Boosting, Trees, and Additive Models

CS 534: Machine Learning

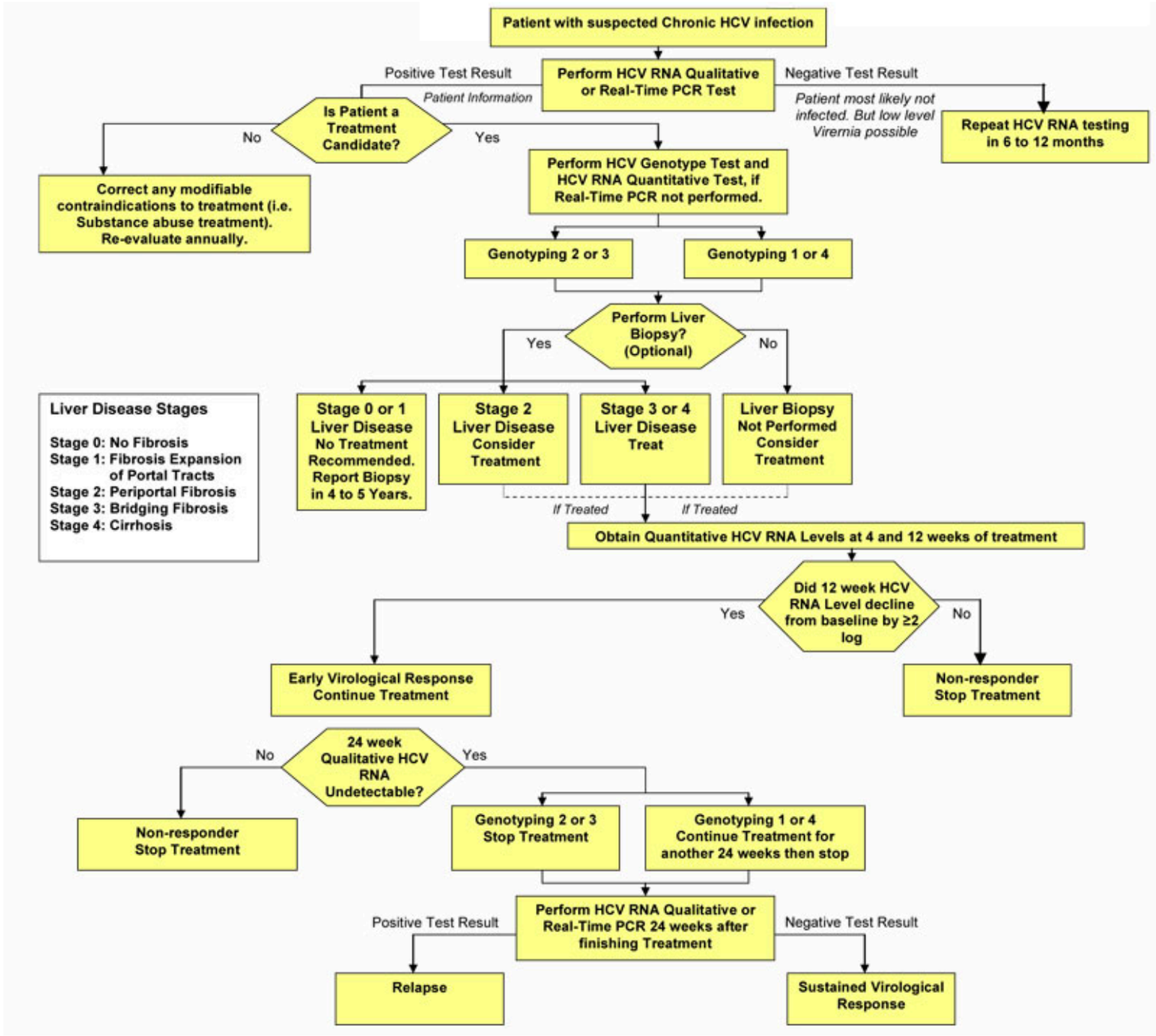
Slides adapted from Lee Cooper, Ryan Tibshirani, Cheng Li, David Sontag, Luke Zettlermoyer, Carlos Guestrin, Andrew Moore, and Yubin Park

Tree-based Models

Tree-based Methods

- Divides the feature space into rectangles
    - Successive binary splits on predictor variables
        - Fits very simple model in each rectangle (e.g., constant)
- Works for both discrete and continuous responses

Real World Inspiration

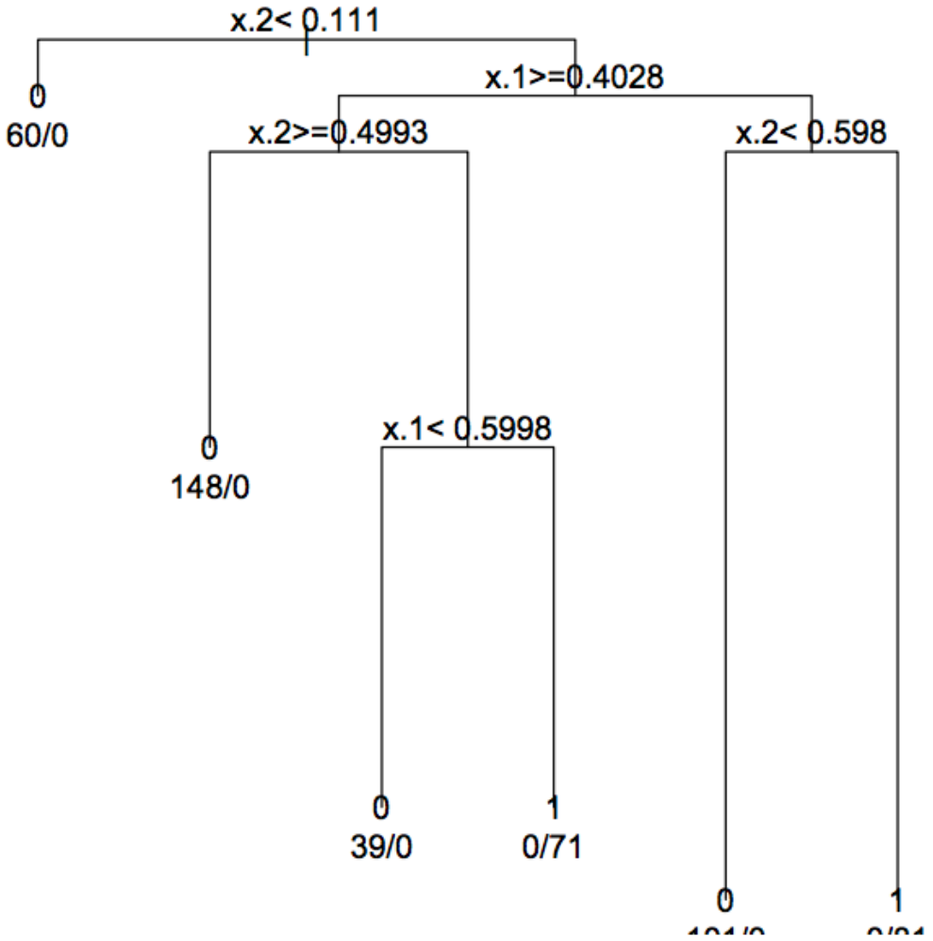


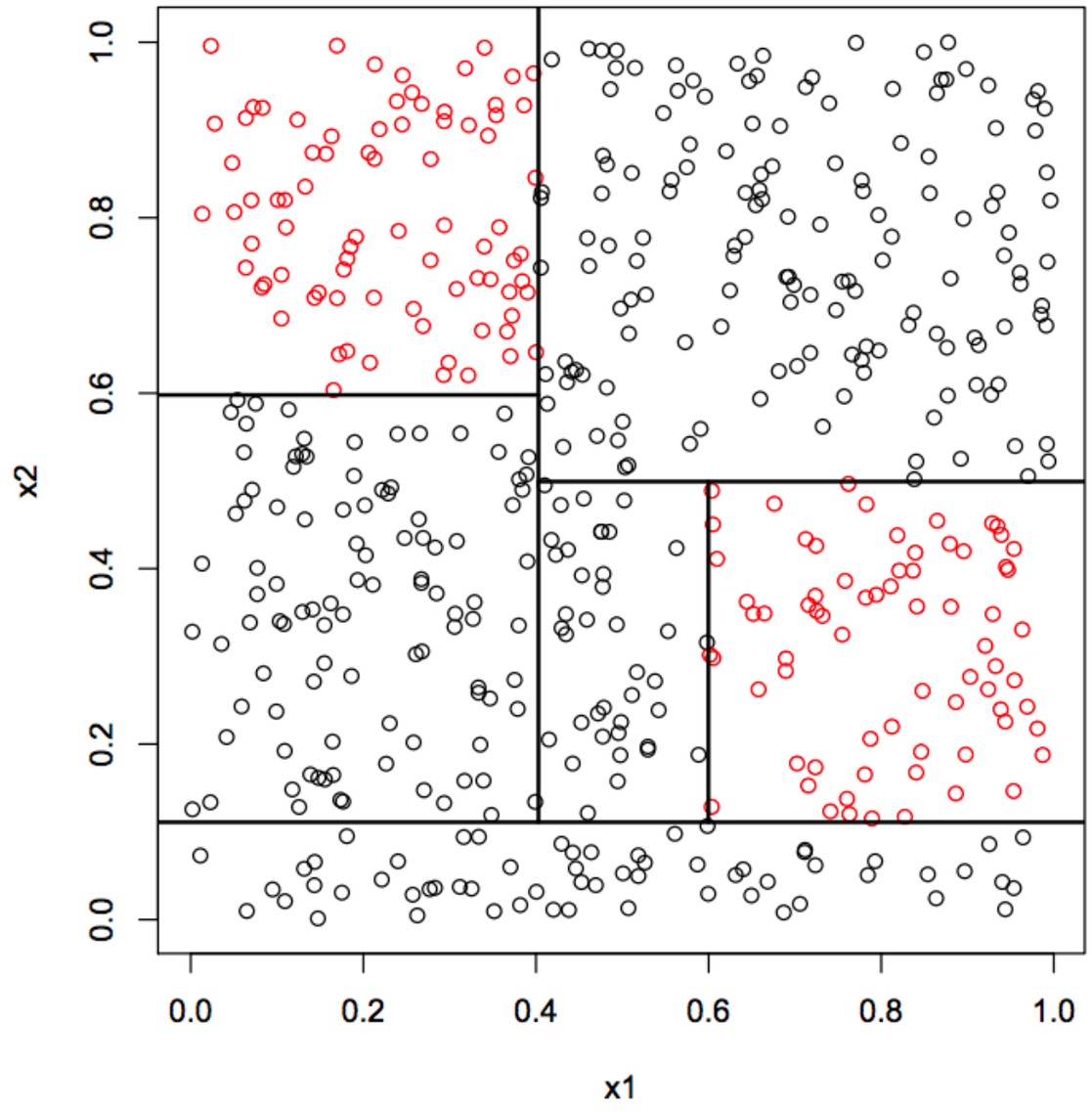
http://hcv.org.nz/wordpress/?tag=treatment-flow-chart

Classification Trees

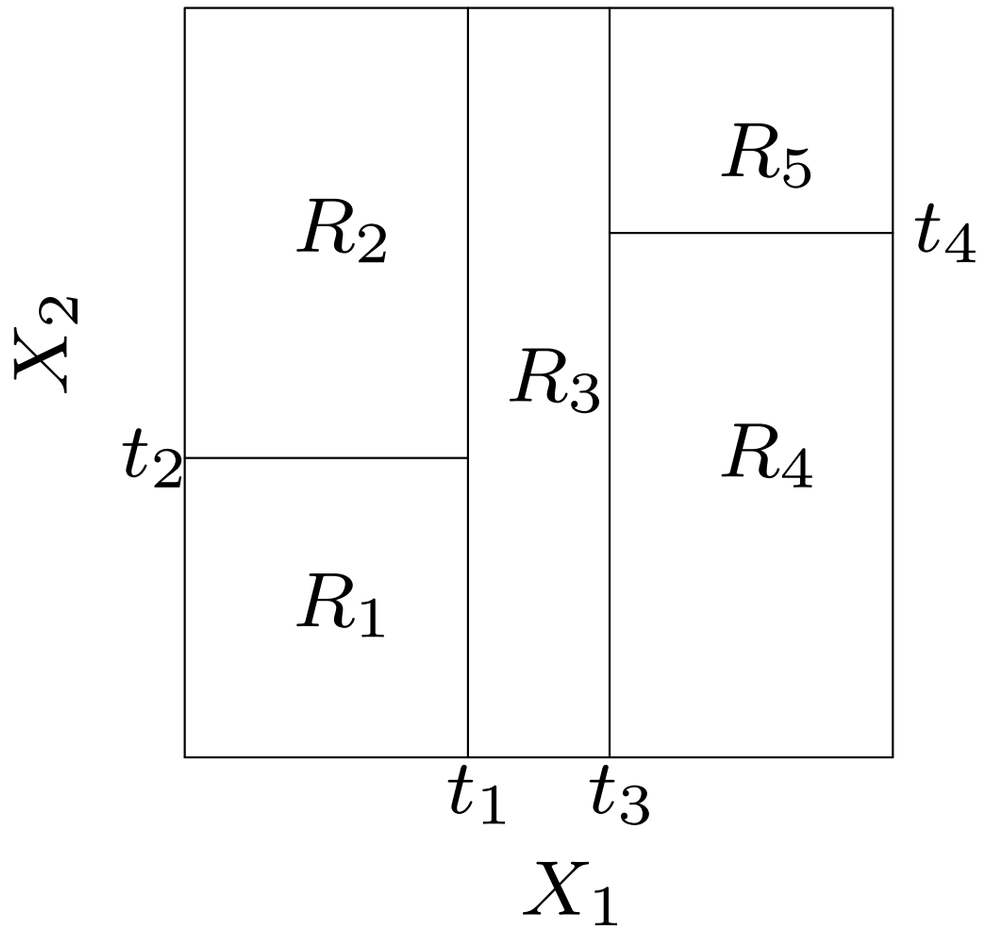
- Popular and interpretable
  - Mimic (to some extent) decision making process
- Can be thought of as defining m regions (rectangles) each corresponding to a leaf of the tree
- Each region is assigned a class label
- Finding region for new point is easy since we just need to scan the tree

