Chapter 9: Additive Models, Trees and Related Methods

DD3364

November 21, 2012

Overview of chapter

- Introduce some specific methods for supervised learning.
  - Generalized Additive Models
  - Trees
  - Multivariate adaptive regression splines
  - Patient Rule Induction Method
  - Hierarchical Mixture of Experts
- Each method assumes a particular structure for the regression function.
- ✓ This structure helps combat the curse of dimensionality.
- ✗ Structure imposed may not be appropriate.

Generalized Additive Models

Definition for regression

- A **generalized additive model** has the form

$$E[Y|X_1, \ldots X_p] = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p)$$

  where the f_j's are smooth, potentially non-parametric, functions.

- In this chapter each f_j is fit using a scatter plot smoother that is
  - cubic smoothing spline
  - kernel smoother
  - ....

Definition for binary classification

- A **generalized additive model** has the form

$$g\left(\frac{P(Y=1\,|\,X)}{P(Y=0\,|\,X)}\right) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p)$$

  where $g(\cdot)$ is a **link function**.

- Common link functions are
  - **Identity**: $g(z) = z$.

    Used for linear and additive models for Gaussian response data.
  - **Logit**: $g(z) = \log(z/(1-z))$.

    Used for modeling of binomial probabilities.
  - **Log**: $g(z) = \log(z)$.

    Used for log-linear or log-additive models for Poisson count data.

Advantages of these generalized additive model

- If f_j's are estimated in a flexible way $\implies$ can reveal non-linear relationship between input X_j and Y.
- Efficient algorithms to fit them if p is not too large.

Fitting Additive Models

Additive Model

$$Y = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + \epsilon$$

where $E[\epsilon] = 0$.

How to find the parameters of the model?

- Have observations $\{(x_i, y_i)\}_{i=1}^n$ then
- **minimize** a **penalized sum-of-squares**:

$$\text{PRSS}(\alpha, f_1, f_2, \ldots, f_n) = \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int_t f_j''(t)^2 dt$$

where each $\lambda_j \geq 0$.

Additive Model

$$Y = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + \epsilon$$

where $E[\epsilon] = 0$.

How to find the parameters of the model?

- Have observations $\{(x_i, y_i)\}_{i=1}^n$ then
- **minimize** a **penalized sum-of-squares**:

$$\text{PRSS}(\alpha, f_1, f_2, \ldots, f_n) = \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int_t f_j''(t)^2 dt$$

where each $\lambda_j \geq 0$.

Fitting the model: One option

How to find the parameters of the model?

Minimize a *penalized sum-of-squares*:

$$\text{PRSS}(\alpha, f_1, f_2, \ldots, f_n) = \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{i=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int_t f_j''(t)^2 dt$$

where each $\lambda_j \geq 0$.

One solution

- Let each $f_j(X_j)$ a cubic smoothing spline with knots at x_{ij} and response y_i for $i = 1, \ldots, n$.
- This solution minimizes $\text{PRSS}(\alpha, f_1, f_2, \ldots, f_n)$.
- However, it is not the only minimizer. α is not identifiable.

How to find the parameters of the model?

Minimize a *penalized sum-of-squares*:

$$\text{PRSS}(\alpha, f_1, f_2, \ldots, f_n) = \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{i=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int_t f_j''(t)^2 dt$$

where each $\lambda_j \geq 0$.

One solution

- Let each $f_j(X_j)$ a cubic smoothing spline with knots at x_{ij} and response y_i for $i = 1, \ldots, n$.
- This solution minimizes $\text{PRSS}(\alpha, f_1, f_2, \ldots, f_n)$.
- However, it is not the only minimizer. α is not identifiable.

Fitting the model: Option 1

- To combat this assume

$$\sum_{i=1}^{n} f_j(x_{ij}) = 0 \quad \text{for } j = 1, \ldots, p$$

- Assumption $\implies \hat{\alpha} = \text{ave}(y_i)$.

- If the data matrix

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{pmatrix}$$

  has **full column rank** then PRSS$(\alpha, f_1, f_2, \ldots, f_n)$ is **convex** and the minimizer is unique. **Hurrah !**

- $\exists$ a simple iterative procedure for finding this solution.

Fitting the model: Option 1

- To combat this assume

$$\sum_{i=1}^{n} f_j(x_{ij}) = 0 \quad \text{for } j = 1, \ldots, p$$

- Assumption $\implies \hat{\alpha} = \text{ave}(y_i)$.

- If the data matrix

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{pmatrix}$$

  has **full column rank** then PRSS$(\alpha, f_1, f_2, \ldots, f_n)$ is **convex** and the minimizer is unique. **Hurrah !**

- $\exists$ a simple iterative procedure for finding this solution.

Backfitting Algorithm for Additive Models

1. **Initialize**:

$$\hat{\alpha} = \frac{1}{n}\sum_{i} y_i, \quad f_j \equiv 0 \ \ \forall j$$

2. **Cycle until convergence**: $j = 1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$

$$\hat{f}_j \longleftarrow \mathcal{S}_j \left[\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \}_{i=1}^{n} \right]$$

$$\hat{f}_j \longleftarrow \hat{f}_j - \frac{1}{n}\sum_{i=1}^{n} \hat{f}_j(x_{ij})$$

where $\mathcal{S}_j \left[\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \}_{i=1}^{n} \right]$ denotes the cubic smoothing spline with knots at x_{ij} and responses $y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})$ for $i = 1, \ldots, n$. Could use other smoothing operators $\mathcal{S}_j$.

Example: Additive Logistic Regression

Generalized Additive Logistic Model:

$$\log\left(\frac{P(Y=1|X)}{P(Y=0|X)}\right) = \alpha + f_1(X_1) + \cdots + f_p(X_p)$$

- Functions $f_1, \ldots, f_p$ estimated by a backfitting algorithm within a Newton-Raphson procedure.
- What does this mean....

Example: Additive Logistic Regression

Generalized Additive Logistic Model:

$$\log\left(\frac{P(Y=1|X)}{P(Y=0|X)}\right) = \alpha + f_1(X_1) + \cdots + f_p(X_p)$$

- Functions $f_1, \ldots, f_p$ estimated by a backfitting algorithm within a Newton-Raphson procedure.
- What does this mean....

Additive Logistic Regression: Estimating its parameters

- **Goal**: maximize the log-likelihood

$$\mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_i = \sum_i [\, y_i \log P(Y=1|x_i) + (1-y_i) \log P(Y=0|x_i) \,]$$

of the training data where $P(Y=1|x_i) = e^{\eta_i}/(1+e^{\eta_i})$ and $\eta_i = \alpha + f_1(x_{i1}) + \cdots + f_p(x_{ip})$

- **How**: Iteratively perform until convergence
  - Let each $\hat{\eta}_i = \hat{\alpha} + \sum \hat{f}_j(x_{ij})$ be the estimate of η_i given the current estimates of the parameters $\alpha, f_1, \ldots, f_p$.
  - Use a Newton-Raphson update step to produce a new estimate, $\hat{\eta}_i^{\text{new}}$, of η_i s.t. $\mathcal{L}_i(\hat{\eta}_i^{\text{new}}) \geq \mathcal{L}_i(\hat{\eta}_i)$ for each i.
  - Fit an additive model to the targets $\hat{\eta}_i^{\text{new}} \;\; \forall i$. Use back-fitting.
  - This produces new estimates of $\hat{\alpha}, \hat{f}_j \; \forall j$

Additive Models: Summary

Pros

- Extension of linear models - more flexible but still interpretable.
- Parameter estimate via Backfitting method is simple.
- Backfitting allows the appropriate fitting method for each input variable.

Cons

- No feature selection is performed.
- Backfitting is not feasible for large p.

For large p forward stagewise fitting (such as boosting) can be a solution...

Additive Models: Summary

Pros

- Extension of linear models - more flexible but still interpretable.
- Parameter estimate via Backfitting method is simple.
- Backfitting allows the appropriate fitting method for each input variable.

Cons

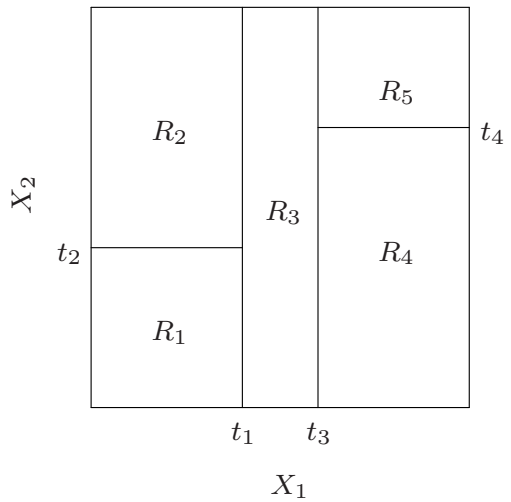
- No feature selection is performed.
- Backfitting is not feasible for large p.

For large p forward stagewise fitting (such as boosting) can be a solution...

Tree Based Methods

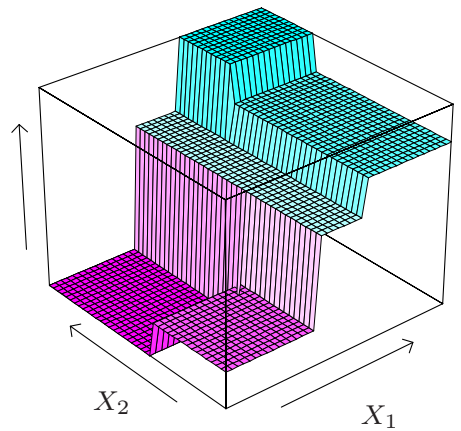
Background: Tree based methods

1. Partition feature space into a set of hyper-rectangles.

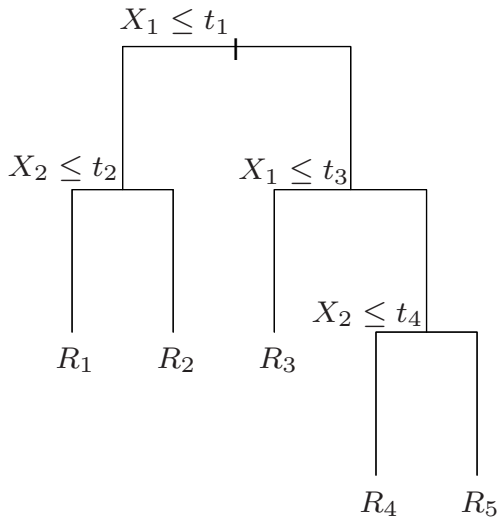


A partition of 2D space with recursive binary splits

2. Fit a simple model in each region of the partition.



- For simplification only consider recursive binary partitions



- The leaves of the tree correspond to the regions $R_1, R_2, \ldots$ in the partition.
- **Regression**: can use a constant model in each region R_m:

$$\hat{f}(X) = \sum_m c_m \, \mathsf{Ind}(\mathbf{x} \in R_m)$$

Regression Trees

- **Aim**: Approximate a regression function $f : \mathbb{R}^p \to \mathbb{R}$ with

$$\hat{f}(x) = \sum_{i=1}^{M} f_m(x)\, \mathsf{Ind}(x \in R_m)$$

where regions $R_1, \ldots, R_M$ partition $\mathbb{R}^p$ and $f_m : \mathbb{R}^p \to \mathbb{R}$.

- **Challenge**: (assuming a specific form for f_m's)

Find M and the regions $R_1, \ldots, R_M$ s.t. $\hat{f} \approx f$

from training data $(x_1, y_1), \ldots, (x_n, y_n)$ with each $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$.

