# Additive Models, Trees, and Related Methods

**Jingwen Zhang**

**Sun Yat-sen University, School of Mathematics**

## Contents

- Regression models providing prediction and classification rules.
- Data analytic tools for understanding the importance of different inputs.

But in real life, effects are often not linear.

**Generalized additive models**

In the regression setting, a generalized additive model has the form

$$\mathrm{E}(Y|X_1, X_2, \ldots, X_p) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p).$$

$X_1, X_2, \cdots, X_p$ represent predictors and $Y$ is the outcome; the sj's are unspecified smooth ("nonparametric") functions.

## Generalized additive models

Fitting such a model relies on the relationship

$$s_k(X_k) = E\left\{Y - \alpha - \sum_{j \neq k} s_j(X_j) | X\right\} \tag{12.2}$$

where $X_k$ is the kth component in $\mathbf{X}$.

For actual data, The average is taken over points whose kth coordinate is in a neighborhood of $x_k^*$.
A second problem—that none of the $s_j$ are actually known—is overcome by iteratively cycling through smoothing steps based on isolations like (12.2) updating $s_k$ using the best current guesses for all $s_j$ for $j \neq k$.

## Backfitting algorithm

Let $Y = (Y_1, Y_2, \cdots, Y_n)^T$, and for each $k$, let $\hat{s}_k^{(t)}$ denote the vector of estimated values of $s_k(x_{ik})$ at iteration $t$ for $i = 1, \cdots, n$. The $n$-vectors of estimated smooths at each observation are updated as follows:

1. Let $\hat{\alpha}$ be the $n$-vector $(\bar{Y}, \cdots, \bar{Y})^T$. Let $t = 0$, where $t$ indexes the iteration number.
2. A reasonable initial guess is to let $\hat{s}_k^{(t)} = (\hat{\beta}_k x_{1k}, \cdots, \hat{\beta}_k x_{nk})^T$ for $k = 1, \cdots, p$, where the $\hat{\beta}_k$ are the linear regression coefficients found when $Y$ is regressed on the predictors.
3. For $\mathrm{k} = 1, \cdots, p$ in turn, let

$$\hat{s}_k^{(t+1)} = smooth_k(r_k)$$

where

$$r_k = Y - \alpha - \sum_{j<k} \hat{s}_j^{(t+1)} - \sum_{j>k} \hat{s}_j^{(t)}$$

the smoothing technique used for the kth smooth may vary with k.

## Backfitting algorithm

4. Increment t and go to step 3.

The algorithm can be stopped when none of the $\hat{s}_k^{(t)}$ change very much—perhaps when

$$\sum_{k=1}^{p} \left(\hat{\mathbf{s}}_k^{(t+1)} - \hat{\mathbf{s}}_k^{(t)}\right)^{\mathrm{T}} \left(\hat{\mathbf{s}}_k^{(t+1)} - \hat{\mathbf{s}}_k^{(t)}\right) \Bigg/ \sum_{k=1}^{p} \left(\hat{\mathbf{s}}_k^{(t)}\right)^{\mathrm{T}} \hat{\mathbf{s}}_k^{(t)}$$

is very small.

Suppose only linear smoothers are used to fit an additive model, and let $S_k$ be the $n \times n$ smoothing matrix for the $k$th component smoother. Then the Then the backfitting algorithm solves the set of equations given by

$$\hat{s}_k = S_k\left(Y - \sum_{j \neq k} \hat{s}_j\right)$$

Writing this set of equations in matrix form yields

$$\begin{pmatrix} \mathbf{I} & \mathbf{S}_1 & \mathbf{S}_1 & \ldots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \mathbf{S}_2 & \ldots & \mathbf{S}_2 \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \mathbf{S}_p & \ldots & \mathbf{I} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{s}}_1 \\ \hat{\mathbf{s}}_2 \\ \vdots \\ \hat{\mathbf{s}}_p \end{pmatrix} = \begin{pmatrix} \mathbf{S}_1\mathbf{Y} \\ \mathbf{S}_2\mathbf{Y} \\ \vdots \\ \mathbf{S}_p\mathbf{Y} \end{pmatrix}, \qquad (12.5)$$

which is of the form $\boldsymbol{Az} = \boldsymbol{b}$ where $\boldsymbol{z} = (\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_p)^{\boldsymbol{T}} = \hat{s}$. Note that $\boldsymbol{b} = \boldsymbol{\Lambda Y}$ where $\boldsymbol{\Lambda}$ is a block-diagonal matrix with the individual $S_k$ matrices along the diagonal.

The backfitting estimating equations $\boldsymbol{A}\hat{s} = \boldsymbol{\Lambda Y}$ will not have a unique solution if there exists any $\gamma$ such that $\boldsymbol{A}\gamma = 0$. In this case, there are many solutions to $\boldsymbol{A}\hat{s} = \boldsymbol{\Lambda Y}$, and backfitting converges to one of them, depending on the starting values.

For two-class classification, We relate the mean of the binary response $\mu(X) = \Pr(Y = 1|X)$ to the predictors via a linear regression model and the logit link function:

$$\log\left(\frac{\mu(X)}{1-\mu(X)}\right) = \alpha + \beta_1 X_1 + \cdots + \beta_p X_p.$$

The additive logistic regression model replaces each linear term by a more general functional form

$$\log\left(\frac{\mu(X)}{1-\mu(X)}\right) = \alpha + f_1(X_1) + \cdots + f_p(X_p),$$

where again each $f_j$ is an unspecified smooth function. The additive logistic regression model is an example of a generalized additive model.

