Tree-based methods

Hastie, Tibshirani, Friedman Ch 9.2, Ch 10

Kevin Murphy Ch. 16.1-16.4

James, Witten, Hastie, Tibshirani Ch 8

CS 6140

Machine Learning

Professor Olga Vitek

April 6, 2017

Decision trees

See handout (no author)

Overview: recursive partitioning

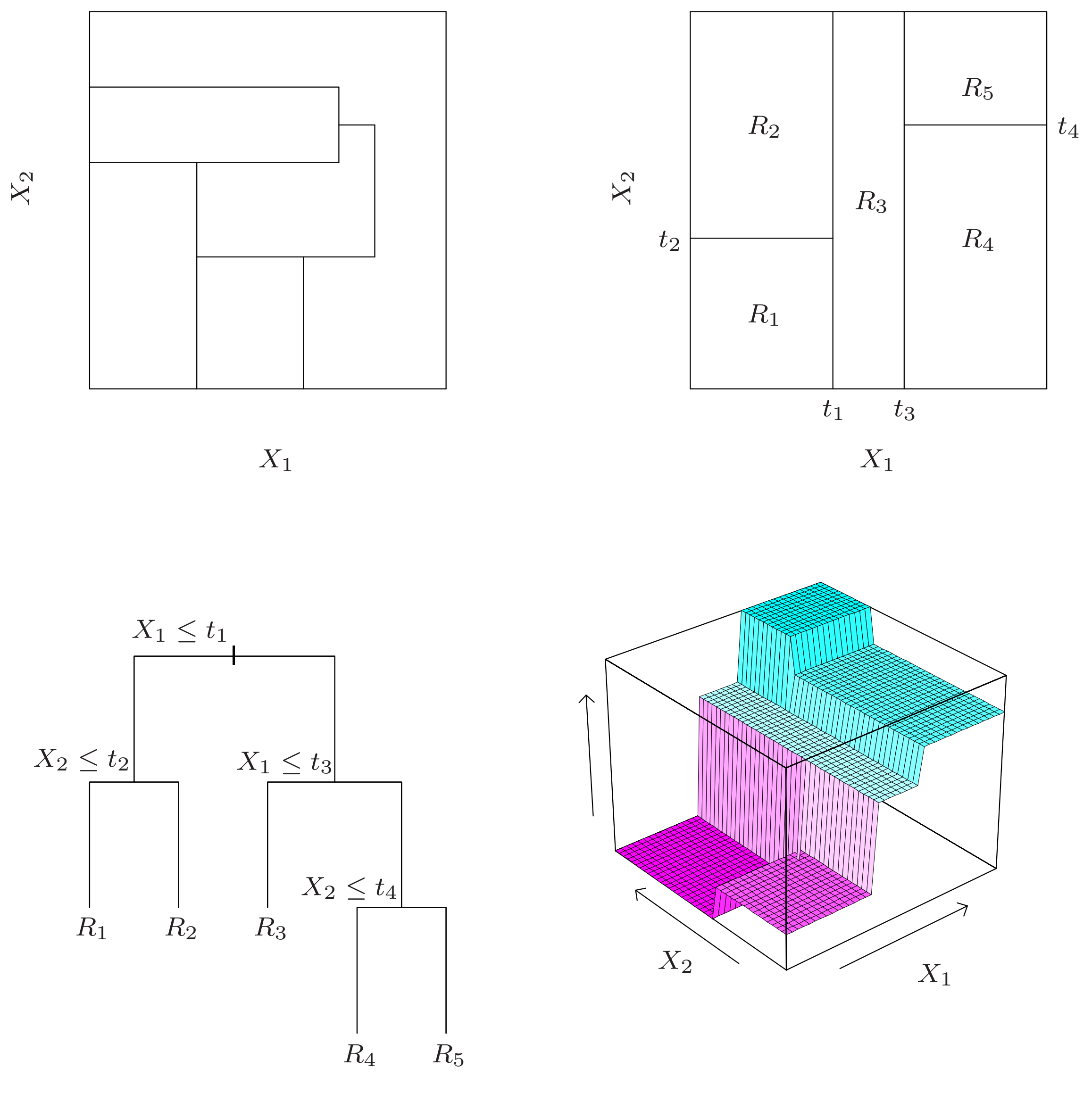


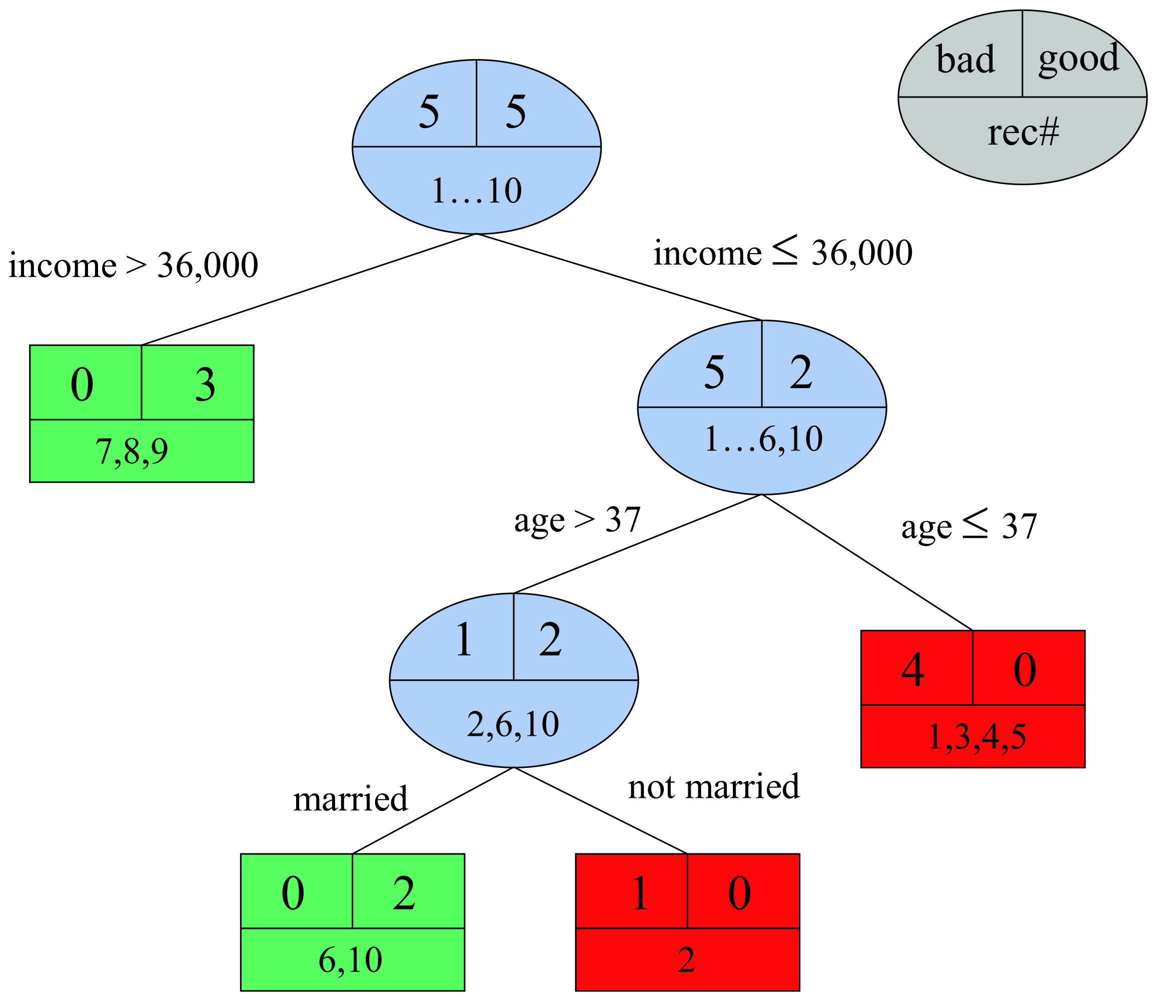
Fig. 9.2. Hastie, Tibshirani, Friedman
The Elements of Statistical Learning 2008

Example: categorical response credit score classification

Record	age	married?	own house	income	gender	class
1	22	no	no	28,000	male	bad
2	46	no	yes	32,000	female	bad
3	24	yes	yes	24,000	male	bad
4	25	no	no	27,000	male	bad
5	29	yes	yes	32,000	female	bad
6	45	yes	yes	30,000	female	good
7	63	yes	yes	58,000	male	good
8	36	yes	no	52,000	male	good
9	23	no	yes	40,000	female	good
10	50	yes	yes	28,000	female	good

Anonymous. See handout.

Credit score classification



Anonymous. See handout.

