Classification and regression trees

Wei-Yin Loh

Classification and regression trees are machine-learning methods for constructing prediction models from data. The models are obtained by recursively partitioning the data space and fitting a simple prediction model within each partition. As a result, the partitioning can be represented graphically as a decision tree. Classification trees are designed for dependent variables that take a finite number of unordered values, with prediction error measured in terms of misclassification cost. Regression trees are for dependent variables that take continuous or ordered discrete values, with prediction error typically measured by the squared difference between the observed and predicted values. This article gives an introduction to the subject by reviewing some widely available algorithms and comparing their capabilities, strengths, and weakness in two examples. *WIREs Data Mining Knowl Discov* 2011 1 14–23 DOI: 10.1002/widm.8

CLASSIFICATION TREES

In a classification problem, we have a training sample of n observations on a class variable Y that takes values 1, 2, ... , k, and p predictor variables, $X_1, \ldots, X_p$. Our goal is to find a model for predicting the values of Y from new X values. In theory, the solution is simply a partition of the X space into k disjoint sets, $A_1, A_2, \ldots, A_k$, such that the predicted value of Y is j if X belongs to A_j, for $j = 1, 2, \ldots, k$. If the X variables take ordered values, two classical solutions are linear discriminant analysis[1] and nearest neighbor classification.[2] These methods yield sets A_j with piecewise linear and nonlinear, respectively, boundaries that are not easy to interpret if p is large.

Classification tree methods yield rectangular sets A_j by recursively partitioning the data set one X variable at a time. This makes the sets easier to interpret. For example, Figure 1 gives an example wherein there are three classes and two X variables. The left panel plots the data points and partitions and the right panel shows the corresponding decision tree structure. A key advantage of the tree structure is its applicability to any number of variables, whereas the plot on its left is limited to at most two.

The first published classification tree algorithm is THAID.[3,4] Employing a measure of node impurity based on the distribution of the observed Y values in the node, THAID splits a node by exhaustively searching over all X and S for the split $\{X \in S\}$ that minimizes the total impurity of its two child nodes. If X takes ordered values, the set S is an interval of the form $(-\infty, c]$. Otherwise, S is a subset of the values taken by X. The process is applied recursively on the data in each child node. Splitting stops if the relative decrease in impurity is below a prespecified threshold. Algorithm 1 gives the pseudocode for the basic steps.

Algorithm 1 *Pseudocode for tree construction by exhaustive search*

1. *Start at the root node.*
2. *For each X, find the set S that minimizes the sum of the node impurities in the two child nodes and choose the split $\{X^* \in S^*\}$ that gives the minimum overall X and S.*
3. *If a stopping criterion is reached, exit. Otherwise, apply step 2 to each child node in turn.*

C4.5[5] and CART[6] are two later classification tree algorithms that follow this approach. C4.5 uses entropy for its impurity function, whereas CART uses a generalization of the binomial variance called the Gini index. Unlike THAID, however, they first grow an overly large tree and then prune it to a smaller size to minimize an estimate of the misclassification error. CART employs 10-fold (default) cross-validation, whereass C4.5 uses a heuristic formula to estimate error rates. CART is implemented in the R system[7] as RPART,[8] which we use in the examples below.

Despite its simplicity and elegance, the exhaustive search approach has an undesirable property. Note that an ordered variable with m distinct values has $(m-1)$ splits of the form $X \leq c$, and an

*Correspondence to: loh@stat.wisc.edu

Department of Statistics, University of Wisconsin-Madison, Madison, WI, USA

DOI: 10.1002/widm.8

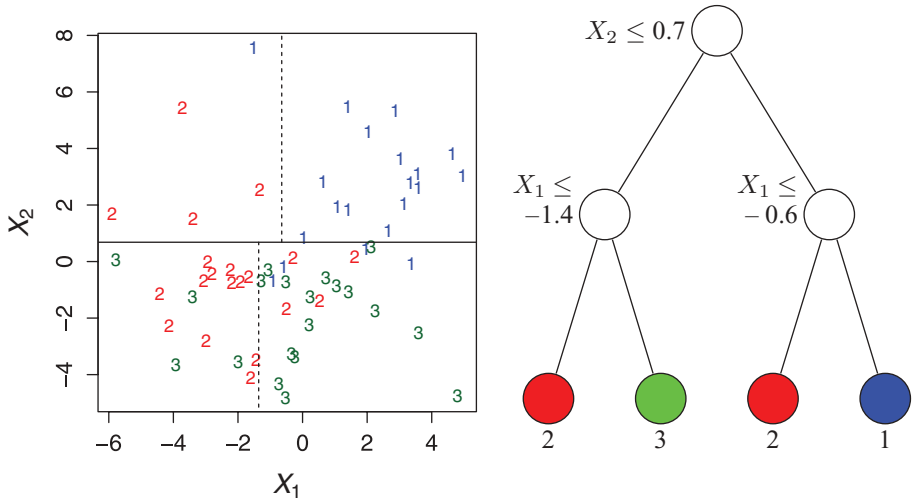

FIGURE 1 | Partitions (left) and decision tree structure (right) for a classification tree model with three classes labeled 1, 2, and 3. At each intermediate node, a case goes to the left child node if and only if the condition is satisfied. The predicted class is given beneath each leaf node.

unordered variable with m distinct unordered values has $(2^{m-1} - 1)$ splits of the form $X \in S$. Therefore, if everything else is equal, variables that have more distinct values have a greater chance to be selected. This selection bias affects the integrity of inferences drawn from the tree structure.