In general, the conditional mean $\mu(X)$ of a response $Y$ is related to an additive function of the predictors via a link function $g$:

$$g[\mu(X)] = \alpha + f_1(X_1) + \cdots + f_p(X_p).$$

Examples of classical link functions are the following:

- $g(\mu) = \mu$ is the identity link, used for linear and additive models for Gaussian response data.
- $g(\mu) = logit(\mu)$ as above for modeling binomial probabilities.
- $g(\mu) = \log(\mu)$ for log-linear or log-additive models for Poisson count data.

$g(\mu) = f(X) + g(Z, W)$ where $g$ is a nonparametric function in two features.

Additive models can replace linear models in a wide variety of settings, for example an additive decomposition of time series,

$$Y_t = S_t + T_t + \varepsilon_t$$

where $S_t$ is a seasonal component, $T_t$ is a trend and $\varepsilon_t$ is an error term.

## Example: Additive Logistic Regression

The most widely used model in medical research is the logistic model for binary data. In this model the outcome $Y$ can be coded as 0 or 1, with 1 indicating an event (like death or relapse of a disease) and 0 indicating no event.

We wish to model $\Pr(Y = 1|X)$, the probability of an event given values of the prognostic factors $X^T = (X_1, \ldots, X_p)$. The generalized additive logistic model has the form

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

**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.

## Example: Predicting Email Spam

We apply a generalized additive model to the spam data.

The data consists of information from 4601 email messages, in a study to screen email for “spam” (i.e., junk email). The response variable is binary, with values email or spam, and there are 57 predictors as described below:

- 48 quantitative predictors—the percentage of words in the email that match a given word. Examples include business, address, internet, free, and george. The idea was that these could be customized for individual users.
- The average length of uninterrupted sequences of capital letters: CAPAVE.
- The length of the longest uninterrupted sequence of capital letters: CAPMAX.
- The sum of the length of uninterrupted sequences of capital letters: CAPTOT.

We coded spam as 1 and email as 0.

A test set of size 1536 was randomly chosen, leaving 3065 observations in the training set.

Before fitting the GAM model, we log-transformed each variable (actually $\log(x + 0.1)$) .

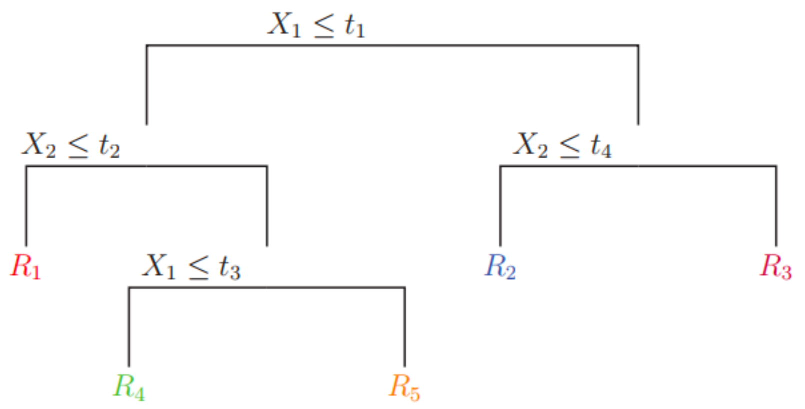
The test error rates are shown in Table; the overall error rate is 5.3%. By comparison, a linear logistic regression has a test error rate of 7.6%.

| True Class | Predicted Class | |
|---|---|---|
| | email (0) | spam (1) |
| email (0) | 58.3% | 2.5% |
| spam (1) | 3.0% | 36.3% |

It is more serious to classify a genuine email message as spam, since then a good email would be filtered out and would not reach the user. We can alter the balance between the class error rates by changing the losses. If we assign a loss $L_{01}$ for predicting a true class 0 as class 1, and $L_{10}$ for predicting a true class 1 as class 0, then the estimated Bayes rule predicts class 1 if its probability is greater than $L_{01} / (L_{01} + L_{10})$.

## Classification and regression trees (CART)

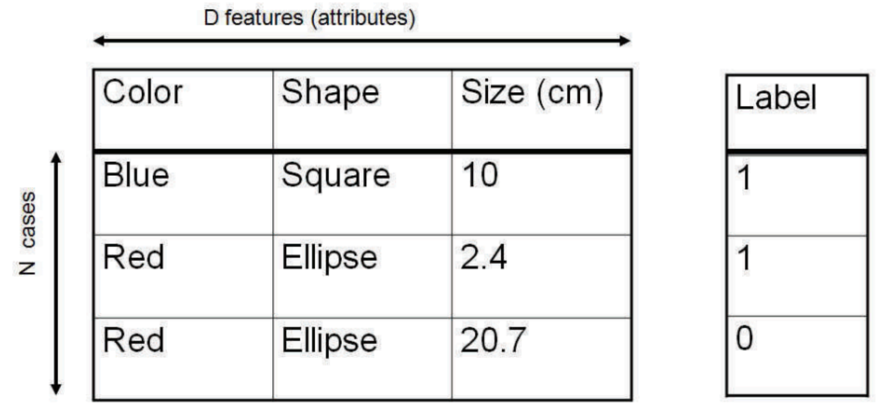
To explain the CART approach, consider the tree in Figure.



We can write the model in the following form

$$f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = \sum_{m=1}^{M} w_m \mathbb{I}(\mathbf{x} \in R_m)$$

A CART model is just a an adaptive basis-function model, where the basis functions define the regions, and the weights specify the response value in each region.

