Lectures 17,18 – Boosting and Additive Trees

Rice ECE697
Farinaz Koushanfar
Fall 2006

Summary

- Bagging and ensemble learning
- Boosting – AdaBoost.M1 algorithm
- Why boosting works
- Loss functions
- Data mining procedures
- Example – spam data

Bagging (Bootstrap Aggregation)

- Training set $D=\{(x_1,y_1),\ldots,(x_N,y_N)\}$
- Sample S sets of N elements from D (with replacement): $D_1, D_2, \ldots, D_S$
- Train on each D_s, s=1,..,S and obtain a sequence of S outputs $f_1(X),..,f_S(X)$
- The final classifier is:

$$\bar{f}(X) = \sum_{s=1}^{S} f_s(X) \qquad \text{Regression}$$

$$\bar{f}(X) = \theta\left(\sum_{s=1}^{S} \text{sign}(f_s(X))\right) \qquad \text{Classification}$$

Bagging: Variance Reduction

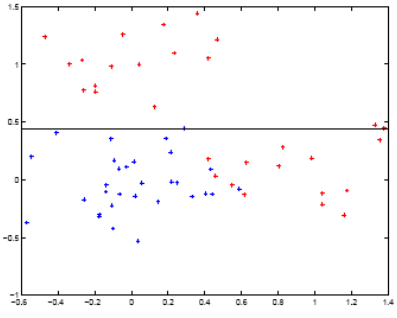
- If each classifier has a high variance (unstable) the aggregated classifier has a smaller variance than each single classifier
- The bagging classifier is like an approximation of the true average computed by replacing the probability distribution with bootstrap approximation

Measuring Bias and Variance in Practice

- Bias and Variance are both defined as expectations:
  - Bias (X) = E_p[f(X)-fbar(X)]
  - Var(X) = EP[(f(X)-fbar(X))^2]
- It is easy to see why bagging reduces variance – averaging
- Bagging is a simple example of an ensemble learning algorithm
- **Ensemble learning:** combine the prediction of different hypothesis by some sort of voting

Boosting

- An ensemble-learning method
- One of the most powerful learning ideas introduced in the past 10+ years
- A procedure that combines many weak classifiers to produce a **powerful committee**



Example: weak learner
T. Jaakkola, MIT

Boosting (Cont'd)

- In an ensemble, the output of an instance is computed by averaging the output of several hypothesis
- Choose the individual classifiers and their ensembles to get a good fit
- Instead of constructing the hypothesis independently, construct them such that new hypothesis focus on instance that were problematic for the previous hypothesis
- Boosting implements this idea!

Main Ideas of Boosting

- New classifiers should focus on difficult cases
  - Examine the learning set
  - Get some "rule of thumb" (weak learner ideas)
  - Reweight the examples of the training set, concentrate on "hard" cases for the previous rule
  - Derive the next rule of thumb!
  - ….
  - Build a single, accurate predictor by combining the rules of thumb
- Challenges: how to reweight? How to combine?

Boosting (Cont'd)

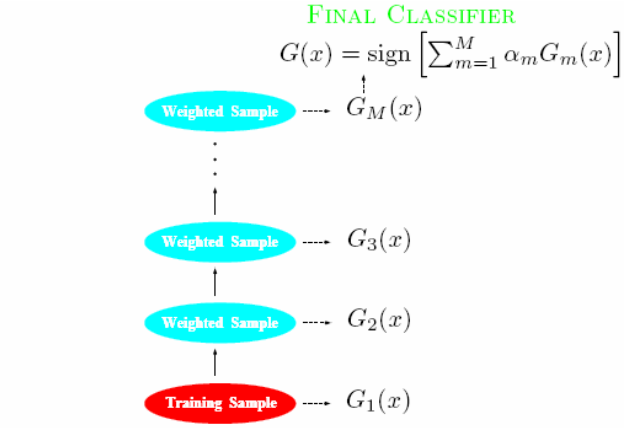


Figure 10.1: *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Ada Boost.M1

- The most popular boosting algorithm – Fruend and Schapire (1997)
- Consider a two-class problem, output variable coded as Y $\in$ {-1,+1}
- For a predictor variable X, a classifier G(X) produces predictions that are in {-1,+1}
- The error rate on the training sample is

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$$

Ada Boost.M1 (Cont'd)

- Sequentially apply the weak classification to repeatedly modified versions of data
- → produce a sequence of weak classifiers $G_m(x)$ m=1,2,..,M
- The predictions from all classifiers are combined via majority vote to produce the final prediction

FINAL CLASSIFIER

$$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

Algorithm AdaBoost.M1

1. Initialize the observ. weights $w_i^{(1)} = 1/N, i = 1, \ldots, N$.
2. For $m = 1$ to M
   - Fit classifier $G_m(x)$ to the training data using weights $w_i^{(m)}$.
   - Compute $err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$.
   - Compute $\alpha_m = \log((1 - err_m)/err_m)$.
   - $w_i^{(m+1)} = w_i^{(m)} \exp[\alpha_m I(y_i \neq G_m(x_i))], i = 1, \ldots, N$.
3. Compute $G(x) = sign(\sum_{i=1}^{M} \alpha_m G_m(x))$.

