1` (1se QUEST linear)	5.0
4	`QL0` (0se QUEST linear)	6.5
5	`ST1` (1se Splus tree)	7.5
6	`IC1` (1se IND CART)	7.5
7	`OCM` (OC1 mixed)	7.5
8	`LMT` (LMDT)	8.5
9	`OCL` (OC1 linear)	9.0
10	`QU1` (1se QUEST univ.)	11.5

Ranking by median number of leaves (cont'd)

Rank	Algorithm	Median
11	`ST0` (0se Splus tree)	13.0
12	`FTU` (FACT univ.)	13.0
13	`IC0` (0se IND CART)	13.0
14	`OCU` (OC1 univ.)	15.0
15	`QU0` (0se QUEST univ.)	20.5
16	`CAL` (CAL5)	29.5
17	`C4T` (C4.5 tree)	32.5
18	`IM` (IND Bayes)	79.5
19	`IB` (IND Bayes)	80.0
20	`IMO` (IND Bayes)	11,989.0
21	`IBO` (IND Bayes)	18,824.0

Variable selection bias of greedy search algorithms

- Biased towards variables with more splits—an M-category variable has $(2^{M-1} - 1)$ splits, an M-valued ordered variable has $(M - 1)$ splits.
- Biased towards variables with more missing values.

Variable selection bias without missing values

		CART				QUEST			
		k				k			
X_i	Dist.	5	10	15	20	5	10	15	20
X_1	$N(0,1)$	.41	.25	.12	.05	.20	.20	.22	.20
X_2	$E(0,1)$	.42	.26	.12	.05	.23	.20	.21	.21
X_3	$U\{4\}$	.04	.02	.01	.00	.21	.21	.20	.20
X_4	$C\{2\}$	.02	.01	.01	.00	.19	.19	.18	.21
X_5	$C\{k\}$	.11	.46	.74	.90	.18	.20	.20	.19

1000 cases, 2 classes, 5 predictors independent of class

$U\{k\}$ is uniform distribution on $\{1, 2, \ldots, k\}$

$C\{k\}$ is an equi-probable k-category predictor

Selection probabilities for QUEST all within 2 SEs of .20

Variable selection bias with missing values

		CART				QUEST			
		Percent missing X_1				Percent missing X_1			
X_i	Dist.	20	40	60	80	20	40	60	80
X_1	$N(0,1)$	.38	.50	.62	.74	.22	.20	.23	.21
X_2	$E(0,1)$	.22	.14	.13	.07	.21	.20	.20	.21
X_3	$U\{4\}$	.03	.02	.01	.01	.19	.20	.19	.20
X_4	$C\{2\}$	.01	.00	.01	.01	.20	.21	.21	.19
X_5	$C\{10\}$	.37	.34	.23	.17	.18	.20	.19	.20

1000 cases, 2 classes, 5 predictors independent of class

$U\{k\}$ is uniform distribution on $\{1, 2, \ldots, k\}$

$C\{k\}$ is an equi-probable k-category predictor

Selection probabilities for QUEST all within 2 SEs of .20

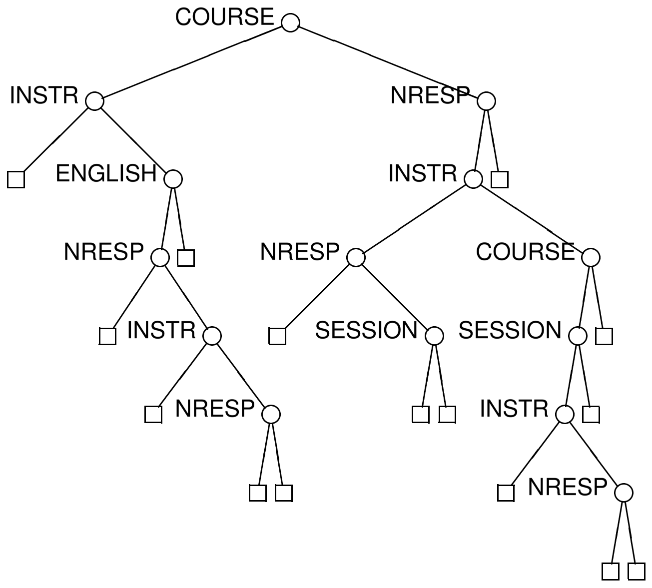
Example: Teaching assistant evaluation data

