9. Additive Models, Trees, and Related Methods

오영민

Introduction

We discuss some specific methods which assume a structured form for unknown regression function, and by doing so they finesse the curse of dimensionality.

- **GAM**: Generalized Additive Models
- **Trees**
- **PRIM**: The Patient Rule Induction Method
- **MARS**: Multivariate Adaptive Regression Splines
- **HME**: Hierarchical Mixtures of Experts

Generalized Additive Models

- Automatic flexible statistical methods that may be used to identify and characterize **nonlinear** regression effects.
- $g(\mu(X)) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$ where $\mu(X) = \mathbb{E}[Y \mid X_1, \ldots X_p]$ and $g(\cdot)$ is link function.
    1. Specify the probability **density of response** variable Y e.g. Gaussian for linear regression, Binomial for binary classification.
    2. Specify the **systematic component**. In this case, $\alpha + f_1(X_1) + \ldots + f_p(X_p)$
    3. Specify the **link function** between the response and systemic component.

- Examples of classical link functions for the exponential family of distributions:
    - $g(\mu) = \mu$ is the identity link, used for linear and additive models for Gaussian response data
    - $g(\mu) = logit(\mu) = \sigma^{-1}(\mu)$ or $g(\mu) = probit(\mu) = \Phi^{-1}(\mu)$ for modeling binomial probabilities
    - $g(\mu) = \log(\mu)$ for log-linear or log-additive models for Poisson count data

Generalized Additive Models

- Example of systematic component:
  - $g(\mu) = X^T\beta + \alpha_k + f(Z)$, a **semi-parametric model** where first term is the parametric part, second is the effect for the k-th level of a qualitative input V, and the non-parametric part $f(Z)$.
  - $g(\mu) = f(X) + g_k(Z)$ where k indexes the level of qualitative input V, $g_k(Z) = g(V, Z)$ is the interaction term.
  - $g(\mu) = f(X) + g(Z, W)$ where g is a non-parametric function in Z, W.

Generalized Additive Models

- Fitting Additive Models: In Regression problem, $Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon$
- The penalized sum of squares:

$$PRSS(\alpha, f_1, \ldots, f_p) = \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j, \ \lambda_j \geq 0$$

- We know that the minimizer of each f_j is a cubic spline with knots at $x_{1j}, \ldots, x_{Nj}$ in chapter 5, but α is not unique.
- The standard convention is to assume that $f_j(x_{1j}) + \ldots + f_j(x_{Nj}) = 0$ for all j.Then $\hat{\alpha} = ave(y_i)$.

Algorithm 9.1 *The Backfitting Algorithm for Additive Models.*

1. Initialize: $\hat{\alpha} = \frac{1}{N} \sum_1^N y_i, \ \hat{f}_j \equiv 0, \forall i, j.$
2. Cycle: $j = 1, 2, \ldots, p, \ldots, 1, 2, \ldots, p, \ldots,$

$$\hat{f}_j \leftarrow \mathcal{S}_j \left[\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \}_1^N \right], \quad \text{(Back-fitting step)}$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N} \sum_{i=1}^{N} \hat{f}_j(x_{ij}). \quad \text{(Mean centering of } \hat{f}_j\text{)}$$

until the functions $\hat{f}_j$ change less than a prespecified threshold.

Generalized Additive Models

- Fitting Additive Models: In logistic model for binary response, $\log \frac{Pr(Y=1\,|\,X)}{Pr(Y=0\,|\,X)} = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon$
- For log-likelihood loss, $f_1, \ldots, f_p$ are estimated by back-fitting algorithm with a Newton-Raphson procedure.

Algorithm 9.2 *Local Scoring Algorithm for the Additive Logistic Regression Model.*

1. Compute starting values: $\hat{\alpha} = \log[\bar{y}/(1-\bar{y})]$, where $\bar{y} = \text{ave}(y_i)$, the sample proportion of ones, and set $\hat{f}_j \equiv 0\ \forall j$.
2. Define $\hat{\eta}_i = \hat{\alpha} + \sum_j \hat{f}_j(x_{ij})$ and $\hat{p}_i = 1/[1+\exp(-\hat{\eta}_i)]$.
   Iterate:
   (a) Construct the working target variable
   $$z_i = \hat{\eta}_i + \frac{(y_i - \hat{p}_i)}{\hat{p}_i(1-\hat{p}_i)}.$$
   (b) Construct weights $w_i = \hat{p}_i(1-\hat{p}_i)$
   (c) Fit an additive model to the targets z_i with weights w_i, using a weighted backfitting algorithm. This gives new estimates $\hat{\alpha}, \hat{f}_j, \forall j$
3. Continue step 2. until the change in the functions falls below a prespecified threshold.

If back-fitting algorithm is replaced by the overall weighted least squares fit, this algorithm is **identical to IRLS** for solving the maximum likelihood equations in linear logistic regression

Trees

- Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one.

