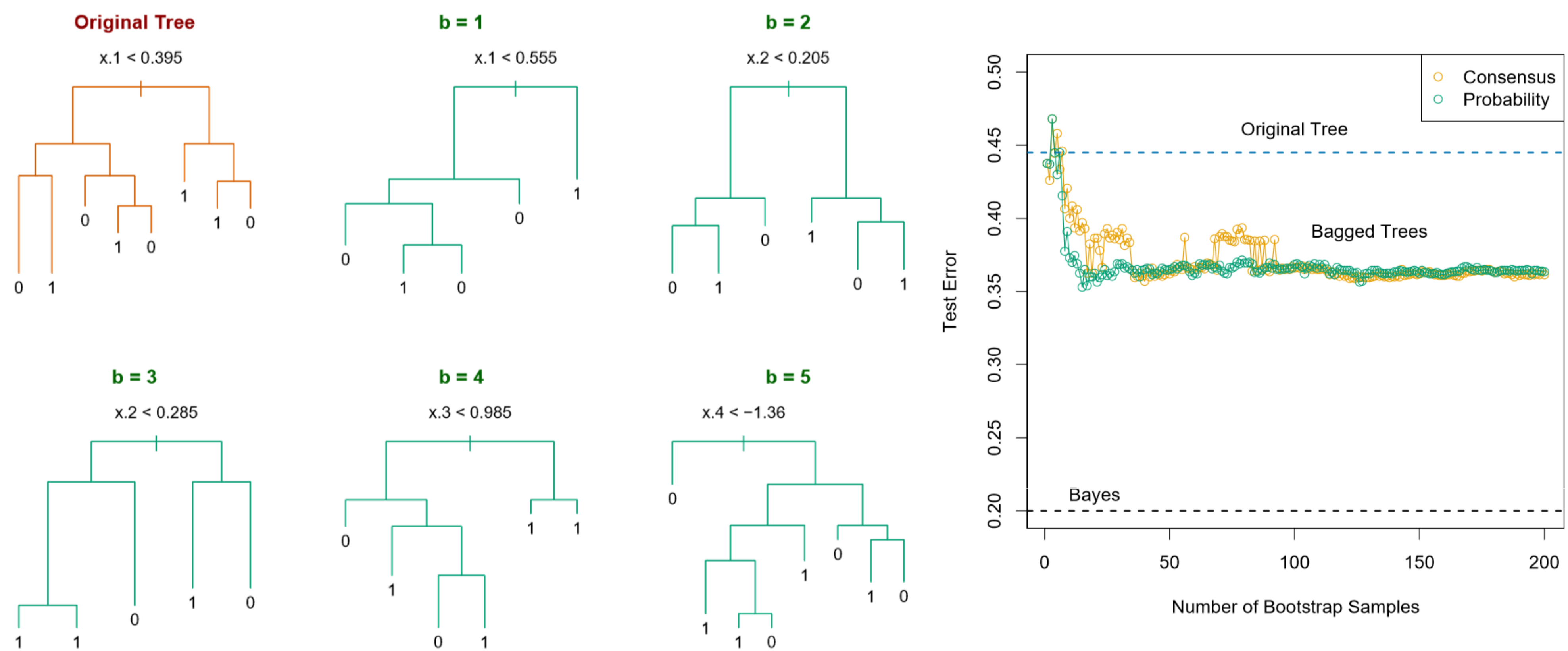
# Model Inference and Averaging

- Baging, Stacking, Random Forest, Boosting



# Bagging – Bootstrap Aggregating

- Bootstrap
  - Repeatedly select n data samples with replacement
  - Each dataset b=1:B is slightly different
    - estimation of test error (out of bag data)
    - estimation of variance of the error
    - NOW: averaging models learned on different samples
    - this reduces variance of tree model.

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

# Bagging Trees

- Decision trees have high variance.
- Bagging often improves tree prediction.

- On each sample, train decision tree

  WITHOUT PRUNNING.
- Predict:
  - Average of tree prediction in regression.
  - In classification:
    - Majority vote: most common prediction.
    - Weighted average: average $p_k$ predictions of trees.

# Out Of Bag Error Estimation

- Reasonable good and does not need crossvalidation.

- For each data sample
  - Predict on trees where the sample was not used for learning,
  - Average the predictions,
  - Calculate error (square, 0-1) for the sample.
- Average the errors.