Example: Simple Classification Tree





Example: Region Split & Tree



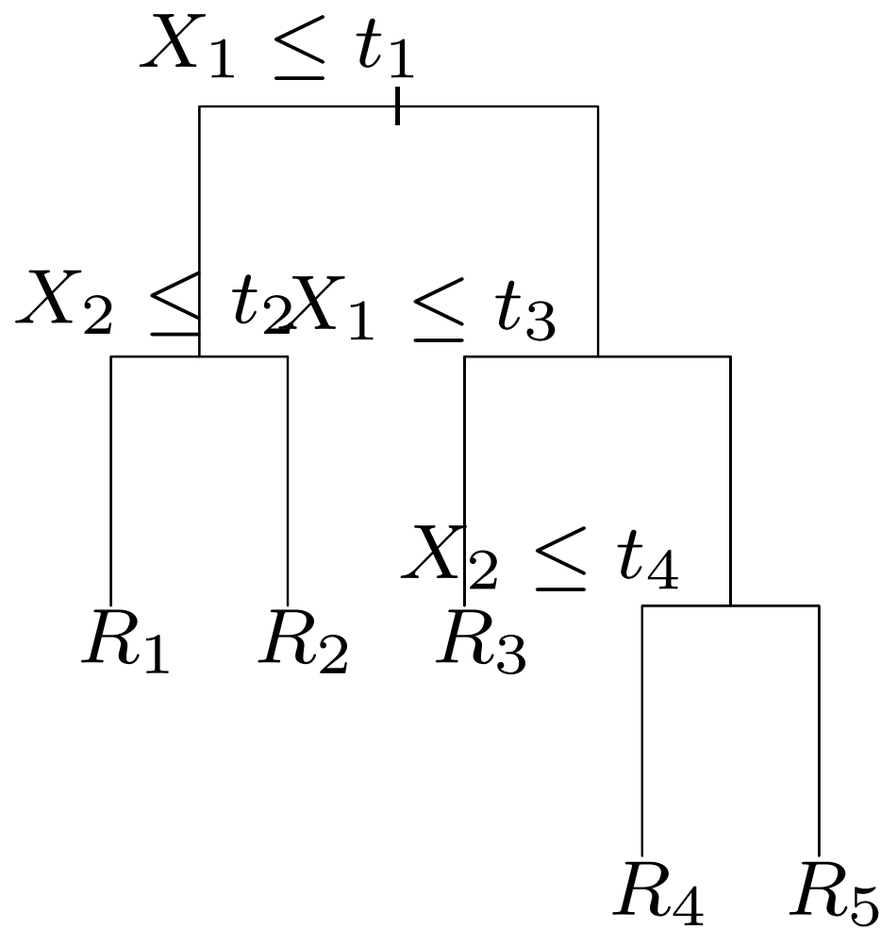


Figure 9.2 (Hastie et al.)

Example: Unachievable Tree Partition

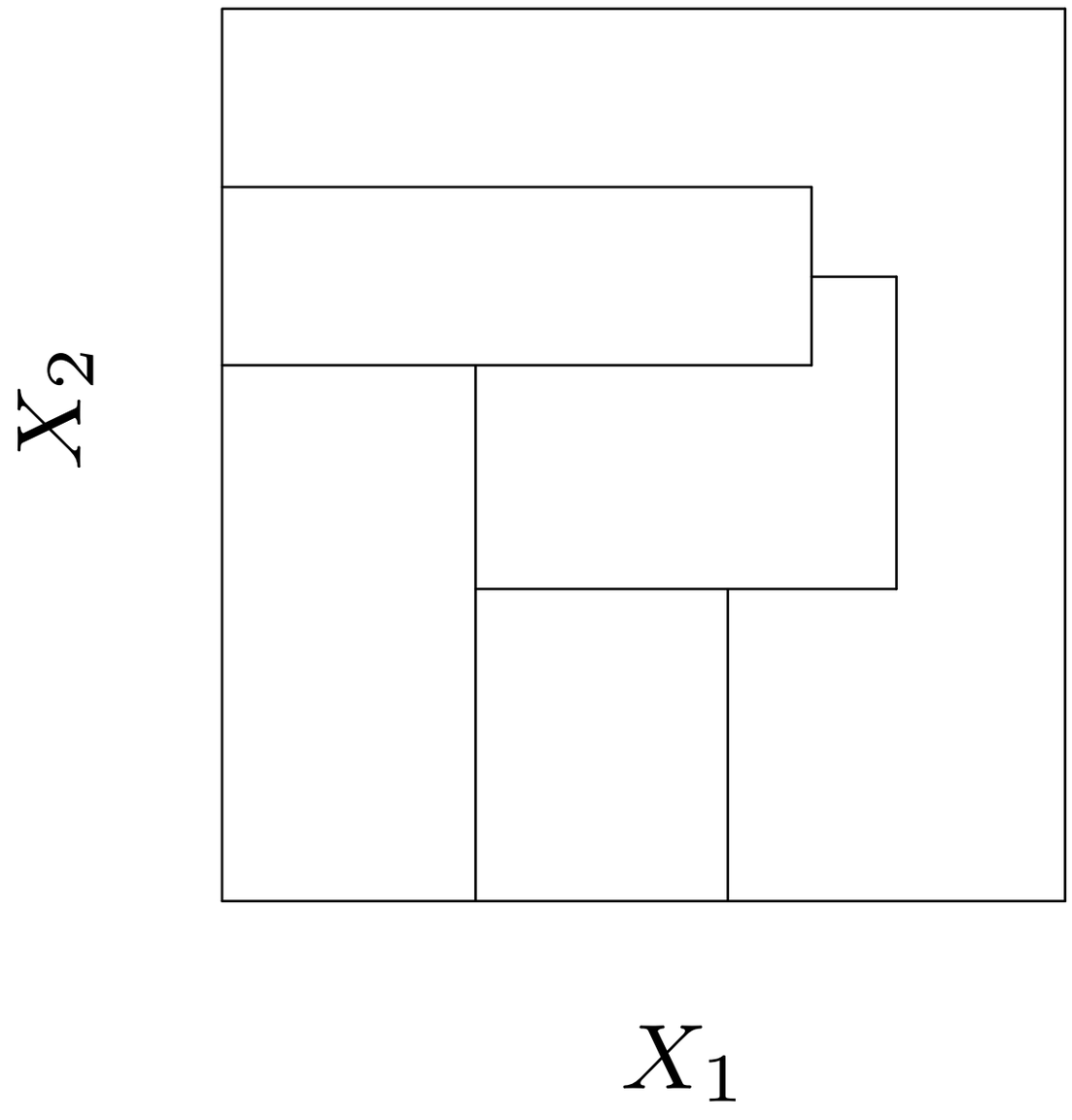


Figure 9.2 (Hastie et al.)

Power of Trees

	Model Assumption	Estimated Prob	Interpretable	Flexible	Predicts Well
LDA	Yes	Yes	Yes	No	Depends on X
LR	Yes	Yes	Yes	No	Depends on X
Tree	No	Yes	Yes	Somewhat	?

Building Trees

- Two main issues arise when building a tree:
  - How to choose the splits
  - How big to grow the tree

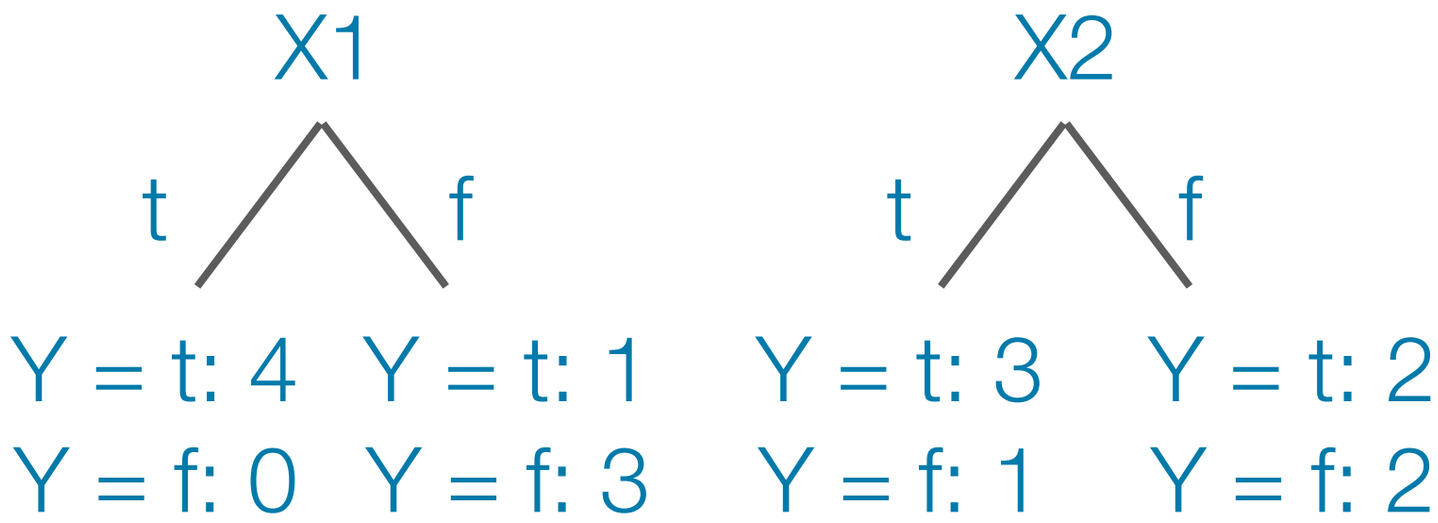
Learning Simplest Decision Tree

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]

- Resort to a greedy heuristic
  - Start from empty tree
  - Split on next best feature
  - Recurse

Splitting: Choosing Good Attribute

- Would we prefer X1 or X2?



X1	X2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

- Idea: Use counts at leaves to define probability distributions to measure uncertainty

Splitting: Uncertainty

- Good split if we are more certain about classification after split
  - Deterministic — good (all true or false)
  - Uniform distribution — bad
  - What about in-between?

Entropy

- Entropy of a random variable

$$H(G) = -\sum_i P(G = g_i) \log_2 P(G = g_i)$$

- Information theory interpretation: number of bits needed to encode a randomly drawn value of Y

- High entropy —> uniform like distribution / flat histogram

- Low entropy —> varied (peaks and valleys) with more predictable values

Splitting Criterion: Information Gain

- Decrease in entropy (uncertainty) after split

$$H(G|X) = -\sum_j P(X = x_j) \sum_i P(G = g_i) \log_2 P(G = g_i)$$

$$IG(X) = H(G) - H(G|X)$$

- Also referred to as cross-entropy or deviance

- Commonly used in ID3 (Iterative Dichotomiser 3) and C4.5 (successor of ID3) algorithms

Splitting Criterion: Gini Index

- Measure of how often a randomly chosen element from set would be incorrectly labeled if it was randomly labeled based on the labels in subset

$$\sum_i P(G = g_i)(1 - P(G = g_i))$$

- Also known as measure of node purity

- Commonly used in CART (Classification and Regression Trees)

Splitting Criterion: Misclassification Error

- Probability of incorrect classification in the node

$$1 - \max_i(P(G = g_i))$$

- Fraction of training observations that does not belong to the most common class

- Less commonly used as not differentiable —> less amendable to numerical optimization

Classification Impurity Measures

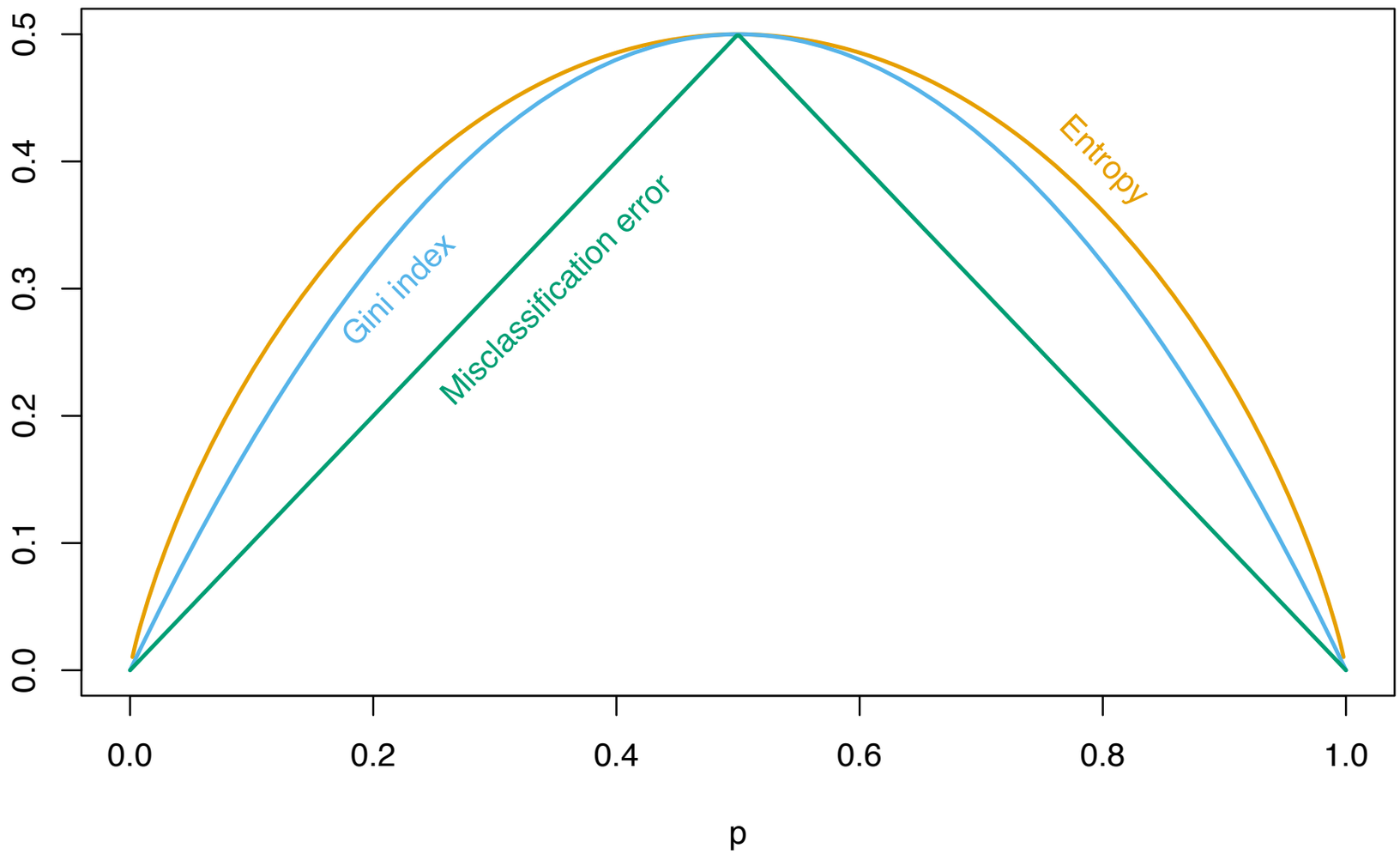


Figure 9.3 (Hastie et al.)