Regression Trees: Piecewise constant regression fns

Let $f_m(x) = c_m$ such that the regression function becomes

$$f(x) = \sum_{i=1}^{M} c_m \, \mathsf{Ind}(x \in R_m)$$

If know the regions $R_1, \ldots, R_M$ then to minimize

$$\underset{c_1,\ldots,c_m}{\arg\min} \sum_{i=1}^{n} \left(y_i - \sum_{m=1}^{M} c_m \, \mathsf{Ind}(x_i \in \mathbb{R}_m) \right)^2$$

one would set

$$\hat{c}_m = \frac{\sum_{i=1}^{n} \sum_{m=1}^{M} y_i \, \mathsf{Ind}(x_i \in R_m)}{\sum_{i=1}^{n} \sum_{m=1}^{M} \mathsf{Ind}(x_i \in R_m)}$$

Regression Trees: **Finding the optimal partition**

The partition that minimizes the sum-of-square training error

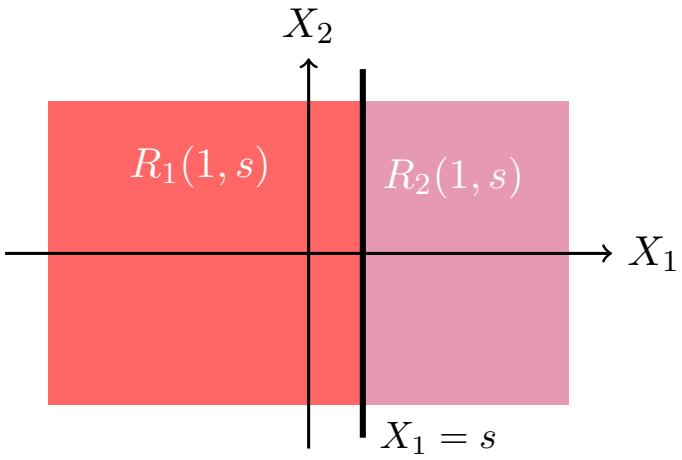
- globally is not feasible to find, ✗
- locally can be found in a greedy fashion. ✓

Regression Trees: Finding the optimal partition

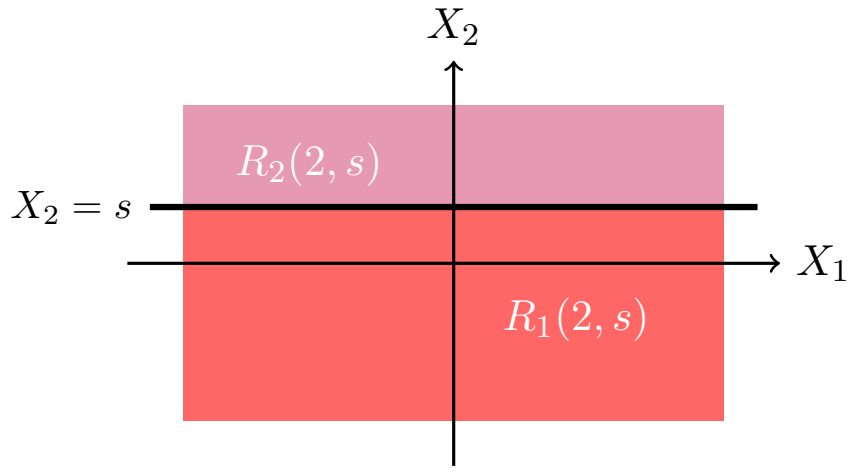
First step of the greedy approach

- Define R_1 and R_2 with a half-plane parallel to an axis of $\mathbb{R}^p$:

$$R_1(j, s) = \{X \mid X_j \leq s\} \quad \textbf{and} \quad R_2(j, s) = \{X \mid X_j > s\}$$



1st coordinate



2nd coordinate

Example Binary Splits

Regression Trees: Finding the optimal partition

First step of the greedy approach

- Let: $R_1(j,s) = \{X \mid X_j \leq s\}$ **and** $R_2(j,s) = \{X \mid X_j > s\}$
- Choose (j,s) to **minimize**: (for observations $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n$)

$$\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2$$

- For a fixed (j,s) the minimum occurs when

$$\hat{c}_k = \textbf{Average}(\, y_i \mid (x_i, y_i) \in \mathcal{X} \text{ and } x_i \in R_k(j,s)\,)$$

  for $k = 1, 2$

- Determination of best pair (j,s) feasible as for each j only have to check $\leq n+1$ values of s.

Regression Trees: Finding the optimal partition

Full Greedy Recursion

- Once the best split (j, s) is found:
  1. Partition the data $\mathcal{X}$

$$\mathcal{X}_1 = \{ (x_i, y_i) \,|\, (x_i, y_i) \in \mathcal{X} \text{ and } x_i \in R_1(j, s) \}$$
$$\mathcal{X}_2 = \{ (x_i, y_i) \,|\, (x_i, y_i) \in \mathcal{X} \text{ and } x_i \in R_2(j, s) \}$$

     based on the two resulting regions.

  2. Repeat the splitting process on both $\mathcal{X}_1$ and $\mathcal{X}_2$.

- The process above is recursively repeated on all the resulting subset of datapoints $\mathcal{X}_i$ until $|\mathcal{X}_i|$ is too small.

- The best splits found in this recursive are recorded in a binary tree.

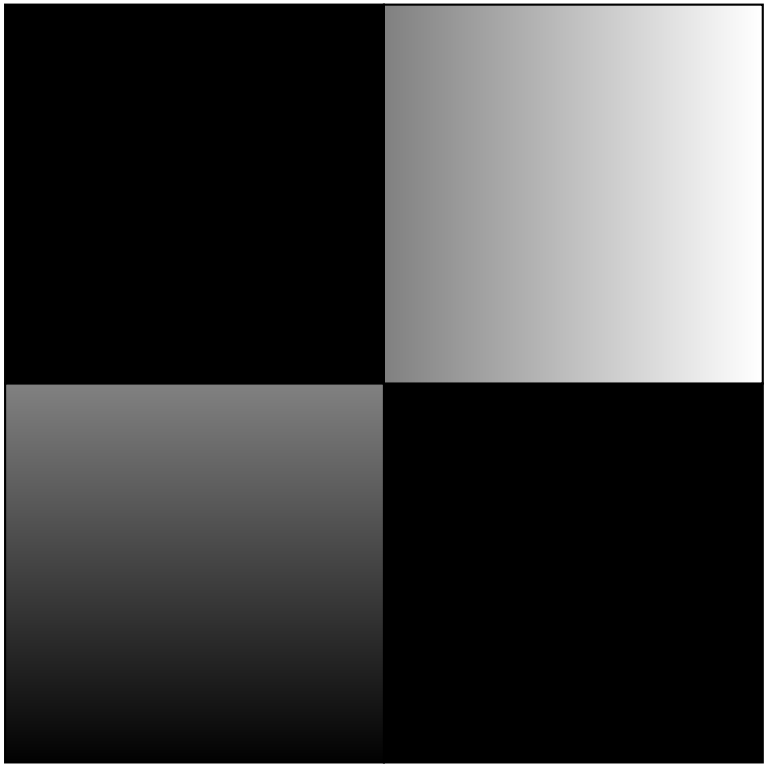
Regression Trees: Finding the optimal partition

Full Greedy Recursion in pseudo-code

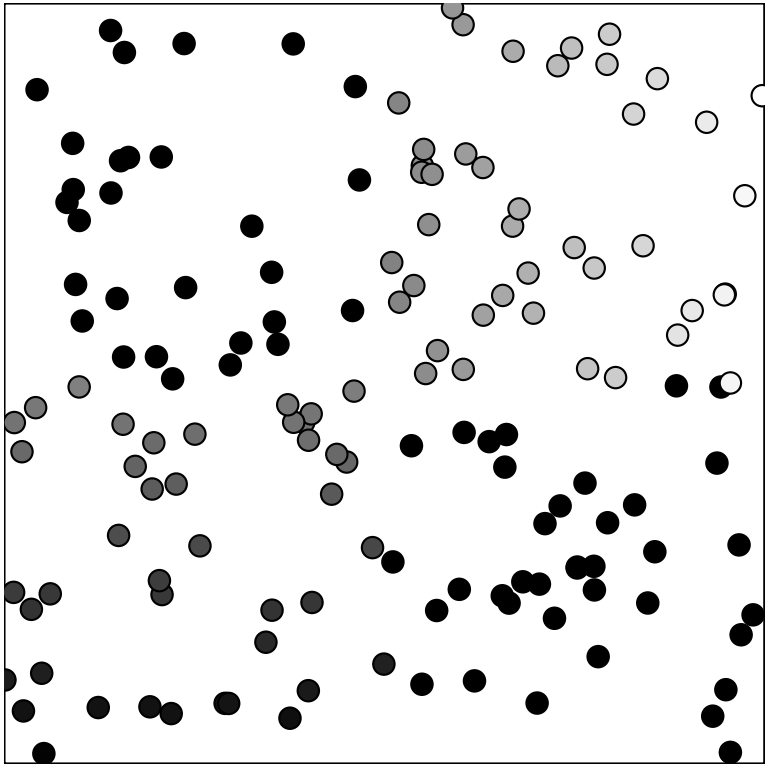
procedure ConstructTree($\mathcal{X}$)
    Initialize an empty binary tree T
    $k \leftarrow -1$
    SplitData($\mathcal{X}$, T, k)
end procedure

procedure SplitData($\mathcal{X}$, T, k)
    $(j, s) =$ FindBestBinarySplit($\mathcal{X}$)
    Add (j, s) as node of T with kth node as its parent
    $\mathcal{X}_1 = \{(x_i, y_i) \mid (x_i, y_i) \in \mathcal{X}$ and $x_i \in R_1(j, s)\}$
    $\mathcal{X}_2 = \{(x_i, y_i) \mid (x_i, y_i) \in \mathcal{X}$ and $x_i \in R_2(j, s)\}$
    $l \leftarrow$ number of node corresponding to (j, s) in T
    if $|\mathcal{X}_1| > n_s$ **then**
        SplitData($\mathcal{X}_1$, T, l)
    end if
    if $|\mathcal{X}_2| > n_s$ **then**
        SplitData($\mathcal{X}_2$, T, l)
    end if
end procedure

Regression Trees: Growing a tree recursively

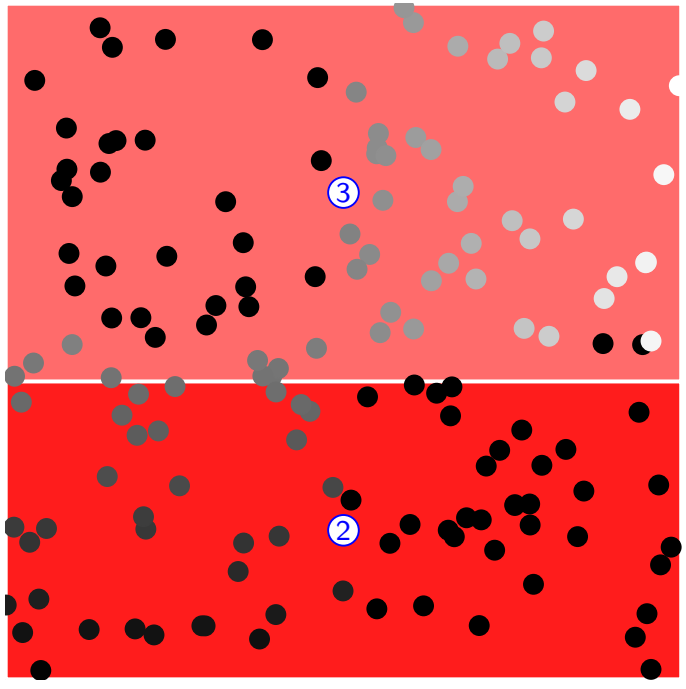


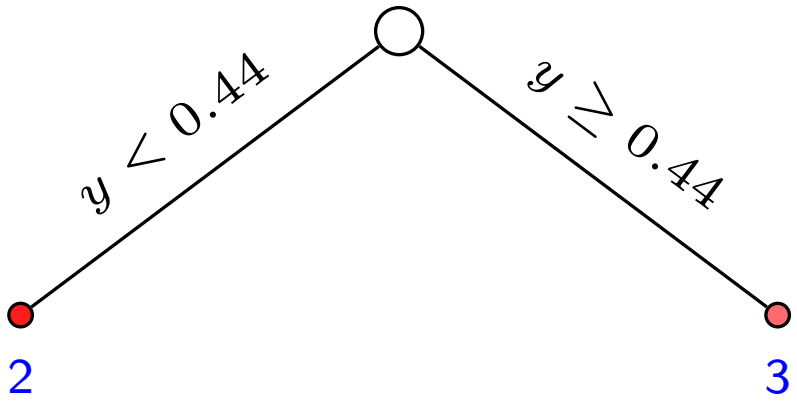
Function to be approximated



Training data (no noise)