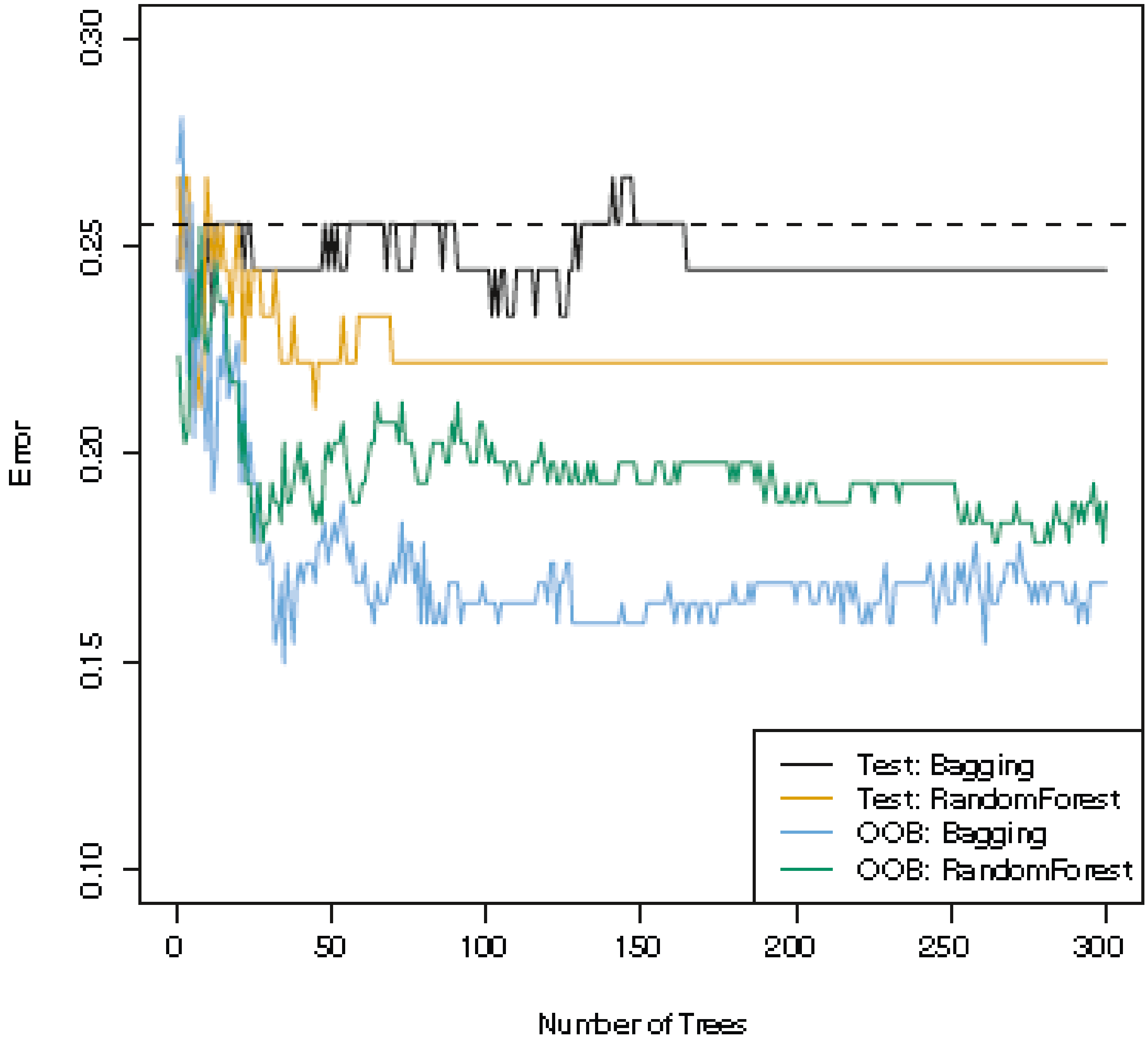
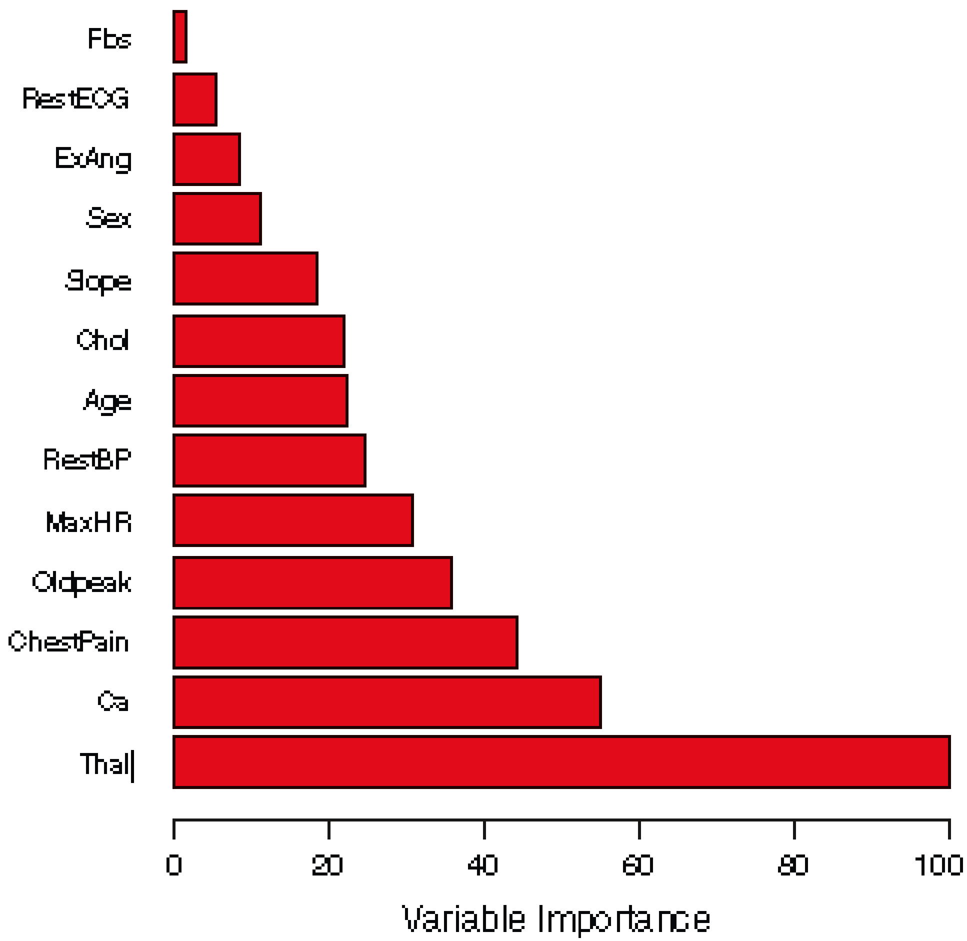