Tree Growing Details

- Select predictor X_j such that splitting predictor space into regions leads to greatest reduction in splitting criterion

$$R_1 = \{\mathbf{X} \in \mathbb{R}^p : X_j = s\}, R_2 = \{\mathbf{X} \in \mathbb{R}^p : X_j \neq s\}$$

- Repeat the process looking for best predictor to split data further within each of the newly defined regions

- Continue in this manner until stopping criterion reached

Stopping Criterion

- Minimum number of instances in a leaf
- Maximum tree depth
- 100% node purity
- Gain tolerance

Predicted Class Probabilities

- Each region R_j contains some subset of training data point

- Predicted probability is just proportion of points in the region belong to class k

$$\hat{p}_g(R_j) = \frac{1}{n_j} \sum_{\mathbf{x}_i \in R_j} \mathbb{1}_{\{y_i = g\}}$$

- Predicted class is the most common class occurring amongst these points

$$g_j = \text{argmax } \hat{p}_g(R_j)$$

Need for Pruning

- Tree growing process may overfit training data leading to poor generalization performance

- Smaller tree with fewer splits might lead to lower variance and better interpretation

- Why not just grow smaller trees?

Pruning Details

- Grow a very large tree and prune it back to get a subtree
- Cost complexity pruning (aka weakest link pruning)

$$C_\alpha(T) = \sum_{j=1}^{|T|} [1 - \hat{p}_{g_j}(R_j)] + \alpha|T|$$

- Tuning parameter determines size of tree and can be chosen via CV
- Prune the weakest leaf one at a time

Example: SPAM Dataset

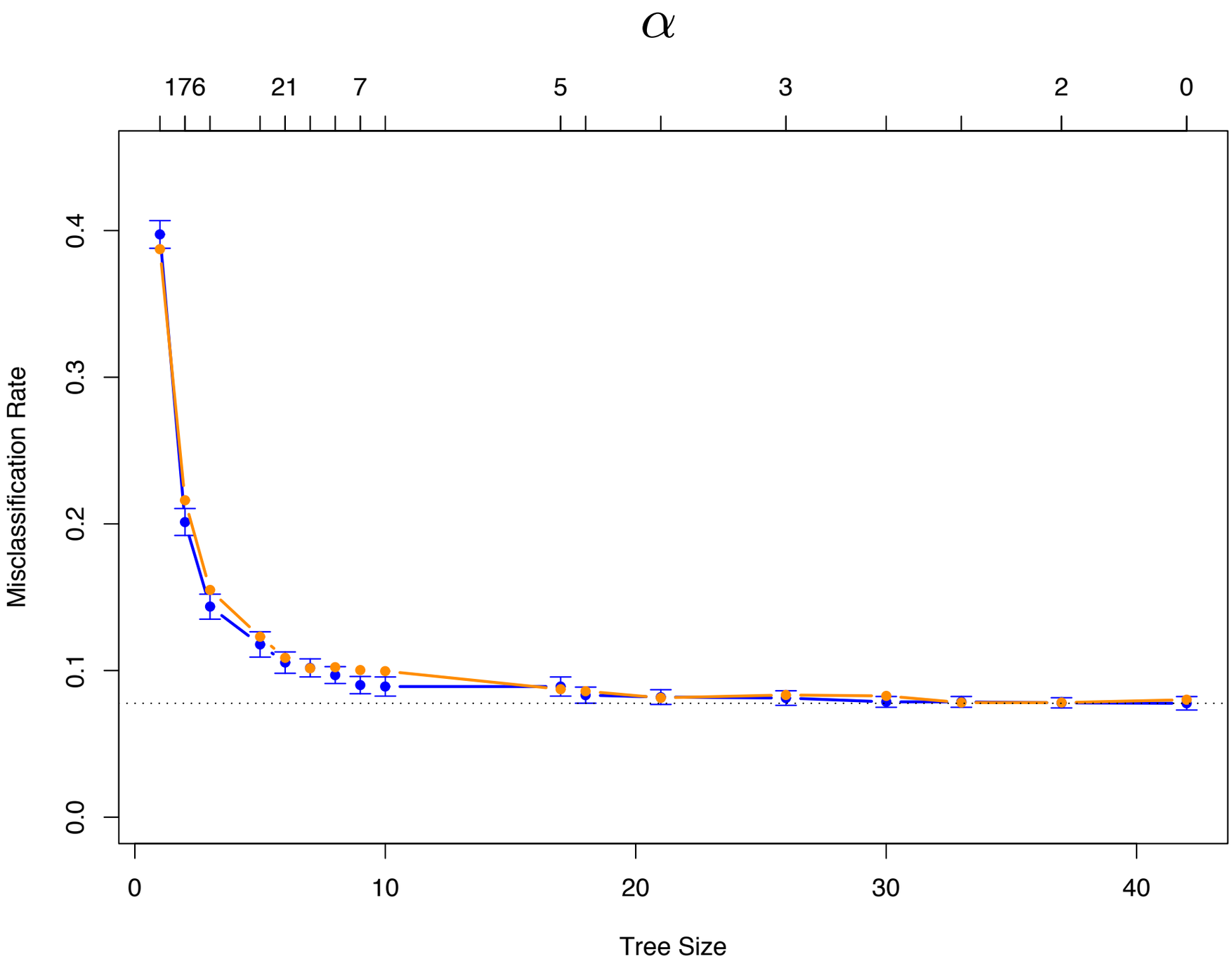


Figure 9.4 (Hastie et al.)

Example: SPAM Decision Tree

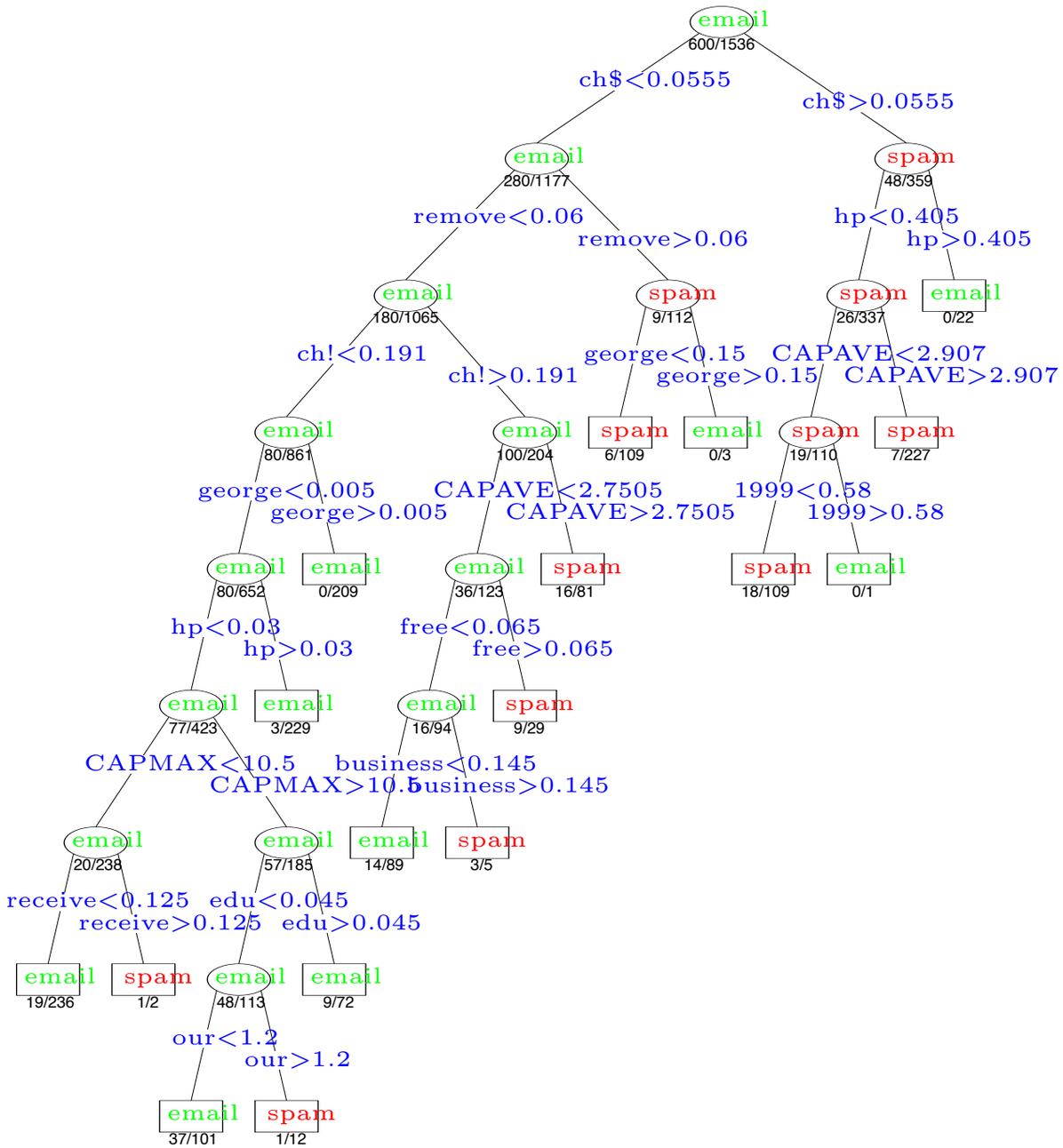
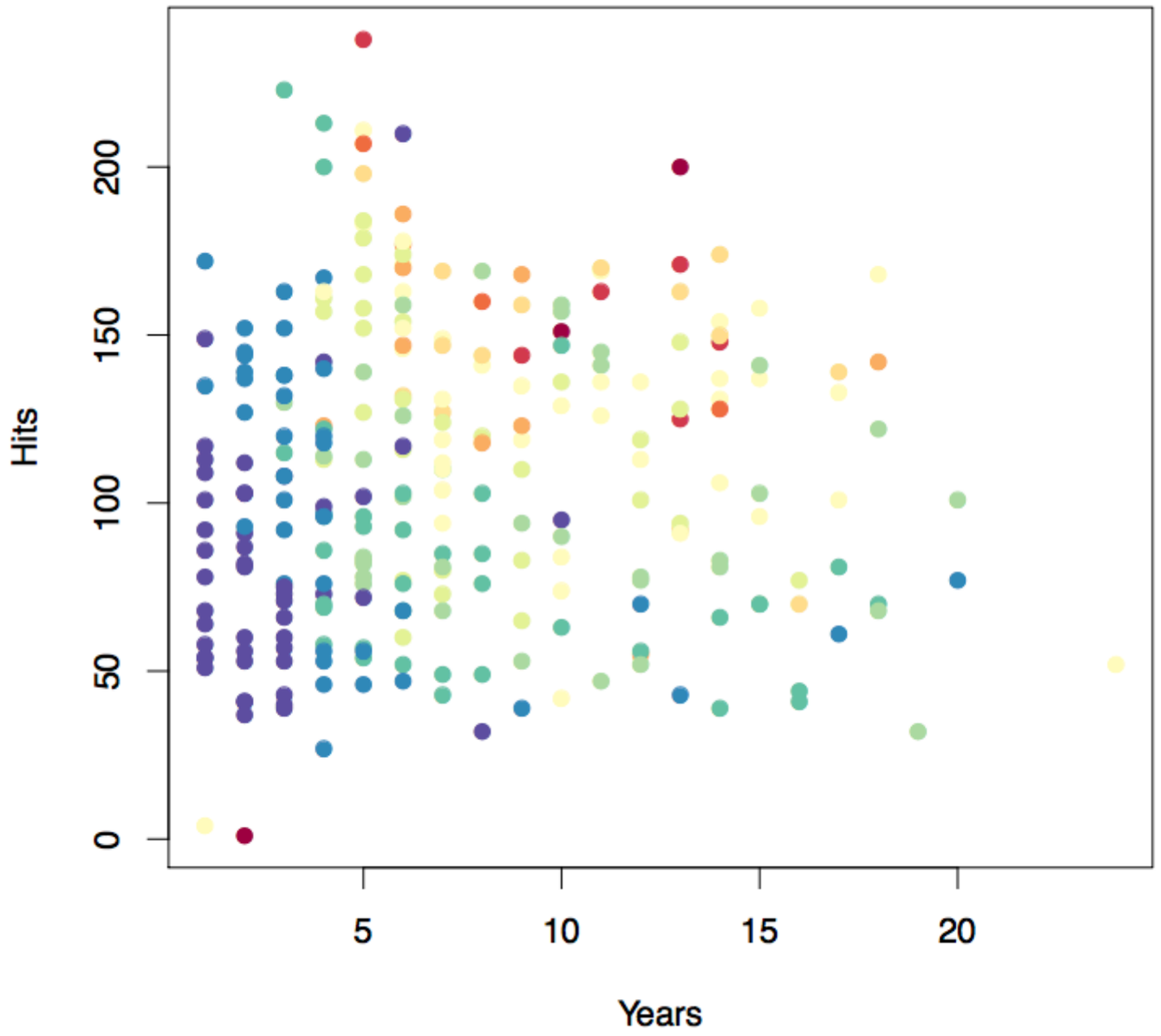