Building on an idea that originated in the FACT[9] algorithm, CRUISE,[10,11] GUIDE,[12] and QUEST[13] use a two-step approach based on significance tests to split each node. First, each X is tested for association with Y and the most significant variable is selected. Then, an exhaustive search is performed for the set S. Because every X has the same chance to be selected if each is independent of Y, this approach is effectively free of selection bias. Besides, much computation is saved as the search for S is carried out only on the selected X variable. GUIDE and CRUISE use chi squared tests, and QUEST uses chi squared tests for unordered variables and analysis of variance (ANOVA) tests for ordered variables. CTree,[14] another unbiased method, uses permutation tests. Pseudocode for the GUIDE algorithm is given in Algorithm 2. The CRUISE, GUIDE, and QUEST trees are pruned the same way as CART.

Algorithm 2 *Pseudocode for GUIDE classification tree construction*

1. *Start at the root node.*
2. *For each ordered variable X, convert it to an unordered variable X′ by grouping its values in the node into a small number of intervals. If X is unordered, set X′ = X.*
3. *Perform a chi squared test of independence of each X′ variable versus Y on the data in the node and compute its significance probability.*
4. *Choose the variable X* associated with the X′ that has the smallest significance probability.*
5. *Find the split set $\{X^* \in S^*\}$ that minimizes the sum of Gini indexes and use it to split the node into two child nodes.*
6. *If a stopping criterion is reached, exit. Otherwise, apply steps 2–5 to each child node.*
7. *Prune the tree with the CART method.*

CHAID[15] employs yet another strategy. If X is an ordered variable, its data values in the node are split into 10 intervals and one child node is assigned to each interval. If X is unordered, one child node is assigned to each value of X. Then, CHAID uses significance tests and Bonferroni corrections to try to iteratively merge pairs of child nodes. This approach has two consequences. First, some nodes may be split into more than two child nodes. Second, owing to the sequential nature of the tests and the inexactness of the corrections, the method is biased toward selecting variables with few distinct values.

CART, CRUISE, and QUEST can allow splits on linear combinations of all the ordered variables, whereas GUIDE can split on combinations of two variables at a time. If there are missing values, CART and CRUISE use alternate splits on other variables when needed, C4.5 sends each observation with a missing value in a split through every branch using

a probability weighting scheme, QUEST imputes the missing values locally, and GUIDE treats missing values as belonging to a separate category. All except C4.5 accept user-specified misclassification costs and all except C4.5 and CHAID accept user-specified class prior probabilities. By default, all algorithms fit a constant model to each node, predicting Y to be the class with the smallest misclassification cost. CRUISE can optionally fit bivariate linear discriminant models and GUIDE can fit bivariate kernel density and nearest neighbor models in the nodes. GUIDE also can produce ensemble models using bagging[16] and random forest[17] techniques. Table 1 summarizes the features of the algorithms.

To see how the algorithms perform in a real application, we apply them to a data set on new cars for the 1993 model year.[18] There are 93 cars and 25 variables. We let the Y variable be the type of drive train, which takes three values (rear, front, or four-wheel drive). The X variables are listed in Table 2. Three are unordered (*manuf*, *type*, and *airbag*, taking 31, 6, and 3 values, respectively), two binary-valued (*manual* and *domestic*), and the rest ordered. The class frequencies are rather unequal: 16 (17.2%) are rear, 67 (72.0%) are front, and 10 (10.8%) are four-wheel drive vehicles. To avoid randomness due to 10-fold cross-validation, we use leave-one-out (i.e., *n*-fold) cross-validation to prune the CRUISE, GUIDE, QUEST, and RPART trees in this article.

Figure 2 shows the results if the 31-valued variable *manuf* is excluded. The CHAID tree is not shown because it has no splits. The wide variety of variables selected in the splits is due partly to differences between the algorithms and partly to the absence of a dominant X variable. Variable *passngr* is chosen by three algorithms (C4.5, GUIDE QUEST); *enginsz*, *fuel*, *length*, and *minprice* by two; and *hp*, *hwympg*, *luggage*, *maxprice*, *rev*, *type*, and *width* by one each. Variables *airbag*, *citympg*, *cylin*, *midprice*, and *rpm* are not selected by any.

When GUIDE does not find a suitable variable to split a node, it looks for a linear split on a pair of variables. One such split, on *enginsz* and *rseat*, occurs at the node marked with an asterisk (*) in the GUIDE tree. Restricting the linear split to two variables allows the data and the split to be displayed in a plot as shown in Figure 3. Clearly, no single split on either variable alone can do as well in separating the two classes there.

Figure 4 shows the C4.5, CRUISE, and GUIDE trees when variable *manuf* is included. Now CHAID, QUEST, and RPART give no splits. Comparing them with their counterparts in Figure 2, we see that the C4.5 tree is unchanged, the CRUISE tree has an additional split (on *manuf*) and the GUIDE tree is much shorter. This behavior is not uncommon when there are many variables with little or no predictive power: their introduction can substantially reduce the size of a tree structure and its prediction accuracy; see, e.g., Ref 19 for more empirical evidence.

