Additive Models, Trees, and Related Methods

Thomas Lonon

Division of Financial Engineering
Stevens Institute of Technology

March 27, 2017

In the regression setting, a generalize additive model has the form:

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

where the $X_i's$ are the predictors, Y is the outcome, and the $f_i's$ are unspecified smooth functions.[2]

Unlike in the basis approach, we don't fit these using least squares. Instead we fit each predictor using a scatterplot smoother (e.g. cubic smoothing spline) for all p.

For example, for the two-class classification, we had the mean of the binary responses $\mu(X) = \mathbb{P}(Y = 1|X)$ and we related it to the predictor via linear regression and the logit 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 the linear terms with a more general function:

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

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

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

Classical Link Functions

- $g(\mu) = \mu$ is the identity link, used for linear and additive models for Gaussian response data
- $g(\mu) = \text{logit}(\mu)$ or $g(\mu) = \text{probit}(\mu)$, the probit link function is for modeling binomial probabilities and is the inverse of the Gaussian cumulative distribution function: $\text{probit}(\mu) = \Phi^{-1}(\mu)$
- $g(\mu) = \log(\mu)$ for log-linear or log-additive models for Poisson count data

[2]

Not all of the functions f_j need be linear (and in fact for qualitative data, we wouldn't want them to be linear).

- $g(\mu) = X^T\beta + \alpha_k + f(Z)$-a semiparametric model, where X is a vector of predictors to be modeled linearly, α_k the effect for the k^{th} level of a qualitative input V, and the effect of the predictor Z is modeled nonparametrically.
- $g(\mu) = f(X) + g_k(Z)$-again k indexes the levels of a qualtitative input V, and thus creates an interaction term $g(V, Z) = g_k(Z)$ for the effect of V and Z.
- $g(\mu) = f(X) + g(Z, W)$ where g is a nonparametric function in two features

[2]

We use the form:

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \varepsilon$$

where ε has mean zero.

Given observations x_i, y_i we utilize a penalized sum of squares:

$$PRSS(\alpha, f_1, f_2, \ldots, f_p) = \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j$$

where $\lambda_j \geq 0$ are tuning parameters

Algorithm 9.1: The Backfitting Algorithm for Additive Models

1. Initialize:

$$\hat{\alpha} = \frac{1}{N}\sum_{i=1}^{N} y_i, \hat{f}_j \equiv 0, \forall i, j$$

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

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

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

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

[2]

Other fitting methods can be accommodated by this algorithm by specifying the appropriate smoothing operators $\mathcal{S}_j$:

- other univariate regression smoothers such as local polynomial regression and kernel methods
- linear regression operators yielding polynomial fits, piecewise constant fits, parametric spline fits, series, and Fourier fits.
- more complicated operators such as surface smoothers for second or higher-order interactions or periodic smoothers for seasonal effects.[2]

If we only consider the operation of the smoother $\mathcal{S}_j$ at training points, it can be represented by an $N \times N$ operator matrix $\mathbf{S}_j$.

The approximate degrees of freedom for the j^{th} predictor could then be found as

$$df_j = \text{trace}[\mathbf{S}_j] - 1$$

In the generalized additive model, the weighted linear regression is simply replaced by a weighted backfitting algorithm.[2]

Algorithm 9.2: Local Scoring Algorithm for the Additive Logistic Regression Model

1. Compute starting values $\hat{\alpha} = \log(\frac{\bar{y}}{1-\bar{y}})$, 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 $\hat{p}_i = \frac{1}{1+\exp(-\hat{\eta}_i)}$. Iterate:
   - 2.1 Construct the working target variable
   $$z_i = \hat{\eta}_i + \frac{y_i - \hat{p}_i}{\hat{p}_i(1-\hat{p}_i)}$$
   - 2.2 Construct weights $w_i = \hat{p}_i(1-\hat{p}_i)$
   - 2.3 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

[2]

Tree-Based Methods

Tree based methods partition the space into a series of rectangles and then fit a simple model on each one