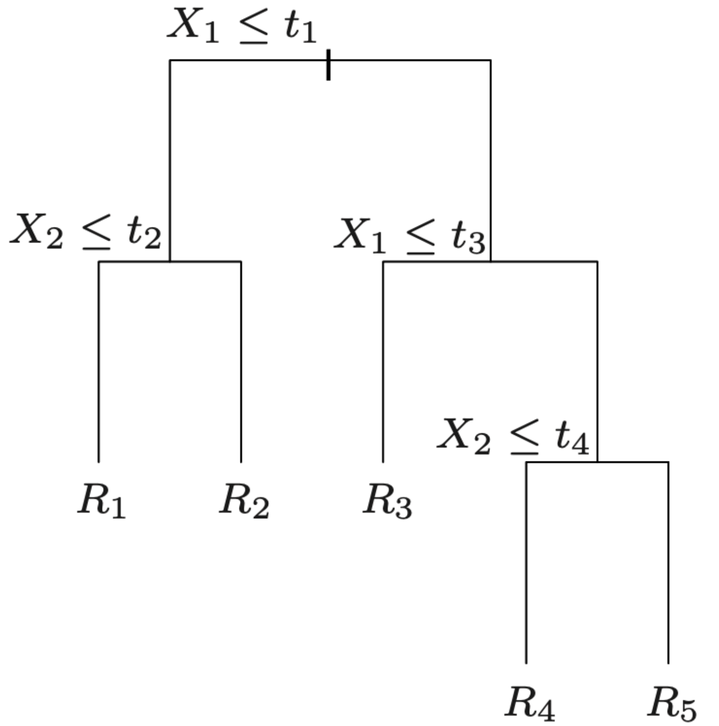
- Popular method: **Classification And Regression Trees(CART)**, ID3, C4.5, C5.0

- To simplify matters, we restrict attention to recursive binary partitions.

    1. Choose the best X_j

    2. Choose the best split point.

- A key advantage of the recursive binary tree is its interpretability.



2-dimensional feature space by recursive binary splitting

Regression Trees

- Given the set of regions $\{R_1, \ldots, R_M\}$, $f(x) = \sum_m c_m I(x \in R_m)$
- For sum of squares error, $\hat{c}_m = ave(y_i | x_i \in R_m)$ in each region R_m
- However, finding the best binary partition of SSE is generally computationally infeasible. -> Use **greedy** algorithm.
- We seek the splitting variable j and its splitting point s that solve

$$(j, s) = argmin_{j,s} \left[min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$ where $R_1(j, s) = \{X | X_j \leq s\}$, $R_2(j, s) = \{X | X_j > s\}$

- Then for each (j, s), $\hat{c}_1 = ave(y_i | x_i \in R_1(j, s))$, $\hat{c}_2 = ave(y_i | x_i \in R_2(j, s))$
- Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions.

Regression Trees

- Model selection

  1. Cross Validation -> There are too many possibilities, so we would still over-fit.

  2. Stop growing the tree when the decrease in RSS doesn't exceed ϵ -> It's not good for greedy algorithm.

- The preferred strategy is to grow a large tree T_0 (e.g. min node size ≥ 5). Then T_0 is pruned using **cost-complexity pruning**.

- **[Weakest link pruning]** Define a subset $T \subset T_0$ to be any tree that can be obtained by pruning T_0 .

  Then, $T_1 = \underset{T \subset T_0}{min} \dfrac{RSS(T) - RSS(T_0)}{|T_0| - |T|}$ where $|T|$ is # of leaf node in T. And iterate this pruning to obtain $T_0, T_1, \ldots, T_S$

  where T_s is the root tree.

Regression Trees

- **[The cost complexity pruning]** Letting $N_m = \#$ of x_i in R_m, $\hat{c}_m = ave(y_i \,|\, x_i \in R_m)$, $Q_m(T) = \dfrac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$
- The cost complexity criterion:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T| = \text{(sum of all RSS in each } R_m) + \alpha \cdot (\#\text{ of nodes in } T),\ \alpha \geq 0$$

- $T_\alpha = \underset{T \in T_0}{argmin} C_\alpha(T)$
- Theorem1. $\forall \alpha \geq 0,\ \exists! T_\alpha\ s.t.\ T_\alpha = \underset{T \in T_0}{argmin} C_\alpha(T)$
- Theorem2. $T_\alpha \in \{T_1, \ldots, T_S\}$
- For $\alpha = 0$, we select T_0 , and for $\alpha = \infty$, we select T_S . Choose the best α by 5- or 10-fold cross validation

Classification Trees

- No major differences between regression and classification tress: Target y taking values $\{1,...,K\}$.
- Define the loss function and estimators in each region $\hat{k}_1, \hat{k}_2$.
- For 0-1 loss,

$$(j,s) = argmin_{j,s}\left[min_{k_1}\sum_{x_i \in R_1(j,s)} I(y_i \neq k_1) + min_{k_2}\sum_{x_i \in R_2(j,s)} I(y_i \neq k_2)\right]$$

- Then for each (j,s), $\hat{k}_1, \hat{k}_2$ are Bayes estimators i.e. **majority vote**:

$$\hat{k}_1 = \underset{k\in\{1,...,K\}}{argmax}\,\hat{p}_{1k} \text{ where } \hat{p}_{1k} = ave(I(y_i = k)\,|\,x_i \in R_1(j,s)),\ k = 1,...,K$$

$$\hat{k}_2 = \underset{k\in\{1,...,K\}}{argmax}\,\hat{p}_{2k} \text{ where } \hat{p}_{2k} = ave(I(y_i = k)\,|\,x_i \in R_2(j,s)),\ k = 1,...,K$$

- Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions : $R_1, R_2, R_3, \ldots$.