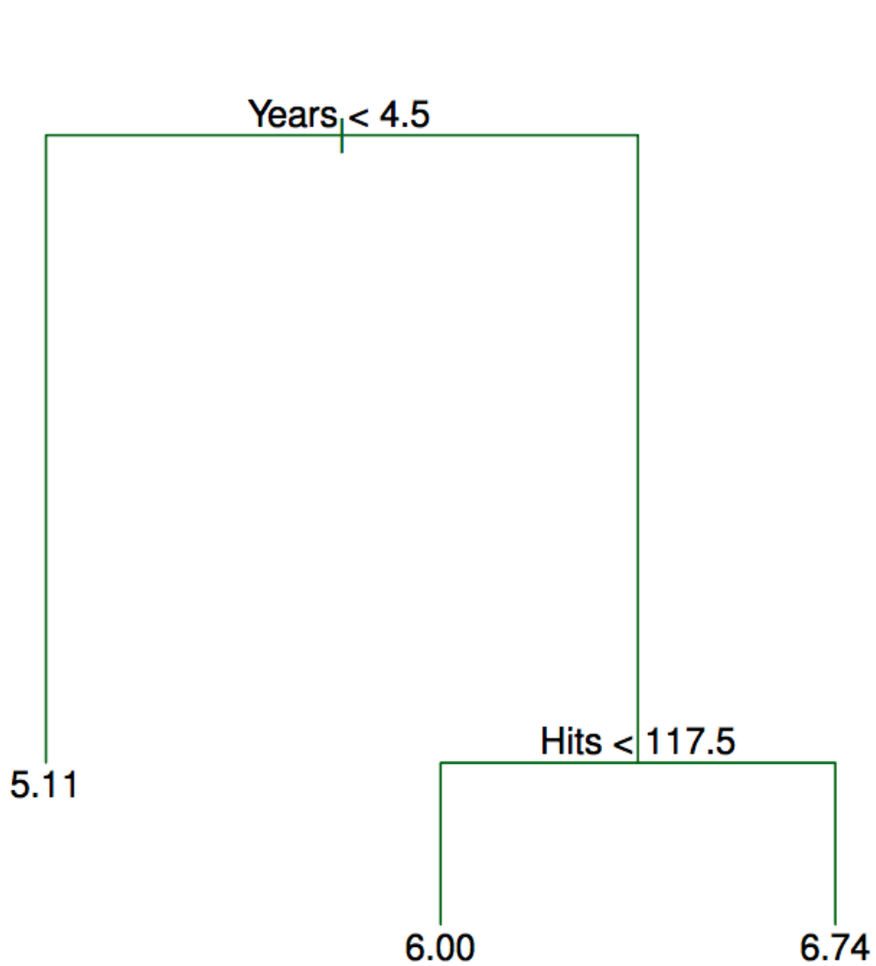
Figure 9.5 (Hastie et al.)

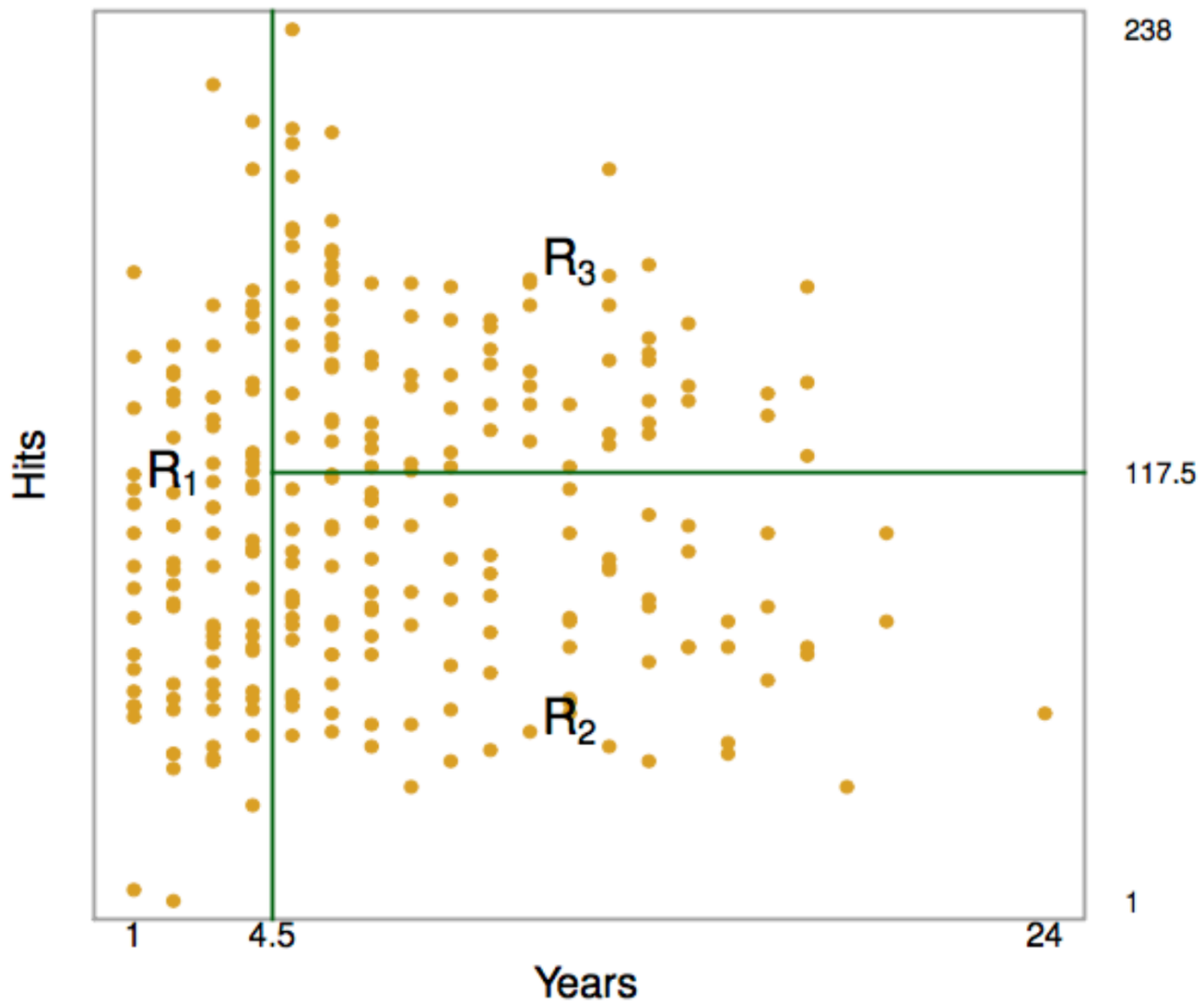
Regression: Baseball Salary Data



Salary is color-coded from low (blue, green) to high (yellow, red)

Regression: Baseball Salary Data





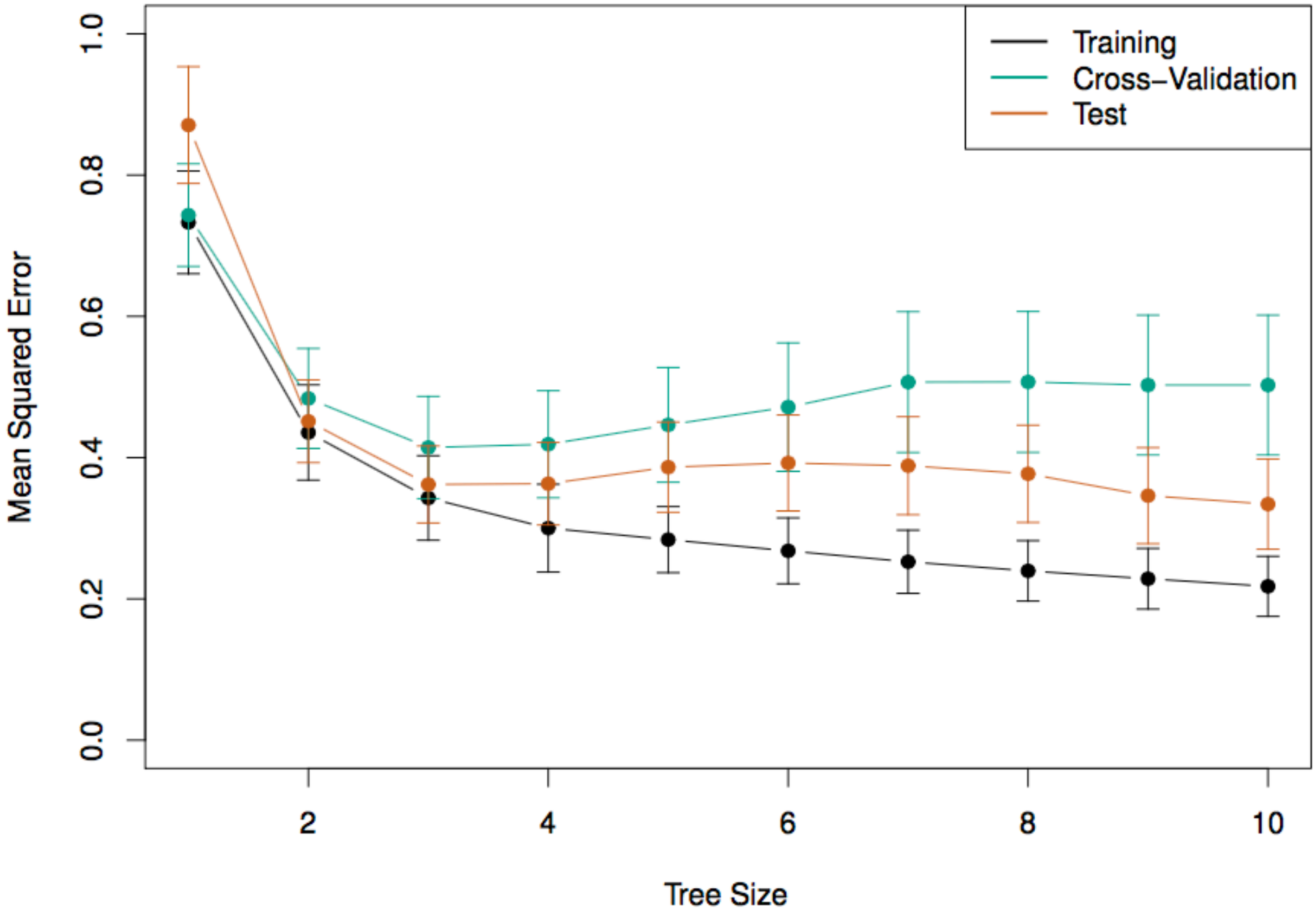
Regression Trees

- Similar idea to classification trees
- Splitting criteria: Minimize RSS in region

$$\sum_j \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Each region predicts a single continuous outcome

Regression Example



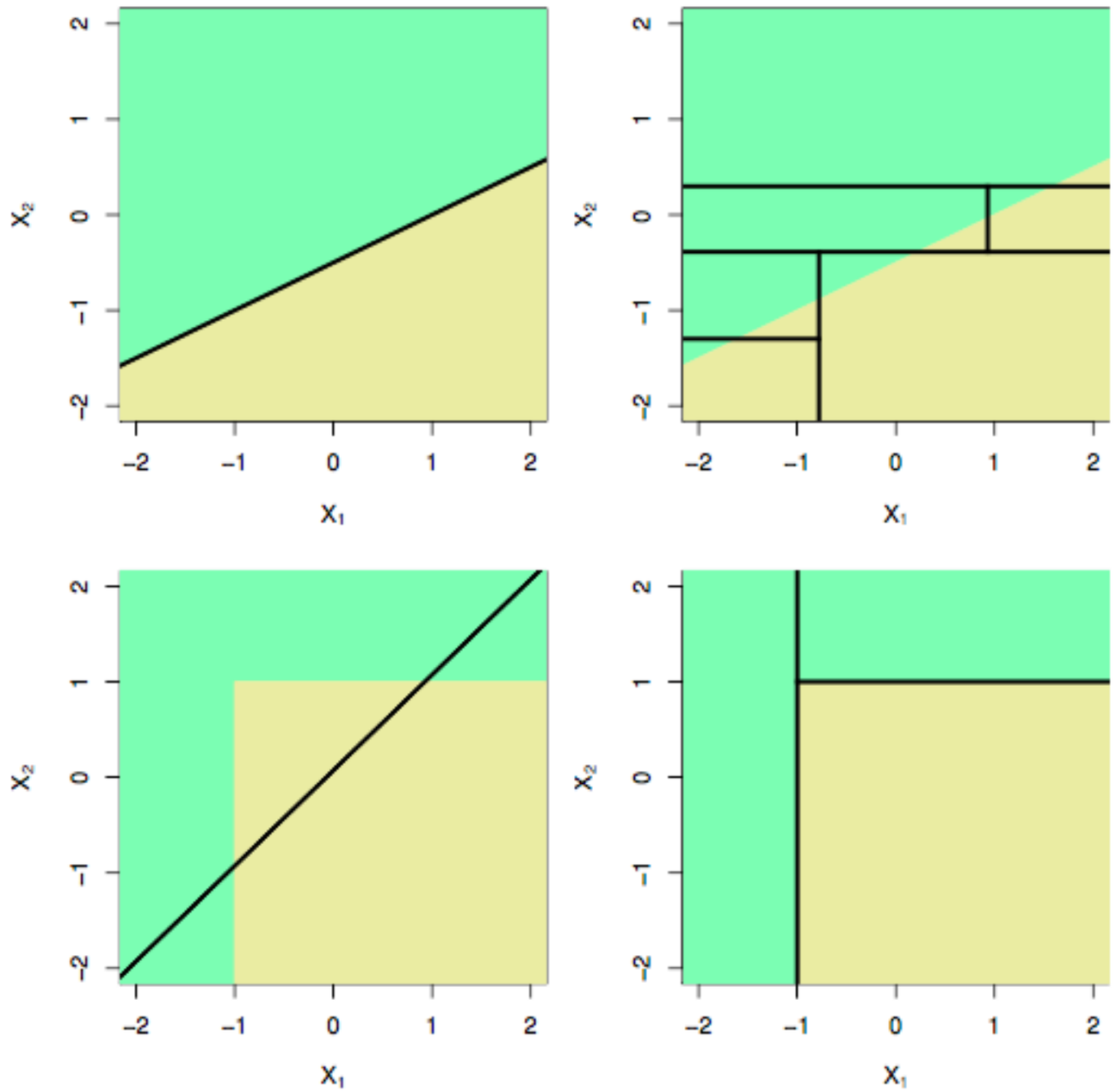
Dealing with Missing Values

- Suppose missing predictor values in some of the variables
- Standard approaches
  - Delete observations w/ missing data
  - Impute (fill in) missing values with mean