Some slides borrowed from http://www.stat.ucl.ac.be/

Example: Adaboost.M1

- The features X1,..,X10 are standard independent Gaussian, the deterministic target is

$$Y = \begin{cases} +1 & \text{if } \sum X_j^2 > \chi_{10}^2(0.5) \\ -1 & \text{otherwise} \end{cases}$$

=9.34 is the median of the chi-square RV with 10 DF

- 2000 training cases, with approximately 1000 cases in each class and 10,000 test observations
- Weak classifier: a two-terminal node tree
- The weak classifiers produce around 46% correct guesses

Example: Adaboost.M1 (Cont'd)

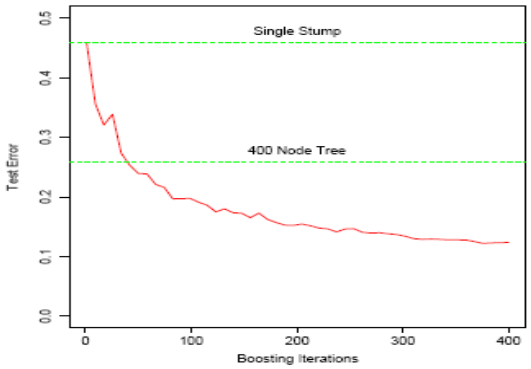


Figure 10.2: *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 400 node classification tree.*

Boosting Fits an Additive Model

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

where $b(x; \gamma_m) = G_m(x) \in \{-1, 1\}$ (for Adaboost) is like a set of elementary "basis functions"

This model is fit by minimizing a loss function L averaged over the training data set:

$$\min_{\{\beta_m, \gamma_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m)\right)$$

Source http://www.stat.ucl.ac.be/

Forward Stagewise Additive Modeling

- An approximate solution to the minimization problem is obtained via forward stagewise additive modeling (greedy algorithm)

1. Initialize $f_0(x) = 0$
2. For $m = 1$ to M
   - Compute
     $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
   - Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Source http://www.stat.ucl.ac.be/

Why adaBoost Works?

- Adaboost is a forward stagewise additive algorithm using the loss function

$$L(y; f(x)) = \exp(-yf(x))$$

with

$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} \exp\left(-y_i(f_{m-1}(x_i) + \beta G(x_i)\right)$

or $(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\beta y_i G(x_i)\right)$

where $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$.

Source http://www.stat.ucl.ac.be/

Why Boosting Works? (Cont'd)

The solution is :

$$G_m = \arg\min_G \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)),$$

$$\beta_m = 1/2 \log \frac{1 - err_m}{err_m},$$

where $err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$.

Source http://www.stat.ucl.ac.be/

Loss Function

A. Exponentional loss $\leftrightarrow$ Binomial negative log-likelihood

Till now

$$f^*(x) = \arg\min_{f(x)} E_{Y|x}(\exp(-Yf(x))) = \frac{1}{2}\log\frac{P(Y=1|x)}{P(Y=-1|x)}$$

$$P(Y=1|x) = \frac{1}{1+\exp(-2f^*(x))}$$

$\rightarrow$ Adaboost is estimating one-half the log-odds of $P(Y=1|x)$. This justifies using its sign as a classification rule.

But the deviance loss criterion admits the same population minimizer interpreting $p(x)$ as in a logistic model.

Source http://www.stat.ucl.ac.be/

Loss Function (Cont'd)

$p(x) = P(Y=1|x) = \frac{\exp(f(x))}{\exp(f(x))+\exp(-f(x))} = \frac{1}{1+\exp(-2f(x))}$.

With $Y' = \frac{Y+1}{2} \in \{0,1\}$,

$$\begin{aligned} l(Y,p(x)) &= Y'\log(p(x)) + (1-Y')\log(1-p(x)), \\ -l(Y,f(x)) &= \log(1+\exp(-2Yf(x))). \end{aligned}$$

Thus

$$\arg\min_{f(x)} E_{Y|x}[-l(Y,f(x))] = \arg\min_{f(x)} E_{Y|x}[\exp(-Yf(x))].$$

Source http://www.stat.ucl.ac.be/

Loss Function (Cont'd)

- Y.f(X) is called the Margin
- In classifications with 1/-1, margin is just like squared error loss (Y-f(X))
- The classification rule implies that observations with positive margin $y_i f(x_i)>0$ were classified correctly, but the negative margin ones are incorrect
- The decision boundary is given by the f(X)=0
- The loss criterion should penalize the negative margins more heavily than the positive ones