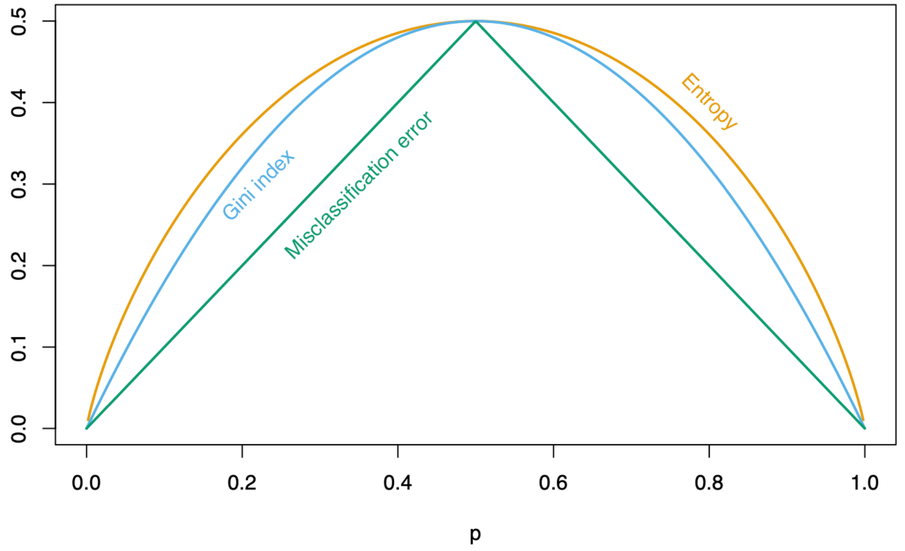
Classification Trees

- **[The cost complexity pruning]** Letting $N_m = \#$ of x_i in R_m, $\hat{k}_m = \underset{k \in \{1,\ldots,K\}}{argmax}\, \hat{p}_{mk}$, $Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i \neq \hat{k}_m) = 1 - \hat{p}_{m\hat{k}_m}$

- Different measures $Q_m(T)$ of node impurity include the following:

  Gini index: $\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$

  Cross entropy or Deviance: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

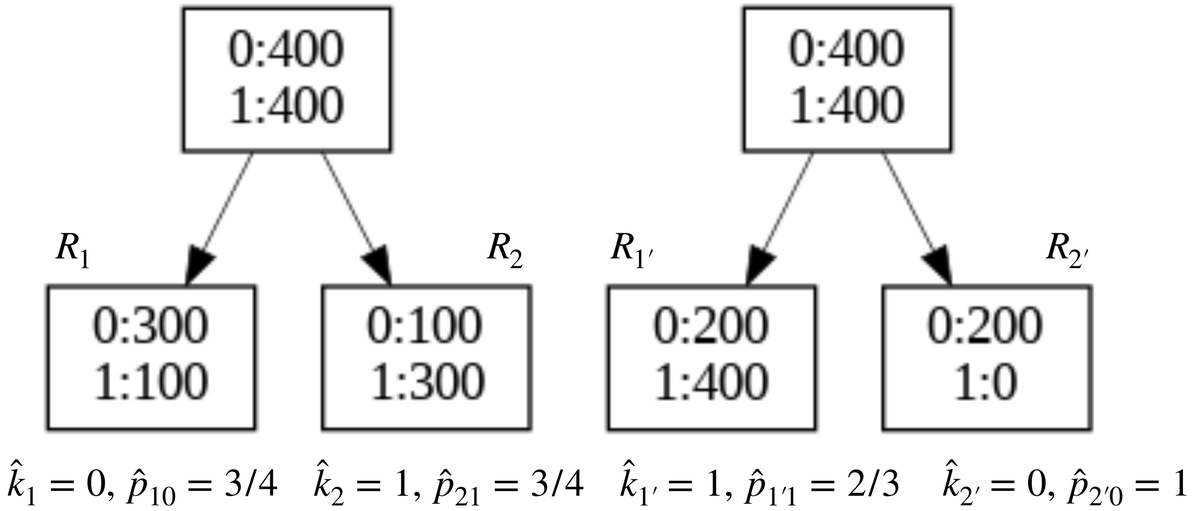


Node impurity measures for binary-classification

- To guide cost-complexity pruning, any of the three measures can be used, but typically it is the misclassification rate

Classification Trees

- Which split is better?



The the second split produces a pure node and is probably preferable.

Misclassification eror rate:

$(400 \times 1/4 + 400 \times 1/4)/800 = 0.25$ in first split,

$(600 \times 1/3 + 200 \times 0)/800 = 0.25$ in second split.

Gini index:

$(400 \times 3/4 \times 1/4 \times 2 \times 2)/800 = 0.375$ in first split,

$(600 \times 2/6 \times 4/6 \times 2 + 200 \times 1 \times 0 \times 2)/800 = 0.333$ in second split.

Cross entropy:

$(400 \times (-0.75 \log 0.75 - 0.25 \log 0.25) \times 2)/800 = 0.562$ in first split,

$(600 \times (-2/3 \log 2/3 - 1/3 \log 1/3) + 200 \times (-1 \log 1 - 0 \log 0)/800 = 0.477$ in second split.

- Cross-entropy and the Gini index are more sensitive to changes in the node probabilities than the misclassification rate.
- It is typical to use the Gini index or cross-entropy for **growing** the tree, while using the misclassification rate when **pruning** the tree.

Other issues in CART

- **Unordered Categorical predictors** $X_j \in \{C_1, . , , , . C_q\}$ **:** There are $2^{q-1} - 1$ possible partitions into 2 groups.

  For 0-1 classification, order the $\{C_1, \dots C_q\}$ by calculate $\sum_{i=1}^{N} I(y_i = 1 \,|\, x_{ij} = C_k)$ for $k = 1,...,q$. Then we split X_j as if it were an ordered predictor. This gives the **optimal split**, in terms of cross-entropy(or gini index), among all possible splits. And it also holds for quantitative y. However, it does not hold for multi-category y. Tree arises severe overfitting if q is large, and such variables should be avoided.