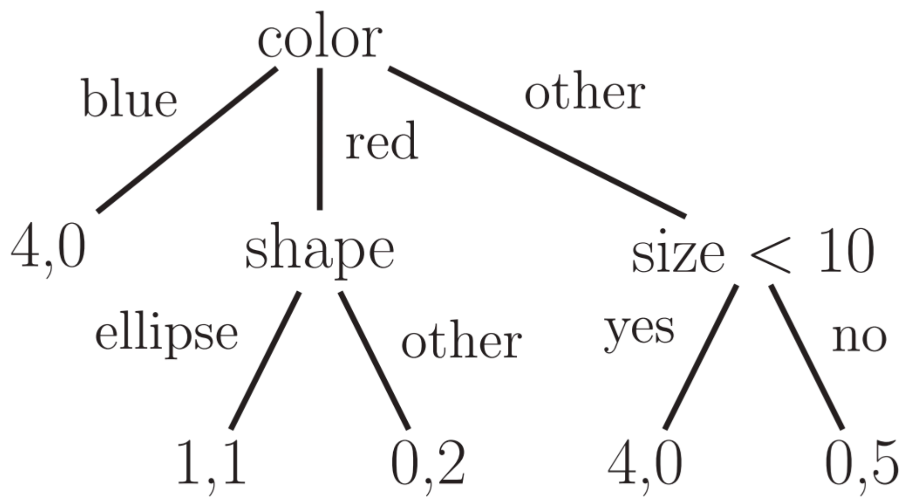


D features (attributes)

N cases

| Color | Shape | Size (cm) |
|---|---|---|
| Blue | Square | 10 |
| Red | Ellipse | 2.4 |
| Red | Ellipse | 20.7 |

| Label |
|---|
| 1 |
| 1 |
| 0 |



We first check the color of the object. If it is blue, we follow the left branch and end up in a leaf labeled "4,0", which means we have 4 positive examples and 0 negative examples which match this criterion. Hence we predict $p(y = 1|x) = 4/4$ if $x$ is blue.

## Growing a tree

The split function chooses the best feature, and the best value for that feature, as follows:

$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} \leq t\}) + \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} > t\})$$

The set of possible thresholds $T_j$ for feature $j$ can be obtained by sorting the unique values of $x_{ij}$. For example, if feature 1 has the values $\{4.5, -12, 72, -12\}$, then we set $T_1 = \{-12, 4.5, 72\}$.

The function that checks if a node is worth splitting can use several stopping heuristics, such as the following:

- is the reduction in cost too small？
- has the tree exceeded the maximum desired depth?
- is the distribution of the response in either $\mathcal{D}_L$ or $\mathcal{D}_R$ sufficiently homogeneous?
- is the number of examples in either $\mathcal{D}_L$ or $\mathcal{D}_R$ too small?

**Regression cost**

In the regression setting, we define the cost as follows:

$$\text{cost}(\mathcal{D}) = \sum_{i \in \mathcal{D}} (y_i - \overline{y})^2$$

where $\bar{y} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} y_i$ is the mean of the response variable in the specified set of data.

## Classification cost

In the classification setting, there are several ways to measure the quality of a split. First, we fit a multinoulli model to the data in the leaf by estimating the class-conditional probabilities as follows:

$$\hat{\pi}_{\mathrm{c}} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} I(y_i = c)$$

where $\mathcal{D}$ is the data in the leaf. Given this, there are several common error measures for evaluating a proposed partition:

- **Misclassification rate.** We define the most probable class label as $\hat{y}_{\mathrm{c}} = \mathrm{argmax}_{\mathrm{c}} \hat{\pi}_{\mathrm{c}}$. The corresponding error rate is then

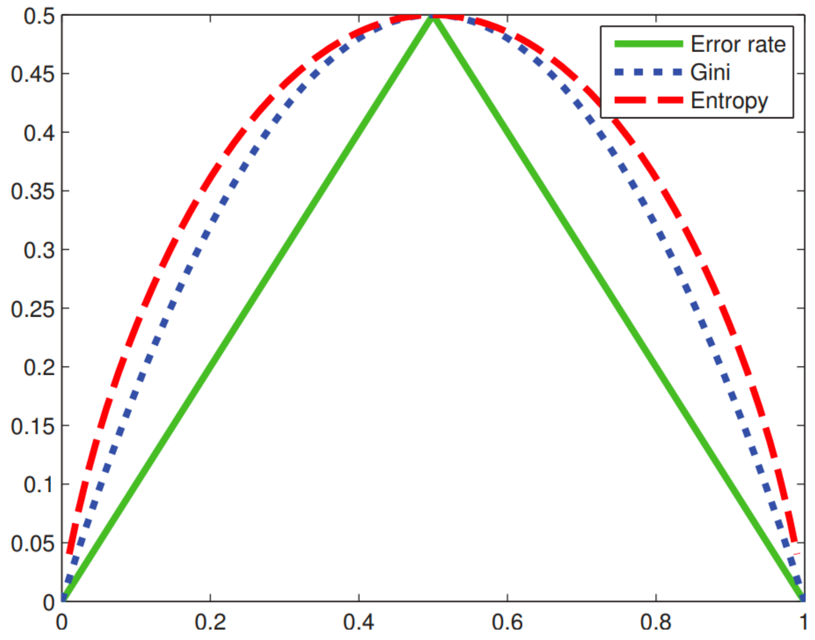
$$\frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} I(y_i \neq \hat{y}) = 1 - \hat{\pi}_{\hat{y}}$$

- **Entropy**

$$H(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi}_c log \hat{\pi}_c$$

- **Gini index**

$$\sum_{c=1}^{C} \hat{\pi}_c(1-\hat{\pi}_c) = \sum_c \hat{\pi}_c - \sum_c \hat{\pi}_c^2 = 1 - \sum_c \hat{\pi}_c^2$$



The horizontal axis corresponds to $p$, the probability of class 1. The entropy measure has been rescaled to pass through (0.5,0.5).