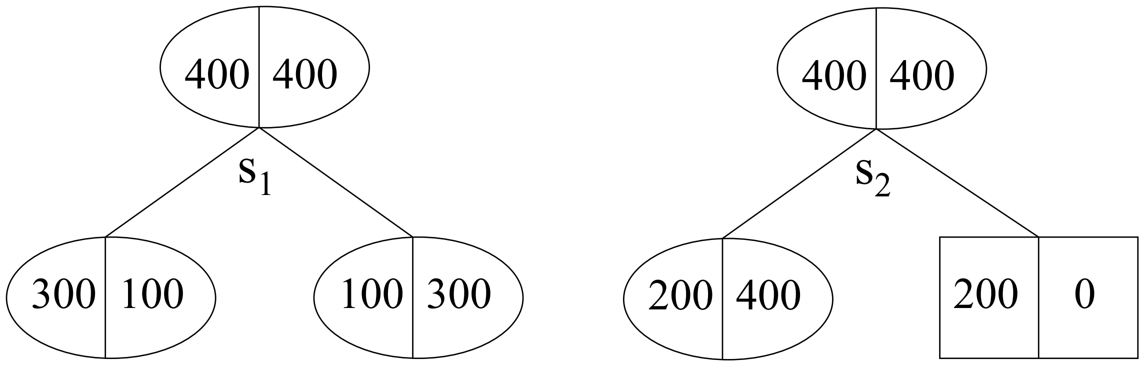
Quality of split

- Choose split to minimize node "impurity"
  - Define as function of relative class frequencies $i(t) = \phi(p_1, p_2, \ldots, p_J)$ with J classes
  - $i(t)$ maximized at $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$
  - $i(t)$ minimized at $(0, 0, \ldots, 1)$ (for some class)
  - $i(t)$ is symmetric function of $(p_1, p_2, \ldots, p_J)$

- Quality of split s at node t
  - $\Delta i(s,t) = i(t) - \pi(l)\, i(l) - \pi(r)\, i(r)$, where
    $\pi(l)$ is the proportion of points sent to the left
    $\pi(r)$ is the proportion of points sent to the right

Measures of node impurity

- Resubstitution error: $i(t) = 1 - \max_j p(j|t)$
  - $p(j|t)$: relative frequency of class j in node t
  - $i(t)$: % of misclassified cases
  - Node impurity simplifies to $\Delta i(s,t) =$ $\max_j p(j|l)\,\pi(l) + \max_j p(j|r)\,\pi(r) - \max_j p(j|t)$
  - Problem: ignores where misclassification occurs
  - Below: same impurity; prefer split to the right
  - Ideally, ϕ would be concave (impurity would decrease faster than linearly)



Left: $\Delta i = 1 - \frac{1}{2} - \frac{1}{2}\left(1 - \frac{3}{4}\right) - \frac{1}{2}\left(1 - \frac{3}{4}\right) = 0.25$

Right: $\Delta i = 1 - \frac{1}{2} - \frac{3}{4}\left(1 - \frac{2}{3}\right) - \frac{1}{4}\left(1 - 1\right) = 0.25$

Concave impurity measures

- Gini index
  - Two classes:

    $$i(t) = p(0|t) \cdot p(1|t) = p(0|t) \cdot (1 - p(0|t))$$

  - Multiple classes

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

  - Variance of Bernoulli drawing from this class

    Impurity reduction $\rightarrow$ variance reduction

- Entropy
  - Two classes:

    $$i(t) = -p(0|t) \cdot log\, p(0|t) - p(1|t) \cdot log\, p(1|t)$$

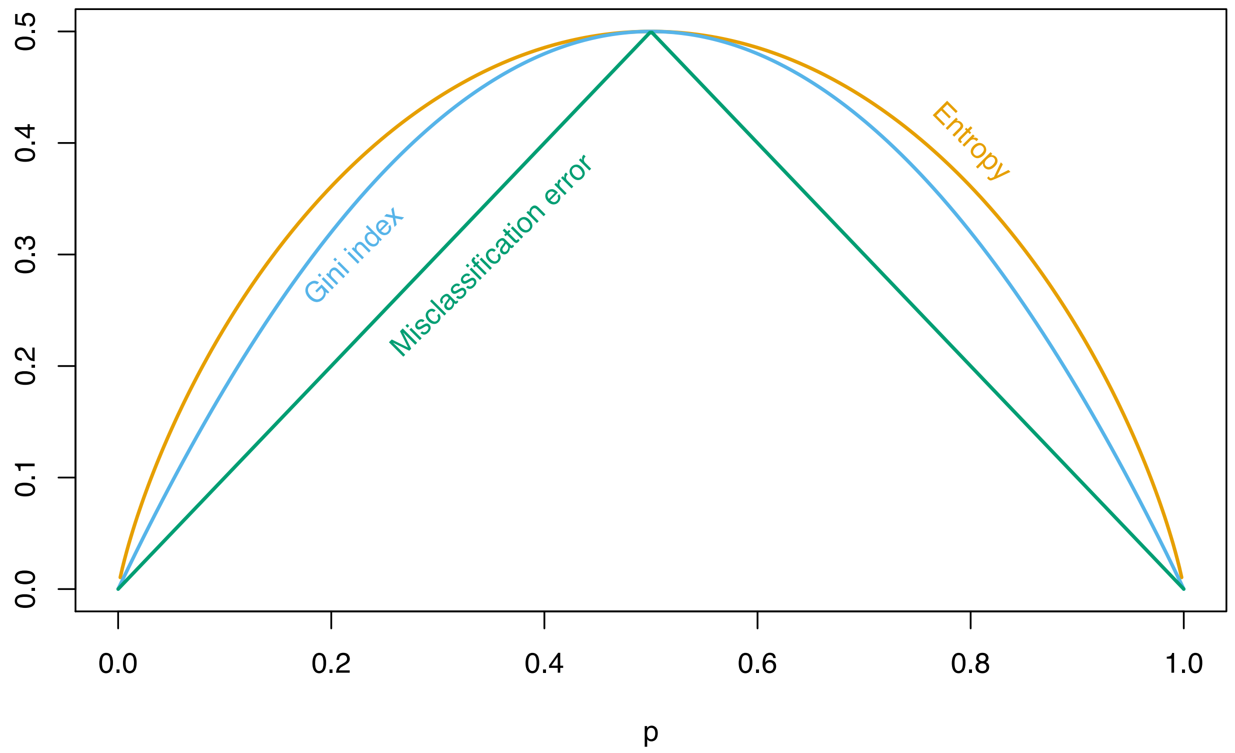
  - Multiple classes

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

  - Average amount of info gathered by drawing (with replacement) a point from the node and recording its class.
    $\rightarrow$ Max impurity reduction = min loss of info.

- Low impurity only for very “pure” nodes

Measures of node impurity



Two-class example; Entropy scaled to max=0.5

General properties:

- $\phi(0) = \phi(1) = 0$
- $\phi(p) = \phi\,(p(1-p)$
- $\phi''(p) < 0,\ 0 < p < 1$

Fig. 9.3. Hastie, Tibshirani, Friedman
The Elements of Statistical Learning 2008

Search for splits (Gini)

Income	Class	Quality (split after) 0.25−
24	B	$0.1(1)(0)+0.9(4/9)(5/9) = 0.03$
27	B	$0.2(1)(0) + 0.8\ (3/8)(5/8) = 0.06$
28	B,G	$0.4(3/4)(1/4) + 0.6(2/6)(4/6) = 0.04$
30	G	$0.5(3/5)(2/5) + 0.5(2/5)(3/5) = 0.01$
32	B,B	$0.7(5/7)(2/7) + 0.3(0)(1) = 0.11$
40	G	$0.8(5/8)(3/8) + 0.2(0)(1) = 0.06$
52	G	$0.9(5/9)(4/9) + 0.1(0)(1) = 0.03$
58	G	

- Split on one predictor at a time
  - X numeric:
    $X \leq constant$ for $constant$ in range of X
    In practice, split wrt distinct values of X
  - X categorical with values in V:
    $X \in S$, S any subset of V
  - Number of possible splits is finite

Anonymous. See handout.

Tree construction

Algorithm: Construct tree

```
nodelist $\leftarrow$ {training sample}
Repeat
    current node $\leftarrow$ select node from nodelist
    nodelist $\leftarrow$ nodelist $-$ current node
    if impurity(current node) $> 0$
    then
        S $\leftarrow$ candidate splits in current node
        s* $\leftarrow \arg\max_{s \in S}$ impurity reduction(s,current node)
        child nodes $\leftarrow$ apply(s*,current node)
        nodelist $\leftarrow$ nodelist $\cup$ child nodes
    fi
Until nodelist $= \varnothing$
```

- Local greedy search; globally suboptimal

  Example: true labels $PxorQ$ can't be classified by splitting on P or Q

- (Partial) remedy: grow maximal tree, then prune

P	Q	P *xor* Q
1	*1*	*0*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

Cost-complexity pruning

- Notation
  - Tree T, and maximal tree T_{max}
  - $R(T)$ resubstitution error on training set
  - $|T|$ tree size (i.e. # of terminal nodes)
  - Total cost of the tree $C_\alpha(T) = R(T) + \alpha|T|$
  - α: parameter penalizing tree complexity

- For every α, there exists a smallest subtree $T(\alpha)$ of T_{max}, such that:
  - No subtree of T_{max} has lower cost than $T(\alpha)$:
    $C_\alpha\left(T(\alpha)\right) = min_{T \leq T_{max}} C_\alpha(T)$
  - If there is a tie, we pick the smallest tree:
    If $C_\alpha(T) = C_\alpha\left(T(\alpha)\right)$, then $T(\alpha) \leq T$

- Consequence
  - It is impossible to have two non-nested subtrees with same min cost
  - Now vary α. Although α is continuous, only need a final set that changes tree structure
    $\{\alpha_1, \alpha_2, \ldots\} \rightarrow T_1 > T_2 > \ldots \{t_1\}$

Cost-complexity pruning

- T_1: min subtree of T_{max} with same resubstitution error
- Cost of T_t:
  - as an intermediate node: $C_\alpha(T_t) = R(T_t) + \alpha|T_t|$
  - as a terminal node: $C_\alpha(t) = R(t) + \alpha \cdot 1$
- The pruned tree has the same cost-complexity as the original tree when $C_\alpha(t) = C_\alpha(T_t)$

$$\begin{aligned} R(T_t) + \alpha|T_t| &= R(t) + \alpha \cdot 1 \\ \alpha &= \frac{R(t) - R(T_t)}{|T| - 1} \end{aligned}$$

  - For any t, when we increase α beyond this level, pruned tree is better
  - Obtain the next tree by pruning the current
  - T_k is the smallest minimizing subtree for $\alpha \in [\alpha_k, \alpha_{k+1})$
- There are $\lfloor 1.5028369^{|T|} \rfloor$ pruned trees