- **The loss matrix** $\mathbf{L}$: It is probably worse to misclassify that a person will not have a heart attack when he/she actually will, than vice versa.

  Define the loss matrix $\mathbf{L} \in \mathbb{R}^{K \times K}$ with $L_{kk'}$ is $loss(\hat{y} = k' \,|\, y = k)$ thus $L_{kk} = 0$ for all k. For gini index, $\sum_{k \neq k'} L_{kk'} \hat{p}_{mk} \hat{p}_{mk'}$. For 0-1 classification, its $(L_{10} + L_{01}) p(1-p)$ which does not work. -> **Observation weighting**: $L_{10} x_i$ for $y_i = 1$, $L_{01} x_i$ for $y_i = 0$ which is to alter the prior probability on the classes. It can be used on multi-class if $L_{kk'}$ does not depend on k', and can be used with the deviance as well.

Other issues in CART

- **The loss matrix $\mathbf{L}$**: Bayes estimator in observation weighting: $\hat{k}_m = \underset{k \in \{1,\ldots,K\}}{argmin} \sum_l L_{lk} \hat{p}_{ml}$.

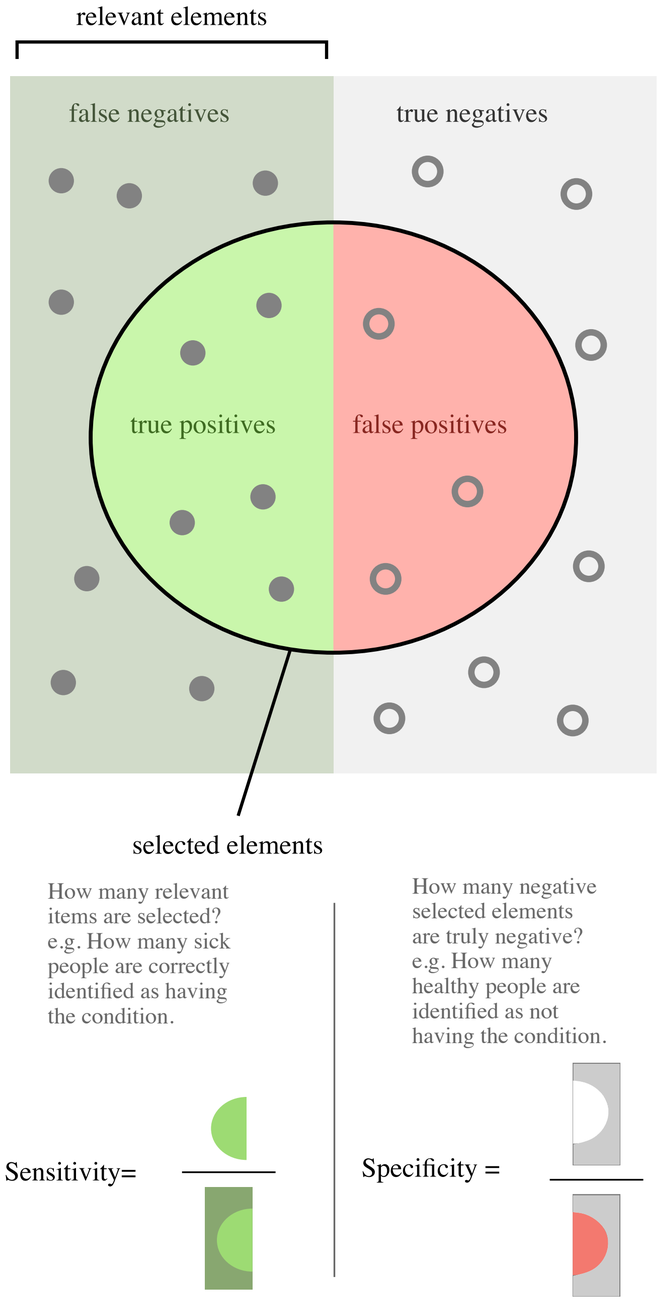
For 0-1 classification,

$$\hat{k}_m = \underset{k \in \{1,\ldots,K\}}{argmin} \sum_l L_{lk} \hat{p}_{ml} = argmin_k \left(L_{10} p,\ L_{01}(1-p) \right) \text{ where } p = \hat{p}_{m1}$$

$$\hat{k}_m = 1 \text{ if } p < \frac{L_{01}}{L_{10} + L_{01}} \text{ and } 0 \text{ o.t.}$$

By varying the relative size of L_{10} and L_{01}, we increase the sensitivity and decrease the specificity of the rule, or vice versa.

To make the specificity to be high, let $L_{01} > 1$ with $L_{10} = 1$.