On the next slide, we see a partition split into five regions. This results in the regression model predicting Y with a constant c_m in region R_m which can be expressed as:

$$\hat{f}(X) = \sum_{m=1}^{5} c_m \mathbb{I}_{\{(X_1, X_2) \in R_m\}}$$

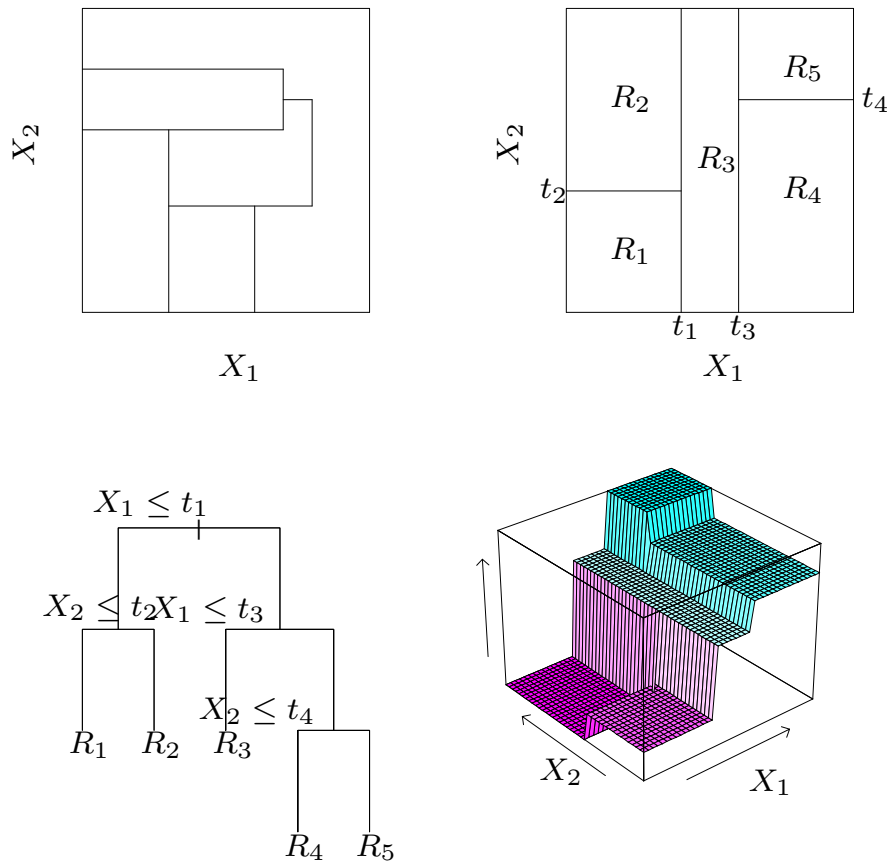

FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

Regression Trees

For data that consists of p inputs and a response for each of N observations: $(x_i, y_i), i = 1, 2, \ldots, N$ with $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$. We need an algorithm to decide on the splitting values and split points.

If we partition into M regions we have:

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{I}_{\{x \in R_m\}}$$

If we use as our criterion the minimization of the sum of squares, we can see that the best $\hat{c}_m$ is going to be the average in the region R_m

To decide where to create the partitions, we use a greedy algorithm. Define the half planes:

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } R_2(j,s) = \{X|X_j > s\}$$

We then need j and s that solve:

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

Cost-Complexity Pruning

Define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0. Index terminal nodes m representing region R_m. Let $|T|$ denote the number of terminal nodes in T.

$$N_m = \#\{x_i \in R_m\}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

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

we define the cost complexity criterion as

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

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 α
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:
   - 3.1 Repeats Steps 1 and 2 on all but the k^{th} fold of the training data
   - 3.2 Evaluate the mean squared prediction error on the data in the left-out k^{th} fold, as a function of α.
     Average the results for each value of α, and pick α to minimize the average error
4. Return the subtree from Step 2 that corresponds to the chosen value of α

[1]

Classification Trees

If the target is taking values $1, 2, \ldots, K$, the only changes to the tree algorithm pertain to the splitting nodes and the pruning.