Table 3 reports the computational times used to fit the tree models on a computer with a 2.66 Ghz Intel Core 2 Quad Extreme processor. The fastest algorithm is C4.5, which takes milliseconds. If *manuf* is excluded, the next fastest is RPART, at a tenth of a second. But if *manuf* is included, RPART takes more than 3 h—a 10^5-fold increase. This is a practical problem with the CART algorithm; because *manuf* takes 31 values, the algorithm must search through $2^{30} - 1$

TABLE 1 | Comparison of Classification Tree Methods. A Check Mark Indicates Presence of a Feature

Feature	C4.5	CART	CHAID	CRUISE	GUIDE	QUEST
Unbiased Splits				√	√	√
Split Type	*u*	*u,l*	*u*	*u,l*	*u,l*	*u,l*
Branches/Split	≥2	2	≥2	≥2	2	2
Interaction Tests				√	√	
Pruning	√	√		√	√	√
User-specified Costs		√	√	√	√	√
User-specified Priors		√		√	√	√
Variable Ranking		√			√	
Node Models	*c*	*c*	*c*	*c,d*	*c,k,n*	*c*
Bagging & Ensembles					√	
Missing Values	*w*	*s*	*b*	*i,s*	*m*	*i*

b, missing value branch; *c*, constant model; *d*, discriminant model; *i*, missing value imputation; *k*, kernel density model; *l*, linear splits; *m*, missing value category; *n*, nearest neighbor model; *u*, univariate splits; *s*, surrogate splits; *w*, probability weights

TABLE 2 | Predictor Variables for the Car Data

Variable	Description	Variable	Description
manuf	Manufacturer (31 values)	*rev*	Engine revolutions per mile
type	Type (small, sporty, compact, midsize, large, van)	*manual*	Manual transmission available (yes, no)
minprice	Minimum price (in $1000)	*fuel*	Fuel tank capacity (gallons)
midprice	Midrange price (in $1000)	*passngr*	Passenger capacity (persons)
maxprice	Maximum price (in $1000)	*length*	Length (inches)
citympg	City miles per gallon	*whlbase*	Wheelbase (inches)
hwympg	Highway miles per gallon	*width*	Width (inches)
airbag	Air bags standard (0, 1, 2)	*uturn*	U-turn space (feet)
cylin	Number of cylinders	*rseat*	Rear seat room (inches)
enginzs	Engine size (liters)	*luggage*	Luggage capacity (cu. ft.)
hp	Maximum horsepower	*weight*	Weight (pounds)
rpm	Revolutions per minute at maximum horsepower	*domestic*	Domestic (U.S./non-U.S.manufacturer)

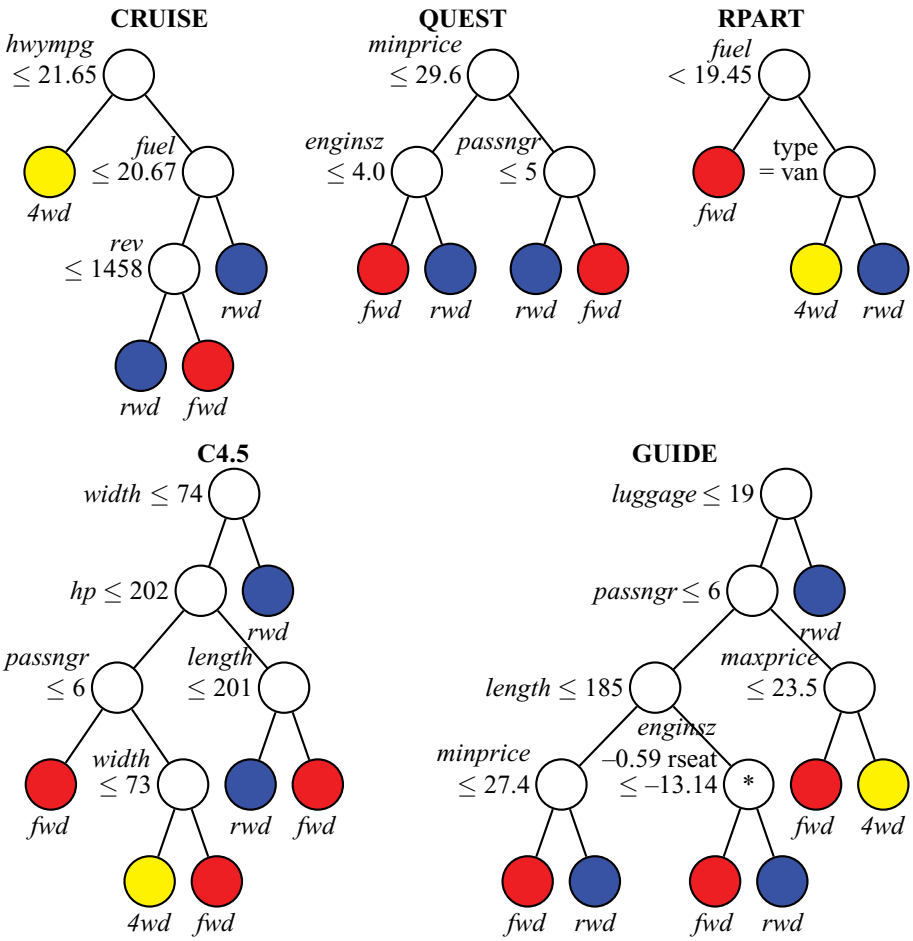

FIGURE 2 | CRUISE, QUEST, RPART, C4.5, and GUIDE trees for car data without *manuf*. The CHAID tree is trivial with no splits. At each intermediate node, a case goes to the left child node if and only if the condition is satisfied. The predicted class is given beneath each leaf node.