- 151 teaching assistant assignments over five semesters
- Response is student evaluation score for TA (grouped into two classes: above or below average)
- Predictor variables are:
  1. `ENGLISH`: Whether or not the TA is a native English speaker (binary)
  2. `COURSE` (26 categories)
  3. `INSTR`: Course instructor (25 categories)
  4. `SESSION`: Summer or regular semester (binary)
  5. `NRESP`: Number of responses (numerically ordered)

Statistical significance of predictors

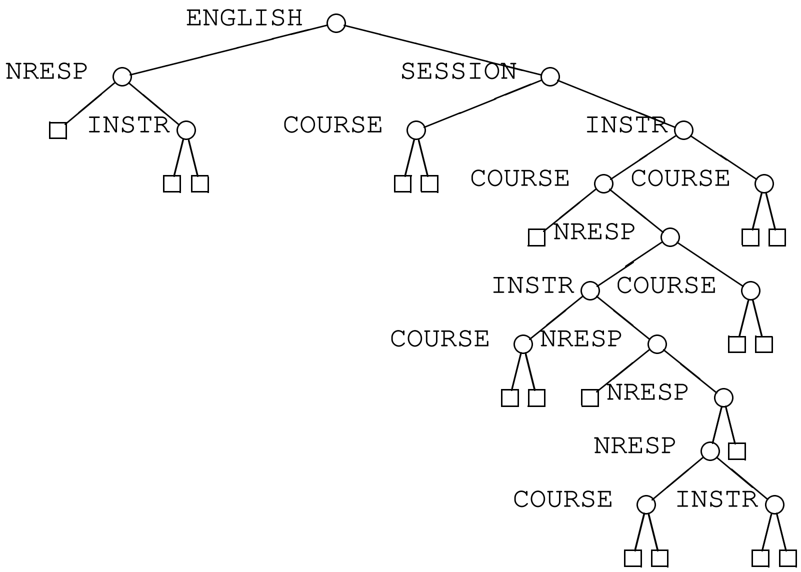
Predictor	P-value
`ENGLISH`	0.005
`SESSION`	0.010
`COURSE`	0.019
`INSTR`	0.171
`NRESP`	0.992

CART 0-SE tree for TA data (1-SE tree null)



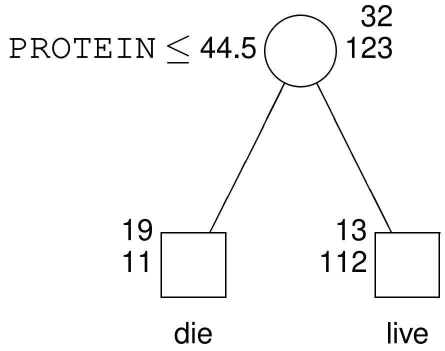
Estimated priors, equal costs, 151-fold CV

QUEST 1-SE tree for TA data



Estimated priors, equal costs, 151-fold CV

CART tree for hepatitis data



Estimated priors, 2:1 misclassification costs,
surrogate splits, N-fold CV. 0-SE and 1-SE rules the same

Concluding remarks

- Many algorithms have good predictive accuracy (*Rashomon effect:* Breiman)
- Some algorithms are fast; some take much time to train
- Some simple algorithms (e.g., LDA) are fast and accurate
- Some tree algorithms produce huge trees
- Tree algorithms that use greedy search have selection bias towards:
  - variables with more splits, and
  - variables with more missing values

  (*Rashomon through tinted lenses*)