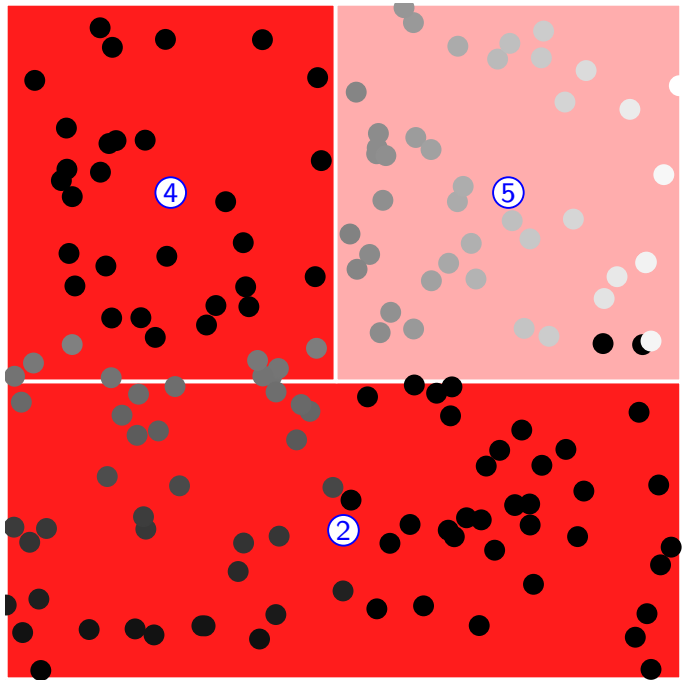
Regression Trees: Growing a tree recursively

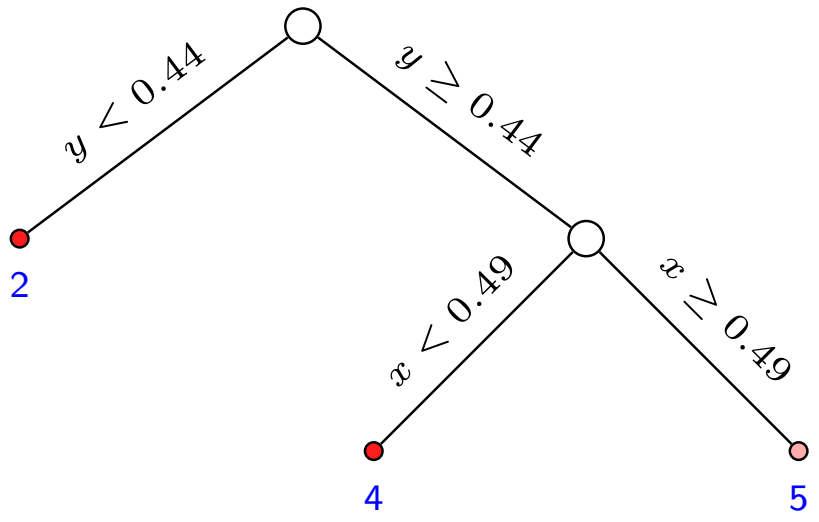




Split 1

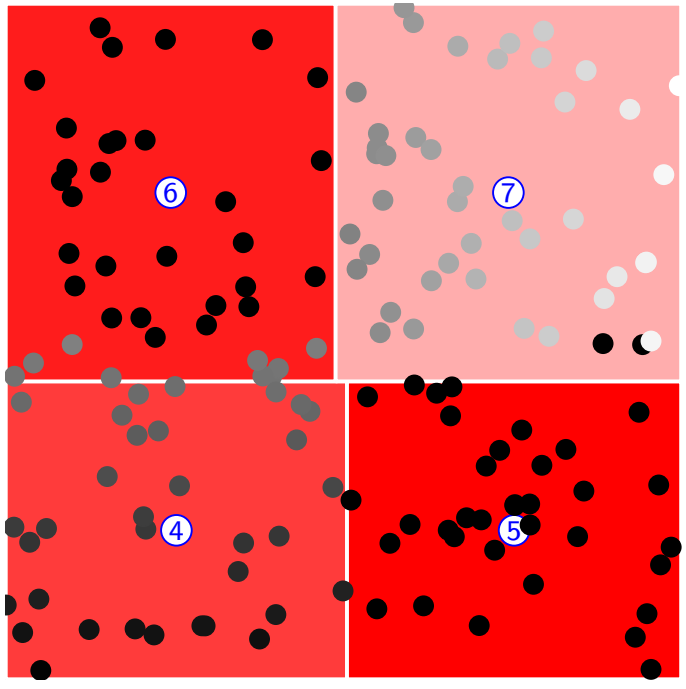
Regression Trees: Growing a tree recursively

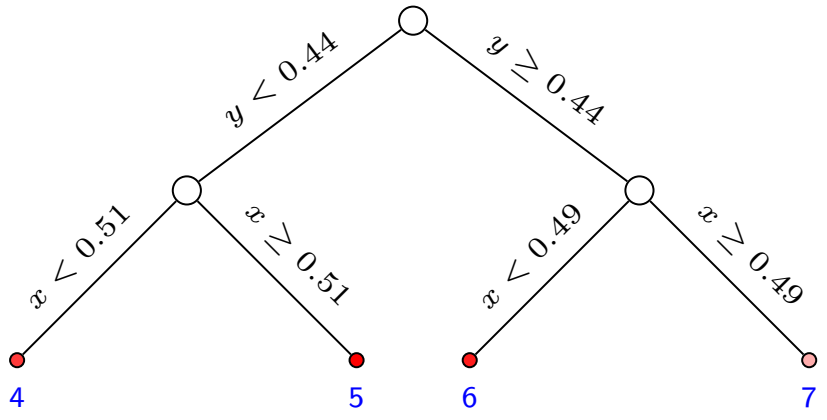




Split 2

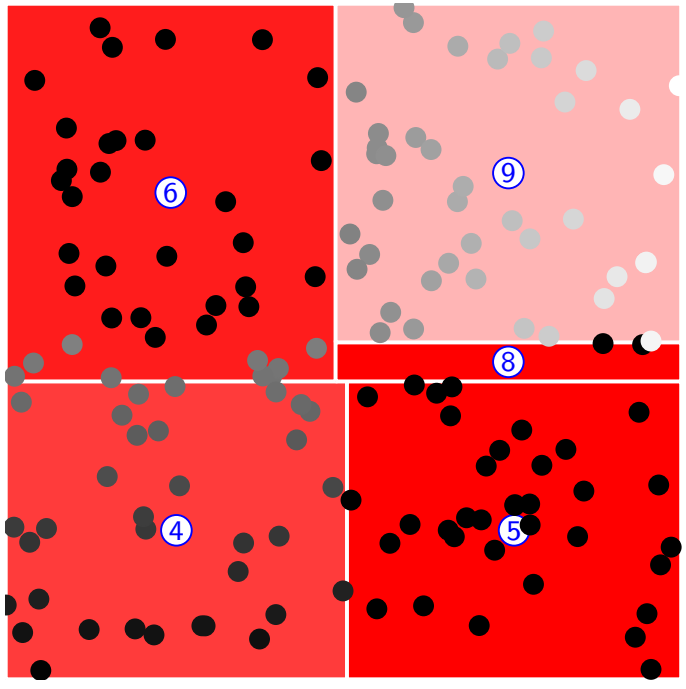
Regression Trees: Growing a tree recursively

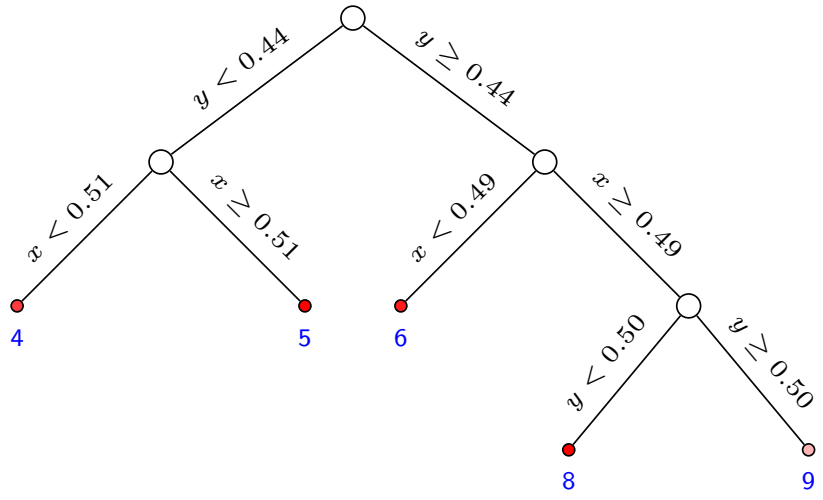




Split 3

Regression Trees: Growing a tree recursively

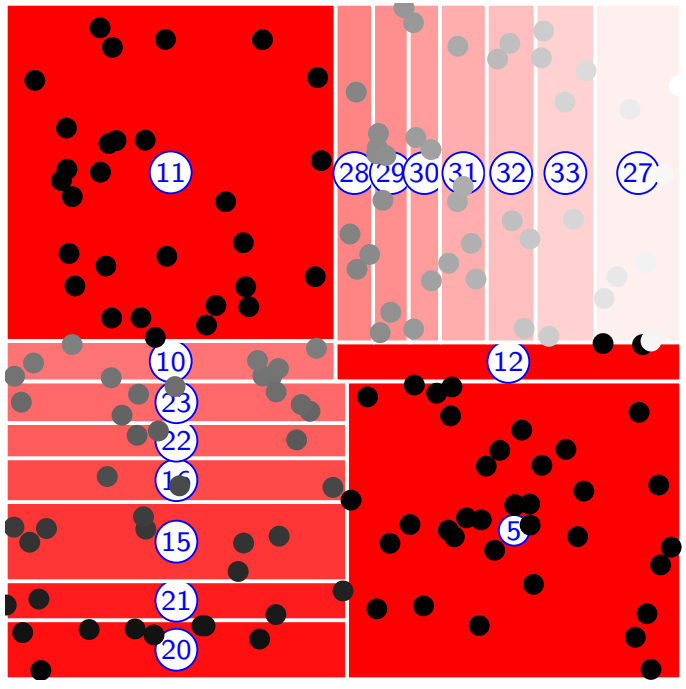


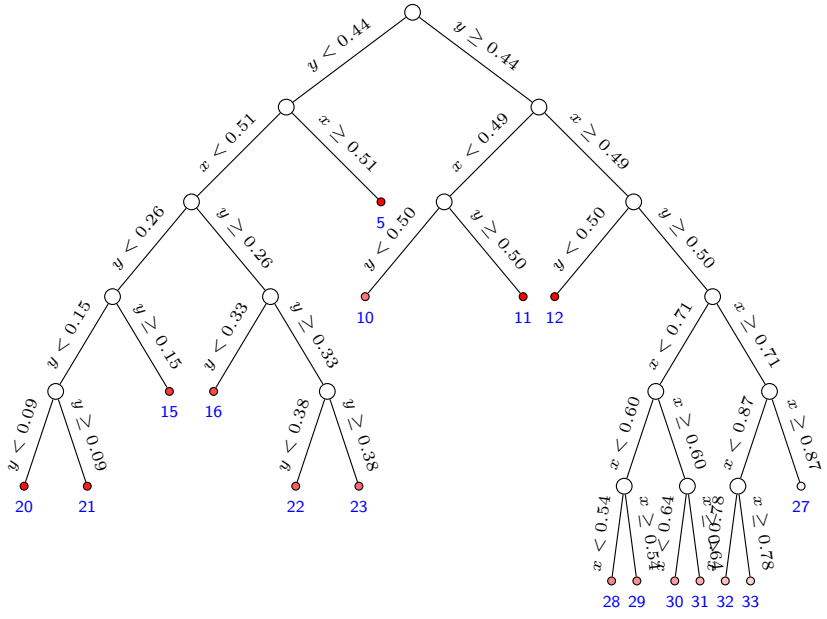


Split 4

AND SO ON UNTIL

Regression Trees: Growing a tree recursively





Split 16

Regression Trees: **Finding the optimal partition**

How large should the tree be ?

- Very large trees may over fit to the data
- Small tree may not capture the structure in the data

Common Solution

- Grow a large tree T_0
- Prune T_0 using **cost-complexity pruning**.