Cost-complexity pruning

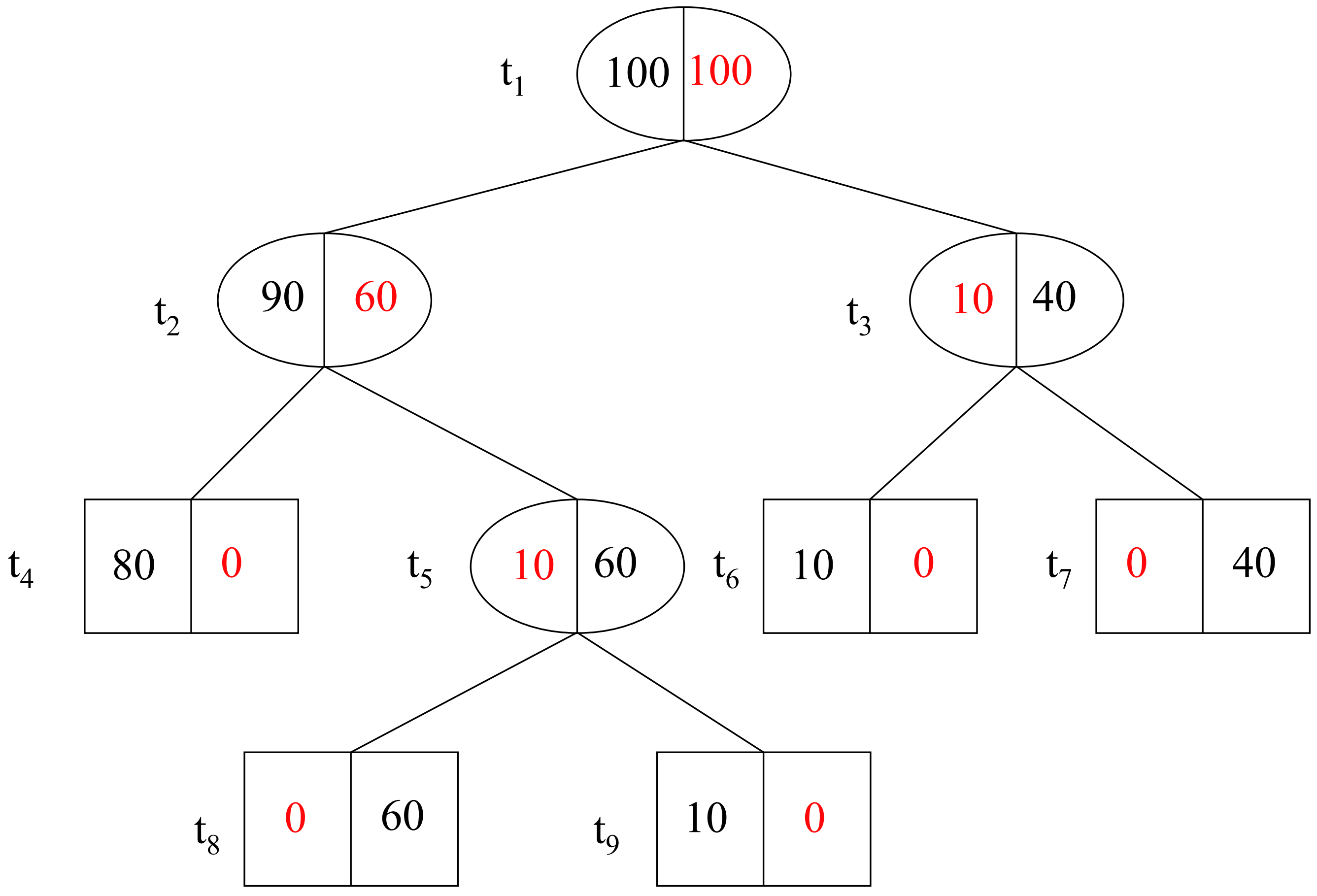
Algorithm: Compute T_1 from T_{max}

```
    $T' \leftarrow T_{max}$
    Repeat
        Pick any pair of terminal nodes $\ell$ and $r$ with common parent $t$ in $T'$
        such that $R(t) = R(\ell) + R(r)$, and set
        $T' \leftarrow T' - T_t$ (i.e. prune $T'$ in $t$)
    Until no more such pair exists
    $T_1 \leftarrow T'$
```

Algorithm: Compute tree sequence

```
    $T_1 \leftarrow T(0)$
    $\alpha_1 \leftarrow 0$
    $k \leftarrow 1$
    While $T_k > \{t_1\}$ do
        For all non-terminal nodes $t \in T_k$
            $g_k(t) \leftarrow \frac{R(t)-R(T_{k,t})}{(|\tilde{T}_{k,t}|-1)}$
        $\alpha_{k+1} \leftarrow \min_t g_k(t)$
        Visit the nodes in top-down order and prune
        whenever $g_k(t) = \alpha_{k+1}$ to obtain $T_{k+1}$
        $k \leftarrow k+1$
    od
```

Example: initial tree



Example: sequential pruning

- $k = 1$

$$\begin{aligned}
g_1(t_5) &= \frac{\frac{70}{200}\cdot\frac{10}{70} - \frac{70}{200}\cdot\frac{60}{70}\cdot 0 - \frac{70}{200}\cdot\frac{10}{70}\cdot 0}{2-1} = \frac{1}{20} \\
g_1(t_3) &= \frac{\frac{50}{200}\cdot\frac{10}{50} - \frac{50}{200}\cdot\frac{10}{50}\cdot 0 - \frac{50}{200}\cdot\frac{40}{50}\cdot 0}{2-1} = \frac{1}{20} \\
g_1(t_2) &= \frac{\frac{150}{200}\cdot\frac{60}{150} - \frac{150}{200}\cdot\frac{80}{150}0 - \frac{150}{200}\cdot\frac{70}{150}\cdot\frac{60}{70}0 - \frac{150}{200}\cdot\frac{70}{150}\cdot\frac{10}{70}0}{3-1} \\
&= \frac{3}{20} \\
&\Rightarrow \text{ prune } t_3 \text{ and } t_5
\end{aligned}$$

- $k = 2$

$$\begin{aligned}
g_2(t_2) &= \frac{\frac{150}{200}\cdot\frac{60}{150} - \frac{150}{200}\cdot\frac{80}{150}\cdot 0 - \frac{150}{200}\cdot\frac{70}{150}\cdot\frac{10}{70}}{2-1} = \frac{5}{20} \\
g_2(t_1) &= \frac{1\cdot\frac{100}{200} - \frac{80}{200}\cdot 0 - \frac{70}{200}\cdot\frac{10}{70} - \frac{50}{200}\cdot\frac{10}{50}}{3-1} = \frac{4}{20} \\
&\Rightarrow \text{ prune } t_1
\end{aligned}$$

- α: $\left\{\alpha_1 = 0,\ \alpha_2 = \frac{1}{20},\ \alpha_3 = \frac{4}{20}\right\}$

Selecting optimal tree

Algorithm:

- Construct a full tree on the full dataset
- Compute $\{\alpha_1, \alpha_2, \ldots\}$
- Obtain $\{T_1, T_2, \ldots, t\}$ for each α
- Define sequence $\{\beta_j = \sqrt{\alpha_{j-1} \cdot \alpha_j}\}$
- Divide data into folds
- Within each fold:
  - Build a sequence of trees over β_j
  - Compute prediction error for left-out observations
- For each β_j, sum # misclassifications over folds
- Select β_j with min # mis-classifications
- Report the β_j sub-tree of the full data

Challenges: Unstable (i.e., high variance), not smooth, can't capture additive structures

Bagging (=bootstrap aggregation)

- Motivation
  - Average prediction on B independent datasets $\downarrow$ prediction variance:

$$\hat{f}_{avg} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x),\ Var\{\hat{f}_{avg}\} = \frac{Var\{\hat{f}^b(x)\}}{B}$$

- Bootstrap (HTF Sec 8.7)
  - Mimics by sampling with replacement B datasets from the original dataset to $\downarrow$ prediction variance
  - Grows maximal trees