**FIGURE 8.8.** *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of $B$, the number of bootstrapped training sets used. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

# Variable Importance Measures

- Mean decrease in RSS, deviance or Gini index (relative to the maimum)



# Variable Importance Measures

- Single tree:

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

  - For each internal node *t*
    - Calculate difference RSS (deviance, 0-1) *i*
    - before and after (weighted) the split
    - For each predictor *l*
    - summ gains of internal nodes with this predictor.
- Set of trees:
  - Average previous measure across trees *M*.

$$\mathcal{I}_\ell^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_\ell^2(T_m)$$

- Report it relative to maximum *100.

# Random Forset

- Like bagging, but
- Each time a split is consider

  a random sample of m predictors is chosen

  only thise are considered for split.
- This makes the trees more different each other.
- A strong predictor cannot take all.

- Recommended *m* for classification $\lfloor \sqrt{p} \rfloor$

  regression $\lfloor p/3 \rfloor$ .

**Algorithm 15.1** *Random Forest for Regression or Classification.*

1. For $b = 1$ to $B$:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.

   (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.

      i. Select $m$ variables at random from the $p$ variables.

      ii. Pick the best variable/split-point among the $m$.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point $x$:

*Regression:* $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

*Classification:* Let $\hat{C}_b(x)$ be the class prediction of the $b$th random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

# Task For You

- Modify previous algorithm to get Bagging Trees.

# Stacking

- Aggregates models of different types:
  - Tree, Neural network, GAM, log. Reg., …
- We learn a model on top of previous ones.
- Simple – to avoid overfitting.
- Linear regression without intercept, i.e. weights.
- Model that minimizes one-leave-out error

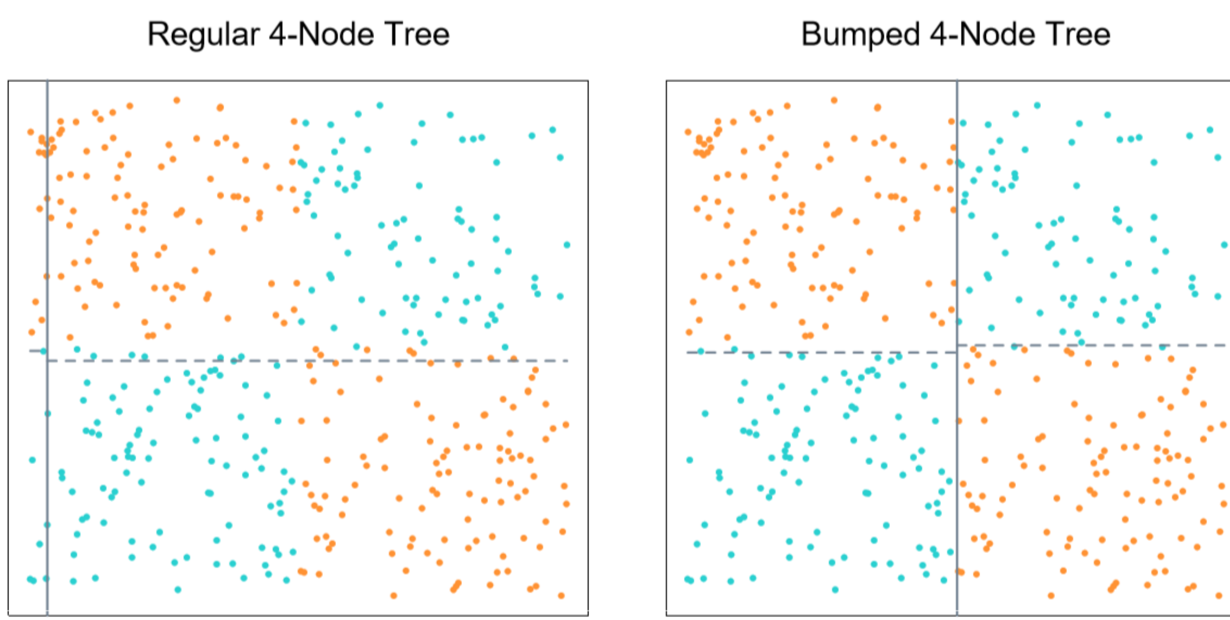
$$\hat{w}^{\mathrm{st}} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x_i) \right]^2$$