Regression Trees: Finding the optimal partition

Cost-complexity pruning

- Pruning T_0 corresponds to collapsing any number of its internal nodes.
- Let $\mathcal{T}_T$ contain the indices of the terminal nodes in tree T.
- Define

$$C_\alpha(T) = \sum_{m \in \mathcal{T}_T} n_m Q_m(T) + \alpha |T|$$

where

$$n_m = \#\{x_i \in R_m\},$$

$$Q_m(T) = \frac{1}{n_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2 \quad \text{with} \quad \hat{c}_m = \frac{1}{n_m} \sum_{x_i \in R_m} y_i$$

Regression Trees: Finding the optimal partition

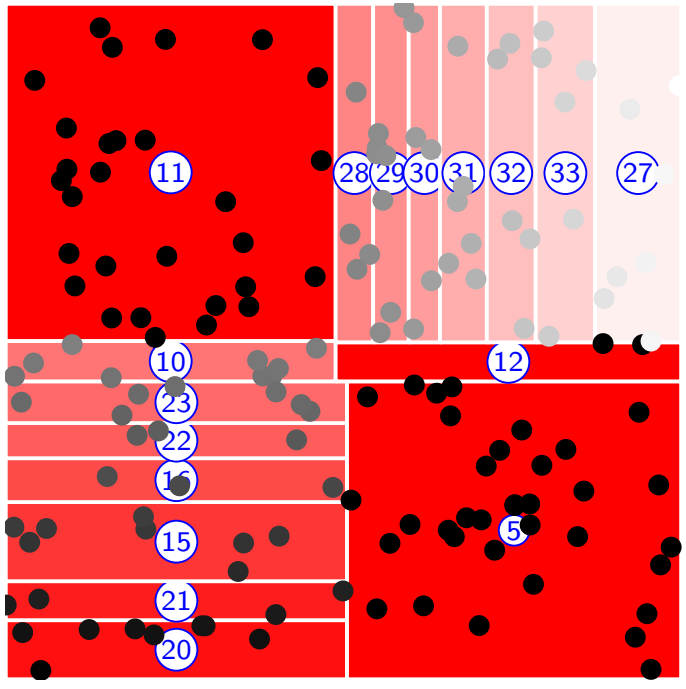
Cost-complexity pruning ctd

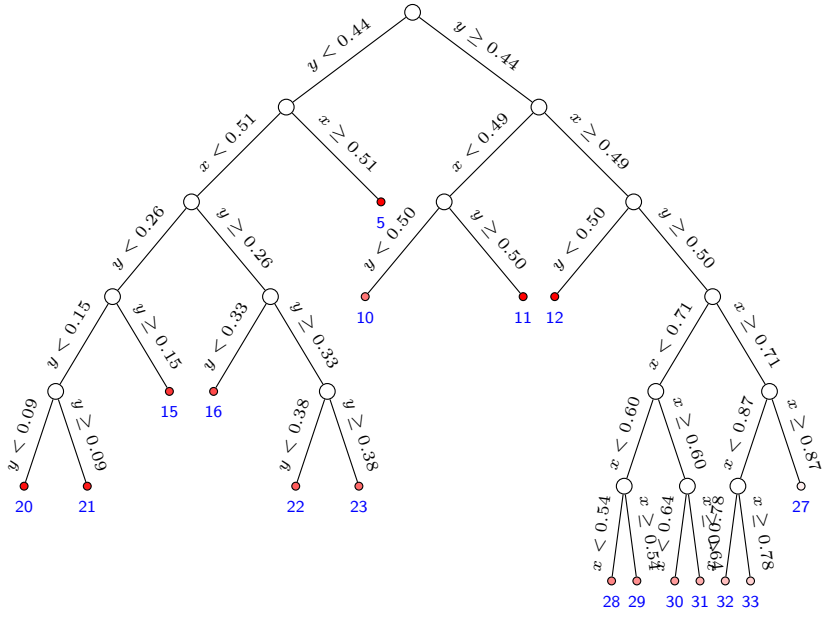
- For a given α find the subtree $T_\alpha \subseteq T$ that minimizes $C_\alpha(T)$.
- **How?** *Weakest Link Pruning*
  - Successively collapse the internal node that produces the smallest per-node increase in

$$\sum_m n_m Q_m(T)$$

    until left with a one node tree.
  - This sequence of collapsed trees contains T_α.