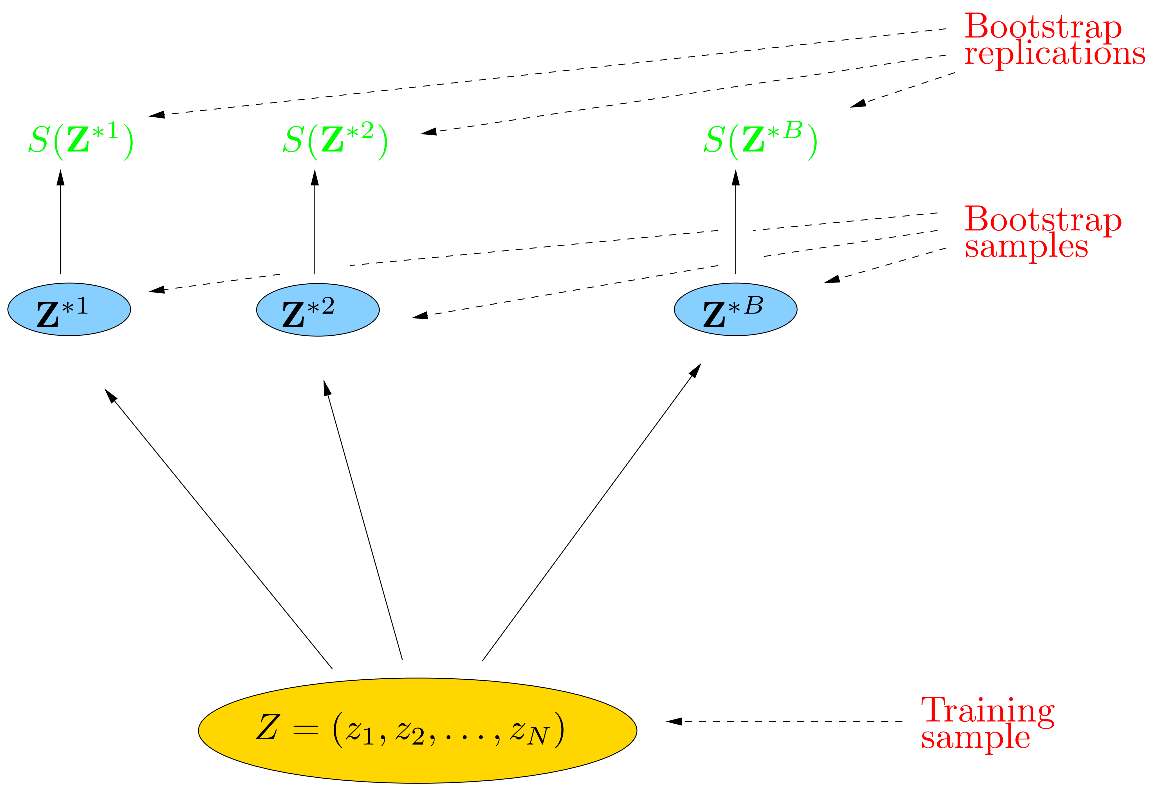
- Features

  + *Out-of-bag error:* in each iteration b, use left-out observations to estimate prediction error (for each observation, average the error over all times when it was left out)

  + *Variable importance:* total amount of decrease in impurity or RSS by splitting on the predictor (averaged over trees)

  − *Loss of interpretation:* a bagged tree $\neq$ a tree

Bootstrap



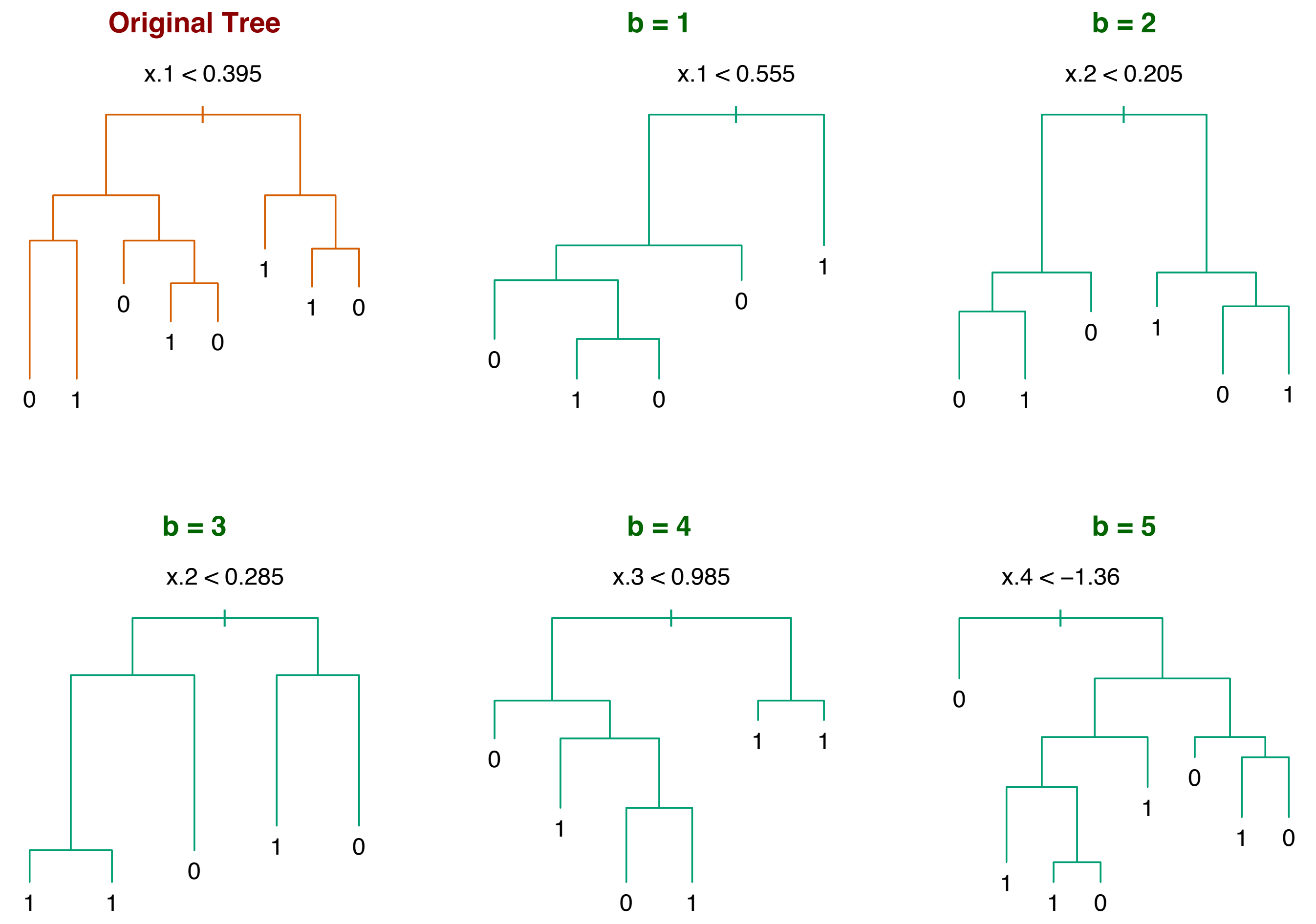
$$\hat{f}_{bag} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Under square loss, for a classifier $\hat{f}^*(x)$ (HTF Sec 8.7.1):

$$\begin{aligned} E\{Y - \hat{f}^*(x)\}^2 &= E\left\{Y - \hat{f}_{bag}(x) + \hat{f}_{bag}(x) - \hat{f}^*(x)\right\}^2 \\ &= E\left\{Y - \hat{f}_{bag}(x)\right\}^2 + E\left\{\hat{f}_{bag}(x) - \hat{f}^*(x)\right\}^2 \\ &\geq E\left\{Y - \hat{f}_{bag}(x)\right\}^2 \end{aligned}$$

Does not always hold for 0-1 cost (misclassification)

Simulation



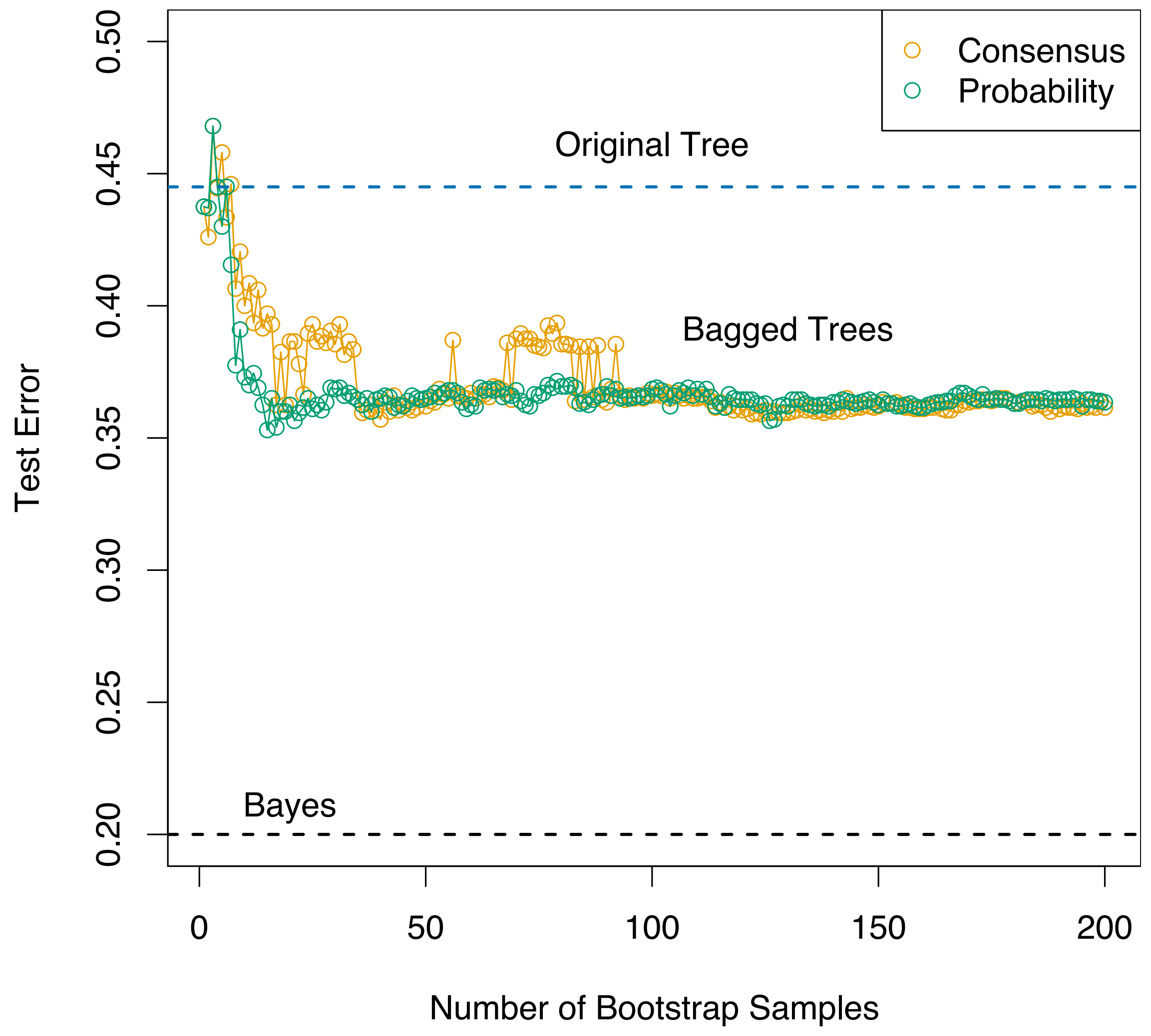
$N = 30$; up to 200 bootstrap samples

True model:

$P\{Y = 1|x_1 \leq 0.5\} = 0.2$ and $P\{Y = 1|x_1 > 0.5\} = 0.8$

HTF Fig. 8.9

Simulation



$N = 30$; up to 200 bootstrap samples

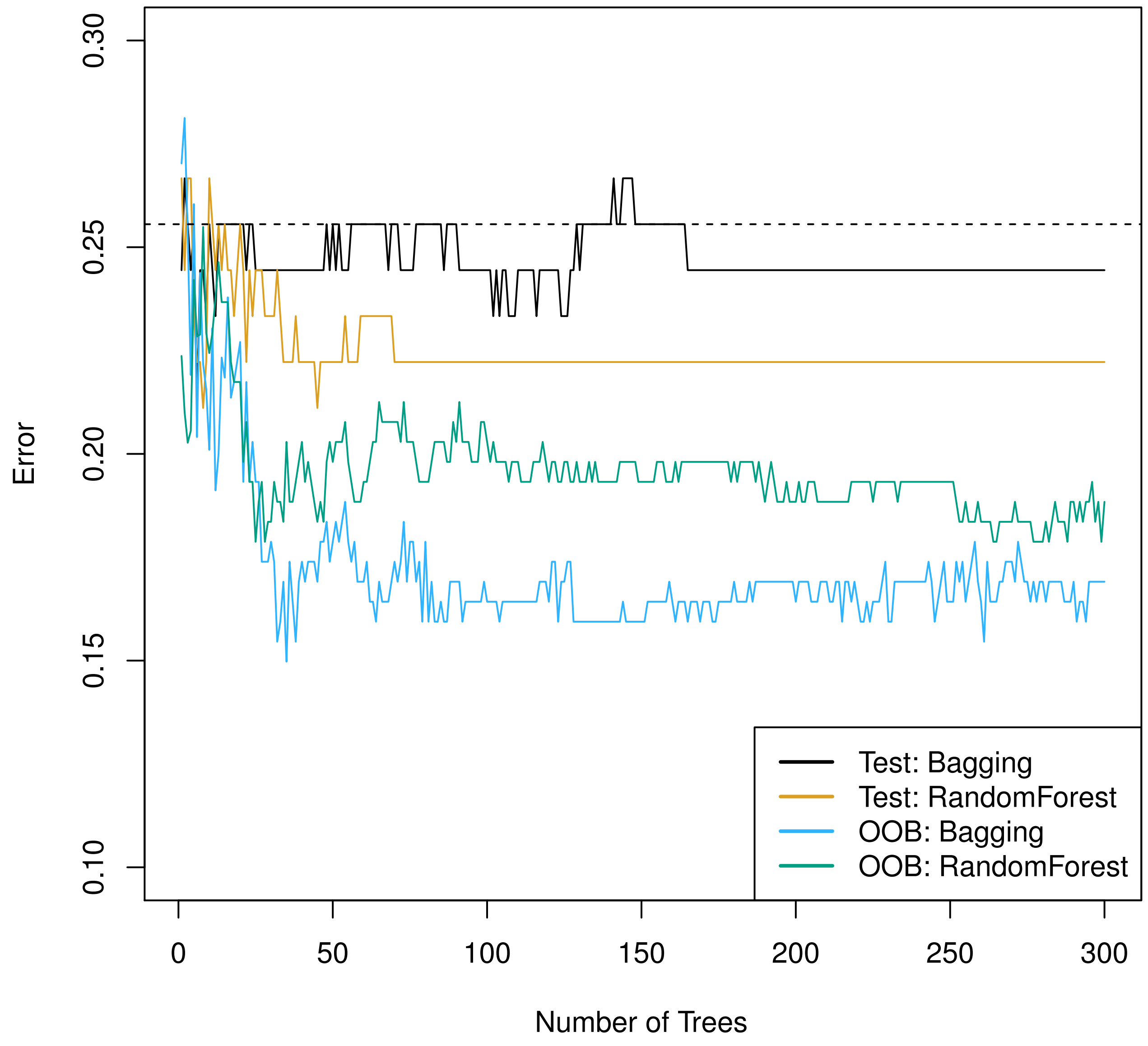
True model:

$P\{Y = 1|x_1 \leq 0.5\} = 0.2$ and $P\{Y = 1|x_1 > 0.5\} = 0.8$

Random forest

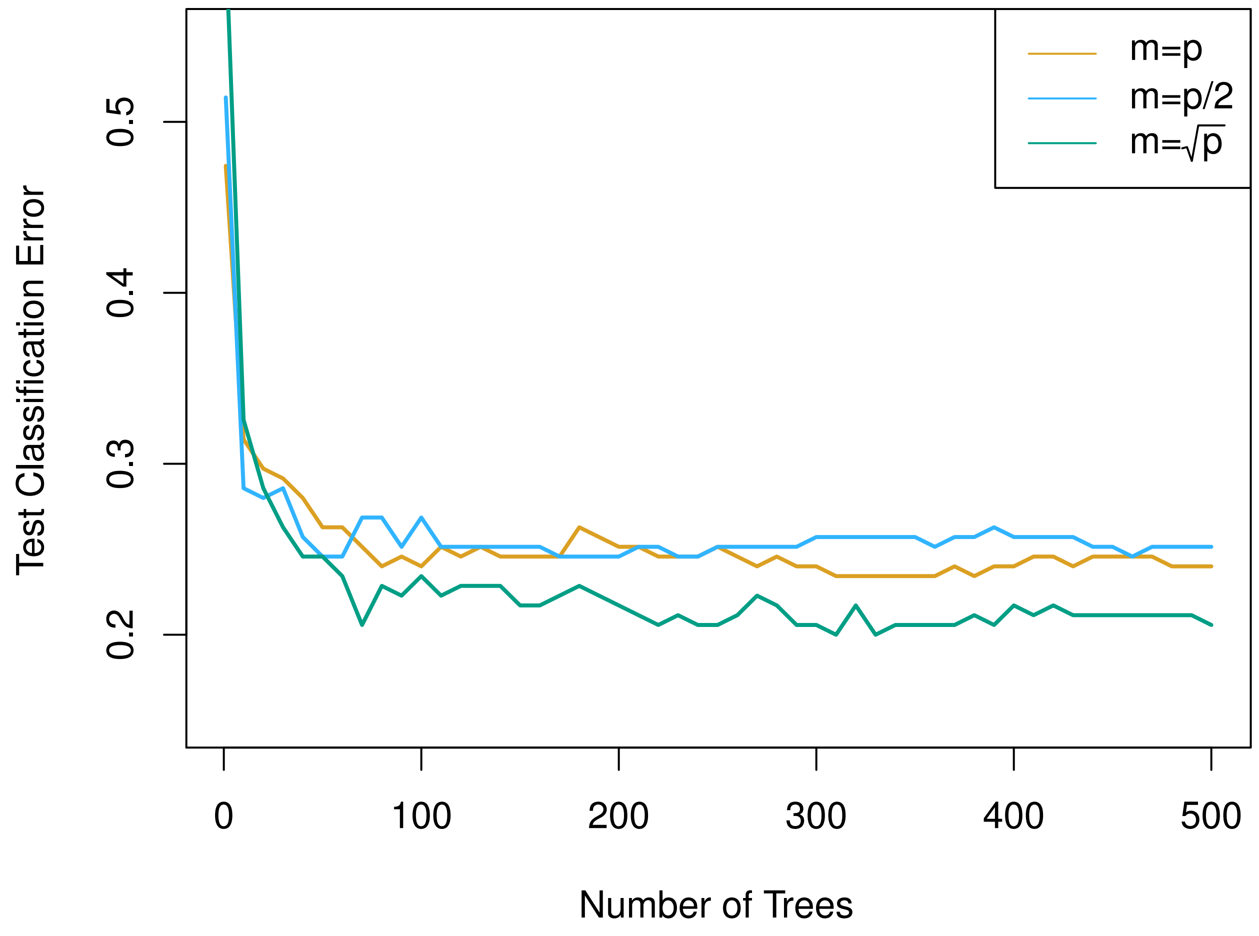
- Motivation
  - In bagging, prediction from trees can be *correlated* in presence of single dominant predictor
  - Ave. correlated predictions does not $\downarrow$ variance
- Random forest
  - Introduces diversity in trees
  - Randomly selects $m \approx \sqrt{p}$ predictors per split
  - On average $(p-m)/p$ splits skip strong predictor
  - Random forest with $(m = p)$ = bagging

Example (experimental data)



Dashed line: single tree. OOB is too optimistic here.
JWHT Fig.8.8

Example (experimental data)