- Final prediction is: $\sum_m \hat{w}_m^{\mathrm{st}} \hat{f}_m(x)$ .

# Stochastic Search: Bumping

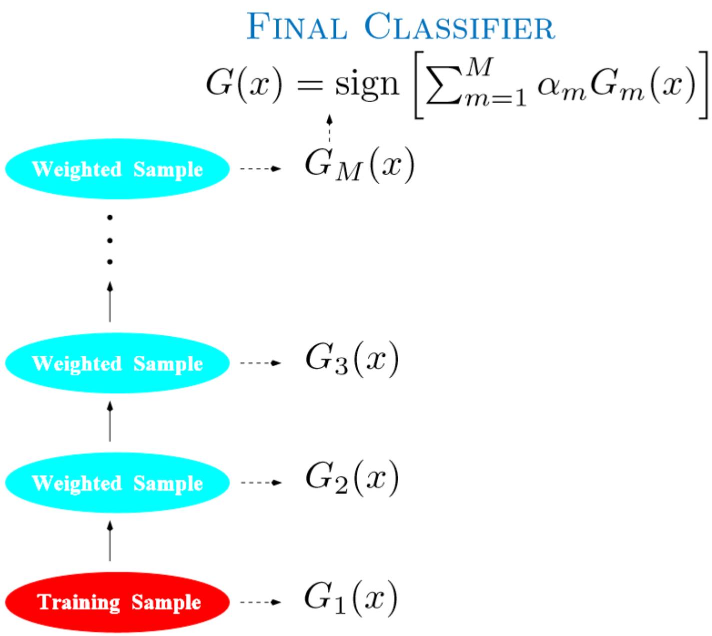
- A try to avoid local minima.
- Select M bootstap samples (and original data),
- train a model on each sample,
- Select the best model (on original training set).

- Hopefully, some sample breaks the tie of XOR.



# Boosting

- Learn ensamble of trees.
- Each time, concentrate on the records BADLY PREDICTED by previous trees,
  - Residuals in regression,
  - Weighted data in classification,
    - high weight of a record means hi
      model.



# Intraction depth, Shrinkage (for Boosting)

Two ways to avoid fast convergence:

- Learn SIMPLE trees, only 2 or a few leaves.
  - Decision stumps: trees with 2 leaves
    - No interaction between predictors is modeled.
  - Interaction of 2 variables – tree depth <=2.

- Use shrinkake parameter λ,
  - simmilarly to LASSO.

Typically 0.01 or 0.001, many trees needed,

λ=1 means no shrinkage, less trees required.

**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
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (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 $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