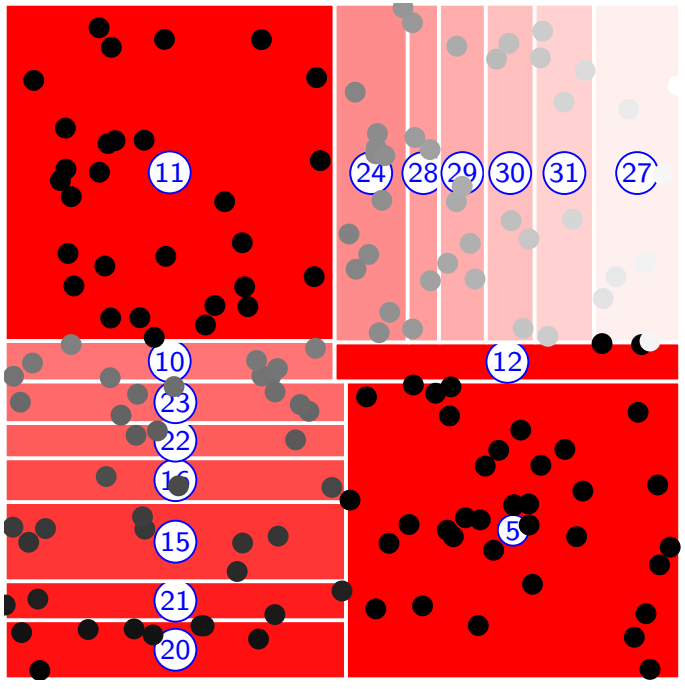
Example: Weakest Link Pruning

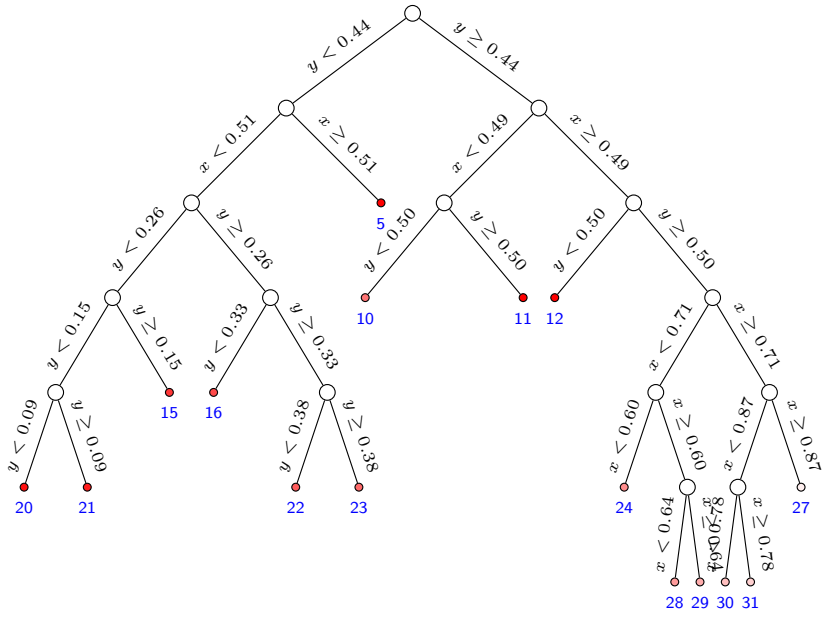




Merge 1

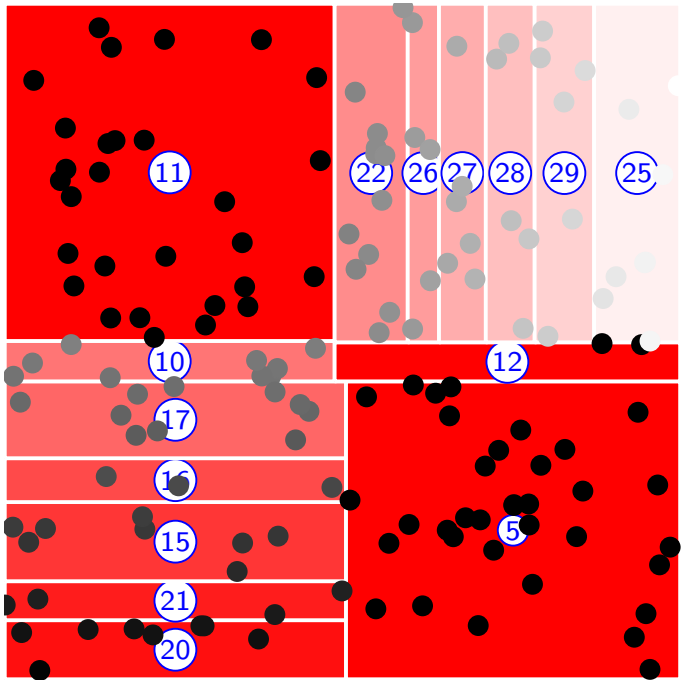
Example: Weakest Link Pruning

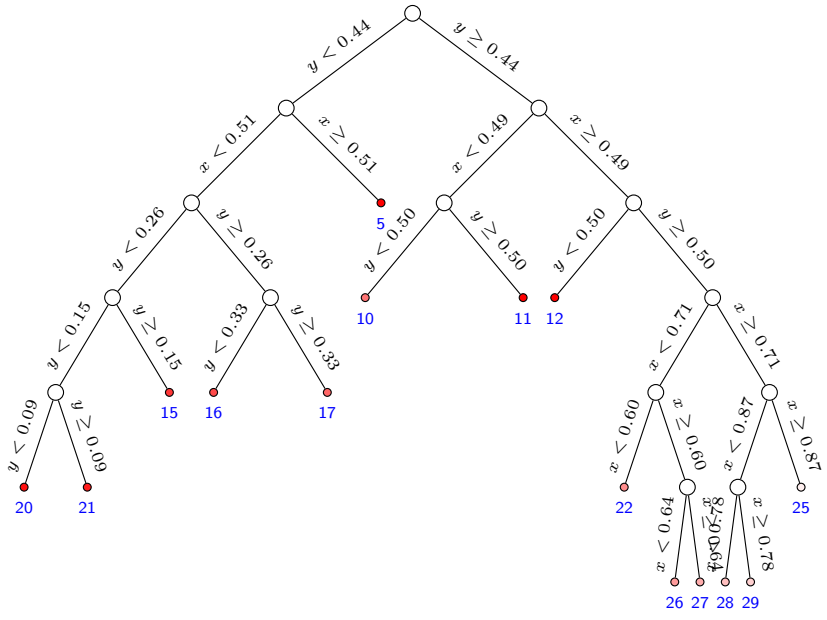




Merge 2

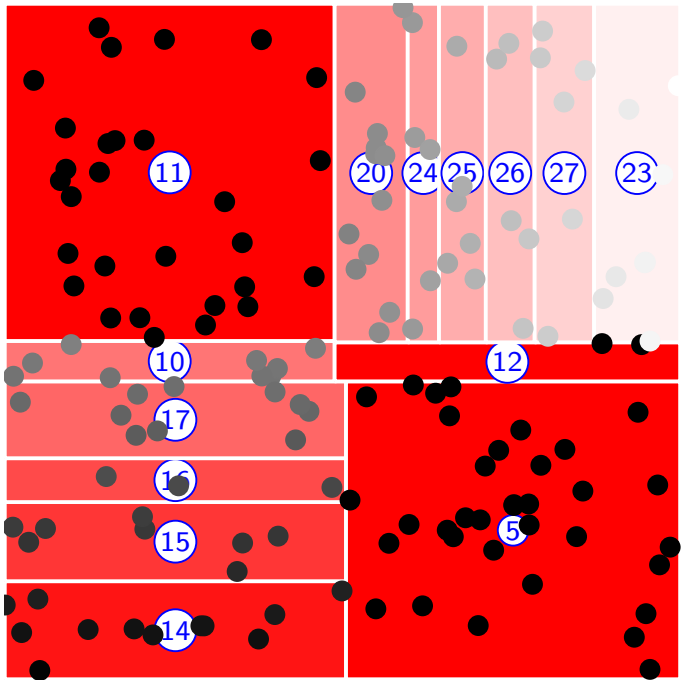
Example: Weakest Link Pruning

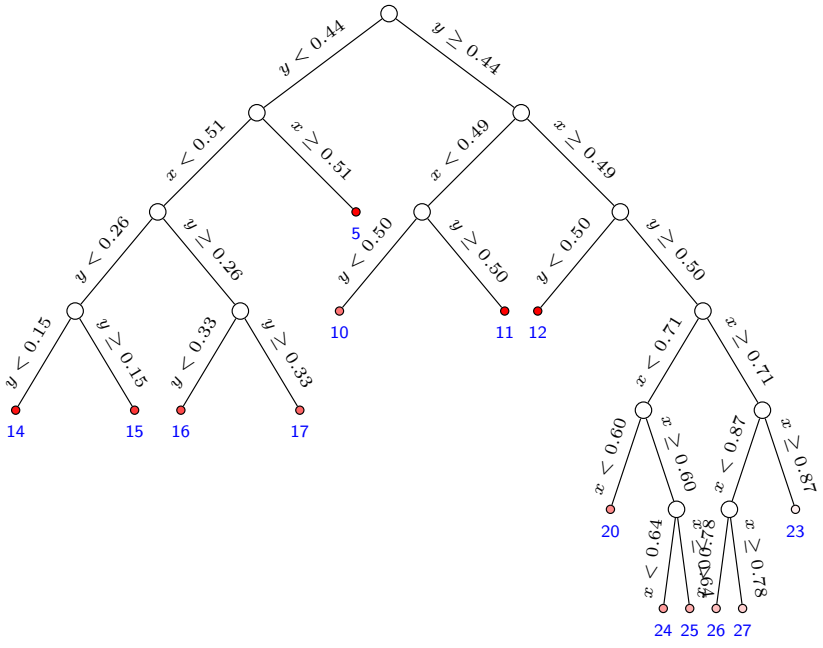




Merge 3

Example: Weakest Link Pruning

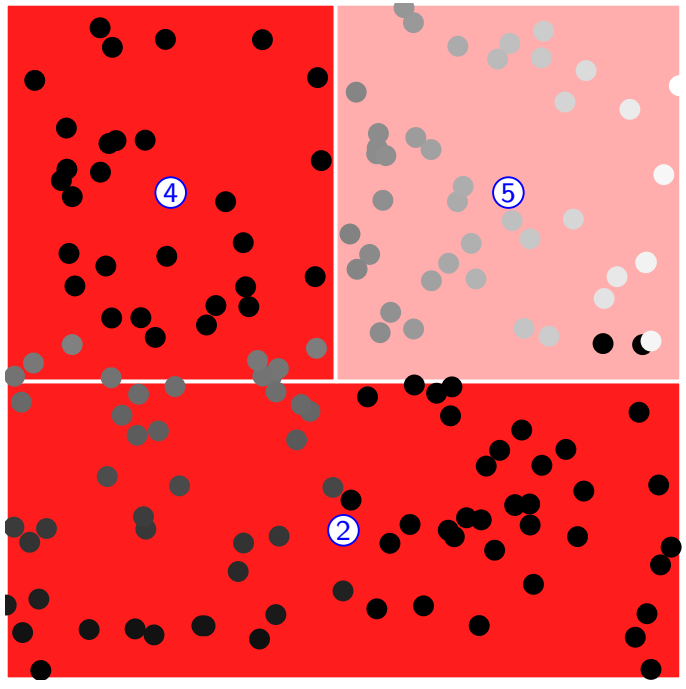


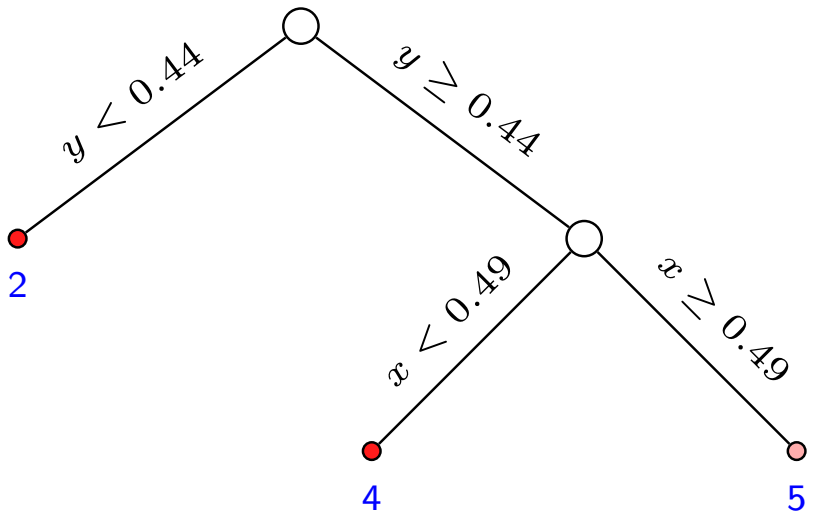


Merge 4

AND SO ON UNTIL

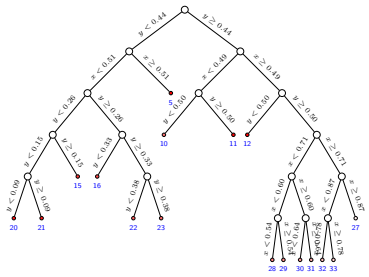
Example: Weakest Link Pruning

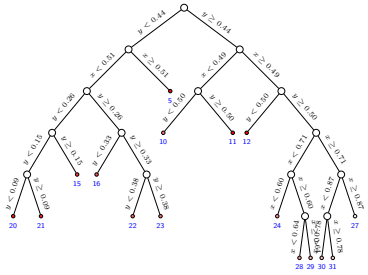


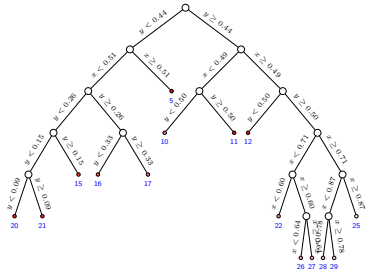


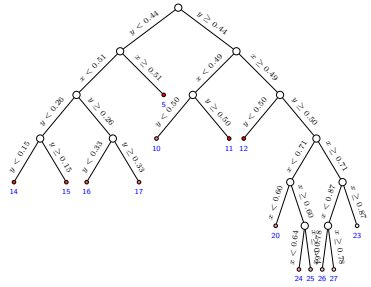
Merge 15

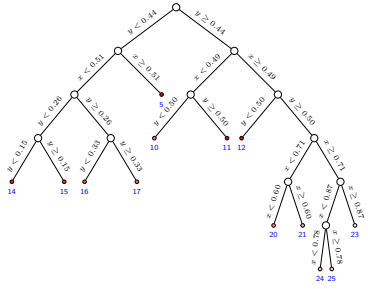
The Sequence of Pruned Trees

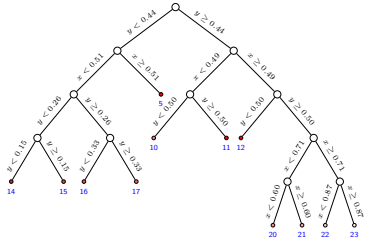


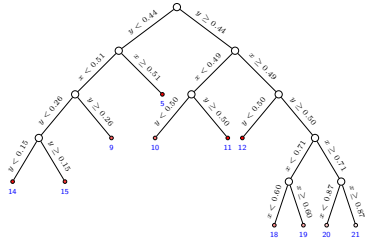


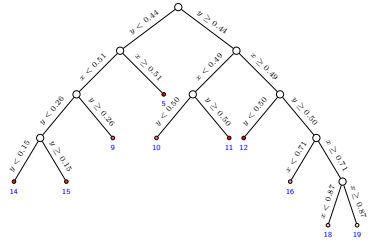


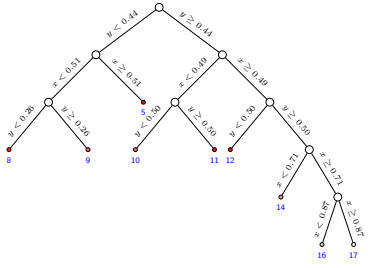


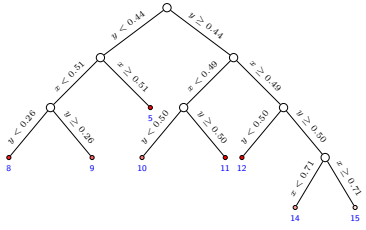


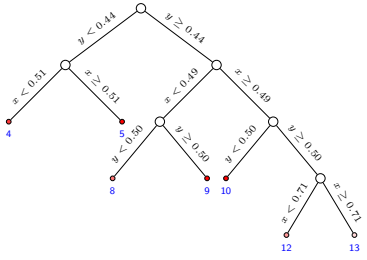


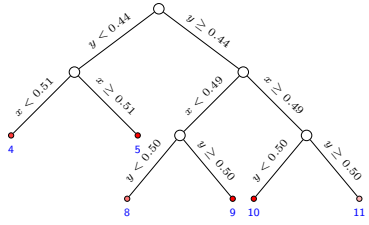


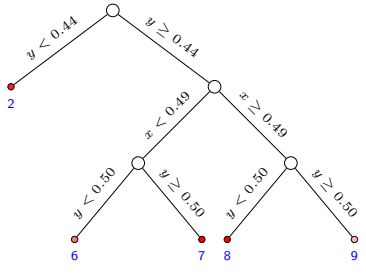


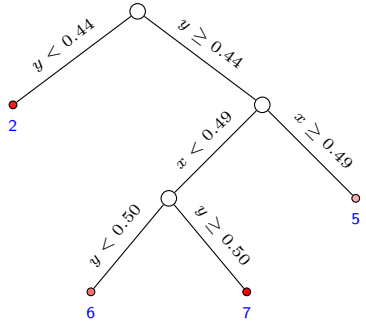


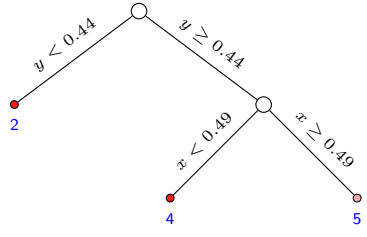




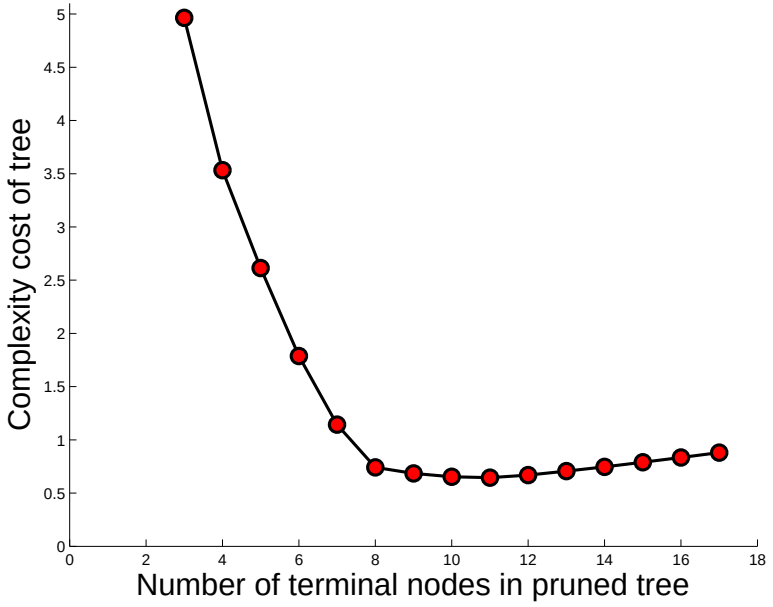






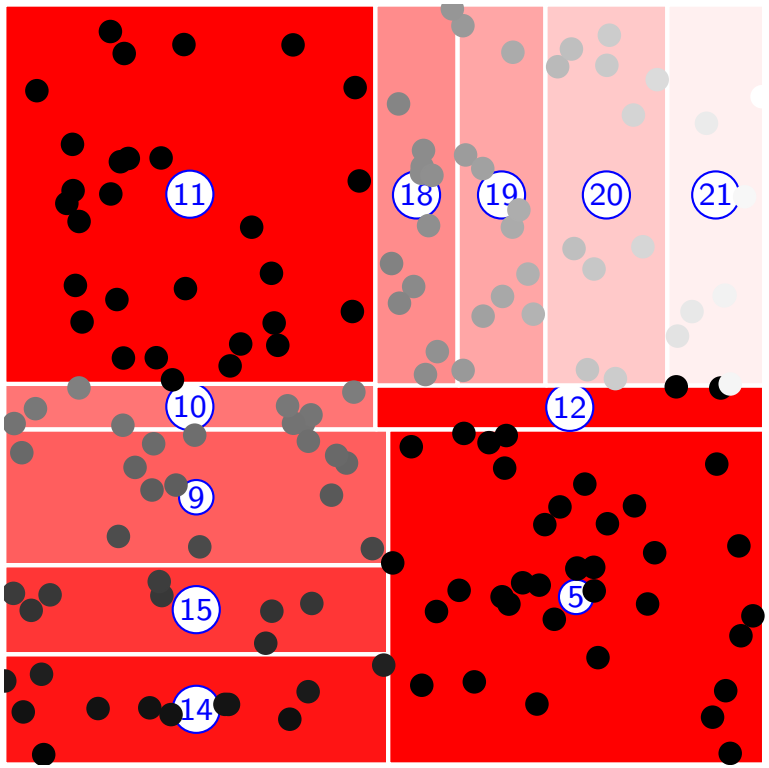


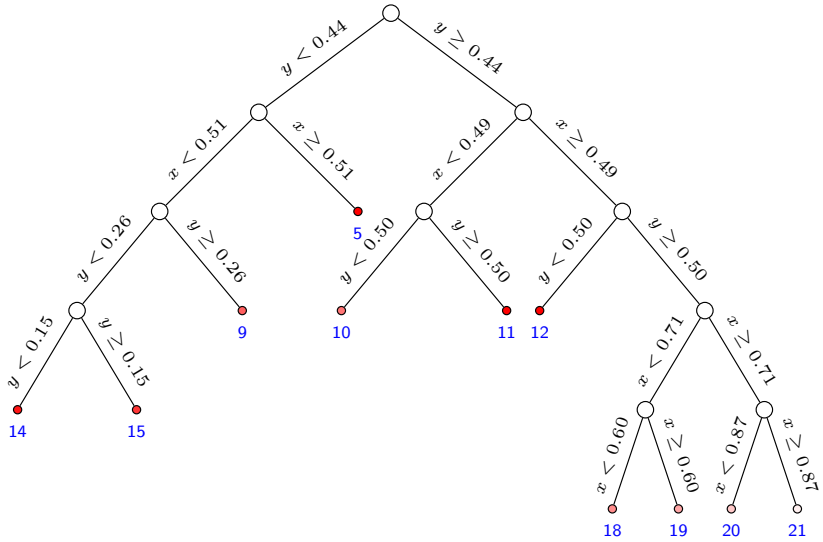
Complexity Cost of Pruned Trees



$C_{.05}(T)$ **for the pruned trees**

Lowest Cost Pruned Tree





Definitions needed for node impurity measures

- In node m, representing a region R_m with n_m observations let

$$\hat{p}_{mk} = \frac{1}{n_m} \sum_{x_i \in R_m} \text{Ind}(y_i = k)$$

  $\hat{p}_{mk}$ is the proportion of class k observations in node m.

- Classify the observation in node m to class

$$k(m) = \arg\max_k \hat{p}_{mk}$$

  the majority class in node m.

Different measures of node impurity

- Misclassification error:

$$\frac{1}{n_m} \sum_{i \in R_m} \mathsf{Ind}(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$$

- Gini index:

$$\sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right)$$

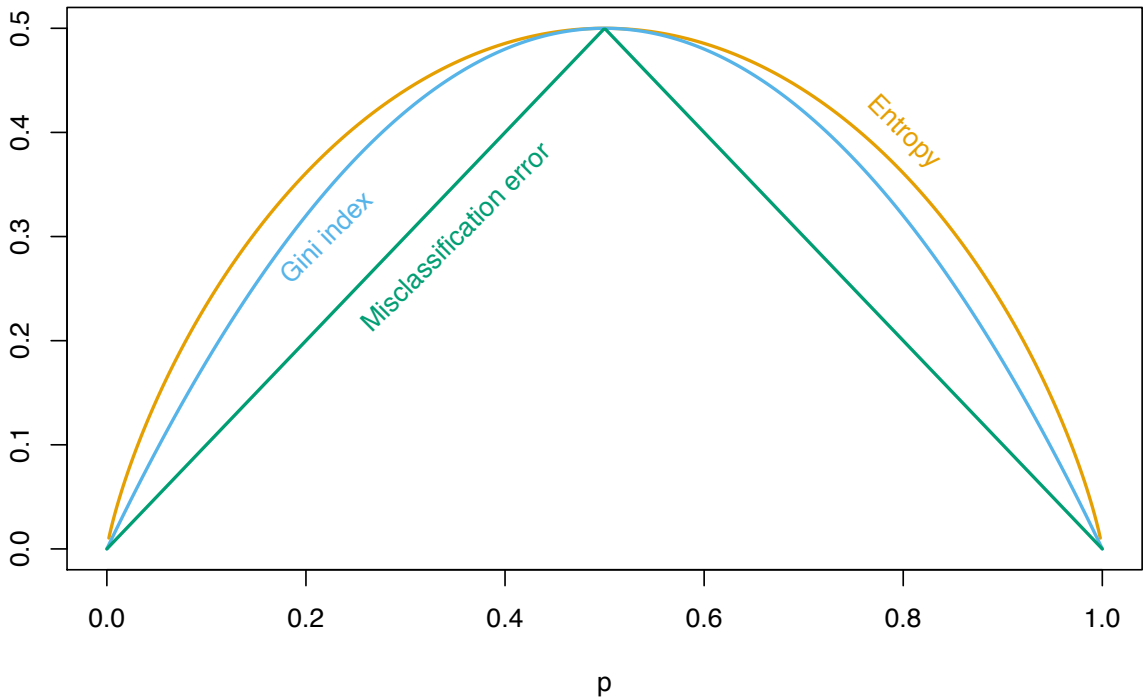
- Cross-entropy or deviance:

$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Illustration of Node Impurity Measures

For binary classification let $p = \hat{p}_{m0}$ then

- Misclassification error: $1 - \max(p, 1-p)$
- Gini index: $2p(1-p)$
- Cross-entropy or deviance: $-p\log(p) - (1-p)\log(1-p)$



p Vs **Impurity measure**

Comments on the impurity measures

- Cost of binary split of node m into nodes m_1 and m_2 is then

$$\frac{1}{n_{m_1}} Q_{m_1} + \frac{1}{n_{m_2}} Q_{m_2}$$

