Tree-based methods

20 Mar 2023

Table of contents

Tree-based methods

- trees are a basic building block of modelign methods ($\sim$ linear regression)
- **greedy** partitioning of parameter space
- efficient updating rules instead of linear algebra
- better at categorical predictors, interactions, missing data
- bias-variance tradeoff, curse of dimensionality, need for hyperparameter tuning … **still apply**

Classification and regression trees

- recursive *binary* splitting
- builds basis of rectangular regions
  - predictions homogeneous within regions
  - could be expressed as indicator variables

CART: machinery

- splitting rule
  - regression: improve SSQ, deviance, …
  - improve misclassification error, Gini coefficient, deviance
  - Gini coefficient: $\sum_k \hat{p}_{mk}(1-\hat{p}_{mk})$ (weighted average $(1-p)$ loss)
  - deviance: $\sum \hat{p}_{mk}(-\log \hat{p}_{mk})$ (weighted average $-\log$ loss)

tree-splitting rule complexity

- only $O(Np)$!
- (more specifically $\sum(\#\text{unique } x_i)$)
- splits only happen at data point values

complexity pruning

- $1/N_m \sum_{x \in R_m} (y_i - \bar{y}_m)^2 + \alpha|T|$
- boils down to (total loss) + α size
- weakest-link pruning (greedy again): collapse least-useful splits first

categorical predictors

- to avoid combinatorial splitting problems, order categories by
  - frequency falling in outcome 1 (binary output)
  - mean response value
  - optimal split for Gini/deviance/cross-entropy/L2 loss
  - multicategory harder
- favors categorical vars with many categories ("such variables should be avoided" … ???)

loss matrix

- allow weighting of misclassification
- e.g. cost of false positive/negative, **or** value of sensitivity/specificity

missing predictor variables

- 'missing' category
- use **surrogate variables** (algorithm?? effects of other splitting variables are already computed?)
- is imputation better?

linear combination splits

- can do generalized discriminant analysis at each split
- weights, split point for $\sum a_j X_j \leq s$
- seems better (Loh and Vanichsetakul 1988) but Breiman and Friedman disagree (Breiman and Friedman 1988)
- highly empirical!

spam example

- 4600 messages, 57 predictors (48 word percentages; punctuation percentages; sequences of capitals)
- earlier: misclassification 7.6% from logistic regression, 5.5% from GAM
- CART: 9.3%
- weighted tree does slightly better at high specificity, but still $\ll$ GAM …

Loh, Wei-Yin, and Nunta Vanichsetakul. 1988. "Tree-Structured Classification via Generalized Discriminant Analysis." *Journal of the American Statistical Association* 83 (403): 715–25. https://doi.org/10.1080/01621459.1988.10478652.

Breiman, Leo, and Jerome H. Friedman. 1988. "Tree-Structured Classification Via Generalized Discriminant Analysis: Comment." *Journal of the American Statistical Association* 83 (403): 725–27. https://doi.org/10.2307/2289296.

316 9. Additive Models, Trees, and Related Methods

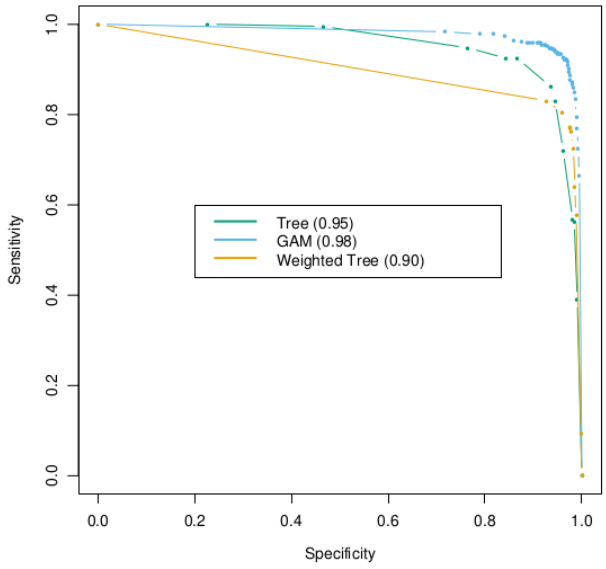


FIGURE 9.6. *ROC curves for the classification rules fit to the* `spam` *data. Curves that are closer to the northeast corner represent better classifiers. In this case the GAM classifier dominates the trees. The weighted tree achieves better sensitivity for higher specificity than the unweighted tree. The numbers in the legend represent the area under the curve.*