Loss Function (Cont'd)

NB: With $K-$class classification, the response Y takes values in the set $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_K\}$. Now

$$G(x) = \mathcal{G}_k \text{ where } k = \arg\max_l p_l(x)$$

With the logistic model, $p_l(x) = \frac{\exp(f_k(x))}{\sum_{l=1}^{K} \exp(f_l(x))}$. Then

$$\begin{aligned} l(y, p(x)) &= -\sum_{k=1}^{K} I(y = \mathcal{G}_k) \log p_k(x) \\ &= -\sum_{k=1}^{K} I(y = \mathcal{G}_k) f_k(x) + \log\left(\sum_{l=1}^{K} \exp(f_l(x))\right) \end{aligned}$$

Source http://www.stat.ucl.ac.be/

Loss Functions for Two-Class Classification

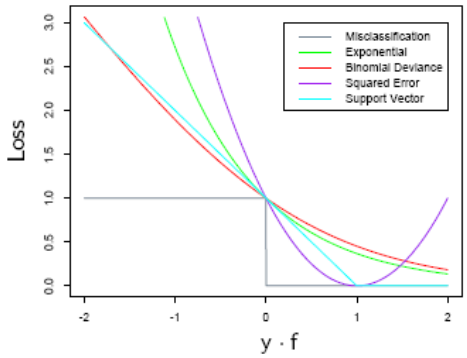


Figure 10.4: *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* f*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:*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

Loss Functions (Cont'd)

C. Loss functions for regression

Classification:	exponential loss	$\leftrightarrow$	deviance
Regression :	squared error loss	$\leftrightarrow$	absolute loss
$L(y, f(x))$:	$(y - f(x))^2$	$\leftrightarrow$	$\lvert y - f(x)\rvert$
Minimizer :	$E(Y\lvert x)$	$\leftrightarrow$	Median $(Y\lvert x)$

They are identical for symmetric distribution but lack of robustness for squared error loss.

Compromise between robustness and efficiency : Huber

$$L(y, f(x)) = \begin{cases} (y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta(|y - f(x)| - \delta/2) & \text{otherwise.} \end{cases}$$

Source http://www.stat.ucl.ac.be/

Loss Function (Comparison)

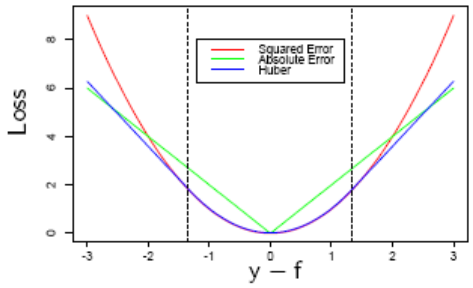


Figure 10.5: *A comparison of three loss functions for regression, plotted as a function of the margin $y - f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y - f|$ is large.*

Data Mining

Data mining requires to deal with :

- Messy data
- Missing values
- Long tailed or highly skewed distributions of the variables
- Outliers
- Predictor variables measured on different scales
- Irrelevant predictor variable

Source http://www.stat.ucl.ac.be/

Data Mining (Cont'd)

- interpretable models
- computation speed

Decision trees come closest to meeting those requirements.
But decision trees are not accurate $\Rightarrow$ solution : use boosting decision trees but some advantages are sacrified like speed, interpretability and robustness.

A multiple additive regression tree is a generalization of tree boosting that attempts to mitigate these problems.

Source http://www.stat.ucl.ac.be/

Some Characteristics of Methods

Characteristic	Neural nets	SVM	Trees	MARS	k-NN, kernels
Natural handling of data of "mixed" type	●	●	●	●	●
Handling of missing values	●	●	●	●	●
Robustness to outliers in input space	●	●	●	●	●
Insensitive to monotone transformations of inputs	●	●	●	●	●
Computational scalability (large N)	●	●	●	●	●
Ability to deal with irrelevant inputs	●	●	●	●	●
Ability to extract linear combinations of features	●	●	●	●	●
Interpretability	●	●	●	●	●
Predictive power	●	●	●	●	●

Table 1: *Some characteristics of different learning methods. Key:* ●*= good,* ●*=fair, and* ●*=poor.*

Spam Data

$$f(x) = \log \frac{Pr(\text{Spam}|x)}{Pr((Email)|x)}$$

- with $J = 2$ terminal node trees : error rate 4.6 %
- with full MART model : error rate 4%