  where Q_{m_1} is the impurity measure of node m_1, slly Q_{m_2}.

- **Cross-entropy** and **Gini** are more sensitive to changes in the node probabilities than **Misclassification rate**.

- **Cross-entropy** and **Gini** measures used to grow trees.

- All measures used to prune tree.

Problems with Trees

- Instability
  - **Trees have high variance** due to hierarchical search process.
  - Errors at top nodes propagate to lower ones.

    $\Longrightarrow$ Small change in training data can give very different splits
- Lack of Smoothness
  - Regression Trees response surface is not smooth.
  - Not good if underlying function is smooth.
- Difficulty in Capturing Additive Structure
  - The **binary tree structure** preludes the discovery of additive structure like

    $$Y = c_1 \,\mathsf{Ind}(X_1 < t_1) + c_2 \,\mathsf{Ind}(X_2 < t_2) + \epsilon$$

    except fortuitously !

Patient Rule Induction Method (PRIM): Bump Hunting

PRIM: Overview

- **Aim**: Locate maximum in the response function.
- **What algorithm does**:

  Finds a rectangular box in the feature space which contains for

  - **Classification:** a clump of points of maximal purity
  - **Regression:** a plateau of high scoring points.

- **How ?**: A greedy search which is more patient than CART.

PRIM: Some definitions

- Box B is defined by the set of inequalities

$$a_j \leq X_j \leq b_j \quad \text{for } j = 1, \ldots, p$$

  where p is the dimension of the feature vectors.

- $B' = \text{NewBox}(B, k, 0, a)$ is defined by the inequalities

$$\begin{aligned} a_j \leq X_j \leq b_j &\quad \text{for } j = 1, \ldots, k-1 \\ a \leq X_k \leq b_k & \\ a_j \leq X_j \leq b_j &\quad \text{for } j = k+1, \ldots, p \end{aligned}$$

- $B' = \text{NewBox}(B, k, 1, b)$ is defined by the inequalities

$$\begin{aligned} a_j \leq X_j \leq b_j &\quad \text{for } j = 1, \ldots, k-1 \\ a_k \leq X_k \leq b & \\ a_j \leq X_j \leq b_j &\quad \text{for } j = k+1, \ldots, p \end{aligned}$$

- Let $n_B = \#$ of training observations in box B.

PRIM: The basic operations

Peeling - Decrease the size of box B for one face

- Define $B' = \text{Peel}(B, k, 0, \alpha)$ to be the box

$$B' = \text{NewBox}(B, k, 0, a)$$

where a is the smallest scalar for $\alpha \in (0, 1)$ s.t.

$$a > a_k \textbf{ and } n_{B'} \leq (1 - \alpha)\, n_B.$$

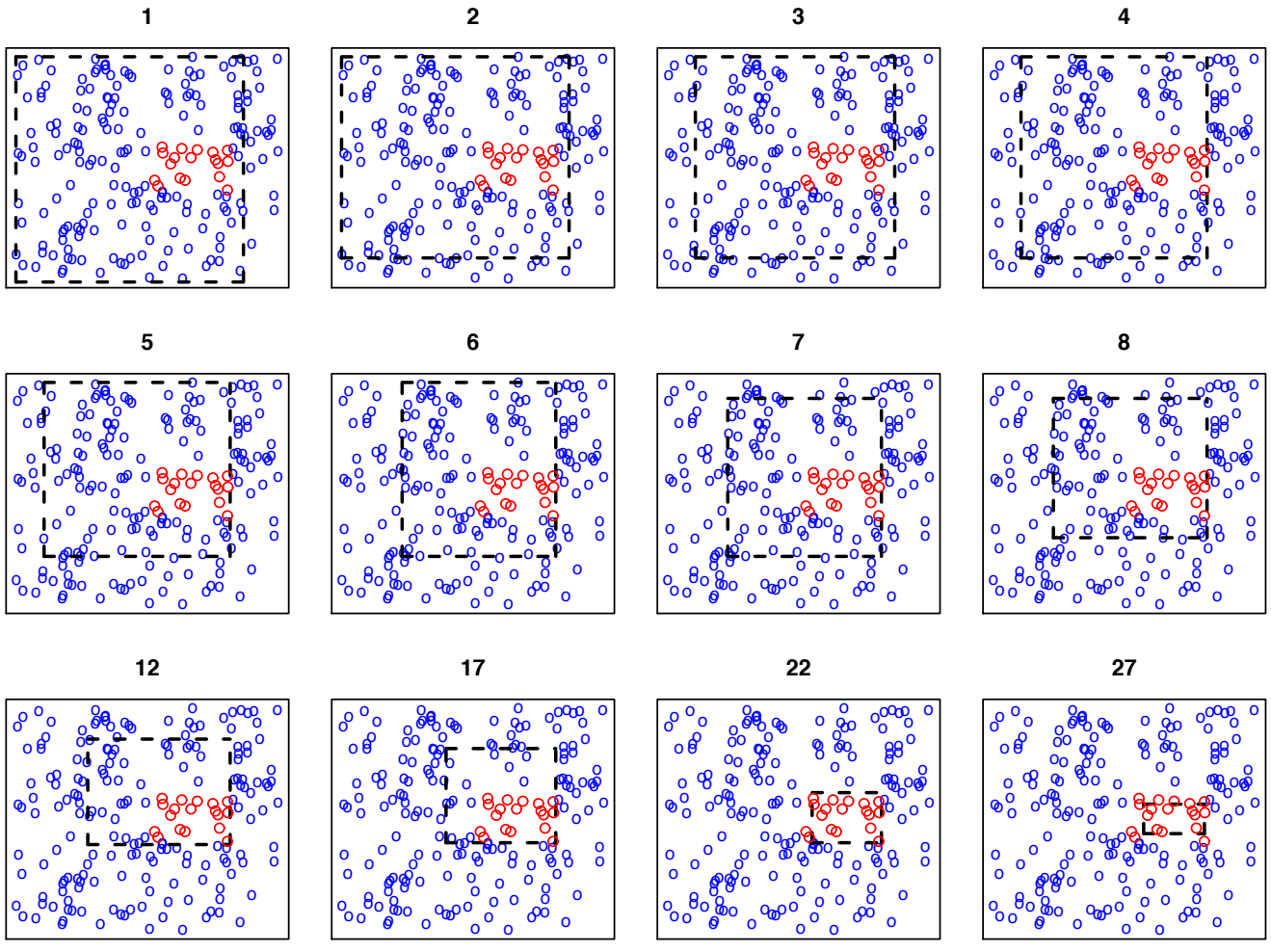
- Define $B' = \text{Peel}(B, k, 1, \alpha)$ to be the box

$$B' = \text{NewBox}(B, k, 1, b)$$

where b is the largest scalar for $\alpha \in (0, 1)$ s.t.

$$b < b_k \textbf{ and } n_{B'} \leq (1 - \alpha)\, n_B.$$

Example of peeling



Have points from two classes Red class and Blue class.