Other issues in CART

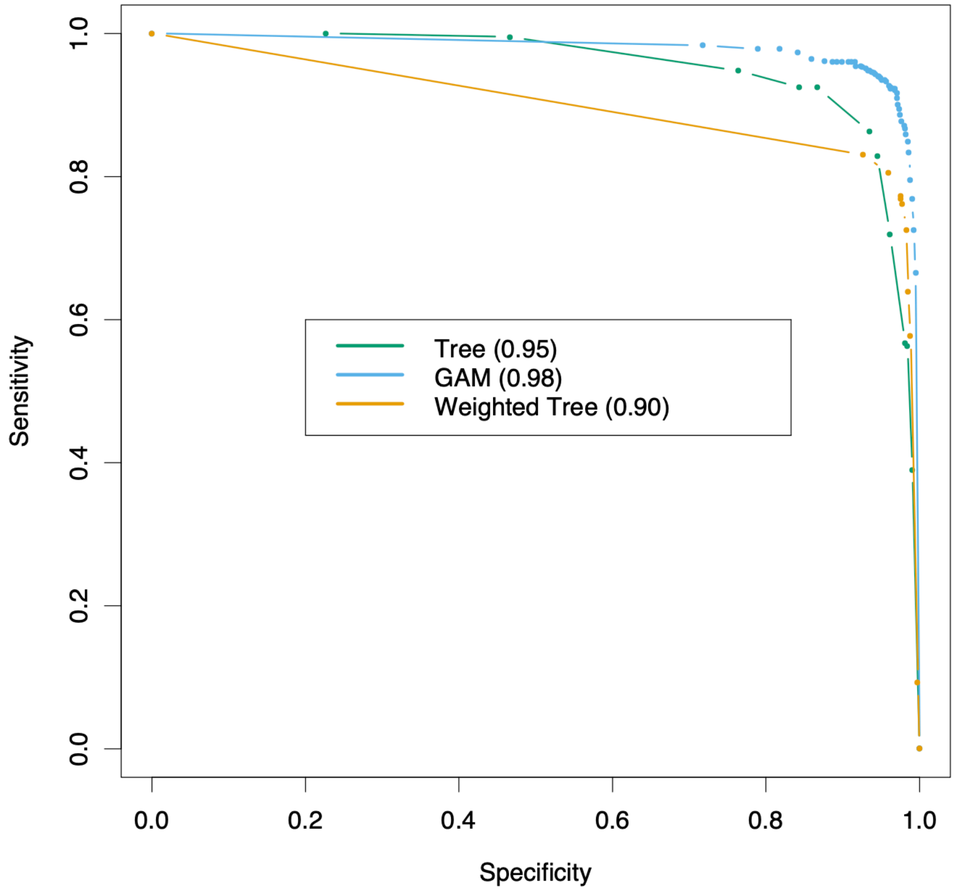
- **The loss matrix $\mathbf{L}$**: The Receiver Operating Characteristic curve **(ROC curve)** summarizes for assessing the trade-off between sensitivity and specificity.

In this case, we plot the ROC curve for the classification rules fit to the spam data. Curves that are **closer to the northeast corner** represent better classifiers.

The area under the ROC curve, used above, is sometimes called the c-statistic.(or Area Under the Curve, AUC).

For evaluating the **contribution of an additional predictor** when added to a standard model, the **c-statistic may not be an informative measure**.

The new predictor can be very significant in terms of the change in model deviance, but show only a small increase in the c-statistic.



Other issues in CART

- **Missing predictor values:**

  A. For classification, we simply make a new category for “missing”.

  B. Construction of the surrogate variables:

     1. In splitting, we use only observations for which that predictor is not missing.

     2. Having chosen the best (j, s), we form a list of surrogate $\{(j_i, s_i)\}_{i=1}^{P-1}$ that most similar split achieved by (j, s).

     3. In prediction $x = (x_1, \ldots, x_P)$, for $R_m(j, s)$ if x_j is missing, then use (j_1, s_1), also x_{j_1} is missing, use (j_2, s_2),…

  Surrogate splits exploit correlations between predictors to try and alleviate the effect of missing data.

- **Why binary splits?:** Multiway splits can be achieved by a series of binary splits and it leaving insufficient data at the next level down.

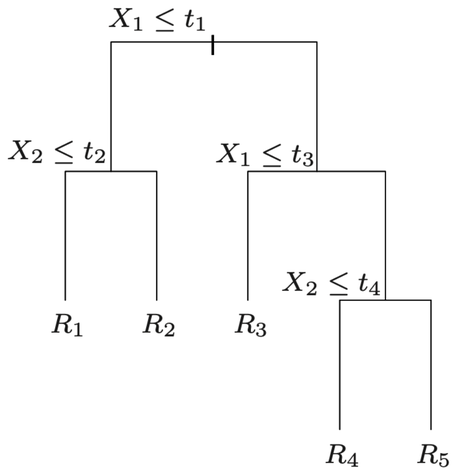
Other issues in CART

- **Instability of Trees:** One major problem with trees is their **high variance** due to hierarchical nature of the process.The effect of an error in the top split is propagated down to all of the splits below it. -> **Bagging** averages many trees.

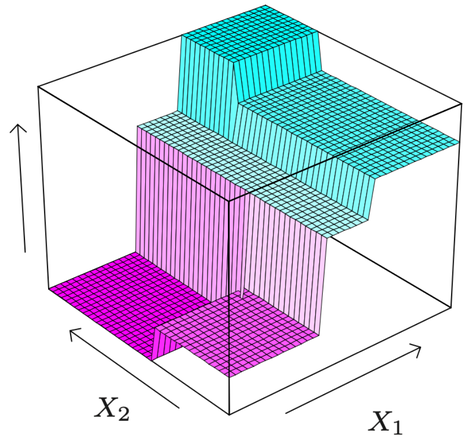
- **Lack of Smoothness:** This can degrade performance in the **regression** setting, where we would normally expect the underlying function to be smooth. MARS can be viewed as a modification of CART designed to alleviate this problem.

- **Difficulty in capturing additive structure:** Suppose that $Y = c_1 I(X_1 < t_1) + c_2 I(X_2 < t_2) + \epsilon$

  If its first split on X_1 near t_1. At the next level down it would split **both nodes** on X_2 at t_2 in order to capture the additive structure. When there are more additive structures, it is more difficult to capture the additive effect.