$\Rightarrow$ interactions

Source http://www.stat.ucl.ac.be/

Spam Data – Importance Spectrum

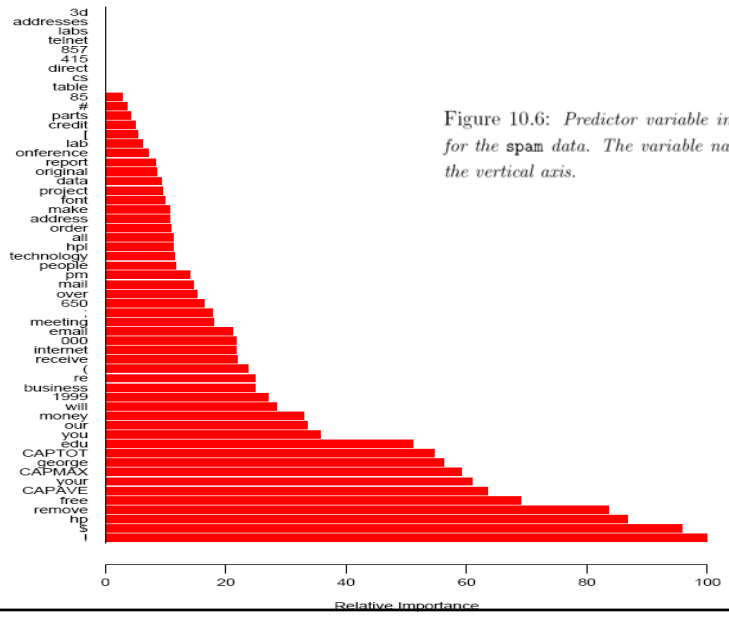


Figure 10.6: *Predictor variable importance spectrum for the* spam *data. The variable names are written on the vertical axis.*

Spam Data – Partial Dependences

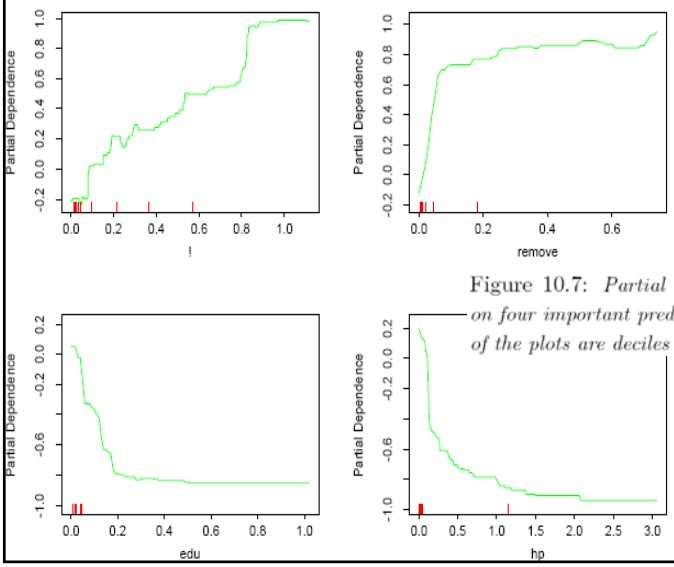


Figure 10.7: *Partial dependence of log-odds of* `spam` *on four important predictors. The red ticks at the base of the plots are deciles of the input variable.*

Spam - 2D Partial Dependency

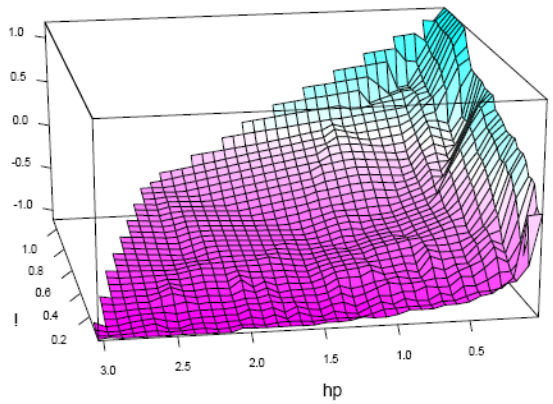


Figure 10.8: *Partial dependence of the log-odds of* `spam` *vs.* `email` *as a function of joint frequencies of* `hp` *and* `!`.

Boosting trees

- interpretable models
- computation speed

Decision trees come closest to meeting those requirements.
But decision trees are not accurate ⇒ solution : use boosting decision trees but some advantages are sacrified like speed, interpretability and robustness.

A multiple additive regression tree is a generalization of tree boosting that attempts to mitigate these problems.

Trees Reviewed!

- Partition of the joint predictor values into disjoint regions R_j, j=1,..,J represented by the terminal nodes
- A constant γ_j is assigned to each region,
- The predictive rule is: $x \in R_j \rightarrow f(x)=\gamma_j$
- The tree is: $T(x;\Theta) = \Sigma_J \gamma_j I(x \in R_j)$
- With parameters $\{R_j, \gamma_j\}$; j=1,…,J
- We find the parameters by minimizing the empirical risk

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j)$$

Optimization problem on Trees

- Finding γ_j given Rj: this is easy
- Finding Rj: this is difficult, we typically approximate. We have talked about the greedy top-down recursive partitioning algorithm
- We have previously defined some smoother approximate loss criterion for growing tree that are easier to work with
- A boosted tree, is sum of such trees,

$$\Rightarrow f_M(x) = \sum_{m=1}^{M} T(x;\theta_m)$$

Boosting Trees

A. Boosting trees

$$T(x;\theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

$$\Rightarrow f_M(x) = \sum_{m=1}^{M} T(x;\theta_m)$$

$\rightarrow$ forward stagewise additive modeling

$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} \tilde{L}(y_i, f_{m-1}(x) + T(x_i;\theta_m))$,

with $\theta_m = \{R_{j,m}, \gamma_{j,m}\}_{j=1}^{J_m}$.