PRIM: The basic operations

Pasting - Increase the size of box B for one face

- Define $B' = \text{ExpandBox}(B, k, 0, \alpha)$ to be the box

$$B' = \text{NewBox}(B, k, 0, a)$$

where a is the largest scalar for $\alpha \in (0, 1)$ s.t.

$$a < a_k \textbf{ and } n_{B'} \geq (1 + \alpha)\, n_B.$$

- Define $B' = \text{ExpandBox}(B, k, 1, \alpha)$ to be the box

$$B' = \text{NewBox}(B, k, 1, b)$$

where b is the smallest scalar for $\alpha \in (0, 1)$ s.t.

$$b > b_k \textbf{ and } n_{B'} \geq (1 + \alpha)\, n_B.$$

Mean response

$$S_B = \frac{\sum_{x_i \in B} y_i}{\sum_{x_i \in B} 1}$$

PRIM algorithm

1. Set $i = 0$ and let $\alpha \in (0, 1)$.
2. Let B_0 be the minimal box containing all the data.
3. **Peeling process**: find sequence of decreasing nested boxes

   while (number of observations in B_i) $\geq n_m$

   - Compute the trimmed boxes $C_k = \text{Peel}(B_i, k, 0, \alpha)$ and $C_{k+p} = \text{Peel}(B_i, k, 1, \alpha)$ for $k = 1, \ldots, p$
   - Choose the C_{j*} with highest response mean.
   - Set $B_{i+1} = C_{j*}$. Set $i = i + 1$.

4. **Pasting process**: find sequence of increasing nested boxes

   For $k = 1, \ldots, p$

   - $C = \text{ExpandBox}(B_i, k, 0, \alpha)$, $D = \text{ExpandBox}(B_i, k, 1, \alpha)$.
   - Set $B_{i+1} = C$ and $i = i + 1$ if $S_C > S_{B_i}$ **and** $S_C > S_D$.
   - Set $B_{i+1} = D$ and $i = i + 1$ if $S_C > S_{B_i}$ **and** $S_D > S_C$.

PRIM algorithm ctd

1. Previous steps produces a sequence of boxes $B_1, \ldots, B_i$.
2. Use cross-validation to choose best box. Call this box B.
3. Remove the data in box B from the dataset.
4. Repeat the peeling and pasting steps and the cross-validation step to obtain a second box.
5. Continue these last two steps to get as many boxes as desired.

MARS: Multivariate Adaptive Regression Splines

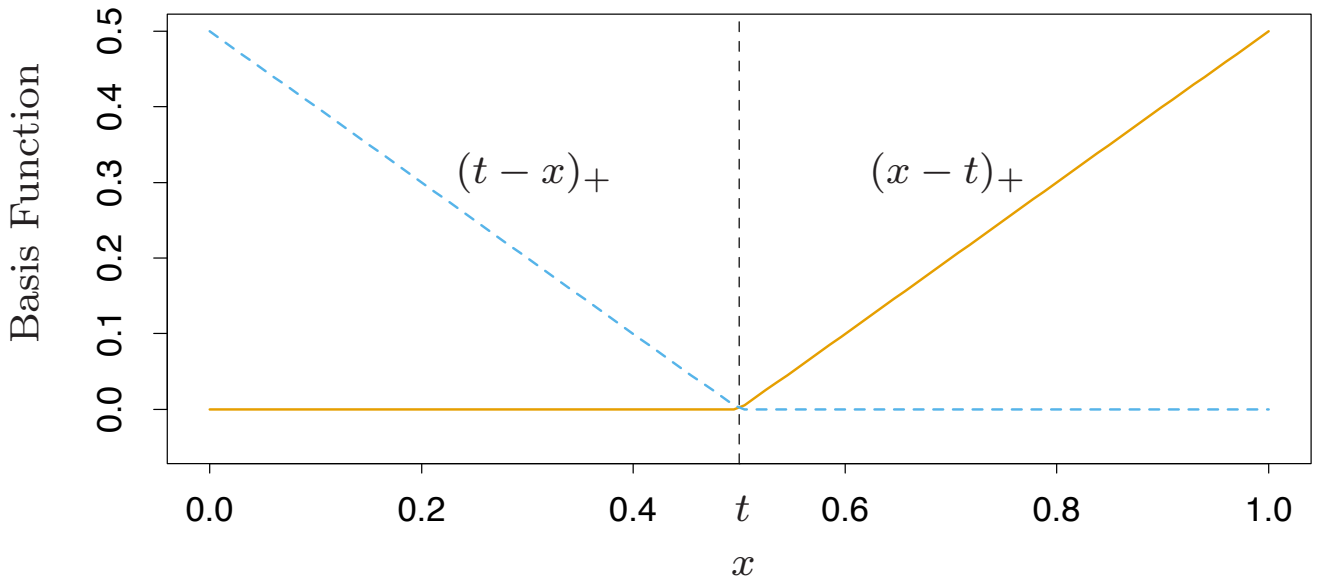
Introduction

- MARS is an adaptive procedure for regression.
- It is suitable for high-dimensional input spaces.
- Can be viewed as
  - a generalization of stepwise linear regression or
  - a modification of CART

Building blocks of MARS

MARS uses expansions in piecewise linear basis functions of the form

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



Basis functions used in MARS

- Have training data $(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i \in \mathbb{R}$ and

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^t \in \mathbb{R}^p$$

- For an input vector $X \in \mathbb{R}^p$ define

$$h_0(X, j, i) = (X_j - x_{ij})_+ \quad \textbf{and} \quad h_1(X, j, i) = (x_{ij} - X_j)_+$$

- Then define a collection of basis functions

$$\mathcal{C} = \{\, h_0(X, j, i),\, h_1(X, j, i) \,\}_{j=1,\ldots,p,\; i=1,\ldots,n}$$

Basis functions used in MARS

- Have training data $(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i \in \mathbb{R}$ and

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^t \in \mathbb{R}^p$$

- For an input vector $X \in \mathbb{R}^p$ define

$$h_0(X, j, i) = (X_j - x_{ij})_+ \quad \textbf{and} \quad h_1(X, j, i) = (x_{ij} - X_j)_+$$

- Then define a collection of basis functions

$$\mathcal{C} = \{\, h_0(X, j, i),\, h_1(X, j, i) \,\}_{j=1,\ldots,p,\; i=1,\ldots,n}$$

Basis functions used in MARS

- Have training data $(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i \in \mathbb{R}$ and

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^t \in \mathbb{R}^p$$

- For an input vector $X \in \mathbb{R}^p$ define

$$h_0(X, j, i) = (X_j - x_{ij})_+ \quad \textbf{and} \quad h_1(X, j, i) = (x_{ij} - X_j)_+$$

- Then define a collection of basis functions

$$\mathcal{C} = \{\, h_0(X, j, i),\, h_1(X, j, i) \,\}_{j=1,\ldots,p,\; i=1,\ldots,n}$$

Form of the regression function

- Then define a collection of basis functions

$$\mathcal{C} = \{\, h_0(X, j, i),\ h_1(X, j, i) \,\}_{j=1,\ldots,p,\ i=1,\ldots,n}$$

- Estimate the **regression function** using functions from $\mathcal{C}$ and product of functions from $\mathcal{C}$

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m\, g(X, \boldsymbol{\alpha}_m)$$

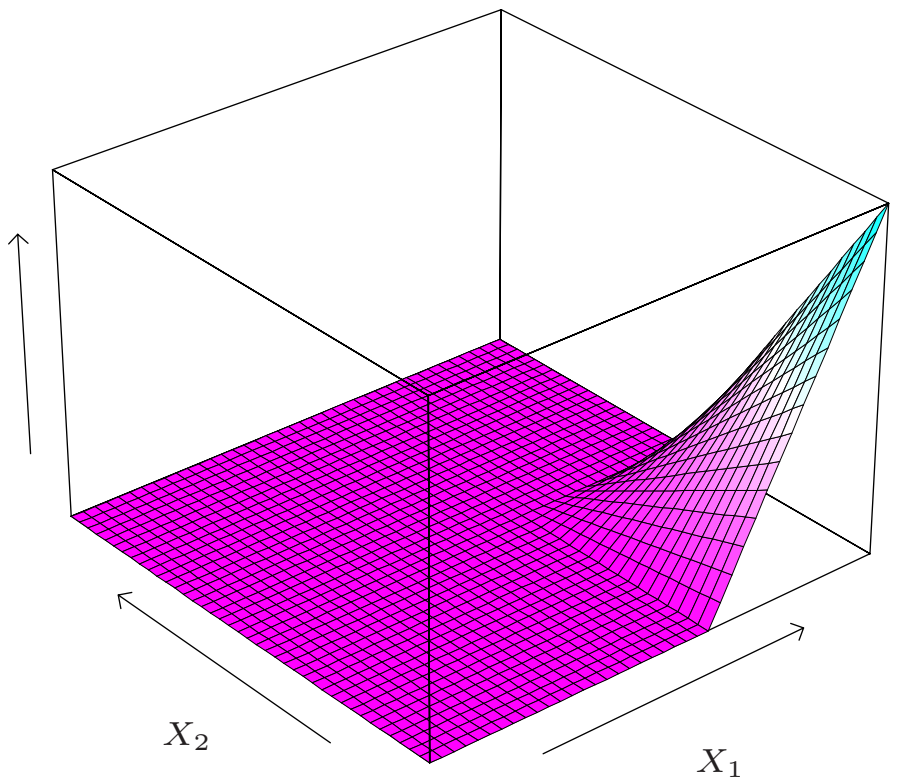
where each $\boldsymbol{\alpha}_m = (n_m, b_1, j_1, i_1, \ldots, b_{n_m}, j_{n_m}, i_{n_m})$ with $b_k \in \{0, 1\}$ such that

$$g(X, \boldsymbol{\alpha}_m) = \prod_{k=1}^{n_m} h_{b_k}(X, j_k, i_k)$$

Example of a product function

Shown below:

$$g(X, \boldsymbol{\alpha}) = h_0(X, 1, 5) \cdot h_1(X, 2, 7) = (X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$$



How to fit such a model? Forward model-building

- **Initially**: Set $g(X, \boldsymbol{\alpha}_0) \equiv 1$ and $\mathcal{M} = \{\, g(X, \boldsymbol{\alpha}_0) \,\}$
- while $|\mathcal{M}| < N$ perform the following:
  - for each $(m, j, i) \in \{1, \ldots, |\mathcal{M}|\} \times \{1, \ldots, p\} \times \{1, \ldots, n\}$
    1. Augment functions in $\mathcal{M}$ - $g(X, \boldsymbol{\alpha}_0), \ldots, g(X, \boldsymbol{\alpha}_{|\mathcal{M}|})$ - with

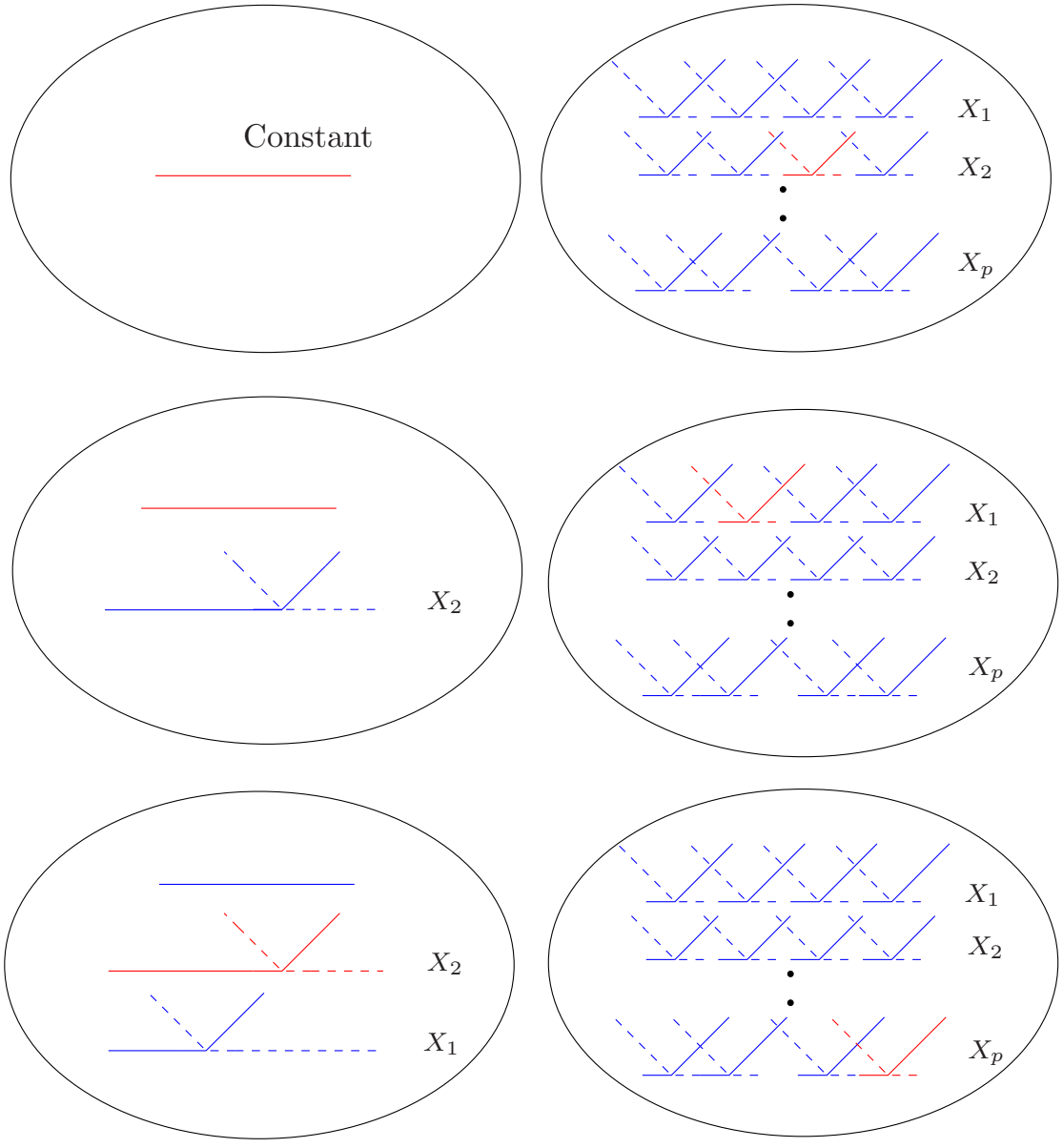
$$g(X, \boldsymbol{\alpha}_m) \cdot h_0(X, j, i) \text{ and } g(X, \boldsymbol{\alpha}_m) \cdot h_1(X, j, i)$$