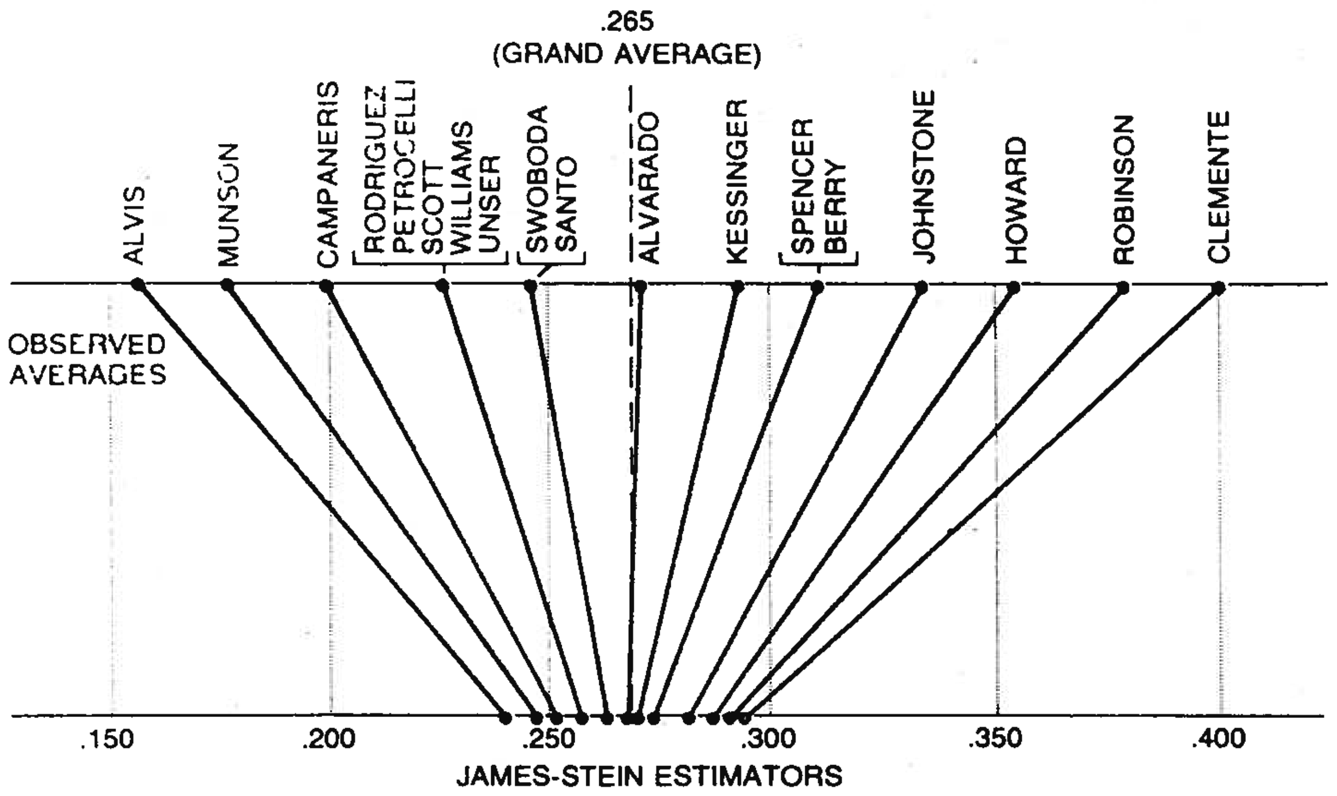
5-class gene expression data set with $p = 500$ predictors. Random forest $(m < p)$ slightly improved over bagging $(m = p)$. Single tree error rate: 45.7 %.

JWHT Fig.8.10

Motivation of shrinkage: James-Stein estimator

- Motivation:
  - Common wisdom: the best guess about the future is average of the past
  - Stein's paradox: average is sub-optimal in multivariate settings
- Set-up: I random variables
  - $Y_1 \overset{iid}{\sim} \mathcal{N}(\mu_1, \sigma^2), \ldots, Y_I \overset{iid}{\sim} \mathcal{N}(\mu_I, \sigma^2)$
  - Goal: jointly estimate $\mu_1, \ldots, \mu_I$ with $\bar{Y}_{1\cdot}, \ldots, \bar{Y}_{I\cdot}$
  - Result: MSE$=E\{||\widehat{\mu} - \mu||^2\}$ is not minimized
- Solution: can reduce MSE by
  - $\widehat{\mu}_i = (1-c) \cdot \bar{\bar{\mu}} + c \cdot \widehat{\mu}_i = \bar{\bar{\mu}} + c \cdot (\widehat{\mu}_i - \bar{\bar{\mu}})$
    – weighted ave. of ith mean and overall mean
  - Here $c = 1 - \frac{(I-3)\cdot\sigma^2}{\sum_{i=1}^{I}(\widehat{\mu}_i - \bar{\bar{\mu}})^2}$
    – For fixed I and σ^2, $c \uparrow$ if between-means variance $(\widehat{\mu}_i - \bar{\bar{\mu}})^2 \uparrow$
- c is the "shrinking" factor

Motivation of shrinkage: James-Stein estimator



$\bar{y} = 0.265$; $c = 0.212$
The shrinkage is $\approx 20\%$

The paper shows that shrunken estimates are closer to the "true" seasonal averages than the average-based estimators.

Efron & Morris, 1977

Extensions of shrinkage

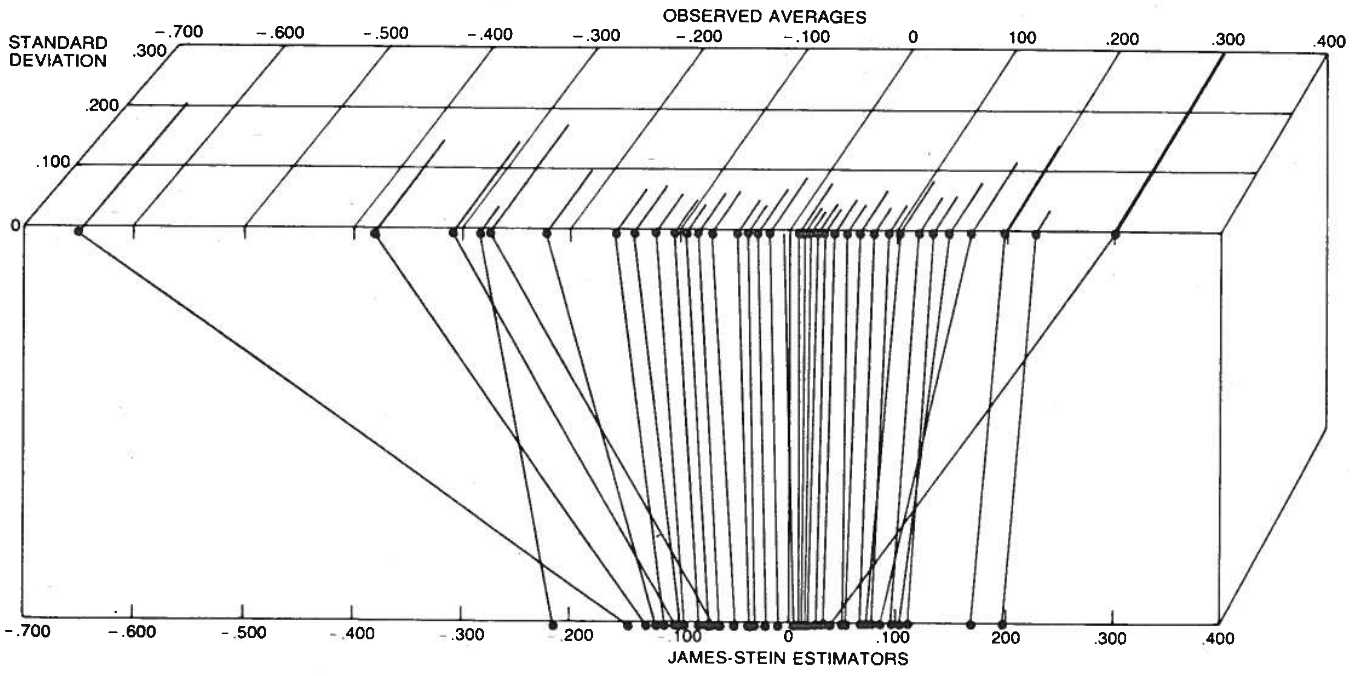
- Shrinking to other target values
  - Any target value besides $\bar{\bar{\mu}}$, e.g. to 0: $\widehat{\mu}_i = c \cdot \widehat{\mu}_i$

- Observations with unequal variance
  - $Y_1 \overset{iid}{\sim} \mathcal{N}(\mu_1, \sigma_1^2), \ldots, Y_I \overset{iid}{\sim} \mathcal{N}(\mu_I, \sigma_I^2)$
  - c inversely related to the variance of the variable
  - More uncertainty $\rightarrow$ more shrinkage
  - If a value is unusually large, it can be more reasonably attributed to random variation than to value of the mean

- Recently extended to non-Normal r.v
- Reduces the overall risk
  - Introduces error when one mean is atypical

- Is equivalent to specifying a prior distribution on the overall mean
  - Sometimes called "empirical Bayes"

Extensions of shrinkage: James-Stein estimator



Shrinking of random variables with unequal variance

Efron & Morris, 1977

Boosting

- Motivation
  - Trees, bagging, RF view obs. as exchangeable
  - Can overlook a few hard-to-classify observations
- Boosting
  - Sequentially grow trees
  - At each step, $\uparrow$ weight of unexplained obs.
  - "Learn slowly"
  - Each next tree depends on the previous tree
- Parameters:
  - Number of trees B
    Large B tends to overfit
  - Weight parameter λ

    λ can be viewed as a shrinkage parameter
  - Number of splits d (interaction depth).