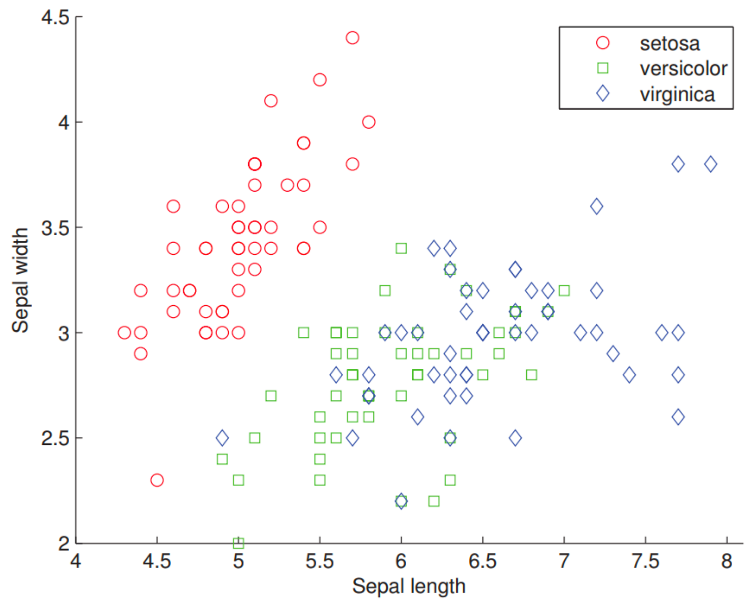
In the two-class case, where $p = \pi_m(1)$, the misclassification rate is $1 - \max(p, 1 - p)$.

We see that the cross-entropy and Gini measures are very similar, and are more sensitive to changes in class probability than is the misclassification rate.

It created the nodes (200,400) and (200,0). Both splits produce a misclassification rate of 0.25. However, the latter seems preferable, since one of the nodes is pure, i.e., it only contains one class. The cross-entropy and Gini measures will favor this latter choice.

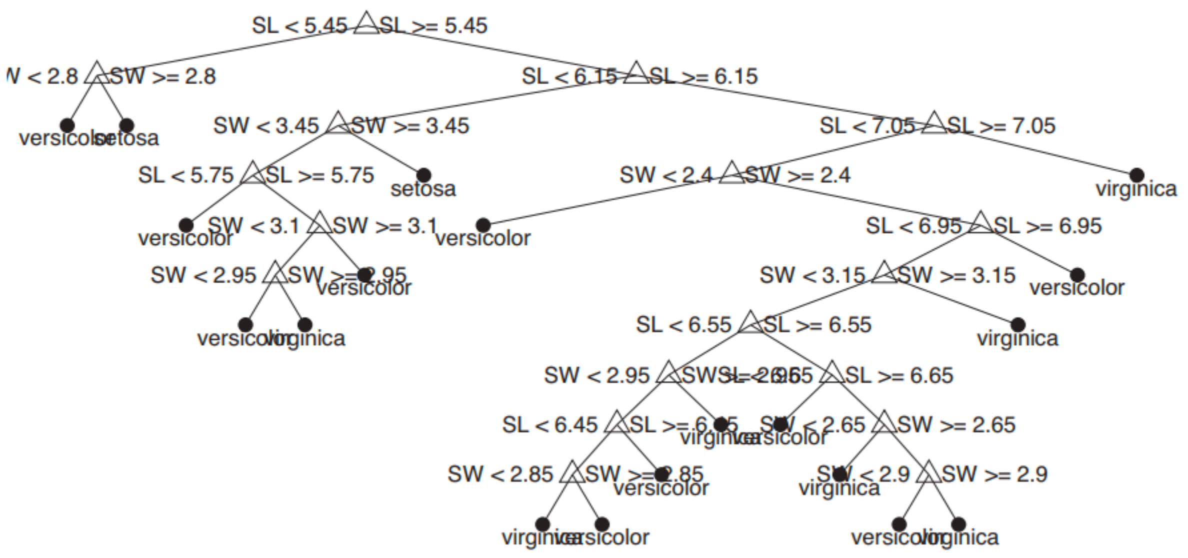
## Example

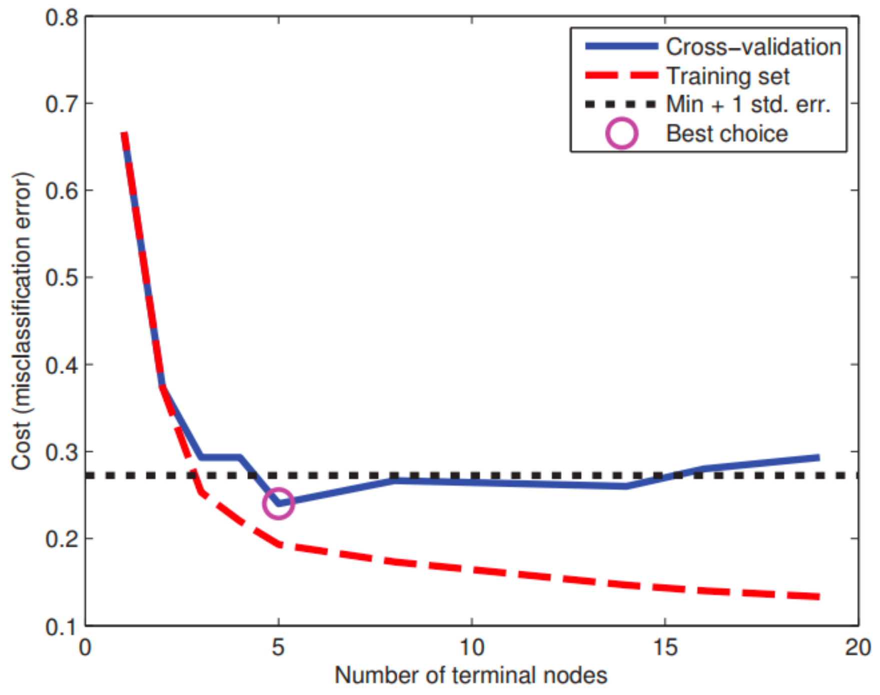
Iris dataset:



(a) Iris data. We only show the first two features, sepal length and sepal width, and ignore petal length and petal width.