**Algorithm 8.2** *Boosting for Regression Trees*

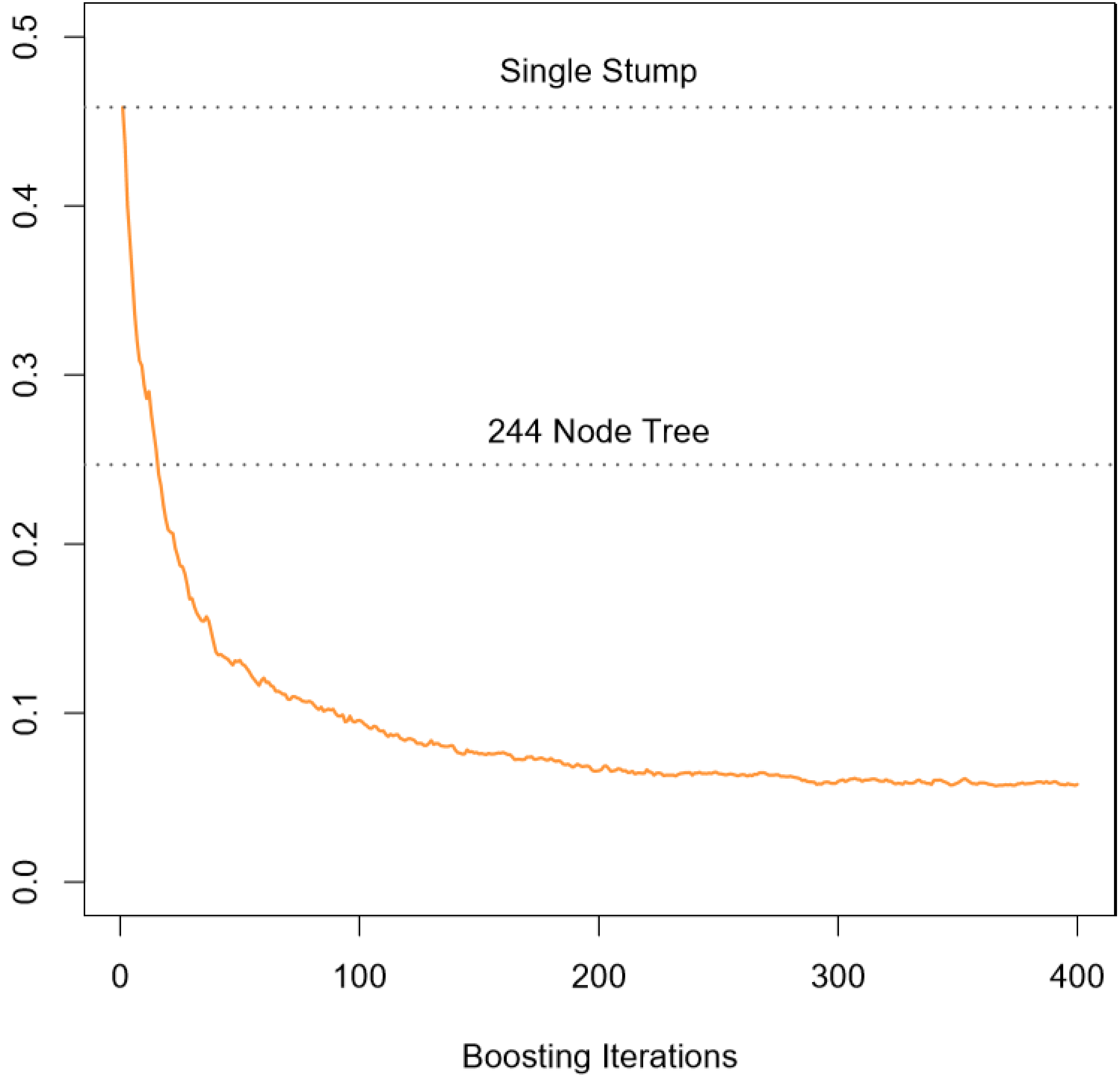
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all $i$ in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with $d$ splits ($d+1$ terminal nodes) to the training data $(X, r)$.
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow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

# 'Skin of Orange' Example (Spere)

- We have 10 predictors,
- Y is defined as:

$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5), \\ -1 & \text{otherwise.} \end{cases}$$

- 2000 samples.
- Decision stump: 45.8% error,
- Boosting: 5.8% error after 400 iterations.



Single Stump
244 Node Tree
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
400
Boosting Iterations

**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 $\hat{f}(x) = f_M(x)$.

**TABLE 10.2.** *Gradients for commonly used loss functions.*

| Setting | Loss Function | $-\partial L(y_i, f(x_i))/\partial f(x_i)$ |
|---|---|---|
| Regression | $\frac{1}{2}[y_i - f(x_i)]^2$ | $y_i - f(x_i)$ |
| Regression | $\|y_i - f(x_i)\|$ | $\text{sign}[y_i - f(x_i)]$ |
| Regression | Huber | $y_i - f(x_i)$ for $\|y_i - f(x_i)\| \leq \delta_m$<br>$\delta_m \text{sign}[y_i - f(x_i)]$ for $\|y_i - f(x_i)\| > \delta_m$<br>where $\delta_m = \alpha\text{th-quantile}\{\|y_i - f(x_i)\|\}$ |
| Classification | Deviance | $k$th component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$ |