    Tree with $d = 1$ = stump
    If $d = 1$, each tree involves one variable
    $\rightarrow$ additive model
- Generalization:
  - Same approach can be applied to other modeling strategies

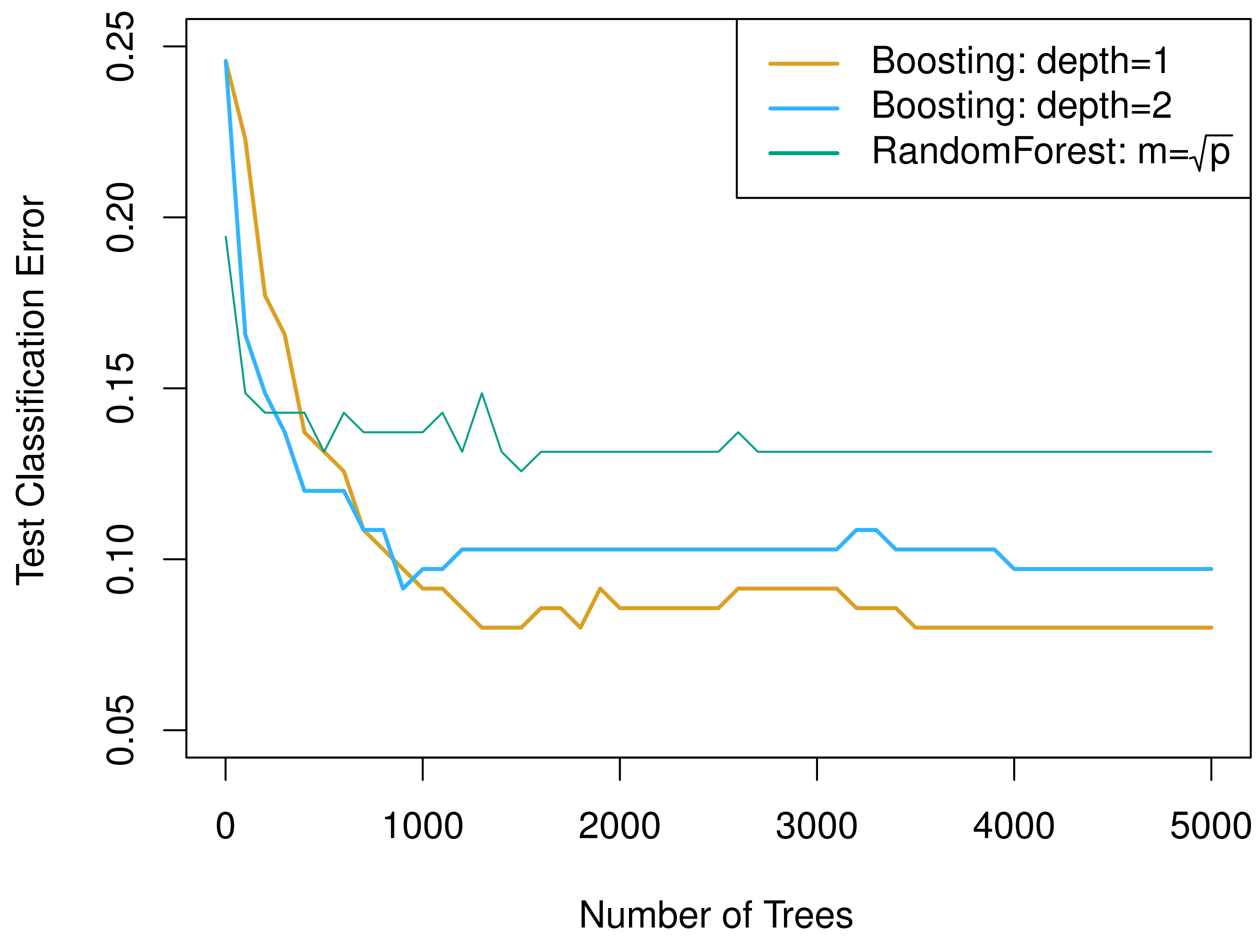
Boosting algorithm

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

JWHT Algorithm 8.2

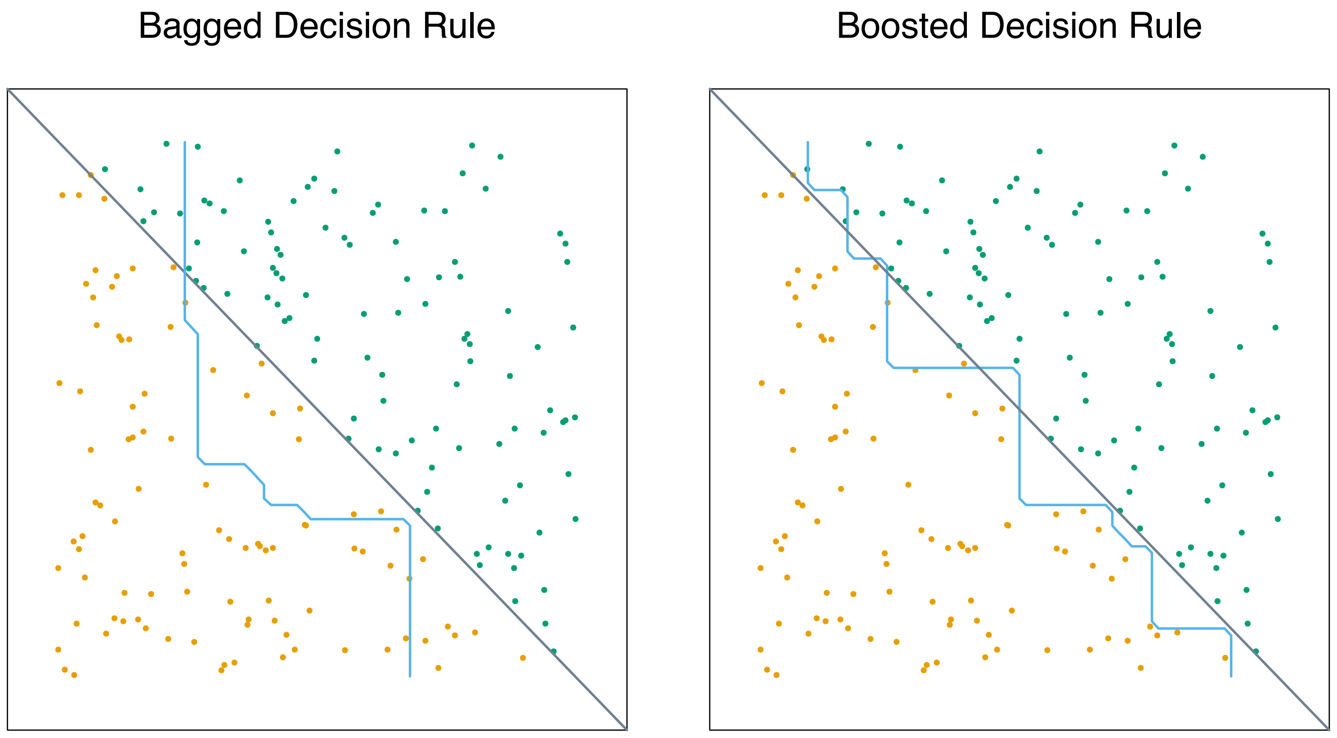
Example (experimental data)



Boosting outperformed Random Forest. However, SE (not shown) is large enough to make the methods comparable

JWHT Fig.8.11

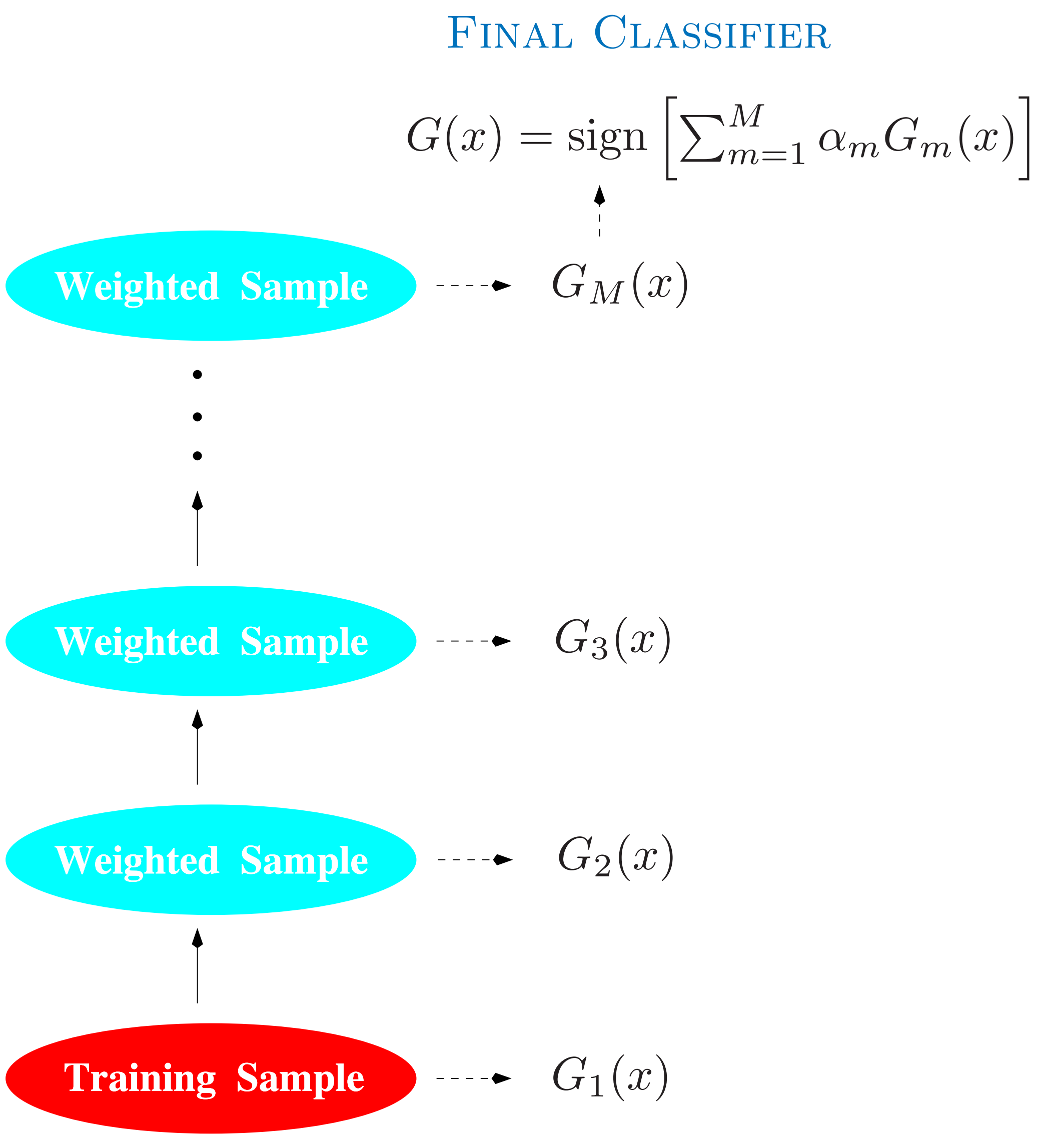
Example (simulated data)



Bagging 0-1 decision rule (i.e., misclassification rate) over $B = 50$ bootstrap samples. The tees split around the middle range of X_1 or X_2, and have little contribution away from the center.