In Figure (b), we show that the CV estimate of the error is much higher than the training set error, indicating overfitting. Below we discuss how to perform a tree-pruning stage to simplify the tree.





(b)

## Tree Pruning

A smaller tree with fewer splits (that is, fewer regions $R_1, \cdots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.

- grow a very large tree $T_0$.
- prune it back in order to obtain a subtree.
- select a subtree that leads to the lowest test error rate.

Given a subtree, we can estimate its test error using cross-validation or the validation set approach. However, estimating the cross-validation error for every possible subtree would be too cumbersome, since there is an extremely large number of possible subtrees. Instead, we need a way to select a small set of subtrees for consideration.

中山大學 SUN YAT-SEN UNIVERSITY

**Cost complexity pruning**

Rather than considering every possible subtree, we consider a sequence of trees indexed by a nonnegative tuning parameter $\alpha$. For each value of $\alpha$ there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree $T$ , $R_m$ is the rectangle (i.e. the subset of predictor space) corresponding to the $m$th terminal node, and $\hat{y}_{R_m}$ is the predicted response associated with $R_m$. The tuning parameter $\alpha$ controls a trade-off between the subtree's complexity and its fit to the training data.

We can select a value of $\alpha$ using a validation set or using cross-validation. We then return to the full data set and obtain the subtree corresponding to $\alpha$.

**Algorithm 8.1** *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of $\alpha$.

3. Use K-fold cross-validation to choose $\alpha$. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the $k$th fold.

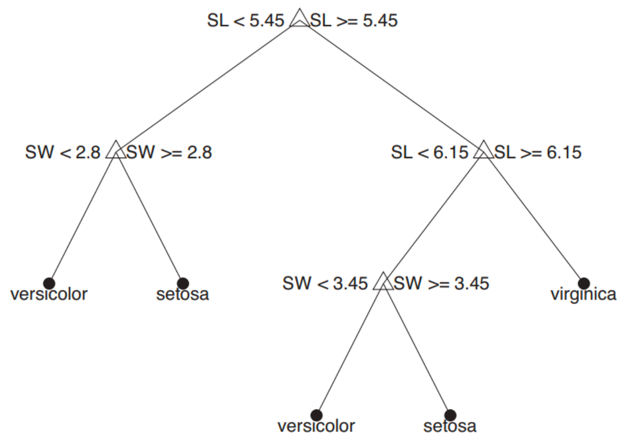
   (b) Evaluate the mean squared prediction error on the data in the left-out $k$th fold, as a function of $\alpha$.

   Average the results, and pick $\alpha$ to minimize the average error.

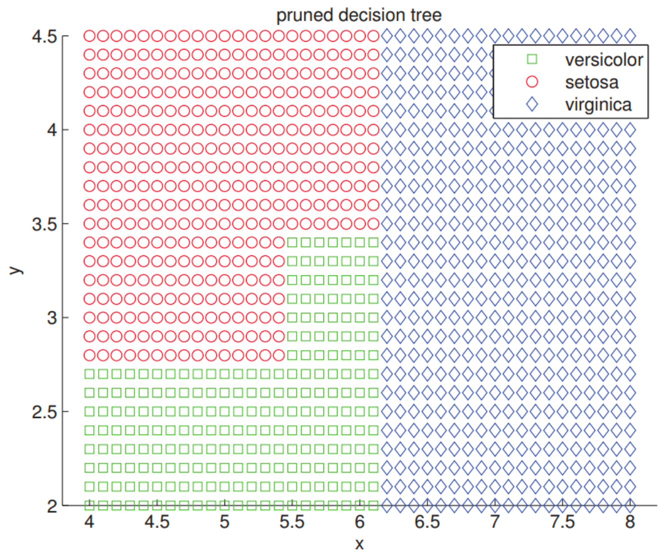
4. Return the subtree from Step 2 that corresponds to the chosen value of $\alpha$.

The standard approach is therefore to grow a “full” tree, and then to perform pruning.

The point with the minimum CV error corresponds to the simple tree in Figure (a).



(a)



(b)

Pruned decision tree for Iris data.