Source http://www.stat.ucl.ac.be/

Boosting Trees (cont'd)

- Finding the regions is more difficult than before
- For a few cases, the problem might simplify!

Given the regions $R_{j,m}$,

$$\hat{\gamma}_{j,m} = \arg\min_{\gamma_{j,m}} \sum_{x_i \in R_{j,m}} L(y_i, f_{m-1}(x) + \gamma_{j,m})$$

Problem : Robust criterion does not give rise to simple fast boosting algorithms.

Boosting Trees (Cont'd)

- For squared error regression, solution is similar to single tree
  - Find the regression tree than best predicts the current residuals y_i-$f_{m-1}(x_i)$ and γ_j is the mean of these residuals in each corresponding region
- For classification and exponential loss, it is the AdaBoost for boosting trees (scaled trees)
  - Find the tree that minimizes the weighted error, with weights $w^{(m)}_i$ defined as before for boosting

Numerical Optimization

- Loss function in using prediction f(x) for y is

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$$

- The goal is to minimize L(f) w.r.t f, where f is the sum of the trees. Ignoring this, we can view minimizing as a numerical optimization $f^{\wedge}=\text{argmin}_f L(f)$
- Where the parameters f are the values of the approximating function $f(x_i)$ at each of the N data points x_i: $f=\{f(x_1),\ldots,f(x_N)\}$,

$$f_M = \sum_{m=1}^{M} h_m, \quad h_m \in \Re^N$$

- Solve it as a sum of component vectors, where $f_0=h_0$ is the initial guess and each successive f_m is induced based on the current parameter vector f_{m-1}

Steepest Descent

1. Choose $h_m = -\rho_m g_m$, where ρ_m is a scalar and g_m is the gradient of L(f) evaluated at $f=f_{m-1}$

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}.$$

2. The step length ρ_m is the solution to

   $\rho_m=\text{argmin}_\rho L(f_{m-1}- \rho g_m)$
3. The current solution is then updated:

   $f_m=f_{m-1}- \rho_m g_m$

Source http://www.stat.ucl.ac.be/

Gradient Boosting

- Forward stagewise boosting is also a very greedy algorithm
- The tree predictions can be thought about like negative gradients
- The only difficulty is that the tree components are not independent

$$t_m = (T(x_1; \theta_m), \ldots, T(x_N; \theta_m))$$

- Search for t_m's corresponding to $\{T(x_i;\Theta_m)\}$ for $x_i \in R_{jm}$
- They are constrained to be the predictions of a J_m-terminal node decision tree, whereas the negative gradient is unconstrained steepest descent
- Unfortunately, the gradient is only defined at the training data points and is not applicable to generalizing $f_M(x)$ to new data
- See Table 10.2 for the gradients of commonly used loss functions!

Gradient Boosting

Ultimate goal: generalize $f_M(x)$ to new data points.
So at the m^{th} iteration, induce a regression tree to the negative gradient :

$$\tilde{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} (-g_{im} - T(x_i; \theta))^2.$$

For K-class classification, K L.S. regression trees are constructed.

$g_{im} = [\frac{\partial L(y_i, f_{1,m}(x_i), \ldots, f_{K,m}(x_i))}{\partial f_{km}(x_i)}] = I(y_i = \mathcal{G}_k) - p_k(x_i)$,
and $p_k(x) = \frac{\exp(f_k(x))}{\sum_{l=1}^{K} \exp(f_l(x))}$.

Source http://www.stat.ucl.ac.be/

Multiple Additive Regression Trees (MART)

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   - For $i = 1, 2, \ldots, N$ compute $g_{im} = [\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f=f_{m-1}}$.
   - Fit a regr. tree to $-g_{im}$ giving terminal regions $R_{j,m}$.
   - For $j = 1, 2, \ldots J_m$:
     $\gamma_{j,m} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(y_i, f_{m-1}(x_i) + \gamma)$.
   - Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{j,m})$.
3. Output $\hat{f}(x) = f_M(x)$.

Source http://www.stat.ucl.ac.be/

MART (Cont'd)

C. Right-sized trees for boosting

Problem: Trees tend to be much too large ($J = \#$ terminal nodes) $\Rightarrow$ Loss in accuracy and computing time.