    2. Use standard linear regression to estimate the $\hat{\beta}_l$'s s.t.

$$f_{\text{try}}(X) = \sum_{l=0}^{|\mathcal{M}|} \hat{\beta}_l \, g(X, \boldsymbol{\alpha}_l) + \hat{\beta}_{|\mathcal{M}|+1} \, g(X, \boldsymbol{\alpha}_m) \cdot h_0(X, j, i) + \hat{\beta}_{|\mathcal{M}|+2} \, g(X, \boldsymbol{\alpha}_m) \cdot h_1(X, j, i)$$

    3. Compute and record training error of $f_{\text{try}}(X)$
  - Let (m^*, j^*, i^*) be triplet producing lowest training error.
  - Add $g(X, \boldsymbol{\alpha}_{m^*}) \cdot h_0(X, j^*, i^*)$ and $g(X, \boldsymbol{\alpha}_{m^*}) \cdot h_1(X, j^*, i^*)$ to $\mathcal{M}$.

Schema of MARS forward model-building procedure



Pruning the model

- If $|\mathcal{M}|$ is large $\implies$ model likely to have overfit.
- Apply a backward deletion process.

  Iteratively remove the individual term which least affects performance.
- Select final model by
  - cross validation or
  - minimizing a criterion which trades-off **model size** and **training error**.

Piecewise linear functions and forward model-building?

- They can operate **locally**. The product

$$\prod_{k=1}^{n_m} h_{b_k}(X, j_k, i_k)$$

  is only non-zero where all the individual components are non-zero.

- Locality $\Longrightarrow$ forward model-building strategy can
  - build up the regression surface parsimoniously,
  - use parameters only where there is need.

  Very important for **high dimensional data**.

- Computational reasons - innermost loop of model building can be made very efficient.

- Hierarchical search avoids unnecessarily complicated terms.

Summary of the Simulation Experiments

Can learn the **underlying model** if it is

- an **additive** one between a subset of the input dimensions and output
- Can do this in the presence of additive noise.

Results not so good if relationship involves **higher order interactions** and **non-linearities**

Relationship of MARS to CART

If the MARS procedure is amended so that

1. Set $h_0(X, j, i) = \mathsf{Ind}(X_j - x_{ij} > 0)$ $\leftarrow$ step function
2. Set $h_1(X, j, i) = \mathsf{Ind}(X_j - x_{ij} \leq 0)$ $\leftarrow$ step function
3. When $g \in \mathcal{M}$ is chosen at one iteration s.t.

$$\mathcal{M} = \mathcal{M} \cup \{\, g(X) \cdot h_0(X, j, i) \,\} \cup \{\, g(X) \cdot h_1(X, j, i) \,\}$$

   remove g from $\mathcal{M}$.

then

MARS forward procedure == CART tree-growing algorithm.

Why is this the case?

- Multiplication of a step-function by a pair of reflected step functions $\equiv$ to splitting a node.
- 3rd change

$$\implies \text{a node cannot be split twice}$$
$$\implies \text{the binary tree representation of CART.}$$

- **Note** the last restriction implies cannot model additive structure well.

Why is this the case?

- Multiplication of a step-function by a pair of reflected step functions $\equiv$ to splitting a node.
- 3rd change

$$\implies \text{a node cannot be split twice}$$
$$\implies \text{the binary tree representation of CART.}$$

- **Note** the last restriction implies cannot model additive structure well.

Hierarchical Mixture of Experts

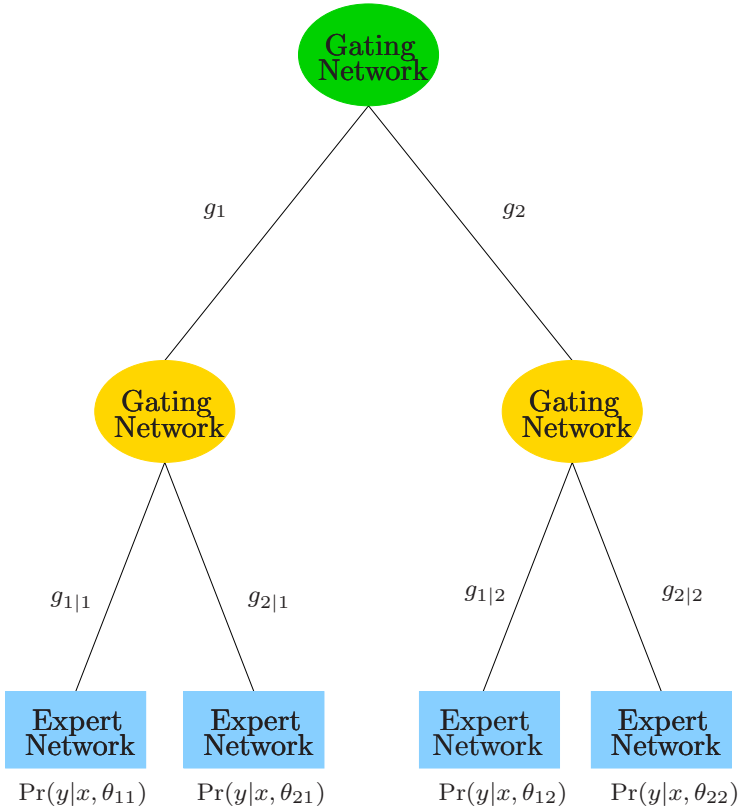
Hierarchical Mixture of Experts (HME)

- Variant of tree-base methods.
- Tree splits are **soft probabilistic ones** as opposed to hard ones.

  This may help

  - **parameter estimation** - optimize a smooth cost function
  - **prediction accuracy** - avoids discontinuities in the response function
- Splits can be multi-way.
- Splits are probabilistic functions of a linear combination of inputs.

HME Terminology

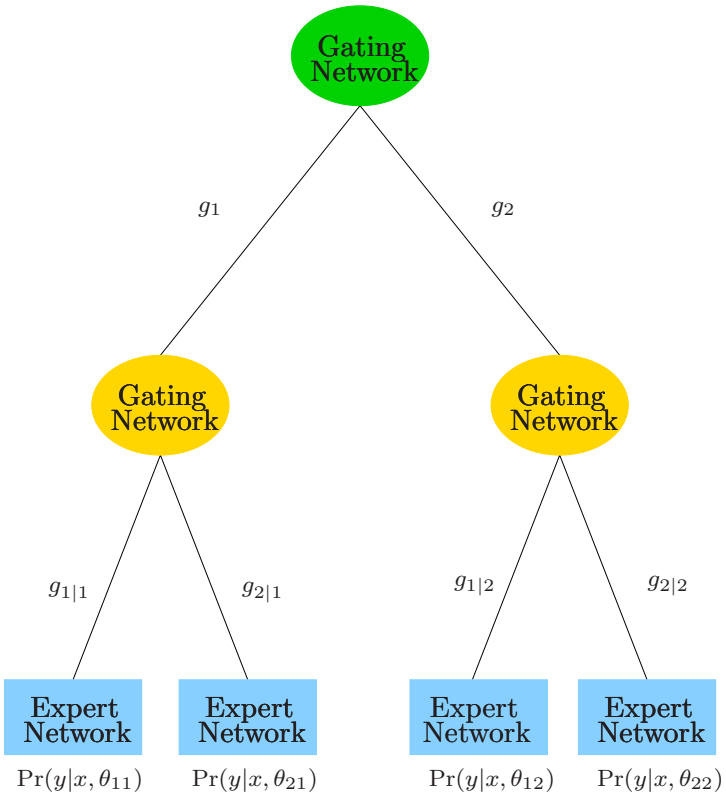


A two-level HME model

The network represents a mixture probability model.

- Terminal nodes called **experts**.
- Each expert represents a prediction of the response.
- Non-terminal nodes called **gating networks**.
- Expert predictions combined by the gating networks.

Details of the HME



A two-level HME model

- Top node is a soft K-way split

$$g_j(x,\gamma_j) = \frac{e^{\gamma_j^t x}}{\sum_{k=1}^K e^{\gamma_k^t x}}, \text{ for } j = 1,\ldots,K$$

$$= \text{prob of assigning } x \text{ to the } j\text{th branch.}$$

- 2nd level has similar soft splits

$$g_{l|j}(x,\gamma_{jl}) = \frac{e^{\gamma_{jl}^t x}}{\sum_{k=1}^K e^{\gamma_{jk}^t x}}, \text{ for } l = 1,\ldots,K$$

$$= \text{prob of assigning to } l\text{th branch given previous assignment to } j\text{th branch.}$$

- Terminal nodes model the response

$$Y \sim P(y \,|\, x, \theta_{jl})$$

Calculations at terminal nodes

- The models used for different problems:

  - **Regression**: **Gaussian linear regression model**

$$P(y \mid x, \theta_{jl}) = \mathcal{N}(\beta_{jl}^t\, x,\ \sigma_{jl}^2) \quad \text{where } \theta_{jl} = (\beta_{jl}, \sigma_{jl}^2)$$

  - **Classification**: **Linear logistic regression model**

$$P(Y = 1 \mid x, \theta_{jl}) = \frac{1}{1 + e^{-\theta_{jl}^t\, x}}$$

Hidden mixture of experts

- The HME represents a mixture probability model.
- The mixture probabilities are determined by the soft splits

$$P(y|x,\Psi) = \sum_{j=1}^{K} g_j(x,\gamma_j) \sum_{l=1}^{K} g_{l|j}(x,\gamma_{lj})\, P(y \,|\, x,\theta_{jl})$$

  where $\Psi = \{\gamma_j, \gamma_{jl}, \theta_{jl}\}$.

- Estimate Ψ by maximizing the **log-likelihood**

$$\max_{\Psi} \sum_{i=1}^{n} \log P(y_i \,|\, x_i, \Psi)$$

  of training data $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^{n}$.

Estimate the parameters of a HME using EM

- Introduce the hidden variables Δ_j^i and $\Delta_{l|j}^i$ to indicate the *underlying* branching decisions made.

- **E-step**: Compute posterior probabilities of Δ_j^i and $\Delta_{l|j}^i$ given $\Psi^{(t)}$.

- **M-step**: Compute $\Psi^{(t+1)}$ by maximization of the expected log-likelihood

$$\Psi^{(t+1)} = E_{\Delta|\mathcal{X},\Psi^{(t)}}[\log L(\Psi; \Delta, \mathcal{X})]$$

HME or CART ?

✓ Advantages of HMEs over CART

- **Smooth final regression function.**
  Soft splits allow for smooth transitions from high to low responses.

- **Easier to optimize for parameters.**
  The log-likelihood is a smooth function and is amenable to numerical optimization.

✗ Disadvantages of HMEs over CART

- **Tree topology ?**
  No good way to find it for HME.

- **Harder to interpret the model**
  Not so clear cut which factors cause which effects.