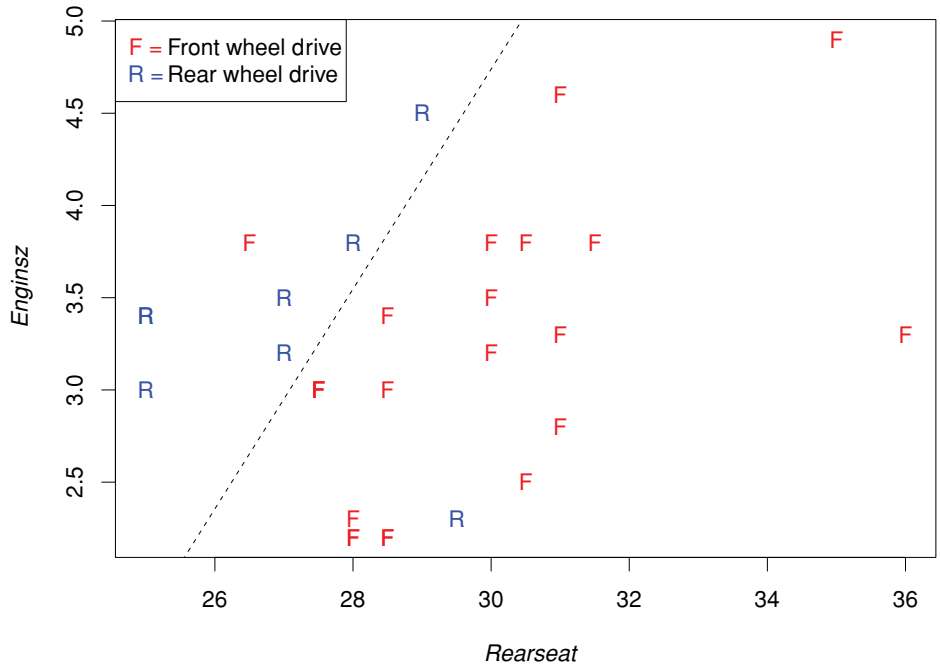

FIGURE 3 | Data and split at the node marked with an asterisk (*) in the GUIDE tree in Figure 2.

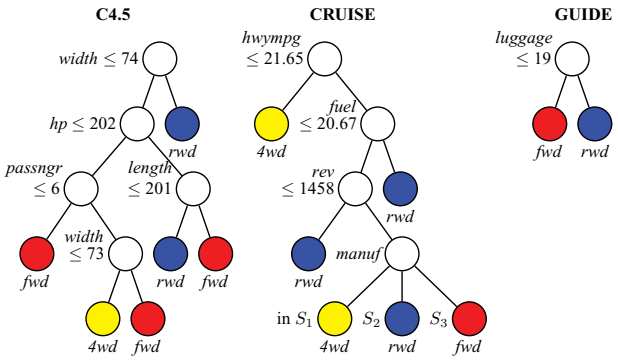

FIGURE 4 | CRUISE, C4.5, and GUIDE trees for car data with *manuf* included. The CHAID, RPART, and QUEST trees are trivial with no splits. Sets S_1 and S_2 are (Plymouth, Subaru) and (Lexus, Lincoln, Mercedes, Mercury, Volvo), respectively, and S_3 is the complement of $S_1 \cup S_2$.

(more than one billion) splits at the root node alone (if Y takes only two values, a computational shortcut[6] reduces the number of searches to just 30 splits). C4.5 is not similarly affected because it does not search for binary splits on unordered X variables. Instead, C4.5 splits the node into one branch for each X value and then merges some branches after the tree is grown. CRUISE, GUIDE, and QUEST also are unaffected because they search exhaustively for splits on unordered variables only if the number of values is small. If the latter is large, these algorithms employ a technique given in Ref 9 that uses linear discriminant analysis on dummy variables to find the splits.

Regression trees

A regression tree is similar to a classification tree, except that the Y variable takes ordered values and a regression model is fitted to each node to give the predicted values of Y. Historically, the first regression tree algorithm is AID,[20] which appeared several years before THAID. The AID and CART regression tree methods follow Algorithm 1, with the node impurity being the sum of squared deviations about the mean and the node predicting the sample mean of Y. This yields piecewise constant models. Although they are simple to interpret, the prediction accuracy of these models often lags behind that of models with more smoothness. It can be computationally impracticable, however, to extend this approach to piecewise linear models, because two linear models (one for each child node) must be fitted for every candidate split.

M5',[21] an adaptation of a regression tree algorithm by Quinlan,[22] uses a more computationally efficient strategy to construct piecewise linear models. It first constructs a piecewise constant tree and then fits a linear regression model to the data in each leaf node. Because the tree structure is the same as that of a piecewise constant model, the resulting trees tend to be larger than those from other piecewise linear tree

TABLE 3 Tree Construction Times on a 2.66 Ghz Intel Core 2 Quad Extreme Processor for the Car Data.

	C4.5	CRUISE	GUIDE	QUEST	RPART
Without *manuf*	0.004s	3.57s	2.49s	2.26s	0.09s
With *manuf*	0.003s	4.00s	1.86s	2.54s	3h 2m

methods. GUIDE[23] uses classification tree techniques to solve the regression problem. At each node, it fits a regression model to the data and computes the residuals. Then it defines a class variable Y' taking values 1 or 2, depending on whether the sign of the residual is positive or not. Finally, it applies Algorithm 2 to the Y' variable to split the node into two. This approach has three advantages: (1) the splits are unbiased; (2) only one regression model is fitted at each node; and (3) because it is based on residuals, the method is neither limited to piecewise constant models nor to the least squares criterion. Table 4 lists the main features of CART, GUIDE, and M5'.