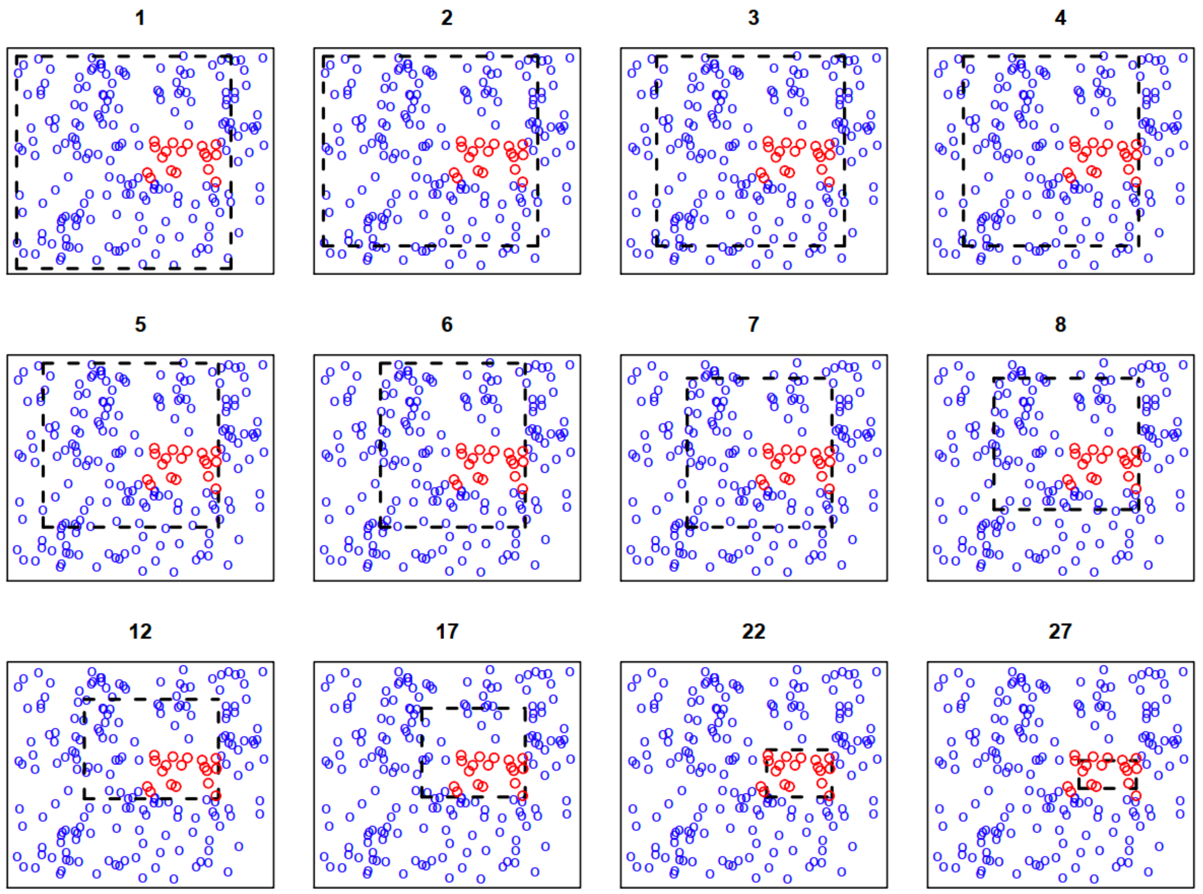
## Advantages and Disadvantages of Trees

- Trees are very easy to explain to people.
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches.
- Trees can be displayed graphically, and are easily interpreted even by  a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

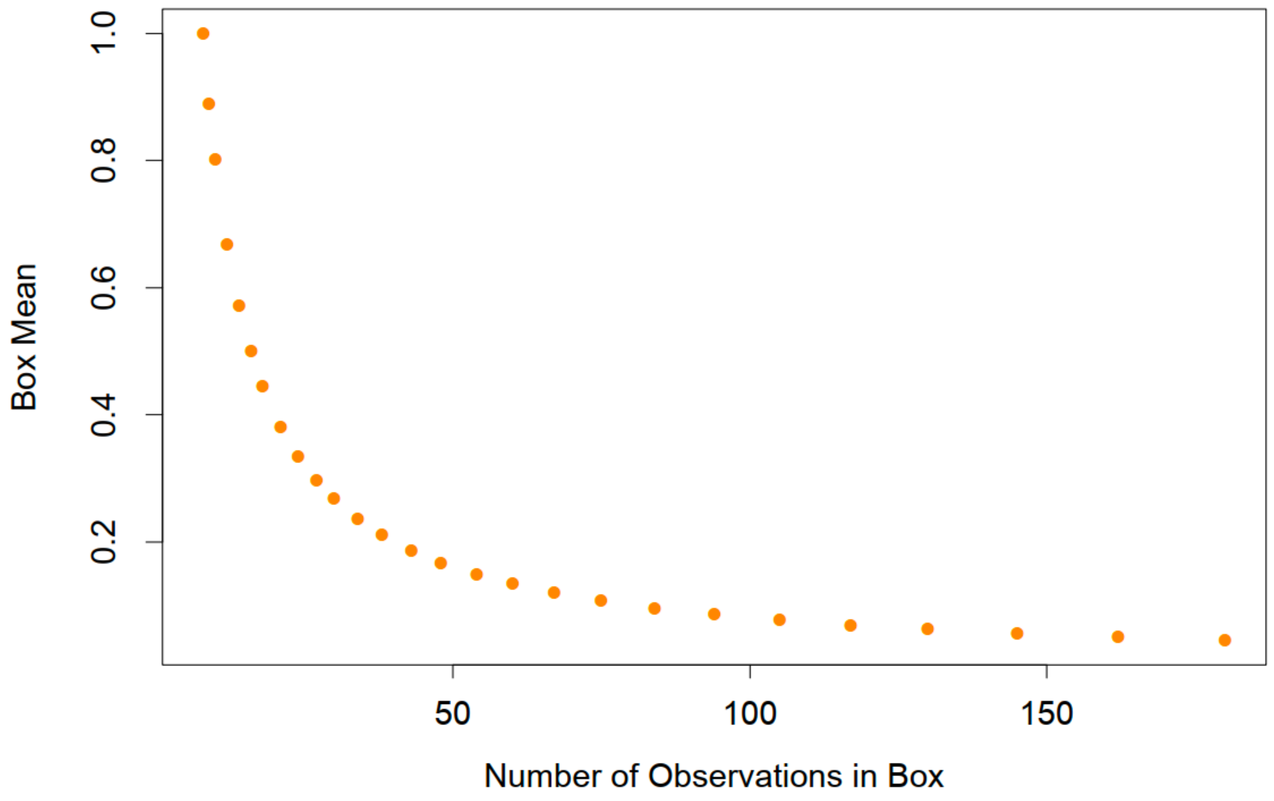
## PRIM: Bump Hunting

The patient rule induction method (PRIM) also finds boxes in the feature space, but seeks boxes in which the response average is high. Hence it looks for maxima in the target function, an exercise known as bump hunting.