The previous node impurity measure Q_m, found with a squared error is no longer suitable. Instead define:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{I}_{\{y_i = k\}}$$

which is the proportion of observation k in node m. We then assign to node m:

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

Measures of Node Impurity

MisClassification error:

$$\frac{1}{N_m} \sum_{i \in R_m} \mathbb{I}_{\{y_i \neq k(m)\}} = 1 - \hat{p}_{mk(m)}$$

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:

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

- Categorical Predictors
- The Loss Matrix
- Missing Predictor Values
- Why Binary Splits?
- Other Tree-Building Procedures
- Linear Combination Splits
- Instability of Trees
- Lack of Smoothness
- Difficulty in Capturing Additive Structure

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 $\alpha = .05$ or .10)
3. Repeat step 2 until some minimal number of observation (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.

[2]

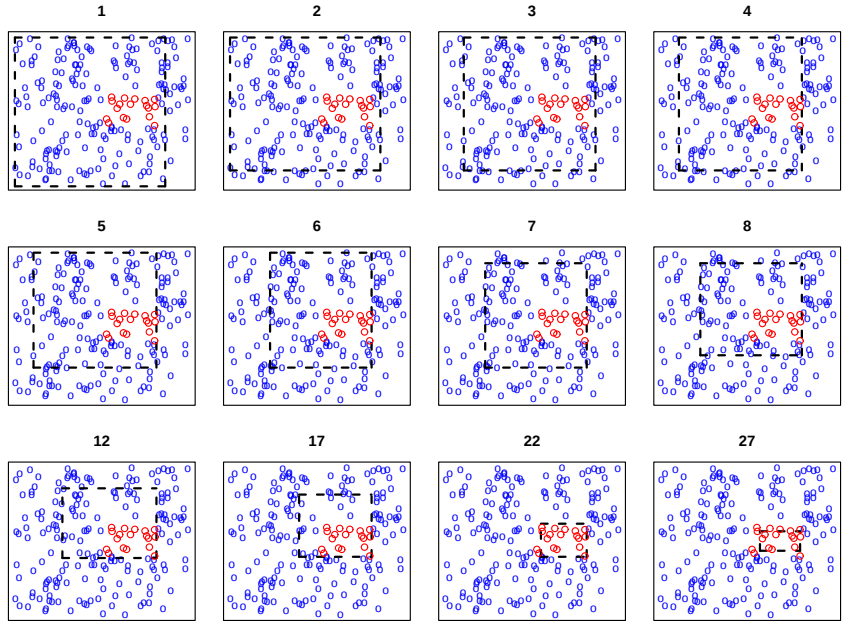

FIGURE 9.7. *Illustration of PRIM algorithm. There are two classes, indicated by the blue (class* 0*) and red (class* 1*) points. The procedure starts with a rectangle (broken black lines) surrounding all of the data, and then peels away points along one edge by a prespecified amount in order to maximize the mean of the points remaining in the box. Starting at the top left panel, the sequence of peelings is shown, until a pure red region is isolated in the bottom right panel. The iteration number is indicated at the top of each panel.*

Multivariate Adaptive Regression Splines

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

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

Each function is piecewise linear, with a knot at value t. These two functions are called a *reflected pair*.

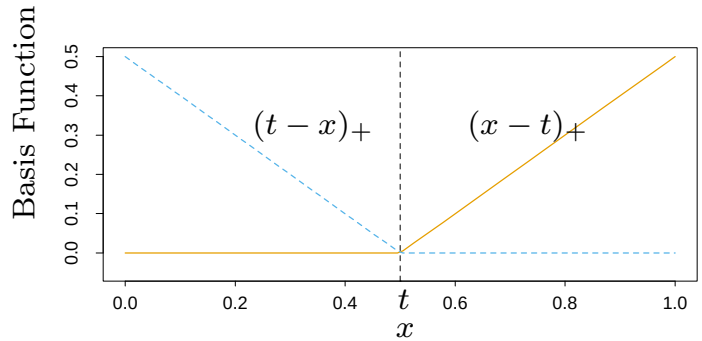

FIGURE 9.9. *The basis functions* $(x-t)_+$ *(solid orange) and* $(t-x)_+$ *(broken blue) used by MARS.*

The idea is to form reflected pairs for each X_j with knots at each observed value x_{ij} of that input. The collection of basis function is then:

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

You then perform forward stepwise linear regression which will have the form:

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

where each $h_m(x)$ is either a function of $\mathcal{C}$ or a product of two or more such functions.