Trees: Dealing with Missing Values

- Two better approaches
  - Categorical predictors — create a new category for "missing"
- Construct surrogate variables
  - Create list of surrogate predictors and split points
  - If missing primary, try the surrogate splits in order

Tree vs Linear Model



Predictive Performance of Trees

- Predictive performance is not that great
- Bias/variance trade-off: larger size means smaller bias, high variance
- High variance overall because trees are quite unstable — small change in observed data can lead to drastically different splits

Trees: Pros & Cons

- (Pro) Easy to explain
- (Pro) Mirrors human decision-making process than other approaches
- (Pro) Graphical display lends it to be easily interpreted by non-expert
- (Pro) Tree can handle qualitative predictors without creating dummy variables
- (Con) Lower predictive accuracy

Additive Models

Additive Model: Regression

- Attractiveness of linear regression
  - Simple
  - Understanding for importance of different inputs
- Quite restrictive— effects often not linear
- Generalized additive model

f_j are unspecified smooth functions

$$E[\mathbf{y}|\mathbf{x}_1, \cdots, \mathbf{x}_p] = \alpha + f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \cdots + f_p(\mathbf{x}_p)$$

Review: Logistic Regression as GLM

- Logistic regression is generalized linear model
- Linear regression model is tied to response via logit link function

$$\mu(\mathbf{x}) = \Pr(y = 1|\mathbf{x})$$

$$\log\left(\frac{\mu(\mathbf{x})}{1-\mu(\mathbf{x})}\right) = \alpha + \beta_1 x_1 + \cdots + \beta_p x_p$$

Generalized Additive Model (GAM)

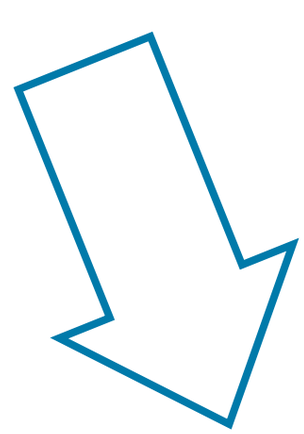
- Generalization of additive linear regression
- Conditional mean of response Y is related to additive function via a link function

$$g[\mu(\mathbf{x})] = \alpha + f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \cdots + f_p(\mathbf{x}_p)$$

Additive Logistic Regression

logistic regression as GLM

$$\log\left(\frac{\mu(\mathbf{x})}{1-\mu(\mathbf{x})}\right) = \alpha + \beta_1 x_1 + \cdots + \beta_p x_p$$



$$\log\left(\frac{\mu(\mathbf{x})}{1-\mu(\mathbf{x})}\right) = \alpha + f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \cdots + f_p(\mathbf{x}_p)$$

additive logistic regression

Common Link Functions

- Identity link: linear and additive models for Gaussian response data

$$g(\mu) = \mu$$

- Logit link: binomial probabilities

$$g(\mu) = \text{logit}(\mu)$$

- Log link: log-linear or log-additive models for Poisson count data

$$g(\mu) = \log(\mu)$$

GAM: Properties

- Local regression to low dimensional projections of the data

- Surface estimation via a collection of one-dimensional functions (each function is analogous to coefficients in linear regression)

- Ability to model nonlinearities in the data automatically

Example: Spam Data

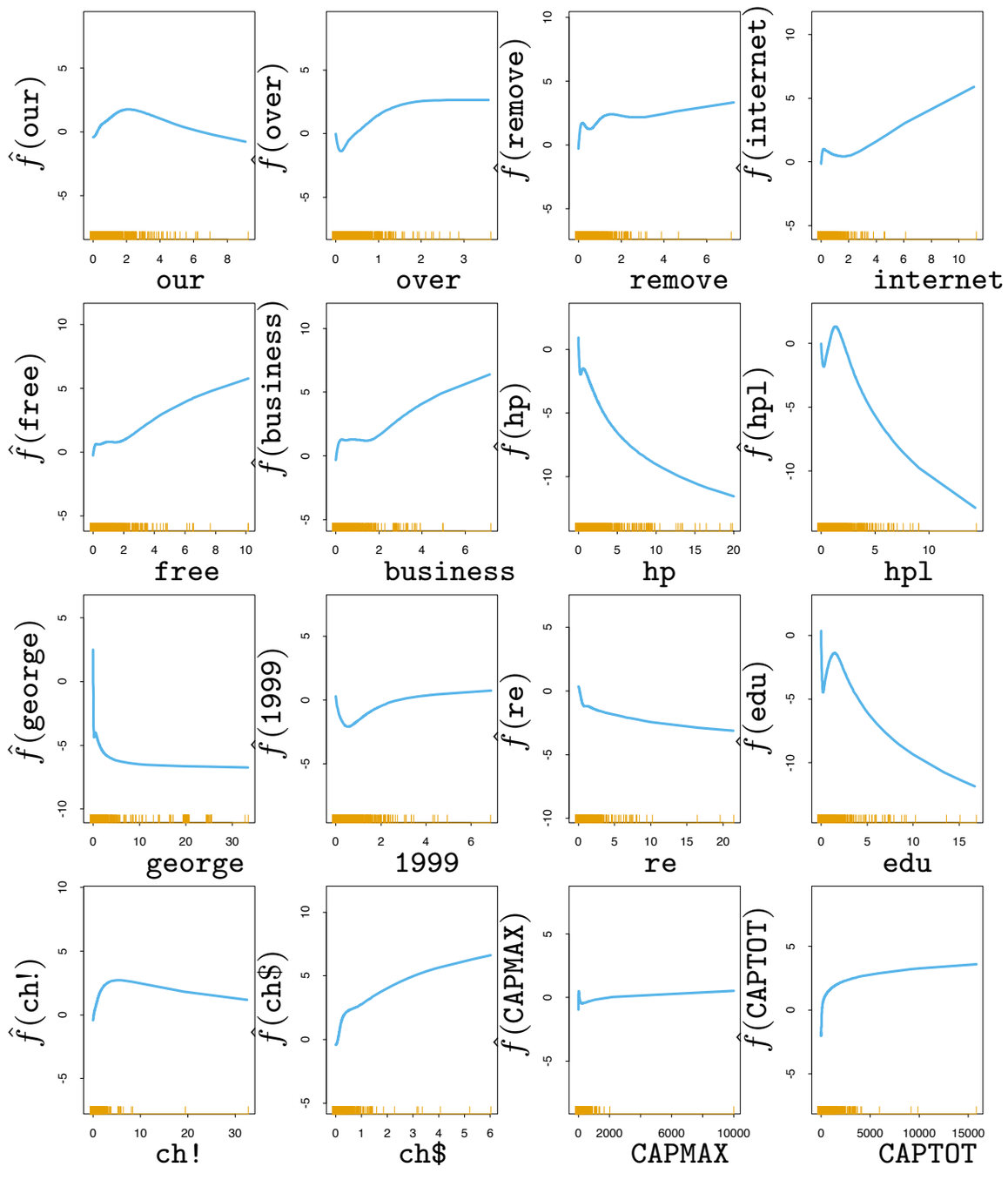


Figure 9.1 (Hastie et al.)

Boosting

Boosting

- Weak learner: model whose error rate is only slightly better than random guessing

- Idea: Combine output of many weak classifiers to produce powerful committee

- Method: Sequentially fit weak learners with later models compensating the shortcomings of the existing learners

Tree with Observation Weights

- Each observation has a weight
  - Higher value means higher importance of correctly classifying this observation
- Tree can be easily adapted to use weights
  - Predictive probability uses weighted class

$$\hat{p}_g(R_j) = \frac{\sum_{\mathbf{x}_i \in R_j} w_i \mathbb{1}_{\{y_i = g\}}}{\sum_{\mathbf{x}_i \in R_j} w_i}$$

AdaBoost

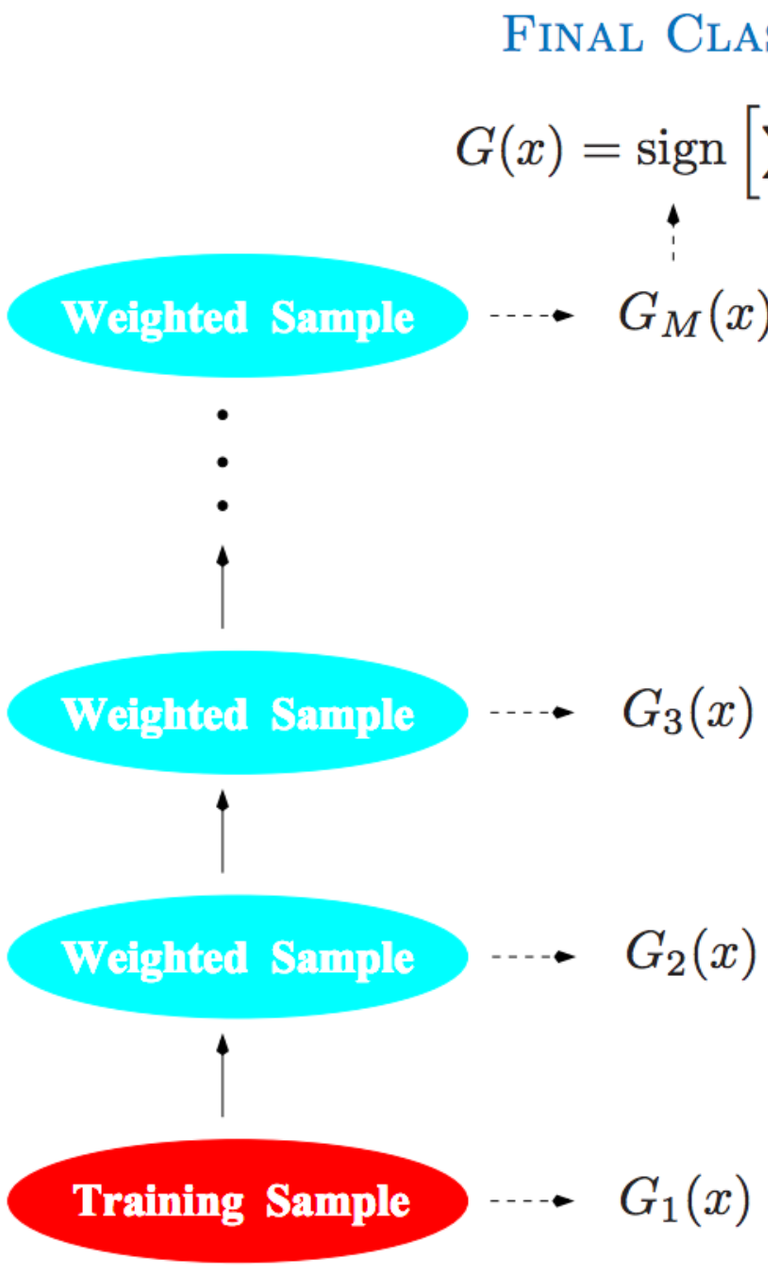
- Most popular boosting algorithm developed by Freund & Schapire (1997)

- Consider two-class problem with the output variable is coded as {+1, -1}

- Each weak classifier $G_m(X)$ produces prediction taking one of the two values

- Combine a weighted sum of M different classifiers

AdaBoost



classifier contribution weights computed by algorithm — higher influence to more accurate classifiers

Figure 10.1 (Hastie et al.)

AdaBoost: Algorithm

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

2. For $m = 1$ to M:

   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

   (b) Compute

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

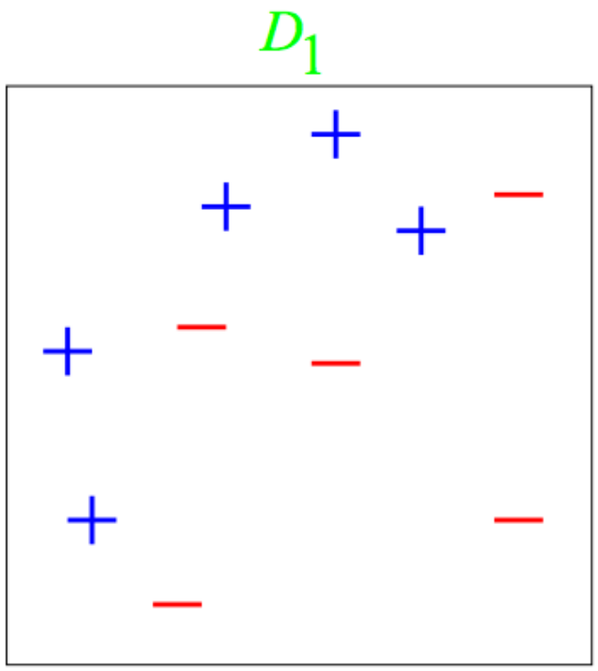
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

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

Algorithm 10.1 (Hastie et al.)