The main box construction method in PRIM works from the top down, starting with a box containing all of the data. The box is compressed along one face by a small amount, and the observations then falling outside the box are peeled off. The face chosen for compression is the one resulting in the largest box mean, after the compression is performed. Then the process is repeated, stopping when the current box contains some minimum number of data points.



There are 200 data points uniformly distributed over the unit square. The color-coded plot indicates the response Y taking the value 1 (red) when $0.5 < X_1 < 0.8$ and $0.4 < X_2 < 0.6$. and zero (blue) otherwise. The panels shows the successive boxes found by the top-down peeling procedure, peeling off a proportion $\alpha = 0.1$ of the remaining data points at each stage.



This figure shows the mean of the response values in the box, as the box is compressed.
After the top-down sequence is computed, PRIM reverses the process, expanding along any edge, if such an expansion increases the box mean. This is called pasting. Since the top-down procedure is greedy at each step, such an expansion is often possible.

**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 $\alpha$ 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.

An advantage of PRIM over CART is its patience. Because of its binary splits, CART fragments the data quite quickly. Assuming splits of equal size, with N observations it can only make $\log 2(N) - 1$ splits before running out of data. If PRIM peels off a proportion α of training points at each stage, it can perform approximately $-\log(N)/\log(1-\alpha)$ peeling steps before running out of data. For example, if $N = 128$ and $\alpha = 0.10$, then $\log 2(N) - 1 = 6$ while $-\log(N)/\log(1-\alpha) \approx 46$. Taking into account that there must be an integer number of observations at each stage, PRIM in fact can peel only 29 times. In any case, the ability of PRIM to be more patient should help the top-down greedy algorithm find a better solution.

## Spam Example

We applied PRIM to the spam data, with the response coded as 1 for spam and 0 for email. The first two boxes found by PRIM are summarized below:

| Rule 1 | Global Mean | Box Mean | Box Support |
|---|---|---|---|
| Training | 0.3931 | 0.9607 | 0.1413 |
| Test | 0.3958 | 1.0000 | 0.1536 |

$$\text{Rule 1}\left\{\begin{array}{rcl}\texttt{ch!} & > & 0.029\\ \texttt{CAPAVE} & > & 2.331\\ \texttt{your} & > & 0.705\\ \texttt{1999} & < & 0.040\\ \texttt{CAPTOT} & > & 79.50\\ \texttt{edu} & < & 0.070\\ \texttt{re} & < & 0.535\\ \texttt{ch;} & < & 0.030\end{array}\right.$$

| Rule 2 | Remain Mean | Box Mean | Box Support |
|---|---|---|---|
| Training | 0.2998 | 0.9560 | 0.1043 |
| Test | 0.2862 | 0.9264 | 0.1061 |

$$\text{Rule 2}\left\{\begin{array}{rcl}\texttt{remove} & > & 0.010\\ \texttt{george} & < & 0.110\end{array}\right.$$

The box support is the proportion of observations falling in the box. The first box is purely spam, and contains about 15% of the test data. The second box contains 10.6% of the test observations, 92.6% of which are spam. Together the two boxes contain 26% of the data and are about 97% spam. The next few boxes (not shown) are quite small, containing only about 3% of the data.

## MARS: Multivariate Adaptive Regression Splines

MARS uses expansions in piecewise linear basis functions of the form $(x-t)_+$ and $(t-x)_+$.

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

We call the two functions a reflected pair in the discussion below. The idea is to form reflected pairs for each input $X_j$ with knots at each observed value $x_{ij}$ of that input. Therefore, the collection of basis functions is

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

The model-building strategy is like a forward stepwise linear regression, but instead of using the original inputs, we are allowed to use functions from the set $C$ and their products. Thus the model has the form

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

where each $h_m(X)$ is a function in $C$, or a product of two or more such functions.

- Start with only the constant function $h_0(X) = 1$, and all functions in the set $C$ are candidate functions.

- At each stage, consider as a new basis function pair all products of a function $h_m$ in the model set $M$ with one of the reflected pairs in $C$. We add to the model $M$ the term of the form

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

  that produces the largest decrease in training error.
- Then the winning products are added to the model and the process is continued until the model set M contains some preset maximum number of terms.

For example, at the first stage we consider adding to the model a function of the form $\beta_1(X_j - t)_+ + \beta_2(t - X_j)_+, t \in \{x_{ij}\}$. Suppose the best choice is $\beta_1(X_2 - x_{72})_+ + \beta_2(x_{72} - X_2)_+$ . Then this pair of basis functions is added to the set M, and at the next stage we consider including a pair of products the form.

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

where for $h_m$ we have the choices

$$\begin{aligned} h_0(X) &= 1, \\ h_1(X) &= (X_2 - x_{72})_+, \text{ or} \\ h_2(X) &= (x_{72} - X_2)_+. \end{aligned}$$

Applying a backward deletion procedure. The term whose removal causes the smallest increase in residual squared error is deleted from the model at each stage, producing an estimated best model $\hat{f}_\lambda$ of each size (number of terms) $\lambda$. One could use cross-validation to estimate the optimal value of λ. This criterion in MARS is defined as

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

The value $M(\lambda)$ is the effective number of parameters in the model: this accounts both for the number of terms in the models, plus the number of parameters used in selecting the optimal positions of the knots.