HTF Fig.8.12

General approach: AdaBoost



$Y \in \{-1, 1\}$, classifier $G(X)$, $\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} I\{y_i \neq G(x_i)\}$

Train many weak classifiers $\rightarrow$ use a "committee" vote

HTF Chapter 10; Fig. 10.2

More generally: Forward stagewise additive modeling

- Basis function expansion

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

  - β_m: expansion coefficients
  - $b(x, \gamma_m)$ simple basis functions, e.g. classifier $G_n \in \{-1, 1\}$

- Loss function
  - Generally, optimize over all parameters (β_m, γ_m) of all basis functions

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

  - Often simplified: optimizing one basis function

$$\min_{\beta, \gamma} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta b(x_i, \gamma)\right)$$

Forward stagewise additive modeling

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

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

For squared-error loss $L(y;\ f(x)) = (y - f(x))^2$:

$$\begin{aligned} L(y_i;\ f_{m-1}(x_i) + \beta\, b(x_i, \gamma)) &= (y_i - f_{m-1}(x_i) - \beta\, b(x_i, \gamma))^2 \\ &= (r_{im} - \beta\, b(x_i, \gamma))^2 \end{aligned}$$

$r_{im} = y_i - f_{m-1}(x_i)$: residual of mth model for ith obs.

$\beta\, b(x_i, \gamma)$: fits to the current residuals

Problem: squared-loss is bad for classification

HTF Chapter 10; Algorithm 10.2

Classification: AdaBoost

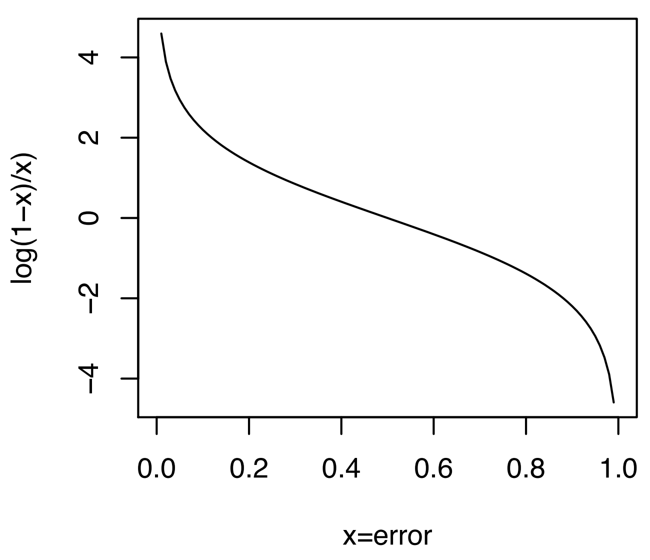
Algorithm 10.1 *AdaBoost.M1.*

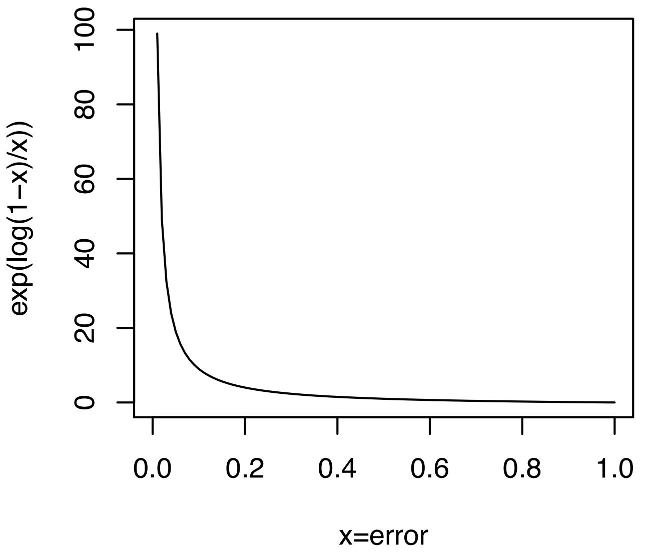
```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}$.
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (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 $G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.
```

- First, all observations have same weights
- At each step, $\uparrow$ weights of misclassified observations by factor $\exp(\alpha_m)$
- It is a basis function expansion.
  Basis functions: individual classifiers $G_m \in \{-1, 1\}$
- Minimize loss function, averaged over training data
- New aspect: w_i

HTF Chapter 10; Algorithm 10.1

AdaBoost weights





Exponential loss

- AdaBoost: equivalent to forward stagewise modeling with exponential loss $L\left(y;\ f(x)\right) = \exp\left\{-y\, f(x)\right\}$
- Optimization problem:
  $Y \in \{-1, 1\}$. For classifier $G_n \in \{-1, 1\}$, solve

$$\begin{aligned}(\beta_m, G_m) &= \min_{\beta,G} \sum_{i=1}^{N} \exp\left[-y_i \left(f_{m-1}(x_i) + \beta G(x)\right)\right] \\ &= \min_{\beta,G} \sum_{i=1}^{N} \exp\left[-y_i f_{m-1}(x_i)\right] \cdot \exp\left[-\beta y_i G(x)\right] \\ &= \min_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} \cdot \exp\left[-\beta y_i G(x)\right] \\ &\quad \text{and } w_i^{(m)} = \exp\left[-y_i f_{m-1}(x_i)\right]\end{aligned}$$

  Weights change with m, do not depend on β or G

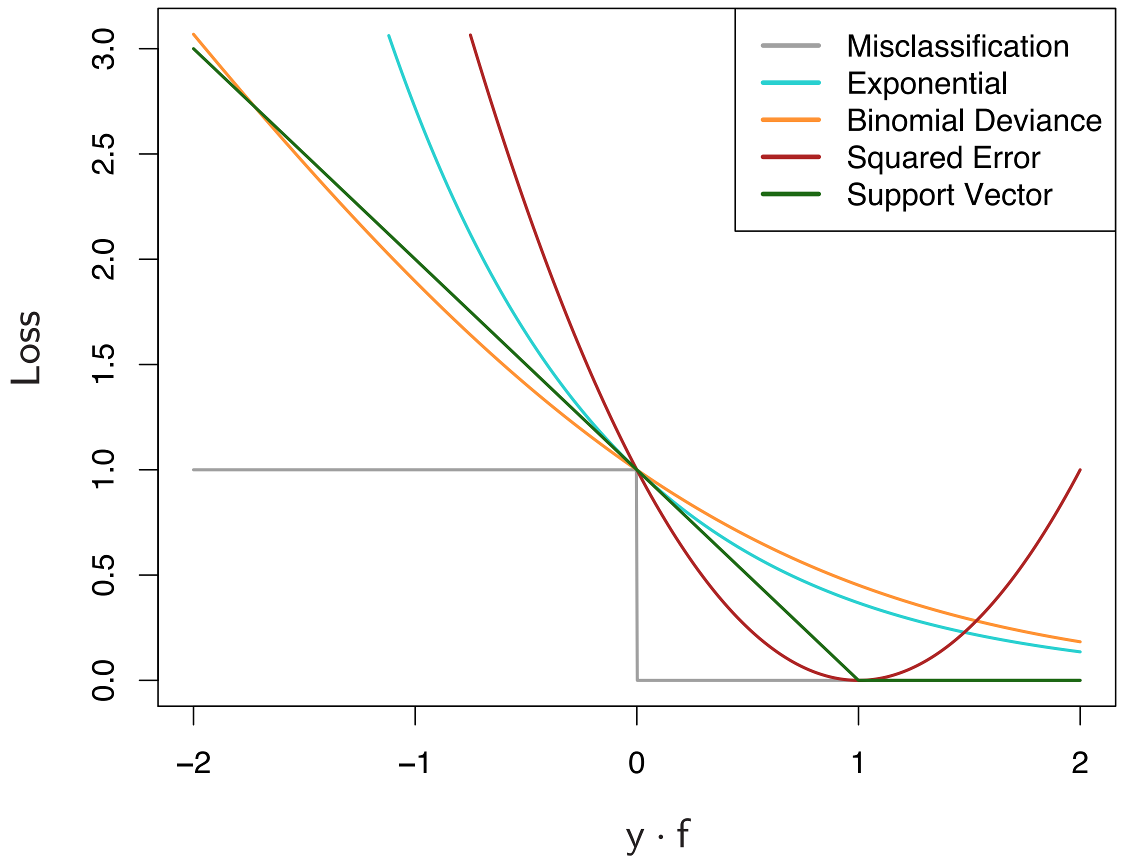
- Solution

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

  Same weights - see HTF Sec. 10.4

- Different loss funct. $\rightarrow$ different properties

Alternative loss functions



Exponential loss:
attractive computational and analytical properties

$y \cdot f(x)$ is the "margin" in SVM

HTF Fig 10.4