In fact, the target function $\eta = \arg\min_f E_{XY} L(Y, f(X))$ can be written in terms of interactions

$$\eta(X) = \sum_j \eta_j(X_j) + \sum_{jk} \eta_{jk}(X_j, X_k) + \sum_{jkl} \eta_{jkl}(X_j, X_k, X_l) + \ldots$$

No interaction effects of level greater than $J - 1$ are possible $\rightarrow$ fixed J (good choice $4 \leq J \leq 8$) .

Source http://www.stat.ucl.ac.be/

MART (Cont'd)

- Besides the size of each tree J, the other meta parameter of MART is M, the number of boosting iterations
- Each iterations reduces the training risk, for M large enough training risk can be made small
  - May lead to overfitting
- Like before, we may need shrinkage!

MART (Cont'd)

D. Shrinkage

$$f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J} \hat{\gamma}_{j,m} I(x \in R_{j,m})$$

ν controls the learning rate.

Smaller values of ν lead to large values of M.

Source http://www.stat.ucl.ac.be/

Penalized Regression

- Consider the set of all possible J-terminal trees $\Im=\{T_k\}$, $K=|\Im|$ that is realized on training data as basis functions in $\Re^p$, linear model is

$$f(x) = \sum_{k=1}^{K} \alpha_k T_k(x)$$

- Penalized least square is required, where $\alpha$is a vector of parameters and $J(\alpha)$ is penalizer!

$$\hat{\alpha}(\lambda) = \arg\min_{\alpha} \{\sum_{i=1}^{N} (y_i - \sum_{k=1}^{K} \alpha_k T_k(x_i))^2 + \lambda J(\alpha)\}$$

- Since we have a large number of basis functions, solving with lasso is not possible
- Algorithm 10.4 is proposed instead

Regularization: Boosting with different sized trees

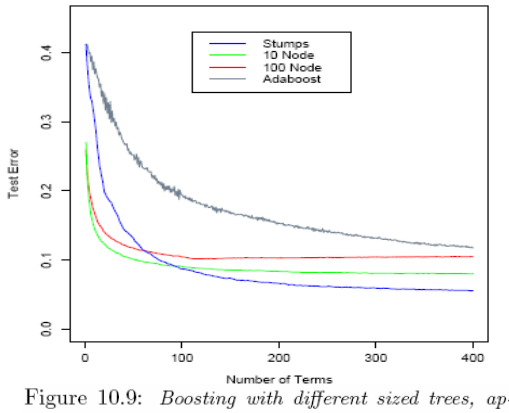


Figure 10.9: *Boosting with different sized trees, applied to the example (10.2) used in Figure 10.2. Since the generative model is additive, stumps perform the best. The boosting algorithm used the binomial deviance loss in Algorithm 10.3; shown for comparison is the AdaBoost algorithm 10.1.*

MART (Example)

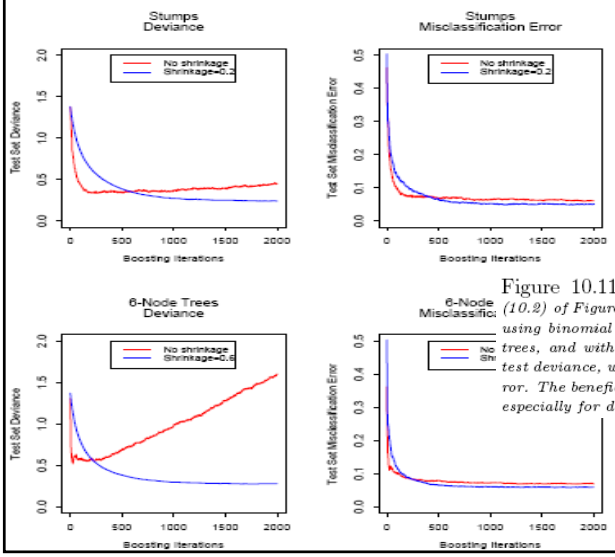


Figure 10.11: *Test error curves for simulated example (10.2) of Figure 10.9, using MART. The models were trained using binomial deviance, either stumps or six terminal-node trees, and with or without shrinkage. The left panels report test deviance, while the right panels show misclassification error. The beneficial effect of shrinkage can be seen in all cases, especially for deviance in the left panels.*