They are zero over part of their range. When they are multiplied together, the result is nonzero only over the small part of the feature space where both component functions are nonzero.

A useful option in the MARS procedure is to set an upper limit on the order of interaction. For example, one can set a limit of two, allowing pairwise products of piecewise linear functions, but not three- or higher way products. This can aid in the interpretation of the final model.

## Hierarchical Mixtures of Experts

A simple two-level HME model in shown in Figure 9.13. The terminal nodes are called experts, and the non-terminal nodes are called gating networks.

The idea is that each expert provides an opinion (prediction) about the response, and these are combined together by the gating networks.

As we will see, the model is formally a mixture model, and the two-level model in the figure can be extend to multiple levels, hence the name hierarchical mixtures of experts.

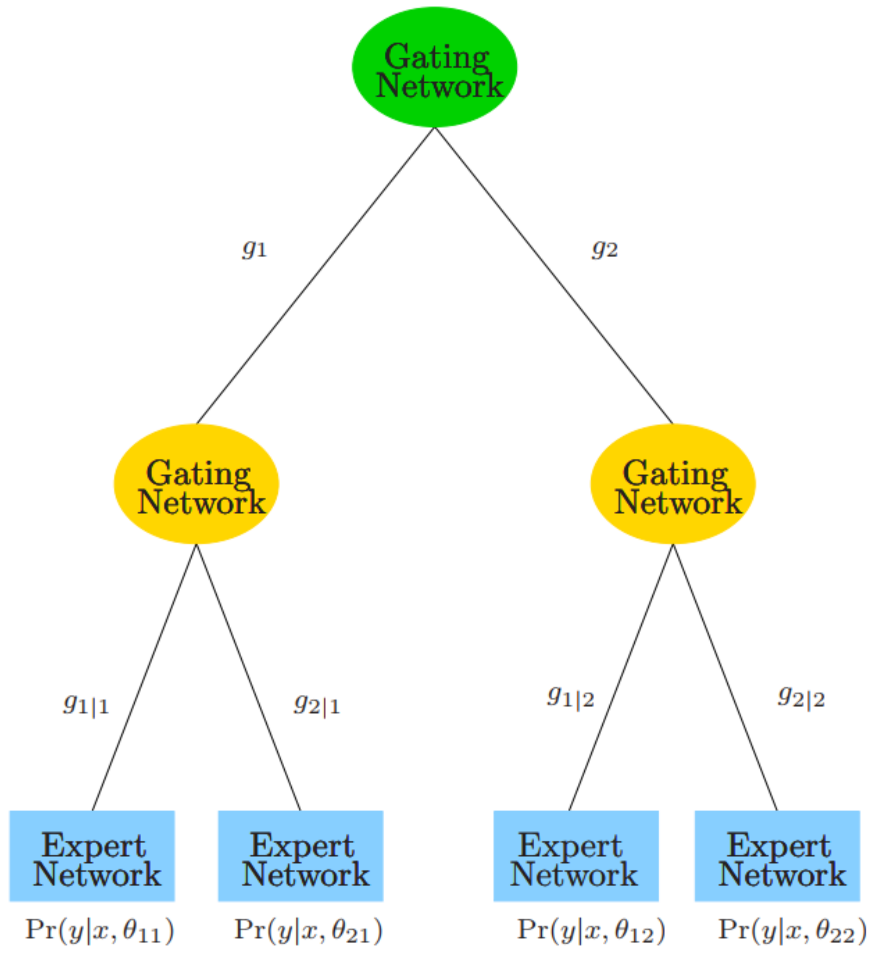


**FIGURE 9.13.** *A two-level hierarchical mixture of experts (HME) model.*

The data is $(x_i, y_i), i = 1, 2, \ldots, N$, with $y_i$ either a continuous or binary-valued response, and $x_i$ a vector-valued input.
Here is how an HME is defined. The top gating network has the output

$$g_j(x, \gamma_j) = \frac{e^{\gamma_j^T x}}{\sum_{k=1}^{K} e^{\gamma_k^T x}}, \; j = 1, 2, \ldots, K,$$

where each $\gamma_j$ is a vector of unknown parameters. This represents a soft K-way split ($K = 2$ in Figure 9.13.)
Each $g_j(x, \gamma_j)$ is the probability of assigning an observation with feature vector $x$ to the jth branch.

At the second level, the gating networks have a similar form:

$$g_{\ell|j}(x, \gamma_{j\ell}) = \frac{e^{\gamma_{j\ell}^T x}}{\sum_{k=1}^{K} e^{\gamma_{jk}^T x}}, \; \ell = 1, 2, \ldots, K.$$

This is the probability of assignment to the $\ell$th branch, given assignment to the $j$th branch at the level above.

At each expert (terminal node), we have a model for the response variable of the form

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

This differs according to the problem.

**Regression:** The Gaussian linear regression model is used, with $\theta_{j\ell} = (\beta_{j\ell}, \theta_{j\ell}^2)$

$$Y = \beta_{j\ell}^T x + \varepsilon \text{ and } \varepsilon \sim N(0, \sigma_{j\ell}^2).$$

**Classification:** The linear logistic regression model is used:

$$\Pr(Y = 1|x, \theta_{j\ell}) = \frac{1}{1 + e^{-\theta_{j\ell}^T x}}.$$

Denoting the collection of all parameters by $\Psi = \{\gamma_j, \gamma_{j\ell}, \theta_{j\ell}\}$, the total probability that $Y = y$ is

$$\Pr(y|x, \Psi) = \sum_{j=1}^{K} g_j(x, \gamma_j) \sum_{\ell=1}^{K} g_{\ell|j}(x, \gamma_{j\ell}) \Pr(y|x, \theta_{j\ell}).$$

*Thanks !*