Example: Toy Data

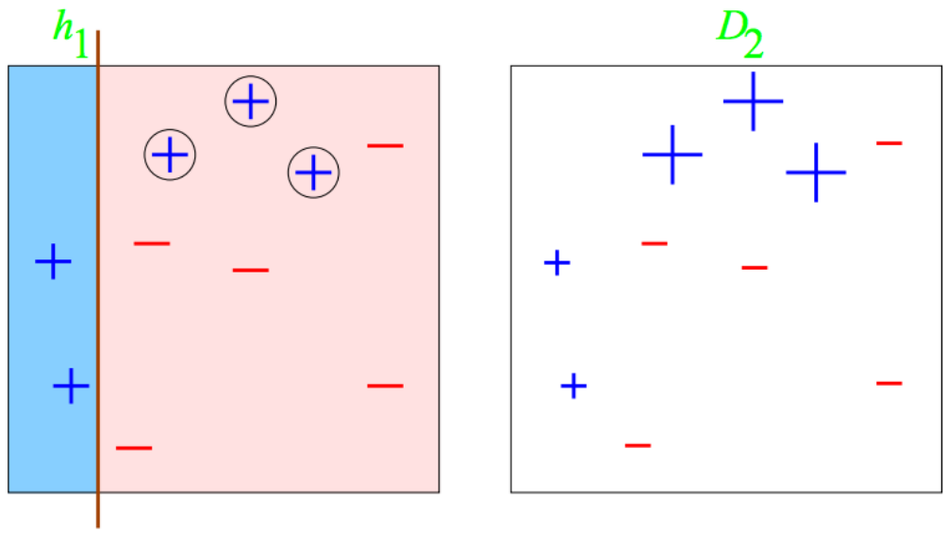


weak classifier: single horizontal or vertical half-plane

http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Example: Toy Data

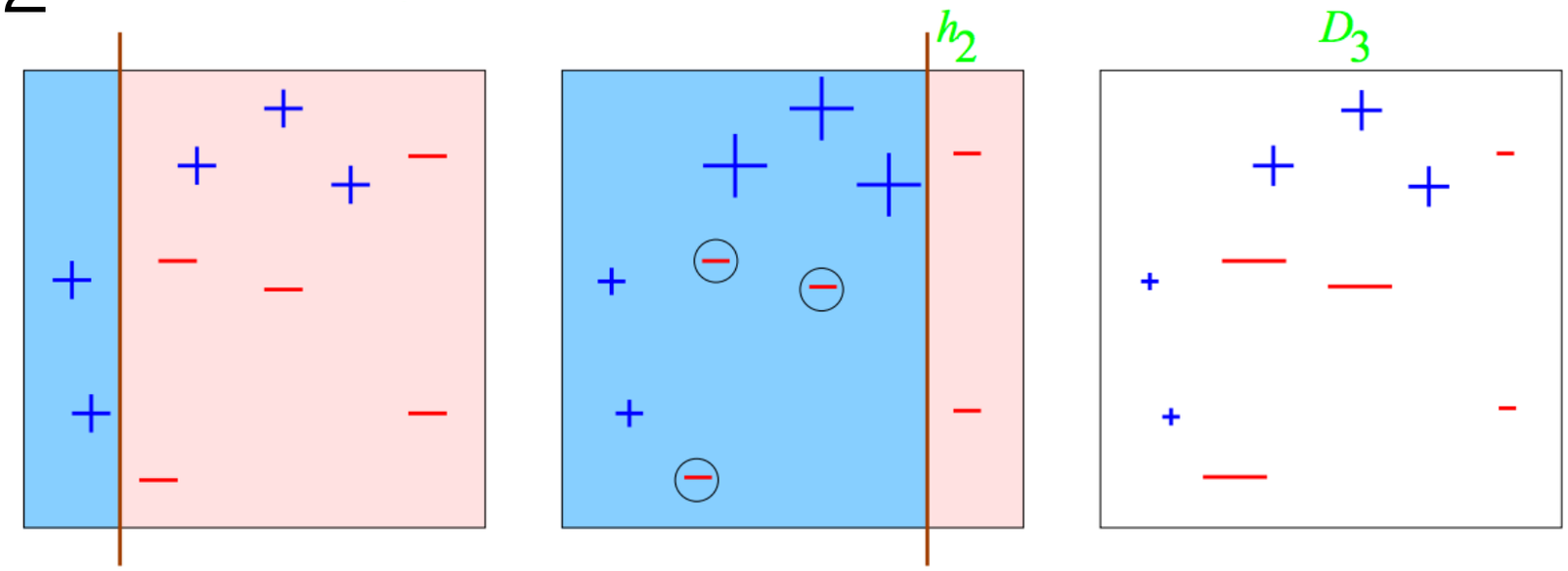
Round 1



Round 2

$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

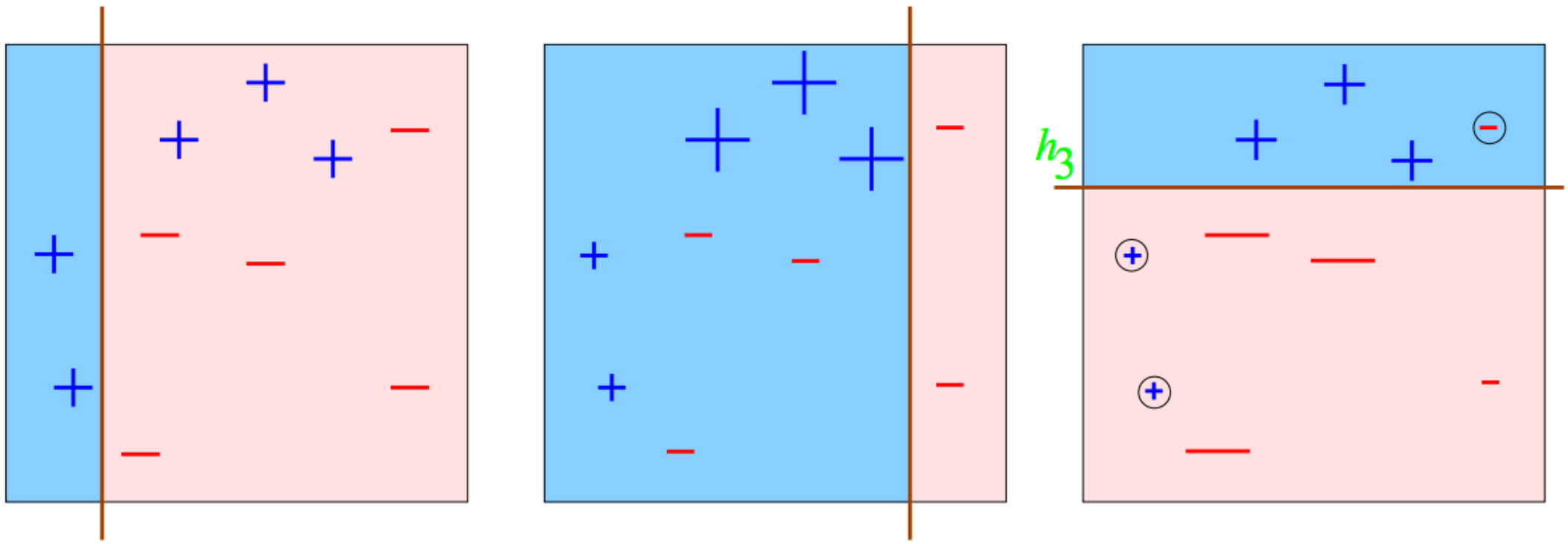


$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

Example: Toy Data

Round 3

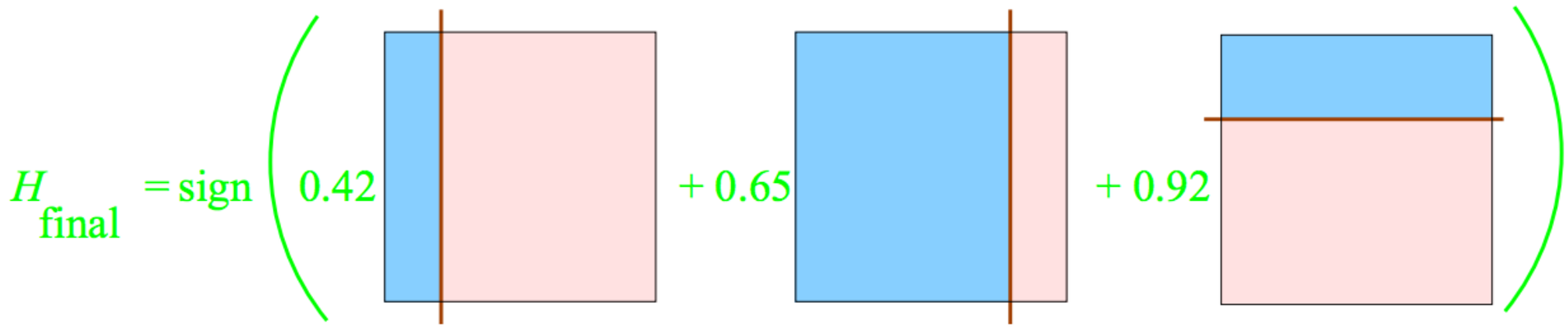


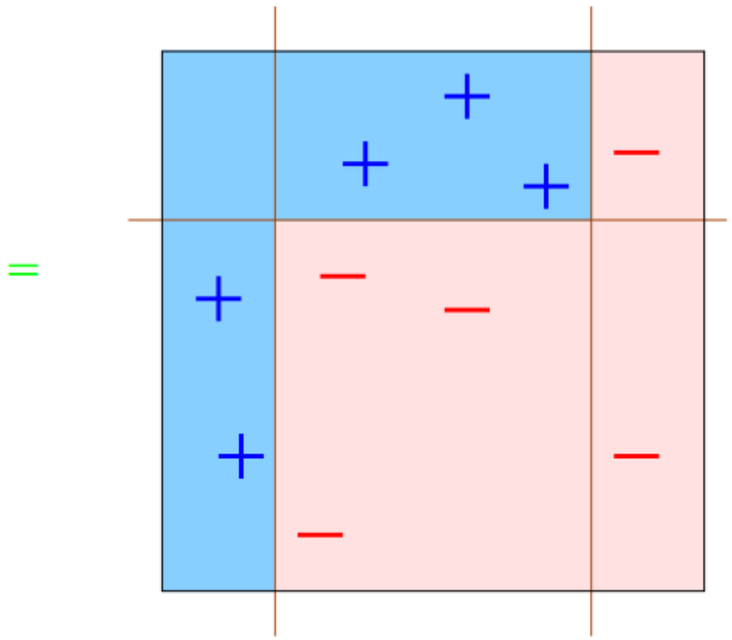
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Example: Toy Data

Final Classifier





Example: Boosting Stumps

- Simulated data with 1000 points draw from known model

- Classification tree with one split (two leaves)

- Misclassification rate of 45.8% for single tree

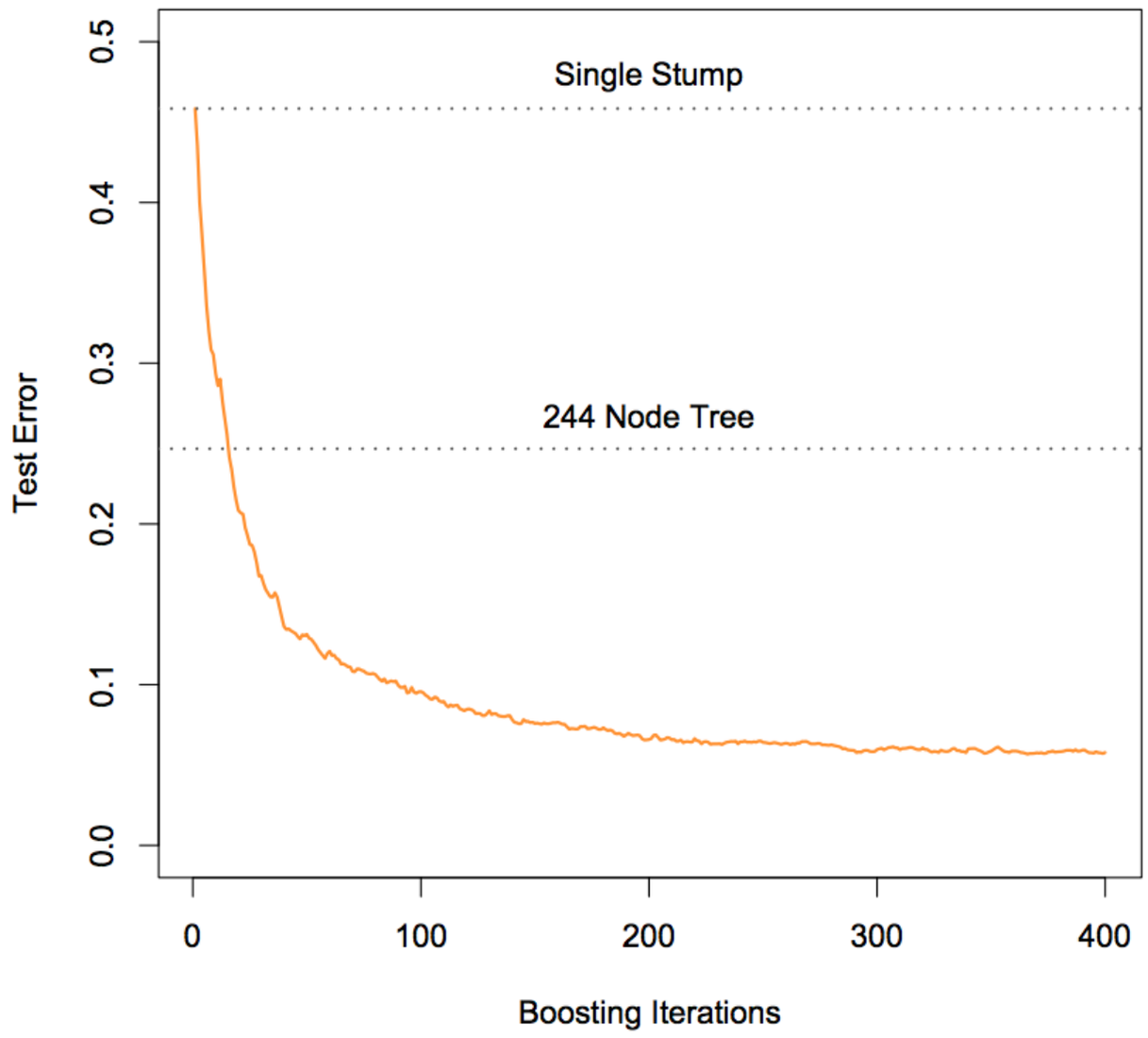


Figure 10.2 (Hastie et al.)