To compare CART, GUIDE, and M5' with ordinary least squares (OLS) linear regression, we apply them to some data on smoking and pulmonary function in children.[24] The data, collected from 654 children aged 3–19 years, give the forced expiratory volume (FEV, in liters), gender (sex, M/F), smoking status (smoke, Y/N), age (years), and height (ht, in.) of each child. Using an OLS model for predicting FEV that includes all the variables, Kahn[25] found that smoke is the only one not statistically significant. He also found a significant age–smoke interaction if ht is excluded, but not if ht and its square are both included. This problem with interpreting OLS models often occurs when collinearity is present (the correlation between age and height is 0.8).

Figure 5 shows five regression tree models: (1) GUIDE piecewise constant (with sample mean of Y as the predicted value in each node), (2) GUIDE best simple linear (with a linear regression model involving only one predictor in each node), (3) GUIDE best simple quadratic regression (with a quadratic model involving only one predictor in each node), (4) GUIDE stepwise linear (with a stepwise linear regression model in each node), and (5) M5' piecewise constant. The CART tree (from RPART) is a subtree of (1), with six leaf nodes marked by asterisks (*). In the piecewise polynomial models (2) and (3), the predictor variable is found independently in each node, and nonsignificant terms of the highest orders are dropped. For example, for model (b) in Figure 5, a constant is fitted in the node containing females taller than 66.2 in. because the linear term for ht is not significant at the 0.05 level. Similarly, two of the three leaf nodes in model (c) are fitted with first-degree polynomials in ht because the quadratic terms are not significant. Because the nodes, and hence the domains of the polynomials, are defined by the splits in the tree, the estimated regression coefficients typically vary between nodes.

TABLE 4 Comparison of Regression Tree Methods. A Check Mark Indicates Presence of a Feature

Feature	CART	GUIDE	M5′
Unbiased Splits		√	
Split type	*u,l*	*u*	*u*
Branches/Split	2	2	≥2
Interaction Tests		√	
Pruning	√	√	√
Variable Importance Ranking	√	√	
Node Models	*c*	*c,m,p,r*	*c,r*
Missing Value Methods	*s*	*a*	*g*
Loss Criteria	*v*	*v,w*	*v*
Bagging & Ensembles		√	

a, missing value category; *c*, constant model; *g*, global mean/mode imputation; *l*, linear splits; *m*, multiple linear model; *p*, polynomial model; *r*, stepwise linear model; *s*, surrogate splits; *u*, univariate splits; *v*, least squares; *w*, least median of squares, quantile, Poisson, and proportional hazards.

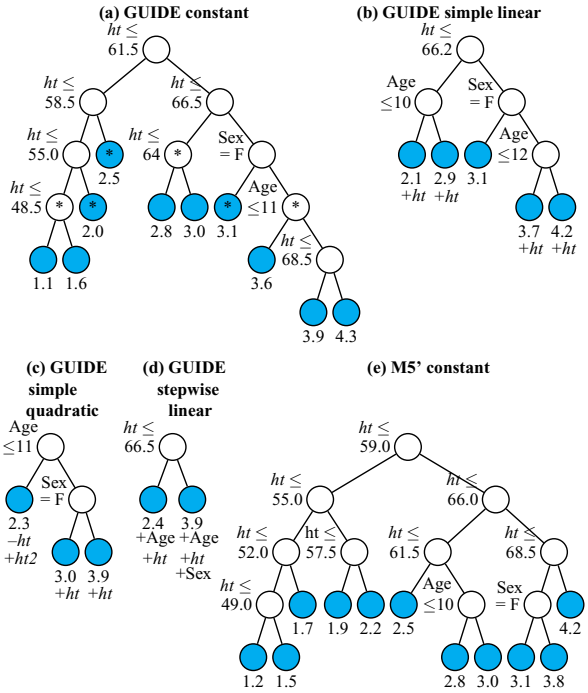

FIGURE 5 | GUIDE piecewise constant, simple linear, simple quadratic, and stepwise linear, and M5' piecewise constant regression trees for predicting FEV. The RPART tree is a subtree of (a), with leaf nodes marked by asterisks (*). The mean FEV and linear predictors (with signs of the coefficients) are printed beneath each leaf node. Variable *ht2* is the square of ht.

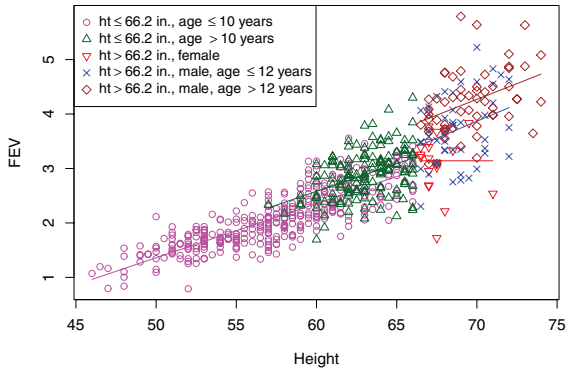

FIGURE 6 | Data and fitted regression lines in the five leaf nodes of the GUIDE piecewise simple linear model in Figure 5(b).