MARS

- like decision trees but piecewise linear (+ interactions) rather than constant
- **hinges** (or "reflected pairs"): 'mini-bases' (pairs of truncated linear spline terms) (total $2Np$)
- stepwise/stagewise fitting
- also include possible *interactions*
  - products of model terms with unused basis elements
  - at each step k we have $2Np \cdot (2k-1)$ choices to evaluate
  - each can be evaluated in $O(1)$
  - principle of **marginality**: only add higher-order terms if lower-order term is already present
- reasonably local basis functions (not as good as B-splines)
- hyperparameters: max interaction depth
- then prune greedily using GCV (denominator: $(1 - M/N)^2$, $M = r + cK$ where r=number of bases, K = number of knots

MARS on spam example

- still slightly worse than GAM …
- GCV chooses model with 60 basis functions

MARS vs CART

- stepwise vs piecewise linear basis
- presence of higher-order interactions

MARS with categorical predictors

- "all possible binary partitions" - really?
- (use ordering trick from CART?)

computational costs

- additive models via backfitting: $pN \log N$ (initial sort) $+mpN$ (backfitting)
  - cf. Np^2 for least-squares
- trees: $pN \log N$ to sort, $pN \log N$ for splits ($\log N$ cycles)
- MARS: $Nm^2 + pmN$ to add a basis function to a model with m terms $\rightarrow NM^3 + pM^2N$ (monitor stopping?)

missing data (with MARS/CART)

- **danger will robinson**
- MCAR, MAR, MNAR ...
- categorical: code as "missing"
- discard incomplete observations
- impute beforehand
- impute/handle as part of learning algorithm
  - impute based on mean/median
  - impute conditional on other observations (MICE)
  - trees: **surrogate splits** (easy to look for next-best split)

random forests

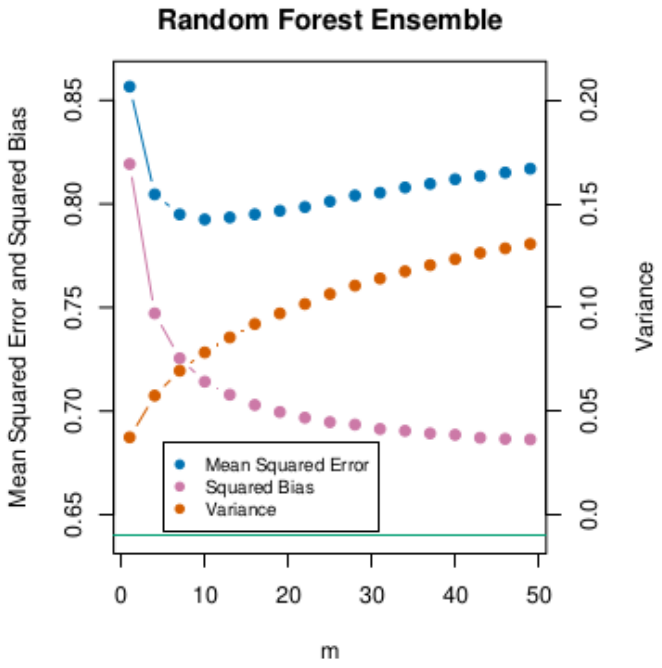
- bootstrap sample data
- grow a tree with a subset of m variables **at each split**
- average prediction from ensemble (mean prediction, or majority vote)
- variance of average of correlated variables $= \sigma^2(\rho + (1 - \rho)/B)$
  - $\rho \approx 0.05$ for bootstrapped **trees**
- subsetting variables reduces correlations between trees

tuning parameters

- suggested $m = \sqrt{p}$ for classification, min node size 1 (stopping point)
  - $p/3$, min node size 5 for regression
- min node size/max depth: "our experience is that using full-grown trees seldom costs much, and results in one less tuning parameter"
- number of trees just needs to be "large enough" (curve flattens quickly)
  - monitor progression for iterative algorithms?
- super-easy to parallelize
- explore tuning parameters: https://github.com/tidymodels/TMwR/issues/356
- out-of-bag samples

comparisons

- with many trees ($B \to \infty$), RF estimation variance shrinks to $\rho(x)\sigma^2(x)$
- correlation increases with m



- RF similar to ridge: shrinks strongly correlated variables toward each other

loss functions

- robustness
- how to pick???