2-dimensional feature space by recursive binary splitting

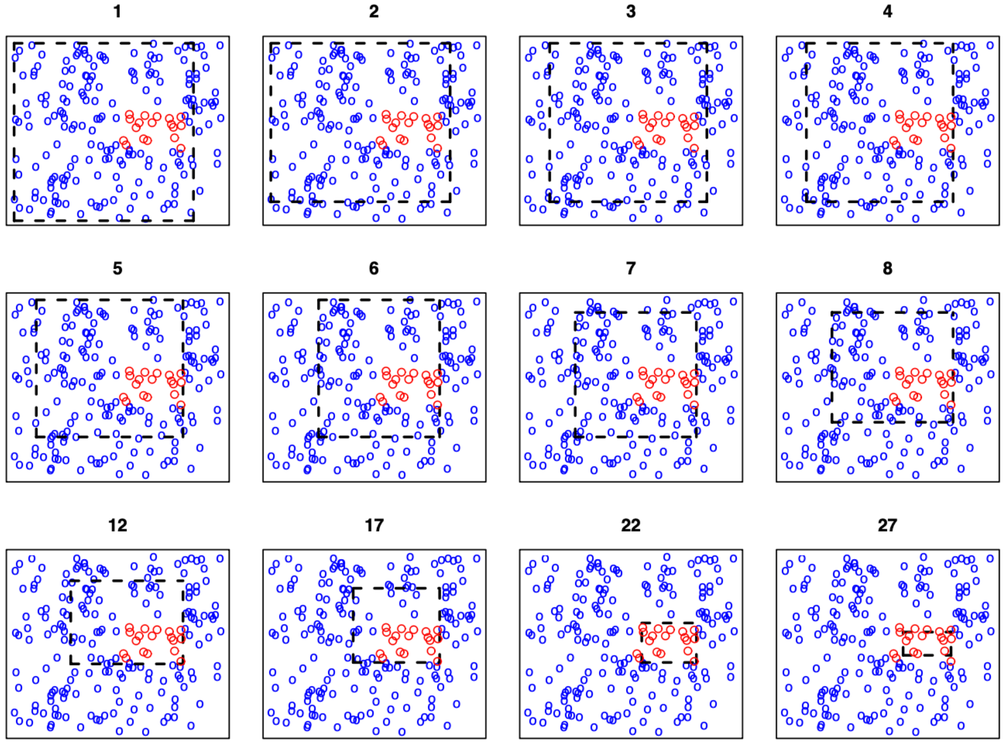


Perspective plot of the prediction surface

The patient rule induction method (PRIM)

- **Regression Tress:** partition the feature space into box-shaped regions, to try to make the response averages in each box as **different as possible**

- **PRIM:** Find the boxes in the feature space, but seeks boxes in which the **response average is high** (If minimization case, multiply -1 to response.)



Algorithm 9.3 *Patient Rule Induction Method.*

1. Start with all of the training data, and a maximal box containing all of the data.

2. Consider shrinking the box by compressing one face, so as to peel off the proportion α of observations having either the highest values of a predictor X_j, or the lowest. Choose the peeling that produces the highest response mean in the remaining box. (Typically $\alpha = 0.05$ or 0.10.)

3. Repeat step 2 until some minimal number of observations (say 10) remain in the box.

4. Expand the box along any face, as long as the resulting box mean increases.

5. Steps 1–4 give a sequence of boxes, with different numbers of observations in each box. Use cross-validation to choose a member of the sequence. Call the box B_1.

6. Remove the data in box B_1 from the dataset and repeat steps 2–5 to obtain a second box, and continue to get as many boxes as desired.

The patient rule induction method (PRIM)

- PRIM is designed for regression; a binary classification can be handled simply by coding it as 0 and 1. There is no simple way to deal with multi-classes simultaneously: one approach is to run PRIM as OVR(One vs. Rest).

- An advantage of PRIM over CART is its **patience**:

  For N observations, CART can only make $\log_2 N - 1$ splits(binary tree). If PRIM peels off a proportion α at each stage, it can perform approximately $-\log N/\log(1-\alpha)$ peeling steps. e.g. $N = 128$, and $\alpha = 0.1$, then $\log_2 N - 1 = 6$ while $-\log N/\log(1-\alpha) \approx 46$. This is an advantage in the greedy algorithm.

Multivariate Adaptive Regression Splines (MARS)

- MARS is an **adaptive** procedure for regression, and is well suited for **high-dimensional problems**.

- MARS takes the form of an **expansion in product spline basis functions**, where the number of basis functions as well as the parameters associated with each one (product degree and knot locations) are **automatically determined by the data**.

- The collection of the **piece wise linear** basis functions $\mathscr{C} = \left\{ (X_j - t)_+, (t - X_j)_+ \right\}_{t \in \{x_{1j},\ldots,x_{Nj}\}, j=1,\ldots,P}$

- The model has the form:

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X) \text{ where } h_m(X) = \prod_{k=1}^{K_m} B_k(X),\ B_k \in \mathscr{C},\ K_m \text{ is \# of splits that gave rise to } h_m(X)$$

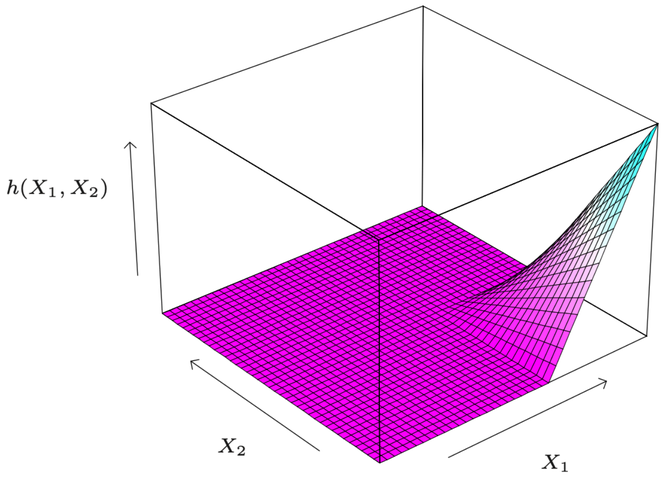
- Thus, $h_m(X) = \prod_{k=1}^{K_m} [s_{km} \cdot (X_{j(k,m)} - t_{km})]_+,\ s_{km} = \pm 1$ can model **interaction** between two or more predictors. (s_{km} determines whether to activate to the right or to the left.)