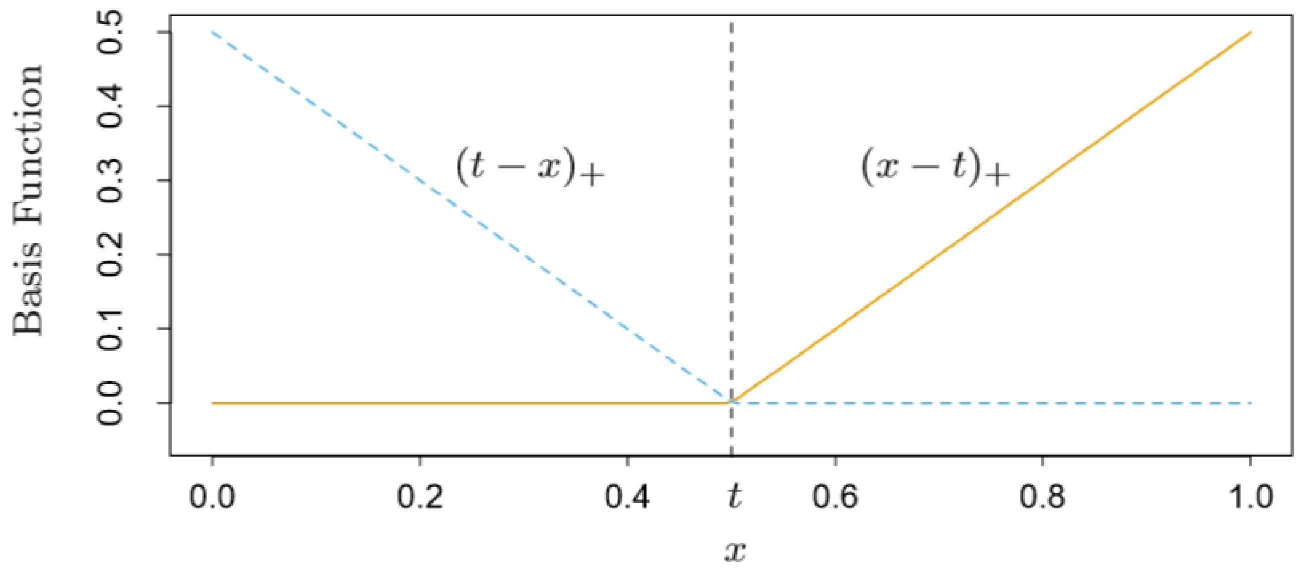
# MARS - Multivariate Additive Regression Splines

- We define set of basis functions (reflected pairs):

$$\mathcal{C} = \{(X_j - t)_+,\ (t - X_j)_+\}_{\substack{t \in \{x_{1j}, x_{2j}, \ldots, x_{Nj}\} \\ j = 1, 2, \ldots, p.}}$$

for each variable, each value in the training data:

$$(x-t)_+ = \begin{cases} x - t, & \text{if } x > t, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad (t-x)_+ = \begin{cases} t - x, & \text{if } x < t, \\ 0, & \text{otherwise.} \end{cases}$$



# MARS – cont.

- we look for a model of the form:

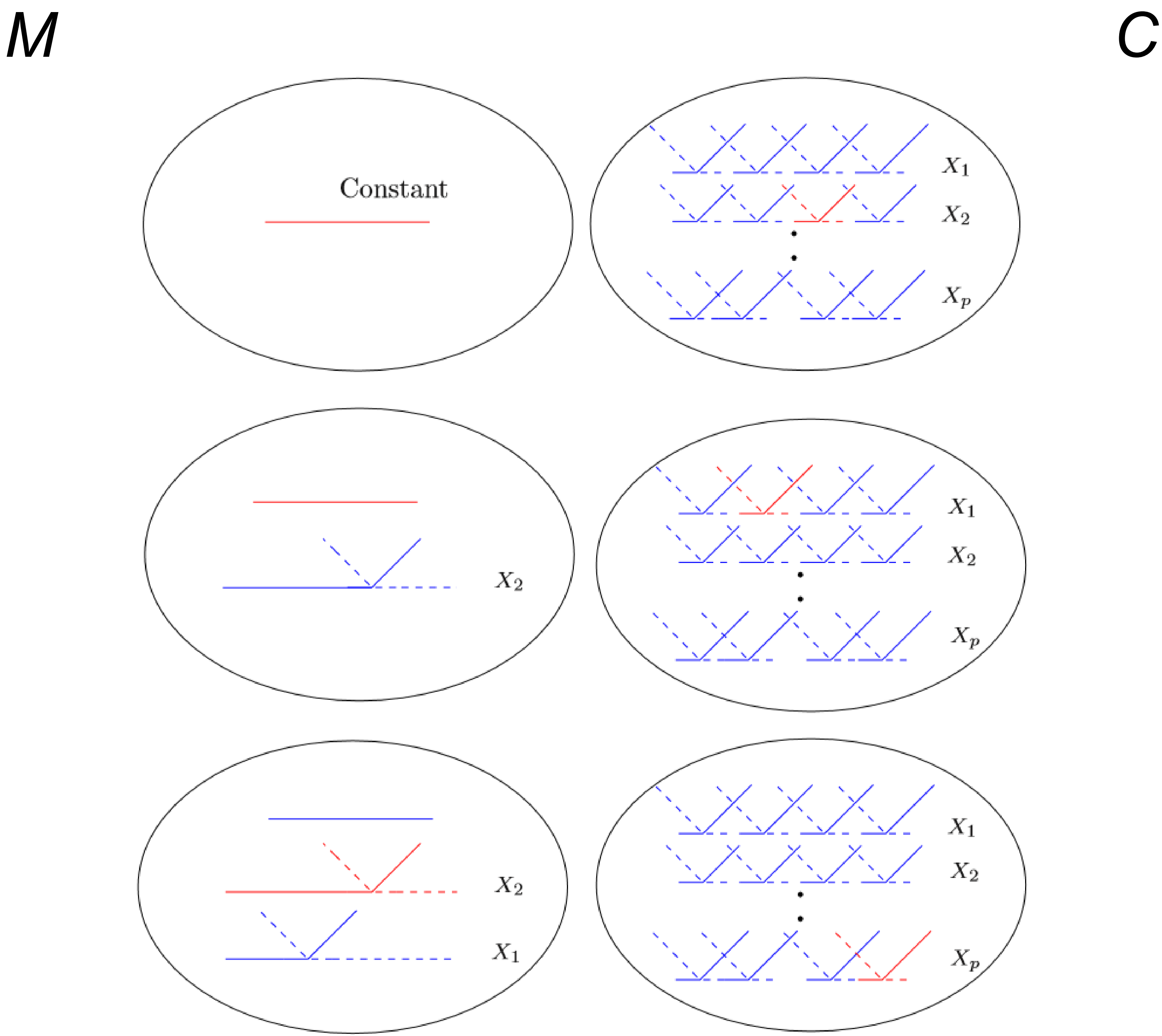
$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$$