Review: Forward Stepwise MLR

- Given continuous response y and predictors X_1, ..., X_p
- Forward stepwise modeling:
  - Choose predictor X_j giving smallest squared error

$$(\mathbf{y} - \hat{\beta}_j \mathbf{X_j})^\top (\mathbf{y} - \hat{\beta}_j \mathbf{X_j})$$

  - Choose predictor X_k giving smallest additional loss

$$(\mathbf{r} - \hat{\beta}_k \mathbf{X_k})^\top (\mathbf{r} - \hat{\beta}_k \mathbf{X_k})$$

  - Repeat last step until stopping criterion reached

Boosting as Additive Model

- Exponential loss function $L(y_i, f(\mathbf{x}_i)) = \exp(-y_i f(\mathbf{x}_i))$
- Analogous stepwise modeling:
  - Find tree and weight giving smallest loss

$$\sum_i \exp(-y_i \beta_j G_j(\mathbf{x}_i))$$

  - Find new tree and weight giving smallest additional loss

$$\sum_i \exp(-y_i (\beta_j G_j(\mathbf{x}_i) + \beta_k G_k(\mathbf{x}_i)))$$

  - Repeat last step

Example: Simulated Data

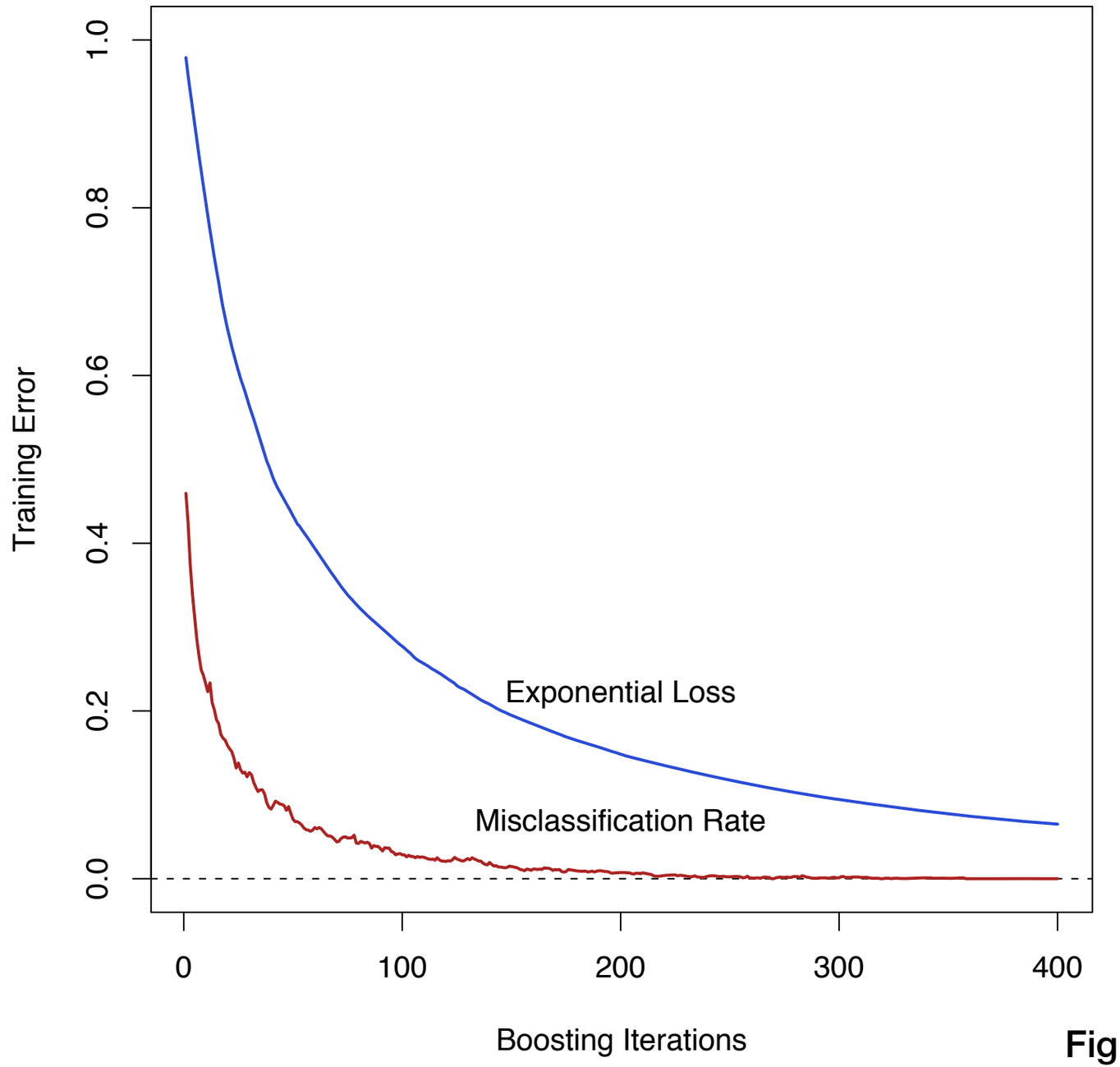


Figure 10.2 (Hastie et al.)

Classification: Loss Functions

	Loss function
Exponential	$\exp(-yf(\mathbf{x}))$
Binomial deviance	$-\log(1+\exp(-2yf(\mathbf{x}))$
Misclassification	$\mathbb{1}_{\{yf(\mathbf{x})<0\}}$
Squared error	$(y-f(\mathbf{x}))^2$
Support vector	$\max(0,1-yf(\mathbf{x}))$

Classification: Loss Functions

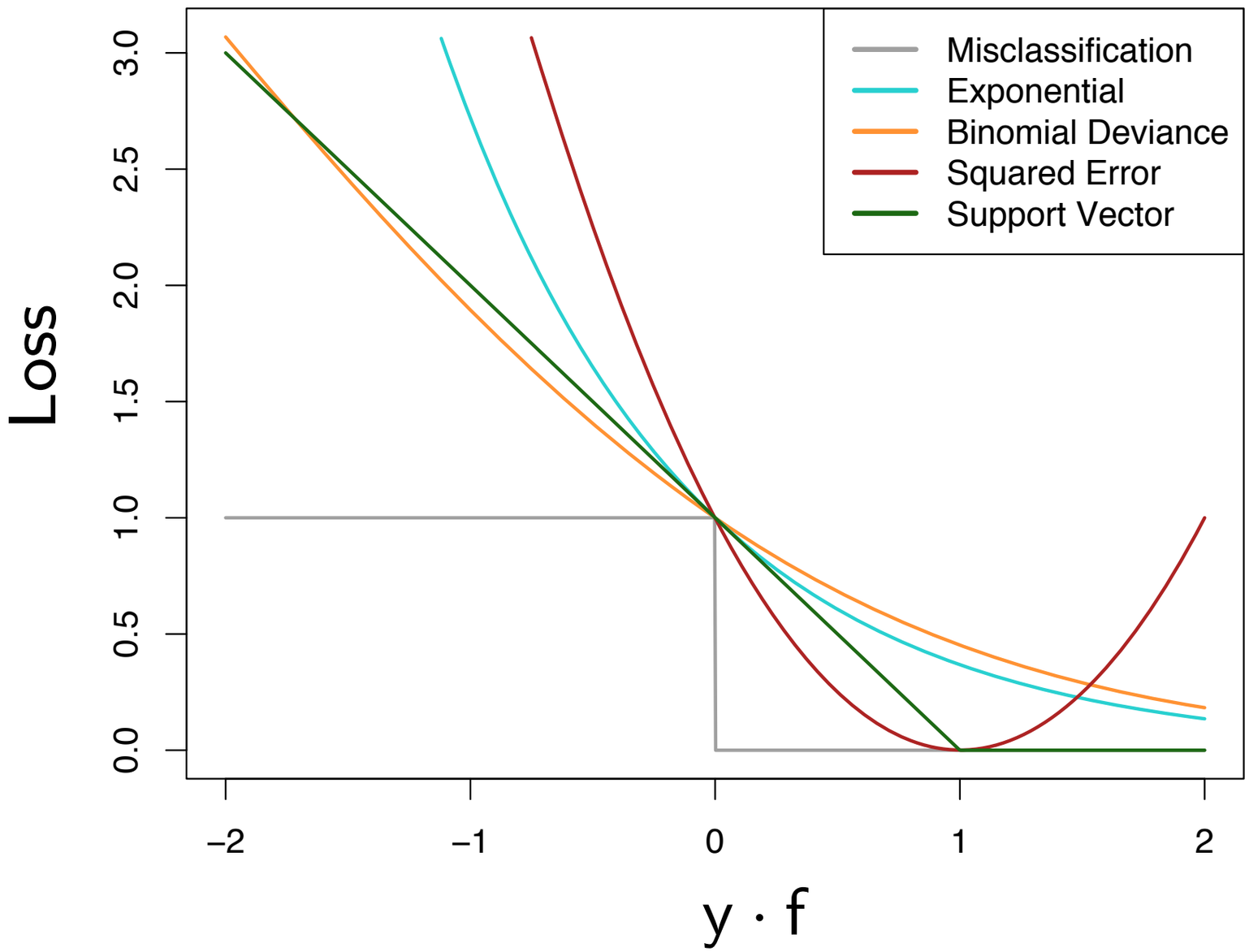


Figure 10.4 (Hastie et al.)

Classification: Margin

- Several of the loss functions are functions of the "margin": $y f(\mathbf{x})$
  - Positive margin are correct
  - Negative margin are misclassified
- Classification algorithms attempt to produce positive margin for each training data point — should penalize negative margins more heavily

Classification: Robust Loss

- Exponential and deviance are continuous approximations to misclassification loss
  - Binomial deviance penalty increases linearly with negative margin
  - Exponential loss penalty increases exponentially with negative margin
- Binomial criterion far more robust than exponential in noisy settings

Gradient Boosting

- Gradient descent + boosting
- Powerful algorithm that can be used for regression, classification, ranking
- Data mining competition winner most likely uses this algorithm

Gradient Boosting: Regression

- We have training data, (x_1, y_1), …, (x_n, y_n), and task to fit model f(x) to minimize square loss

- Friend gives you a model f, with some mistakes

- You are not allowed to remove anything from f or change any parameter in f

- You are allowed to add additional models h to f, so new prediction is f(x) + h(x)

Gradient Boosting: Regression

- Simple solution:

$$\begin{aligned} f(x_1) + h(x_1) &= y_1 \\ f(x_2) + h(x_2) &= y_2 \\ &\vdots \\ f(x_n) + h(x_n) &= y_n \end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned} h(x_1) &= y_1 - f(x_1) \\ h(x_2) &= y_2 - f(x_2) \\ &\vdots \\ h(x_n) &= y_n - f(x_n) \end{aligned}$$

Can this be done using any regression tree?

Gradient Boosting: Regression

- Fit a regression tree to new data, $(x_1, y_1\text{-}f(x_1)), \ldots, (x_n, y_n\text{-}f(x_n))$

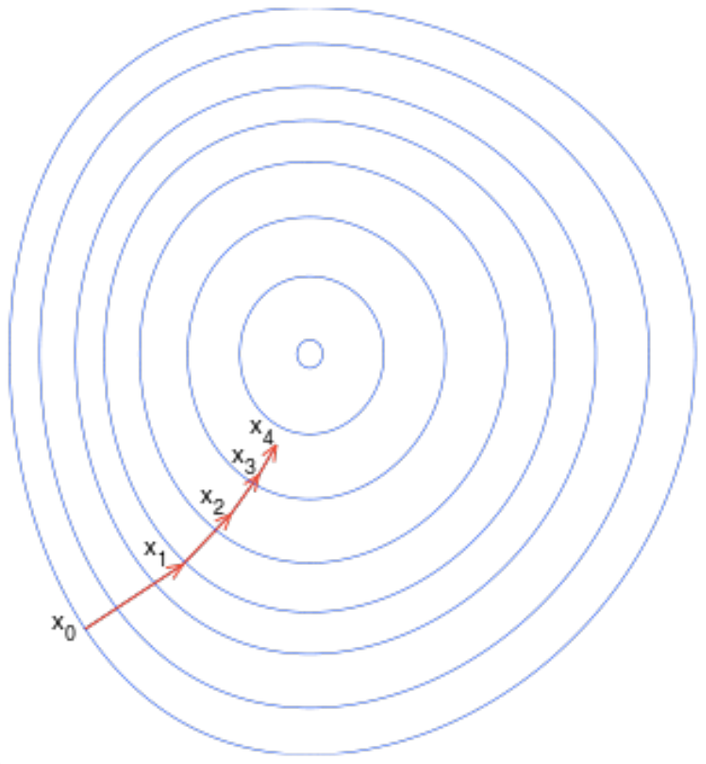
- $y_i\text{-}f(x_i)$ are residuals — parts that existing model f cannot do well

- If new model f + h is still not satisfactory, reiterate again

How does this relate to gradient descent?

Review: Gradient Descent

- Simplest and extremely popular



- Idea: Take a step proportional to the negative of the gradient

$$\theta_i := \theta_i - \eta \frac{\partial L}{\partial \theta_i}$$

Gradient Boosting: Regression

- Loss function:

$$L(\mathbf{y}, f(\mathbf{X})) = \sum_i \frac{1}{2}(y_i - f(\mathbf{x}_i))^2$$

- Treat f($\mathbf{x}_i$) as parameters and take derivatives

$$\frac{\partial L}{\partial f(\mathbf{x_i})} = f(\mathbf{x_i}) - y_i$$

- Interpret residuals as negative gradients

$$f(\mathbf{x_i}) := f(\mathbf{x_i}) + y_i - f(\mathbf{x_i})$$

$$f(\mathbf{x_i}) := f(\mathbf{x_i}) - 1\frac{\partial L}{\partial \mathbf{x}_i}$$