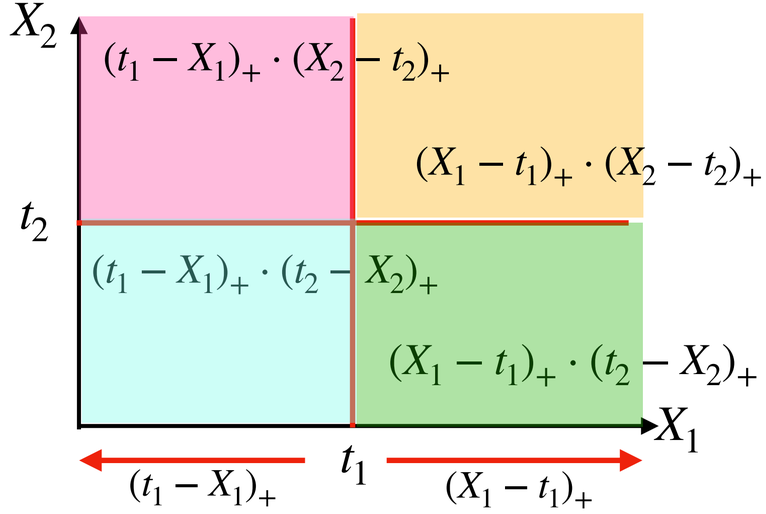
- β_m are estimated by standard linear regression.(Minimize the RSS)

Multivariate Adaptive Regression Splines (MARS)

(The forward pass)

1. For example, at the first stage we consider adding to the model a function of the form: $\beta_1(X_j - t)_+ + \beta_2(t - X_j)_+; t \in \{x_{ij}\}$
2. Suppose that the best choice is $\hat{\beta}_1(X_2 - x_{72})_+ + \hat{\beta}_2(x_{72} - X_2)_+$, Update $\mathscr{M} = \{1\} \cup \{(X_2 - x_{72})_+, (x_{72} - X_2)_+\}$
3. Then at the next stage we consider including a pair of products the form: $\hat{\beta}_3 h_m(X) \cdot (X_j - t)_+ + \hat{\beta}_4 h_m(X) \cdot (t - X_j)_+,\ t \in x_{ij}$
   - For $h_m(X)$ we have the choices: $h_m(X) = 1$ (**main effect** of X_j), $h_m(X) = (X_2 - x_{72})_+$ or $(x_{72} - X_2)_+$ (**interaction effect**)
4. Update $\mathscr{M} = \mathscr{M} \cup \{h_m(X)(t_m - X_{j(m)})_+, h_m(X)(X_{j(m)} - t_m)_+\}$. At the end, the model has the form: $f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$





Nonzero part of feature space in each product.

Multiway products are built up from products involving terms already in the model. -> A high-order interaction will likely only exist if some of its lower- order “footprints” exist as well (**hierarchical** strategy in MARS)

Multivariate Adaptive Regression Splines (MARS)

(The backward pass) The forward pass typically overfits the data, and so a backward deletion procedure is applied.

- Deleting the less effective term (smallest increase in RSS) at each stage until it finds the best model $\hat{f}_\lambda$ where λ is # of terms.
- How to choose the best model?
  - Generalized Cross Validation: $GCV(\lambda) = \frac{1}{(1 - M(\lambda)/N)^2} \|\mathbf{y} - \hat{f}_\lambda(\mathbf{x})\|^2$
  - $M(\lambda)$ is the effective number of parameters in the model; (# of terms in $\hat{f}_\lambda$) + (#of params used in selecting the knots).
  - There are 3 parameters for selecting a knot in a piecewise linear regression.
  - If r linearly independent basis in the model, and K knots were selected in the forward pass, $M(\lambda) = r + cK$ where $c = 3$.
  - We choose the model along the backward sequence that minimizes $GCV(\lambda)$.

Other issues in MARS

- **MARS for classification:**

  In binary classification, codes the output as 0/1 and treat the problem as regression. For multi-class $\{1,...,K\}$, codes $y_k = I(y = k)$, $k = 1,...,K$ and performs a multi-response MARS regression. However, potential **making problems** with this approach. -> "**Optimal scoring**" method is a generally superior approach.

  PolyMARS: In each forward stage, quadratic approximation to the multinomial log-likelihood to search for the next basis (**multiple logistic regression** framework)

- **Relationships of MARS to CART:** $\hat{f}(x) = \sum_m a_m B_m(x)$

  MARS **:** $B_m(X) = h_m(X) = \prod_{k=1}^{K_m} [s_{km} \cdot (X_{j(k,m)} - t_{km})]_+,\ s_{km} = \pm 1$ Multiplying a piecewise linear basis functions by a pair of reflected **piecewise linear basis.**

  CART: $B_m(X) = I(x \in R_m) = \prod_{k=1}^{K_m} I[s_{km} \cdot (X_{j(k,m)} - t_{km}) \geq 0],\ s_{km} = \pm 1$ Multiplying a step function by a pair of reflected **step functions** is equivalent to binary splitting a node at the step.