Because the total model complexity is shared between the tree structure and the set of node models, the complexity of a tree structure often decreases as the complexity of the node models increases. Therefore, the user can choose a model by trading off tree structure complexity against node model complexity. Piecewise constant models are mainly used for the insights that their tree structures provide. But they tend to have low prediction accuracy, unless the data are sufficiently informative and plentiful to yield a tree with many nodes. The trouble is that the larger the tree, the harder it is to derive insight from it. Trees (1) and (5) are quite large, but because they split almost exclusively on ht, we can infer from the predicted values in the leaf nodes that FEV increases monotonically with ht.

The piecewise simple linear (2) and quadratic (3) models reduce tree complexity without much loss (if any) of interpretability. Instead of splitting the nodes, ht now serves exclusively as the predictor variable in each node. This suggests that ht has strong linear and possibly quadratic effects. On the contrary, the splits on age and sex point to interactions between them and ht. These interactions can be interpreted with the help of Figures 6 and 7, which plot the data values of FEV and ht and the fitted regression functions with a different symbol and color for each node. In Figure 6, the slope of ht is zero for the group of females taller than 66.2 in., but it is constant and nonzero across the other groups. This indicates a three-way interaction involving age, ht, and sex. A similar conclusion can be drawn from Figure 7, in which there are only three groups, with the group of children aged 11 years or below exhibiting a quadratic effect of ht on FEV. For children above the age of 11 years, the effect of ht is linear, but males have, on average, about a half liter more FEV than females. Thus, the effect of sex seems to be due mainly to children older than 11 years. This conclusion is reinforced by the piecewise stepwise linear tree in Figure 5(d), in which a stepwise linear model is fitted to each of the two leaf nodes. Age and ht are selected as linear predictors in both leaf nodes, but sex is selected only in the node corresponding to children taller than 66.5 in., 70% of whom are above 11 years old.

Figure 8 plots the observed versus predicted values of the four GUIDE models and two OLS models containing all the variables, without and with the square of height. The discreteness of the predicted values from the piecewise constant model is obvious, as is the curvature in plot (e). The piecewise simple quadratic model in plot (c) is strikingly similar to plot (f), where the OLS model includes squared ht. This suggests that the two models have similar prediction accuracy. Model (3) has an advantage over model (6), however, because the former can be interpreted through its tree structure and the graph of its fitted function in Figure 7.

This example shows that piecewise linear regression tree models can be valuable in providing visual information about the roles and relative importance

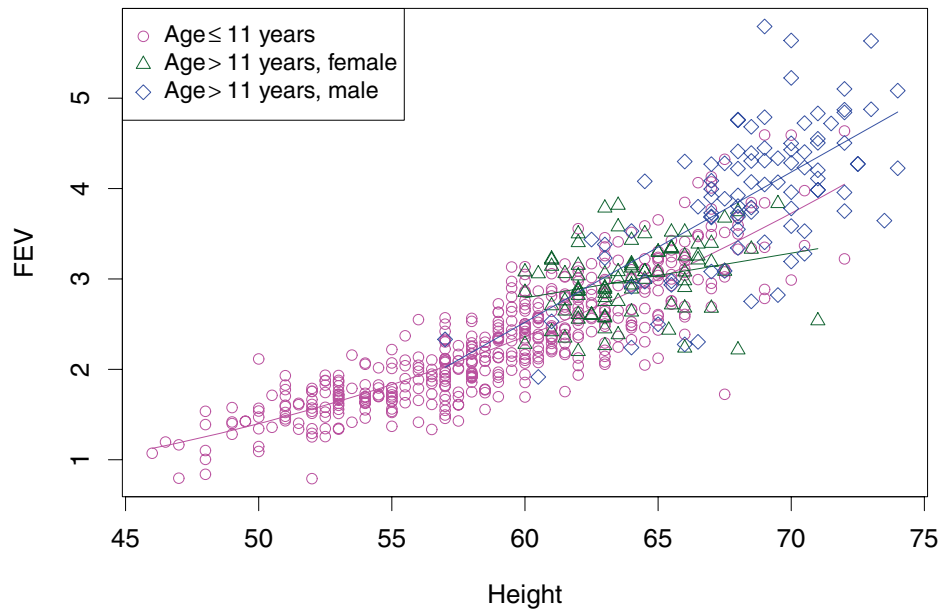

FIGURE 7 | Data and fitted regression functions in the three leaf nodes of the GUIDE piecewise simple quadratic model in Figure 5(c).