Lasso vs. Stagewise linear regression

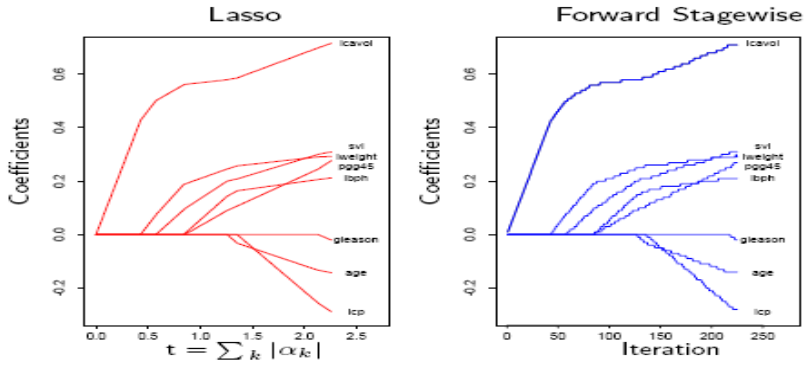


Figure 10.12: *Profiles of estimated coefficients from linear regression, for the prostate data studied in Chapter 3. The left panel shows the results from the lasso, for different values of the bound parameter* $t = \sum_k |\alpha_k|$. *The right panel shows the results of the stagewise linear regression algorithm 10.4, using* $M = 250$ *consecutive steps of size* $\varepsilon = .01$.

Interpretation

- Single decision trees are often very interpretable
- Linear combination of trees loses this important feature
- We often learn the relative importance or contribution of each input variable in predicting the response
- Define a measure of relevance for each predictor X_l, sum over the J-1 internal nodes of the tree

Interpretation (Cont'd0

A. Relevance of a predictor X_l.
In a single tree:

$$I_l^2(T) = \sum_{t=1}^{J-1} \hat{i}_t^2 I(v(t) = l),$$

with $\hat{i}_t^2$ maximal estimated improvement in squared error risk.
In a boosting tree:

$$I_l^2 = \frac{1}{M} \sum_{m=1}^{M} I_l^2(T_m).$$

Source http://www.stat.ucl.ac.be/

Interpretation (Cont'd)

For K-class classification:

$$I_{lk}^2 = \frac{1}{M}\sum_{m=1}^{M} I_l^2(T_{k,m}),$$

is the relevance of X_l in separating the class k from the other classes.

$\Rightarrow$ overall relevance of X_l :

$$I_l^2 = \frac{1}{K}\sum_{k=1}^{K} I_{lk}^2.$$

Source http://www.stat.ucl.ac.be/

Interpretation (Cont'd)

B. Partial dependence plots

$\mathbf{X_S}$ of dim. $l < p$ ($\mathbf{X_S} \subset \mathbf{X} = (\mathbf{X_1}, \ldots, \mathbf{X_p})$),

$S \subset \{1, 2, \ldots, p\}$ and $S \cup C = \{1, 2, \ldots, p\}$.

So $f(\mathbf{X}) = \mathrm{f}(\mathbf{X_S}, \mathbf{X_C})$.

Define

$$f_S(\mathbf{X_S}) = \mathbf{E_{X_S}}\mathrm{f}(\mathbf{X_S}, \mathbf{X_C}).$$

Estimation : $\hat{f}_S(\mathbf{X_S}) = \frac{1}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{N}} \mathrm{f}(\mathbf{X_s}, \mathbf{x_{ic}})$.

$f_S(\mathbf{X_S})$ is the effect of $\mathbf{X_S}$ on $f(\mathbf{X})$ after accounting (averaged) effects of the other variables $\mathbf{X_C}$ on $f(\mathbf{X})$.

Source http://www.stat.ucl.ac.be/

Illustration (California Housing)

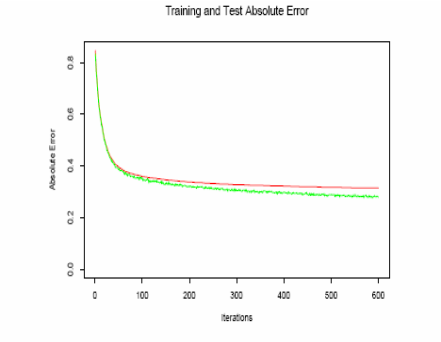


Figure 10.13: *Average-absolute error as a function of number of iterations for the California housing data.*

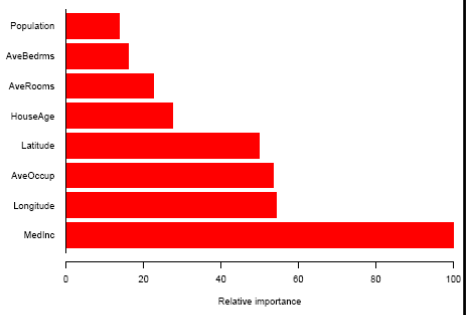


Figure 10.14: *Relative importance of the predictors for the California housing data.*

Illustration – California Housing Data

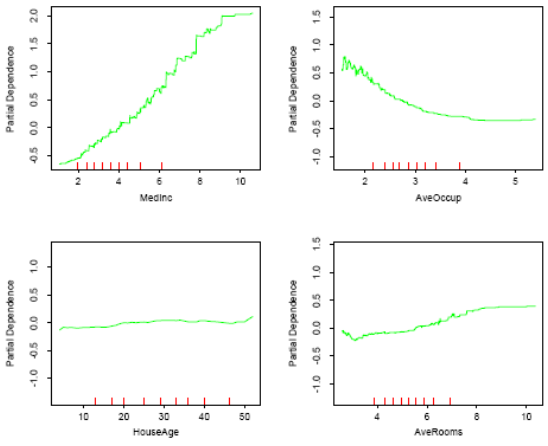


Figure 10.15: *Partial dependence of housing value on the nonlocation variables for the California housing data.*