  This makes difficult it difficult for CART to model additive structures.

Hierarchical Mixtures of Experts (HME)

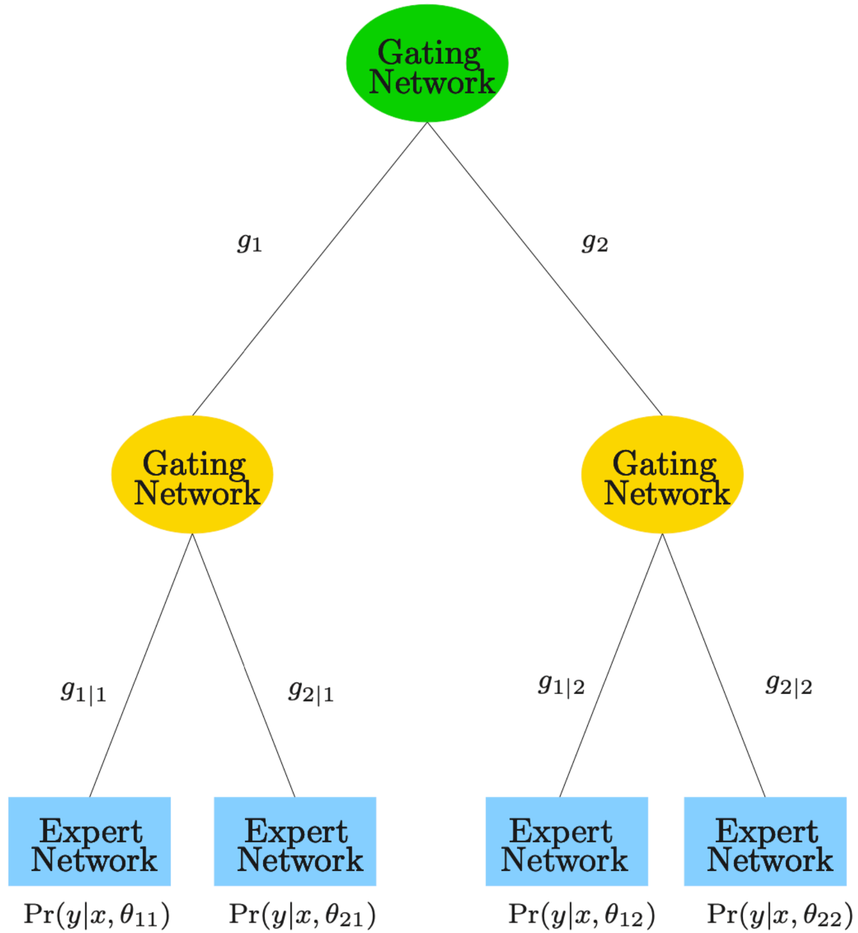
- HME is similar to Tree-bases method, but the main difference is that the splits are not hard decisions but rather **soft-probabilities** depending on its predictors by using **softmax** function.

- The idea is that each experts(terminal nodes) provide a prediction about the response, and these are combined together by the gating networks(non-terminal nodes).

- Difference between HMEs and CART:

  A linear model is fit in each terminal node, instead of constant as in CART.

  The splits can be multi-way, not just binary.



A 2-level and 2-way splits HME

Hierarchical Mixtures of Experts (HME)

- Consider the regression problem i.e. The response is either a continuous or binary-valued. Given the training set $\{(x_i, y_i)\}_{i=1}^N$,

- For K-way splits,

$$g_j(x, \gamma_j) = \frac{\exp(\gamma_j^T x)}{\sum_{k=1}^K \exp(\gamma_k^T x)}, j = 1,...,K \text{ at first stage which is softmax function}$$

$$g_{l|j}(x, \gamma_{jl}) = \frac{\exp(\gamma_{jl}^T x)}{\sum_{k=1}^K \exp(\gamma_{jk}^T x)}, l = 1,...,K \text{ at second stage for branch } j = 1,...,K$$

At each expert(terminal node), $Y \sim Pr(y \mid x, \theta_{jl})$

For countinuous case, model the Gaussian linear regression, $Y = \beta_{jl}^T x + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_{jl}^2),\ \theta_{jl} = (\beta_{jl}, \sigma_{jl})$

For binary case, model the logistic regression, $Pr(Y = 1 \mid x, \theta_{jl}) = \frac{1}{1 + \exp(-\theta_{jl}^T x)}$

- Unknown parameters: $\Psi = \{\gamma_j, \gamma_{jl}, \theta_{jl}\}$ -> Maximize the log-likelihood of the data: $\max_{\Psi} \sum_i \log Pr(y_i \mid x_i, \Psi)$

$$Pr(y \mid x, \Psi) = \sum_{j=1}^K g_j(x, \gamma_j) \sum_{l=1}^K g_{l|j}(x, \gamma_{jl}) Pr(y \mid x, \theta_{jl})$$ which is a mixture model i.e. hard to maximize -> The EM algorithm.

Hierarchical Mixtures of Experts (HME)

- Soft splits can capture situations where the transition from low to high response is gradual.
- The log-likelihood is a smooth function of the unknown weights and hence is amenable to numerical optimization.
- There are no methods for **finding a good tree topology** for the HME model, as there are in CART
- The emphasis in the research on HMEs has been on prediction rather than **interpretation of the final model**.