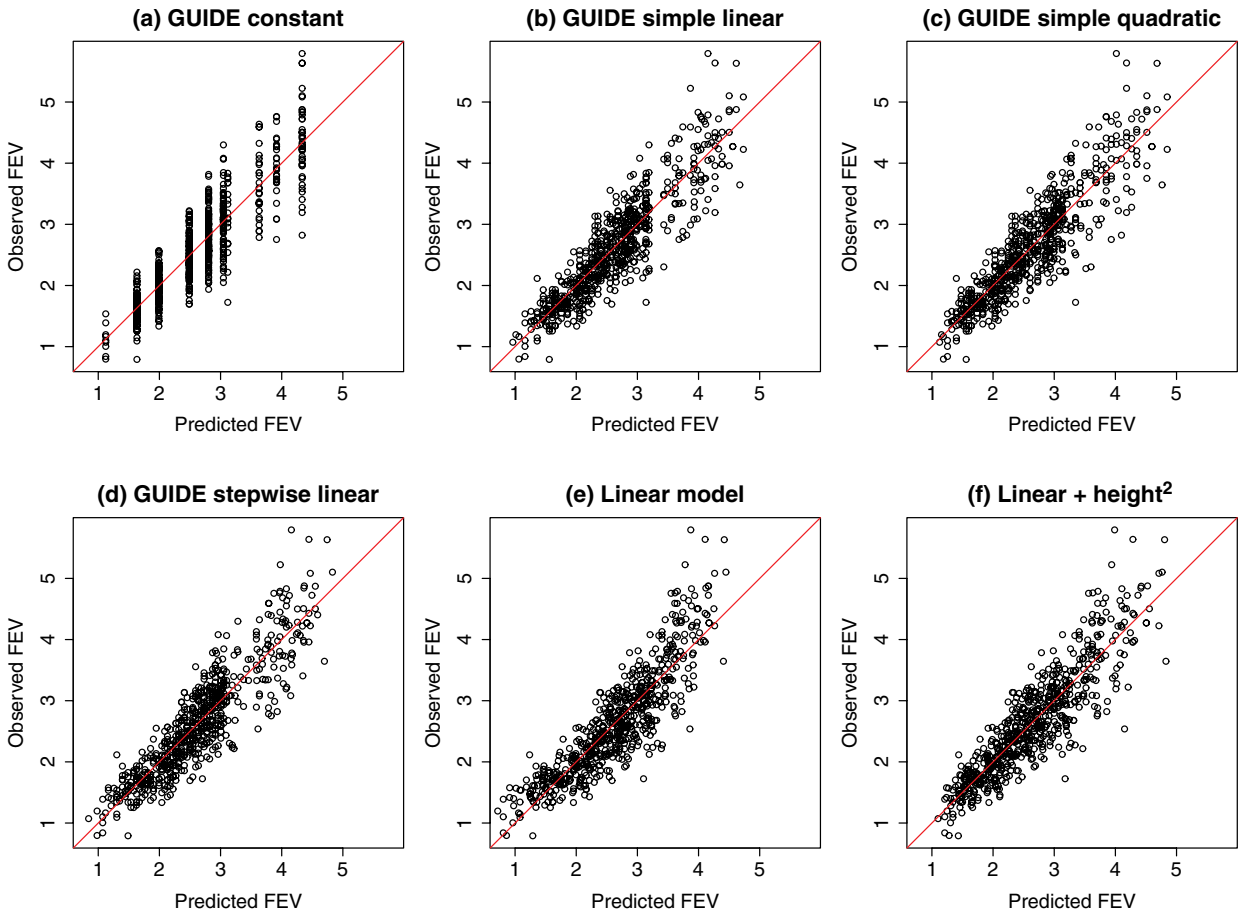

FIGURE 8 | Observed versus predicted values for the tree models in Figure 5 and two ordinary least squares models.

of the predictor variables. For more examples, see Refs 26 and 27.

CONCLUSION

On the basis of the published empirical comparisons of classification tree algorithms, GUIDE appears to have, on average, the highest prediction accuracy and RPART the lowest, although the differences are not substantial for univariate splits.[12] RPART trees often have fewer leaf nodes than those of CRUISE, GUIDE, and QUEST, whereas C4.5 trees often have the most by far. If linear combination splits are used, CRUISE and QUEST can yield accuracy as high as the best nontree methods.[10,11,28] The computational speed of C4.5 is almost always the fastest, whereas RPART can be fast or extremely slow, with the latter occurring when Y takes more than two values and there are unordered variables taking many values (owing to its coding, the RPART software may give incorrect results if there are unordered variables with more than 32 values). GUIDE piecewise linear regression tree models typically have higher prediction accuracy than piecewise constant models. Empirical results[29] show that the accuracy of the piecewise linear trees can be comparable to that of spline-based methods and ensembles of piecewise constant trees.

Owing to space limitations, other approaches and extensions are not discussed here. For likelihood and Bayesian approaches, see Refs 30 and 31, and Refs 32 and 33, respectively. For Poisson and logistic regression trees, see Refs 34 and 35, and Refs 36 and 37, respectively. For quantile regression trees, see Ref 38. For regression trees applicable to censored data, see Refs 39–43. Asymptotic theory for the consistency of the regression tree function and derivative estimates may be found in Refs 26, 34, and 44. The C source code for C4.5 may be obtained from *http://www.rulequest.com/Personal/*. RPART may be obtained from *http://www.R-project.org*. M5' is part of the WEKA[21] package at *http://www.cs.waikato.ac.nz/ml/weka/*. Software for CRUISE, GUIDE and QUEST may be obtained from *http://www/stat.wisc.edu/~loh/*.

NOTE

CART is a registered trademark of California Statistical Software, Inc.