Illustration – California Housing Data

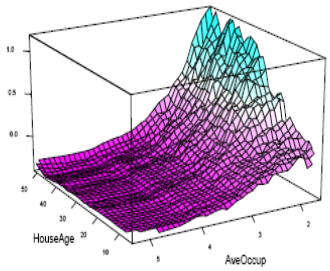


Figure 10.16: *Partial dependence of house value on median age and average occupancy. There appears to be a strong interact effect between these two variables.*

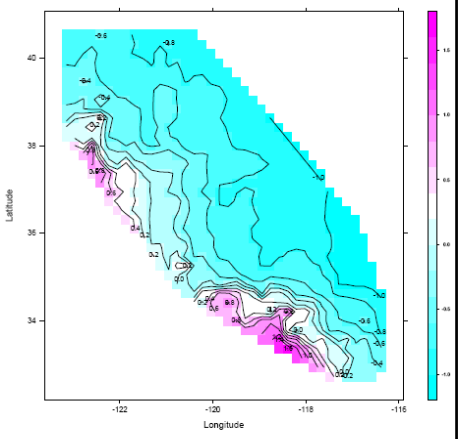


Figure 10.17: *Partial dependence of median house value on location in California.*

Illustration (Demographic Data)

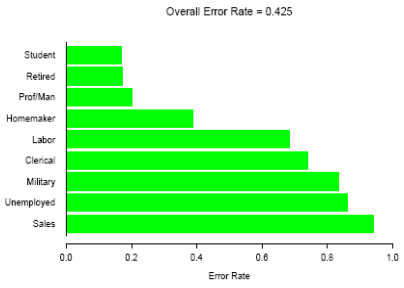


Figure 10.18: *Error rate for each occupation in the demographics data.*

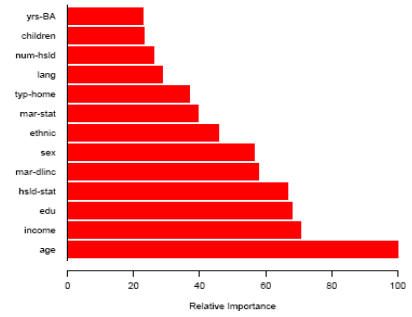


Figure 10.19: *Relative importance of the predictors as averaged over all classes for the demographics data.*

Illustration (Demographic Data)

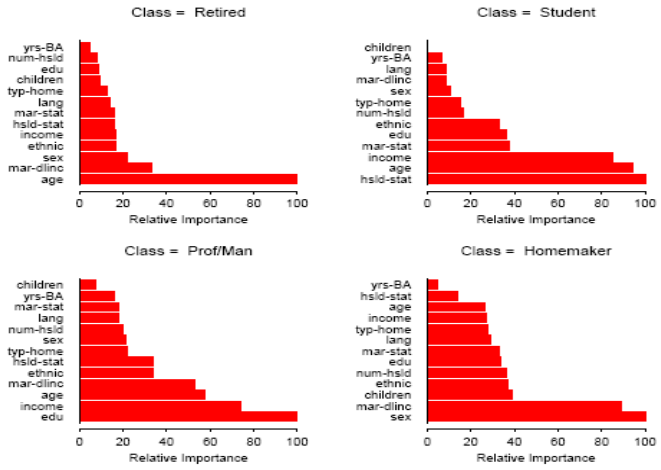


Figure 10.20: *Predictor variable importances separately for each of the four classes with lowest error rate for the demographics data.*

Illustration (Demographic Data)

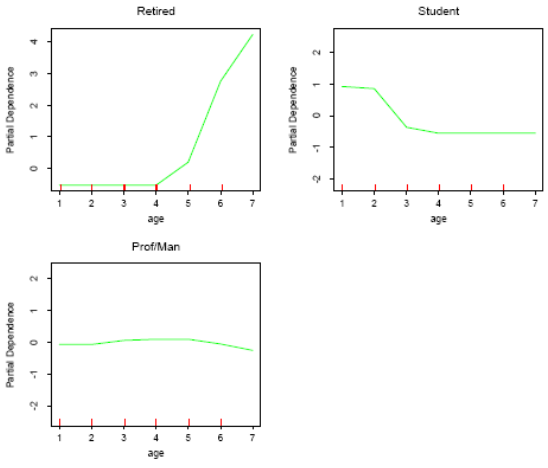


Figure 10.21: *Partial dependence of the odds of three different occupations on age, for the demographics data.*