- where $h_m(X)$ are functions in *C*

  only some! to avoid overfitting.

- We itteratively extend the model set $\mathcal{M}$ and add to the model terms of the form:

$$\hat{\beta}_{M+1} h_\ell(X) \cdot (X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X) \cdot (t - X_j)_+, \; h_\ell \in \mathcal{M}$$


M
C
Constant
$X_1$
$X_2$
$X_p$
$X_1$
$X_2$
$X_p$
$X_2$
$X_1$
$X_2$
$X_p$
$X_2$
$X_1$

# Non-linear Model Functions

- We may consider also product of function from M with a candidate from C

$$h_m(X) \cdot (X_j - t)_+ \quad \text{and} \quad h_m(X) \cdot (t - X_j)_+,\ t \in \{x_{ij}\}$$

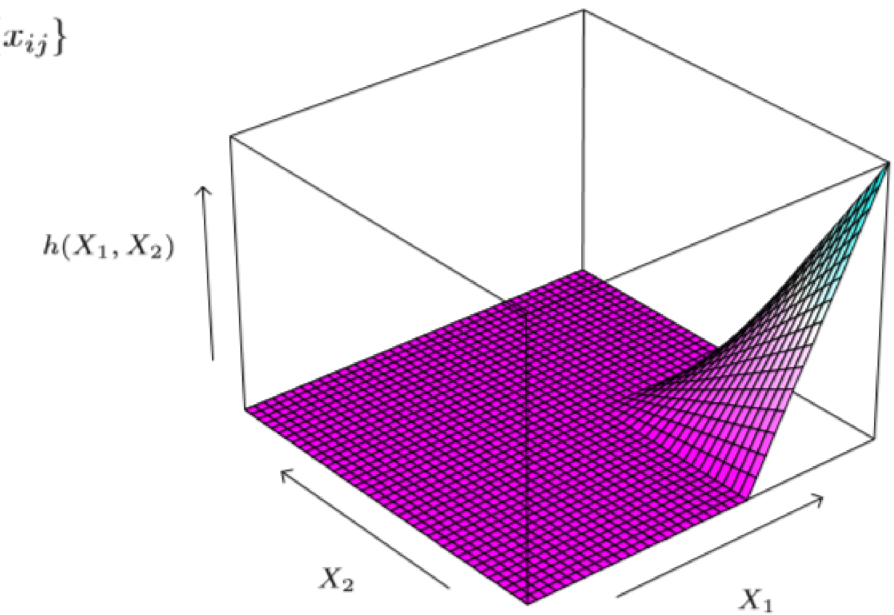
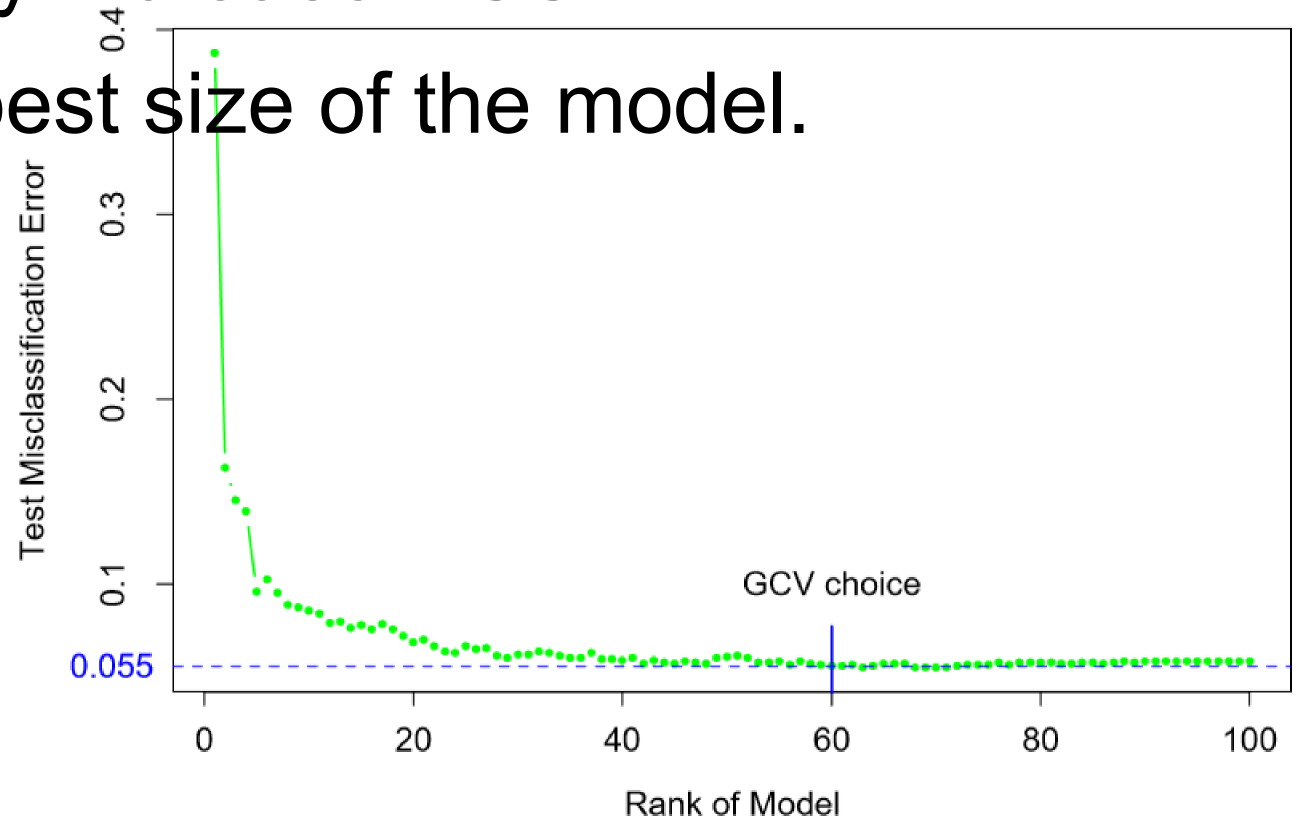