REFERENCES

1. Fisher RA. The use of multiple measurements in taxonomic problems. *Ann Eugen* 1936, 7:179–188.
2. Cover T, Hart P. Nearest neighbor pattern classification. *IEEE Trans Inf Theory* 1967, 13:21–27.
3. Fielding A, O'Muircheartaigh CA. Binary segmentation in survey analysis with particular reference to AID. *The Statistician* 1977, 25:17–28.
4. Messenger R, Mandell L. A modal search technique for predictive nominal scale multivariate analysis. *J Am Stat Assoc* 1972, 67:768–772.
5. Quinlan JR. *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann; 1993.
6. Breiman L, Friedman JH, Olshen RA, Stone CJ. *Classification and Regression Trees*. CRC Press; 1984.
7. R Development Core Team, *R: a language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009.
8. Therneau TM, Atkinson B. *RPART: recursive partitioning*. R port by B. Ripley. R package version 3.1-41, 2008.
9. Loh WY, Vanichsetakul N. Tree-structured classification via generalized discriminant analysis (with discussion). *J Am Stat Assoc* 1988, 83:715–728.
10. Kim H, Loh WY. Classification trees with unbiased multiway splits. *J Am Stat Assoc* 2001, 96:589–604.
11. Kim H, Loh WY. Classification trees with bivariate linear discriminant node models. *J Comput Graphical Stat* 2003, 12:512–530.
12. Loh WY, Chen C, Hordle W, Unwin A, eds. Improving the precision of classification trees. *Ann Appl Stat* 2009, 3:1710–1737.
13. Loh WY, Shih Y. Split selection methods for classification trees. *Stat Sin* 1997, 7:815–840.
14. Hothorn T, Hornik K, Zeileis A. Unbiased recursive partitioning: a conditional inference framework. *J Comput Graphical Stat* 2006, 15:651–674.
15. Kass GV. An exploratory technique for investigating large quantities of categorical data. *Appl Stat* 1980, 29:119–127.

16. Breiman L. Bagging predictors. *Machine Learning* 1996, 24:123–140.
17. Breiman L. Random forests. *Machine Learning* 2001, 45:5–32.
18. Lock RH. 1993 New car data. *Journal of Statistics Education*, 1993. 1(1). *www.amstat.org/publications/jse/v1n1/datasets.lock.html*
19. Doksum K, Tang S, Tsui KW. Nonparametric variable selection: the EARTH algorithm. *J Am Stat Assoc* 2008, 103:1609–1620.
20. Morgan JN, Sonquist JA. Problems in the analysis of survey data, and a proposal. *J Am Stat Assoc* 1963, 58:415–434.
21. Witten I, Frank E. *Data Mining: practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann; 2005.
22. Quinlan JR. Learning with continuous classes. *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence* 1992, 343–348.
23. Loh WY. Regression trees with unbiased variable selection and interaction detection. *Stat Sin* 2002, 12:361–386.
24. Rosner B. *Fundamentals of Biostatistics*, 5th ed. Duxbury MA: Pacific Grove; 1999.
25. Kahn M. An exhalent problem for teaching statistics. *J Stat Education* 2005, 13(2). *www.amstat.org/publications/jse/v13n2/datasets.kahn.html*.
26. Kim H, Loh WY, Shih YS, Chaudhuri P. Visualizable and interpretable regression models with good prediction power. *IIE Trans* 2007, 39:565–579.
27. Loh WY. Regression by parts: fitting visually interpretable models with GUIDE. In: Chen C, Hordle W, Unwin A, eds. *Handbook of Data Visualization*. New York: Springer; 2008, 447–469.
28. Lim TS, Loh WY, Shih YS. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning* 2000, 40:203–228.
29. Loh WY, Chen CW, Zheng W. Extrapolation errors in linear model trees. *ACM Trans on Knowledge Discov from Data* 2007, 1(2).
30. Ciampi A. Generalized regression trees. *Comput Stat Data Anal* 1991, 12:57–78.
31. Su X, Wang M, Fan J. Maximum likelihood regression trees. *J Comput Graphical Stat* 2004, 13:586–598.
32. Chipman HA, George EI, McCulloch RE. Bayesian CART model search (with discussion). *J Am Stat Assoc* 1998, 93:935–960.
33. Denison DGT, Mallick BK, Smith AFM. A Bayesian CART algorithm. *Biometrika* 1998, 85:362–377.
34. Chaudhuri P, Lo WD, Loh WY, Yang CC. Generalized regression trees. *Stat Sin* 1995, 5:641–666.
35. Loh WY. Regression tree models for designed experiments. *IMS Lecture Notes-Monograph Series* 2006, 49:210–228.
36. Chan KY, Loh WY. LOTUS: an algorithm for building accurate and comprehensible logistic regression trees. *J Comput Graphical Stat* 2004, 13:826–852.
37. Loh WY. Logistic regression tree analysis. In: Pham H, ed. *Handbook of Engineering Statistics*. London: Springer; 2006, 537–549.
38. Chaudhuri P, Loh WY. Nonparametric estimation of conditional quantiles using quantile regression trees. *Bernoulli* 2002, 8:561–576.
39. Ahn H. Tree-structured exponential regression modeling. *Biometrical* 2007, 36:43–61.
40. Ahn H, Loh WY. Tree-structured proportional hazards regression modeling. *Biometrics* 1994, 50:471–485.
41. Cho HJ, Hong SM. Median regression tree for analysis of censored survival data. *IEEE Trans Syst, Man Cybern, Part A* 2008, 383:715–726.
42. LeBlanc M, Crowley J. Relative risk trees for censored survival data. *Biometrics* 1992, 48:411–424.
43. Chaudhuri P, Huang MC, Loh WY, Yao R. Piecewise-polynomial regression trees. *Stat Sin* 1994, 4:143–167.
44. Segal MR. Regression trees for censored data. *Biometrics* 1988, 44:35–47.

FURTHER READING

E. Alpaydin. *Introduction to Machine Learning*. 2nd ed. Boston: MIT Press; 2010.