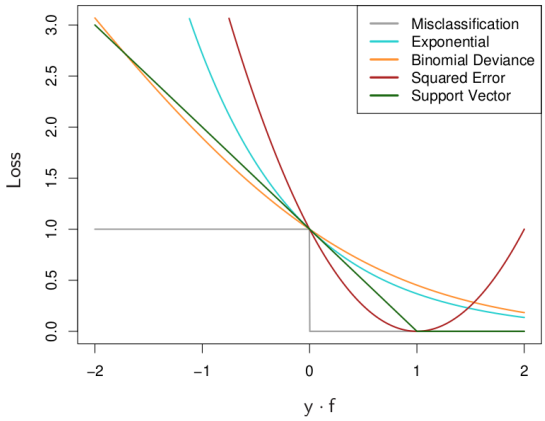


FIGURE 10.4. *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* f*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:* $(y - f)^2$*; and support vector:* $(1 - yf)_+$ *(see Section 12.3). Each function has been scaled so that it passes through the point* $(0, 1)$*.*

boosting

- "deep" methods
- still an additive model
- **stagewise** rather than simultaneous
- stagewise ≈ stepwise, but not recomputing previous coefficients/models

boosting: AdaBoost

- using {-1, 1} scores
- fit a classifier with current weights w_i
- compute **average error** (== avg weights of correct predictions)
- α_m = log-odds of average correctness
- weights of *incorrectly predicted* samples multiplied by odds of avg correctness
- prediction is $\text{sign}(\sum \alpha_m G_m(x))$
- can use probability mapped to [-1, 1] instead of classification

- corresponds to **exponential loss** $\exp(-yf)$; deviance is $\log(1+\exp(-2yf))$

boosting (generally)

- algorithm (stagewise):
  - fit a 'weak learner' to *pseudo-residuals*
  - update model based on the *sum* of the previous model plus the current weak learner
- pseudo-residuals: $-\partial L/\partial \hat{y}$
  - $= 2(y-\hat{y})$ for MSE
  - related to *generalized scoring* for GLMs etc.
  - == **gradient** of loss function
- weakest tree: "stump" (== "fork" ?)
- robust criteria don't give rise to fast algorithms

gradient boosting

- works for any differentiable loss function
- find gradient, line search

gradient tree boosting algorithm

(ESL, Bujokas (2022))

Bujokas, Eligijus. 2022. "Gradient Boosting in Python from Scratch." *Medium*. https://towardsdatascience.com/gradient-boosting-in-python-from-scratch-788d1cf1ca7.

- fit a decision tree (learner) to pseudo-residuals (= modeling the *gradient of the loss*)
- find the step size γ to apply to the new learner:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$
$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum L(y_i, F_{m-1}(x) + \gamma h_m(x))$$

where $h_m(x)$ is the new prediction for x

- for MSE $\gamma_m = \frac{1/N}{\sum}(h_m(x)(\hat{y}-y))/\sum(h_m^2)$

- for regression trees, h_m is **constant** in each region
- Friedman 'TreeBoost' optimizes γ for each region
- $F_m = F_{m-1} + \sum_j \gamma_{jm} 1_{Rjm}(x)$
- γ_{jm} reduces to the mean $(\hat{y} - y)$ for MSE

hyperparameters

- tree size
  - 2 = 'stump'
  - stumps == **additive** models
  - J determines maximum interaction depth
  - ESL say $4 \leq J \leq 8$ is good, rarely > 10, ≈ 6 usually OK
- early stopping (M) ("how many iterations without an improvement in the objective function occur before training should be halted") (Prechelt 2012)
- learning rate/**shrinkage**
- stochastic boosting

Prechelt, Lutz. 2012. "Early Stopping - but When?" In *Neural Networks: Tricks of the Trade*, edited by Grégoire Montavon and Klaus-Robert Müller, 53–67. Lecture Notes in Computer Science. http://page.mi.fu-berlin.de/~prechelt/Biblio/stop_tricks1997.pdf.

extreme gradient boosting

- use a "Newton" step (**elementwise** second-order approximation)
  - compute gradient and curvature of L wrt $\hat{y}$
  - irrelevant/same as gradient descent for MSE, Huber loss, L1 loss …
- (Sigrist 2018; Cho 2018)
- Implementation-dependent stuff? Sparsity, out-of-memory implementation, etc.