Gradient Boosting: Algorithm

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

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

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Regression: Loss Functions

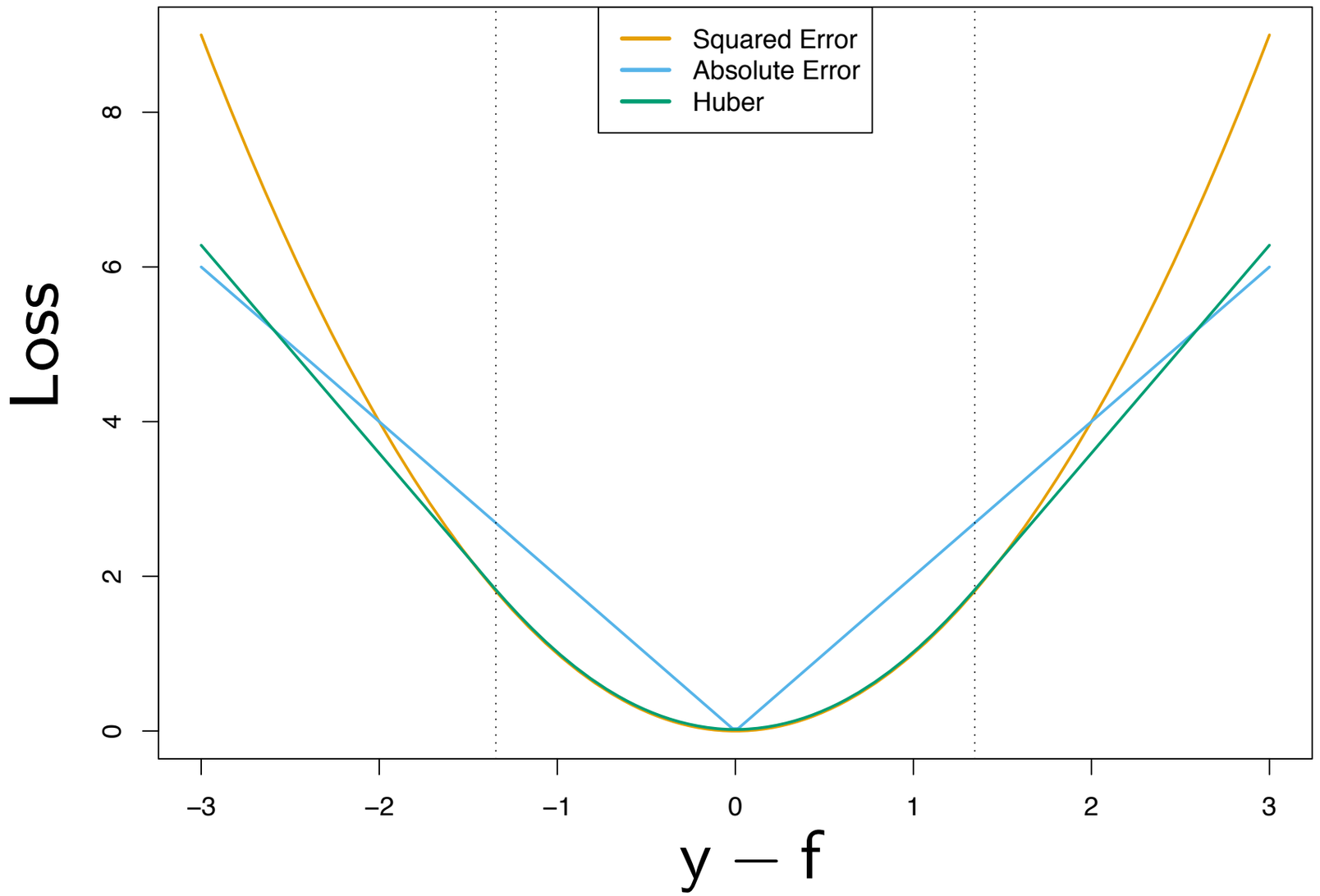


Figure 10.5 (Hastie et al.)

Regression: Robust Loss

- Squared error puts more emphasis on observations with large deviation than absolute loss

- Squared error is less robust and performance degrades for long-tailed error distributions and mislabelings

- Huber loss has strong resistance to gross outliers with efficiency of least squares for Gaussian errors

Gradient Boosting: Regression

	Loss function	Negative gradient
Squared Loss	$\frac{1}{2}(y_i - f(\mathbf{x_i}))^2$	$y_i - f(\mathbf{x}_i)$
Absolute Loss	$\lvert y - f(\mathbf{x})\rvert$	$\text{sign}(y_i - f(\mathbf{x}_i))$
Huber Loss	$\begin{cases} \frac{1}{2}(y_i - f(\mathbf{x_i}))^2 & \lvert y - f(\mathbf{x})\rvert \leq \delta \\ \delta(\lvert y - f(\mathbf{x})\rvert - \delta/2) & \lvert y - f(\mathbf{x})\rvert > \delta \end{cases}$	$\begin{cases} y_i - f(\mathbf{x}_i) & \lvert y - f(\mathbf{x})\rvert \leq \delta \\ \delta\text{sign}(y_i - f(\mathbf{x}_i)) & \lvert y - f(\mathbf{x})\rvert > \delta \end{cases}$

Gradient Boosting: Classification

- Regression setting can be easily adapted for classification
- Generalization of Adaboost to general classification loss functions

Gradient Boosting vs AdaBoost

- Boosting: Fit additive model in a forward stage-wise manner where each stage, new weak learner compensates shortcomings of existing models

- Gradient boosting — “shortcomings” identified by gradient

- Adaboost — “shortcomings” identified by high-weight data points

Size of Trees

- Best method is to grow small trees with no pruning

- Right size will depend on level of interaction between variables

- Generally 2-8 leaves works well

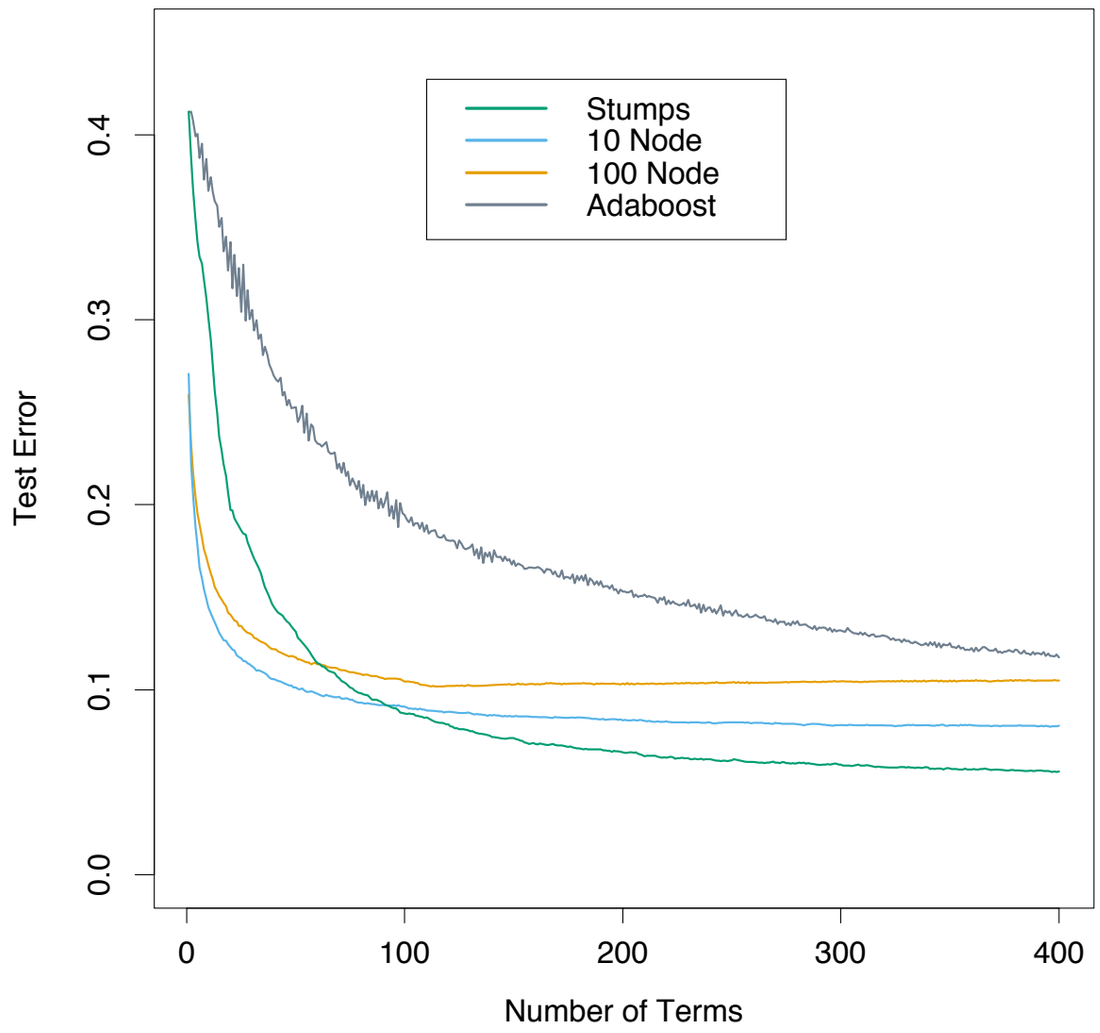


Figure 10.9 (Hastie et al.)

Trees: Variable Importance

- Squared importance for variable j

$$\mathrm{Imp}_j^2(\hat{f}^{\mathrm{tree}}) = \sum_{k=1}^{m} \hat{d}_k \mathbb{1}_{\{\text{split at node } k \text{ is on variable j}\}}$$

- m is number of internal modes (non-leaves)

- $\hat{d}_k$ is the improvement in training misclassification error from making the nth split

Boosting: Variable Importance

- Average squared importance over all fitted trees

$$\mathrm{Imp}_j^2(\hat{f}^{\mathrm{boost}}) = \frac{1}{M}\sum_{m=1}^{M}\mathrm{Imp}_j^2(\hat{f}_m^{\mathrm{tree}})$$

- Stabilizes variable importances —> more accurate than for single tree

- Relative importance: Scale largest importance to 100 and scale all other variable importances accordingly

Example: Spam Data

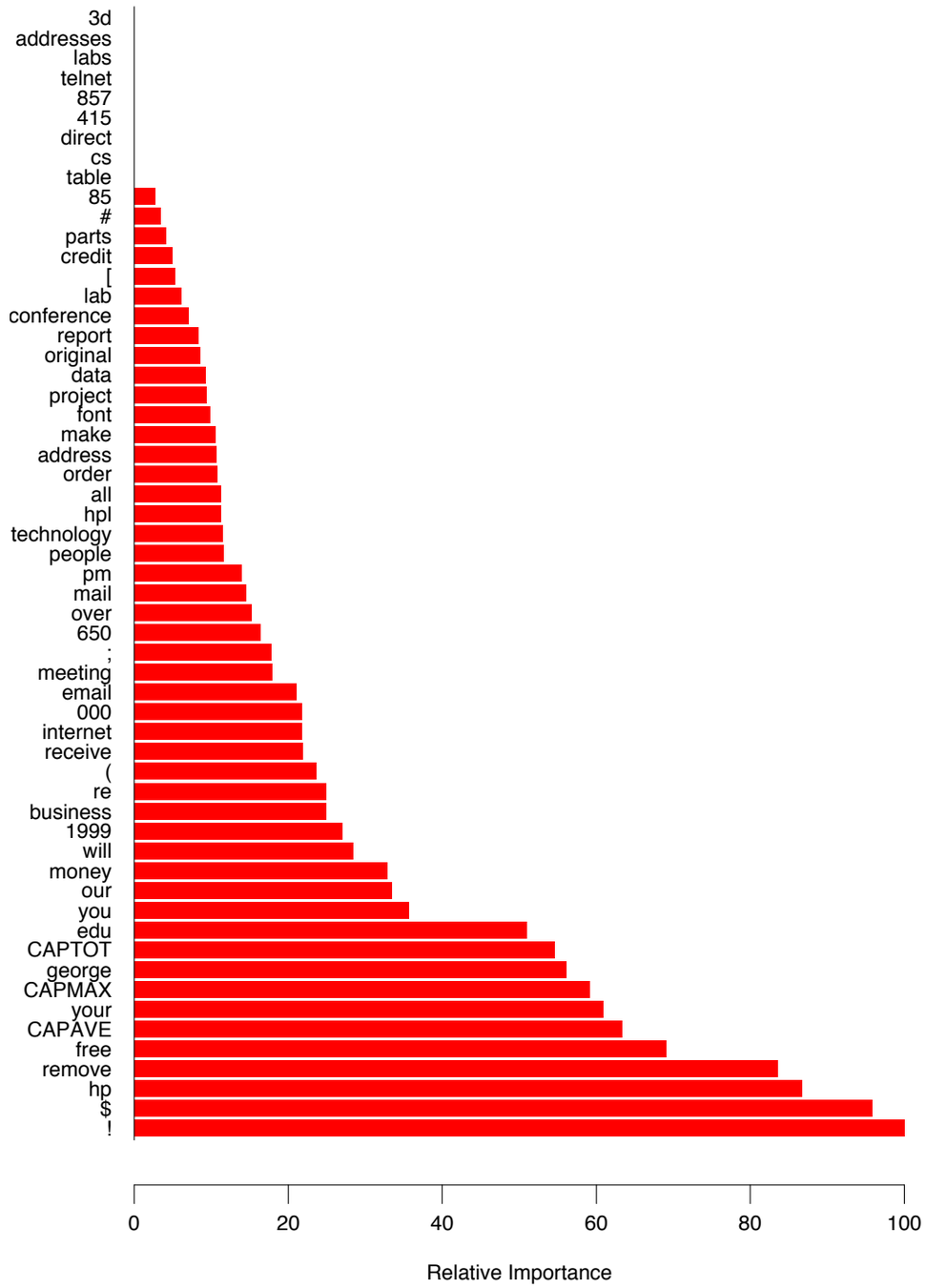


Figure 10.6 (Hastie et al.)

Boosting: Advantages

- Provably effective — reduces both bias and variance
- Easy — limited number of parameters to tune
- Flexible — combine with any learning algorithm
- Versatile — can be used with data that is numeric, discrete, textual

Boosting: Disadvantages

- Lack of interpretability — lose benefit of classification tree
- Serializability — computation can be difficult and hard to perform in parallel
- Performance dependent on weak learner and data — can fail if weak classifiers are too complex or too weak
- Noise susceptibility — empirically seems susceptible to uniform noise

Extreme Gradient Boosting (XGBoost)

Homesite Quote Conversion, Winners' Interview: 3rd place, Team New Model Army | CAD & QuY

Kaggle Team | 02.29.2016

Prudential Life Insurance Assessment, Winner's Interview: 2nd place, Bogdan Zhurakovskyi

Kaggle Team | 03.14.2016

Airbnb New User Bookings, Winner's Interview: 2nd place, Keiichi Kuroyanagi (@Keiku)

Kaggle Team | 03.17.2016

Telstra Network Disruption, Winner's Interview: 1st place, Mario Filho

Kaggle Team | 03.23.2016

What these various data mining competitors have in common: all used XGBoost

CS 534 [Spring 2017] - Ho

Why XGBoost?

- Implements basic idea of GBM with some tweaks
  - Regularization of base tree
  - Approximate split finding
  - Weighted quantile sketch
  - Sparsity-aware split finding
  - Cache-aware block structure for out of core computation

"XGBoost scales beyond billions of examples using far fewer resources than existing systems."

–T. Chen and C. Guestrin