**FIGURE 9.11.** *The function $h(X_1, X_2) = (X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$, resulting from multiplication of two piecewise linear MARS basis functions.*

# Overfitted Model – What to do?

- The model is usually overfitted.
- We delete backwards functions from M

  that minimally increase RSS.
- We select the best size of the model.



# Generalized Crossvalidation

- If the crossvalidation is too time consuming, we use an estimate:

$$\mathrm{GCV}(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - M(\lambda)/N)^2}$$

- effective number of parameters: $M(\lambda)$
- r number of lin. indep. elems. of M $M(\lambda) = r + cK$
- K number of knots in M

$c = 3$

**Algorithm 10.4** *Gradient Boosting for K-class Classification.*

1. Initialize $f_{k0}(x) = 0,\ k = 1, 2, \ldots, K$.
2. For $m$=1 to $M$:
   (a) Set
   $$p_k(x) = \frac{e^{f_k(x)}}{\sum_{\ell=1}^{K} e^{f_\ell(x)}},\ k = 1, 2, \ldots, K.$$
   (b) For $k = 1$ to $K$:
      i. Compute $r_{ikm} = y_{ik} - p_k(x_i),\ i = 1, 2, \ldots, N$.
      ii. Fit a regression tree to the targets $r_{ikm},\ i = 1, 2, \ldots, N$, giving terminal regions $R_{jkm},\ j = 1, 2, \ldots, J_m$.
      iii. Compute
      $$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jkm}} r_{ikm}}{\sum_{x_i \in R_{jkm}} |r_{ikm}|(1 - |r_{ikm}|)},\ j = 1, 2, \ldots, J_m.$$
      iv. Update $f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jkm} I(x \in R_{jkm})$.
3. Output $\hat{f}_k(x) = f_{kM}(x),\ k = 1, 2, \ldots, K$.

# MART – Multivariate Additive Trees

- Gradient tree learning
- Symetric logistic transform

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}}$$

constraint $\sum_{k=1}^{K} f_k(x) = 0.$

- Multinomial deviance loss function

$$\begin{aligned} L(y, p(x)) &= -\sum_{k=1}^{K} I(y = \mathcal{G}_k) \log p_k(x) \\ &= -\sum_{k=1}^{K} I(y = \mathcal{G}_k) f_k(x) + \log \left( \sum_{\ell=1}^{K} e^{f_\ell(x)} \right). \end{aligned}$$