Sigrist, Fabio. 2018. "Gradient and Newton Boosting for Classification and Regression." *arXiv.org*. https://doi.org/10.48550/arXiv.1808.03064.

Cho, Philip Hyunshu. 2018. "Does Xgboost Do Newton Boosting?" *GitHub*. https://github.com/dmlc/xgboost/issues/3227.

stochastic boosting

- subsample data at each stage (e.g. $\eta = 0.5$)

stagewise modeling

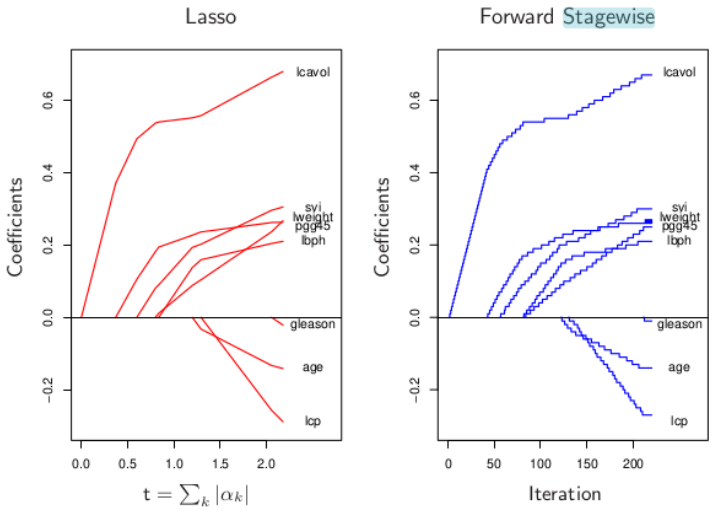


FIGURE 16.1. *Profiles of estimated coefficients from linear regression, for the prostate data studied in Chapter 3. The left panel shows the results from the lasso, for different values of the bound parameter* $t = \sum_k |\alpha_k|$. *The right panel shows the results of the stagewise linear regression Algorithm 16.1, using* $M = 220$ *consecutive steps of size* $\varepsilon = .01$.

BART

- Chipman, George, and McCulloch (2010)
- again a sum of trees
- smart, data-informed ("empirical Bayesian") priors
- fitting procedure is MCMC
- defaults are good, don't need much tuning

Chipman, Hugh A., Edward I. George, and Robert E. McCulloch. 2010. "BART: Bayesian Additive Regression Trees." *The Annals of Applied Statistics* 4 (1). https://doi.org/10.1214/09-AOAS285.

regularizing priors

- informative but non-biasing
- shrink toward 'null value', e.g. ridge/lasso penalties

BART priors

- tree depth, splitting point, overall residual variance
- componentwise/independent priors on each model
- tree depth: $\alpha(1+d)^{-\beta}$
  - $\alpha = 0.95$, $\beta = 2$: modal $d = 2$, most $d \leq 5$
- splitting priors: uniform over split points (observed values) and parameters
- split point/mean: scaled Normal prior
  - overall mean prediction is sum of means across m trees
  - $m\mu_\mu \pm k\sqrt{m}\sigma_\mu = y_{\min}$
  - e.g. $k = 2$ for $\approx 95\%$ range in (ymin, ymax)
- residual variance: inverse-chi-squared with scale matching σ
  - $\hat{\sigma}$ is marginal std dev *or* linear regression std dev
  - choose df (3-10) so that upper q quantile is at $\hat{\sigma}$
- overfitting is unlikely, so choose m "big enough" ($m = 200$)

MCMC rules

- **Gibbs sampling**
- draw σ from inverse gamma (conjugate prior)
- T_j: grow a terminal node (0.25), prune (0.25), change a nonterminal rule (0.4), swap a rule between parent and child (0.1)
- resample M_j values from a normal (and recompute residuals)
- initialize with m stumps

MCMC stuff

- burn-in (typically 200 steps)
- sampling
- single chain (typically 1000 steps)