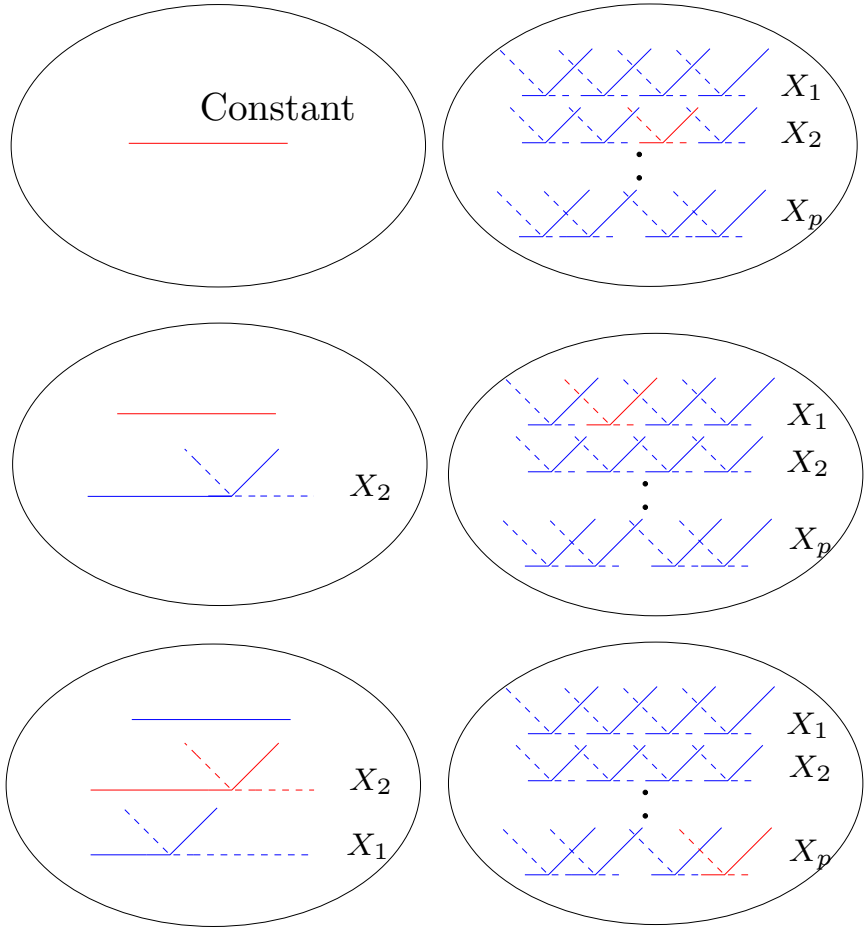

FIGURE 9.10. *Schematic of the MARS forward model-building procedure. On the left are the basis functions currently in the model: initially, this is the constant function* $h(X) = 1$. *On the right are all candidate basis functions to be considered in building the model. These are pairs of piecewise linear basis functions as in Figure 9.9, with knots* t *at all unique observed values* x_{ij} *of each predictor* X_j. *At each stage we consider*

Moving forward through the model in this way, we end up overfitting the data. So a backward deletion procedure is applied. In order to determine the optimum value of λ (the number of terms) we use generalized cross-validation.

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

where $M(\lambda)$ is the effective number of parameters in the model (both the number of terms and the number of parameters in choosing the knots)

Hierarchical Mixture of Experts

Variant of tree-based methods, where the tree splits are not hard decisions, but soft probabilistic ones.

In this approach, the terminal nodes are referred to as *experts* and the non-terminal nodes are referred to as *gating networks*.

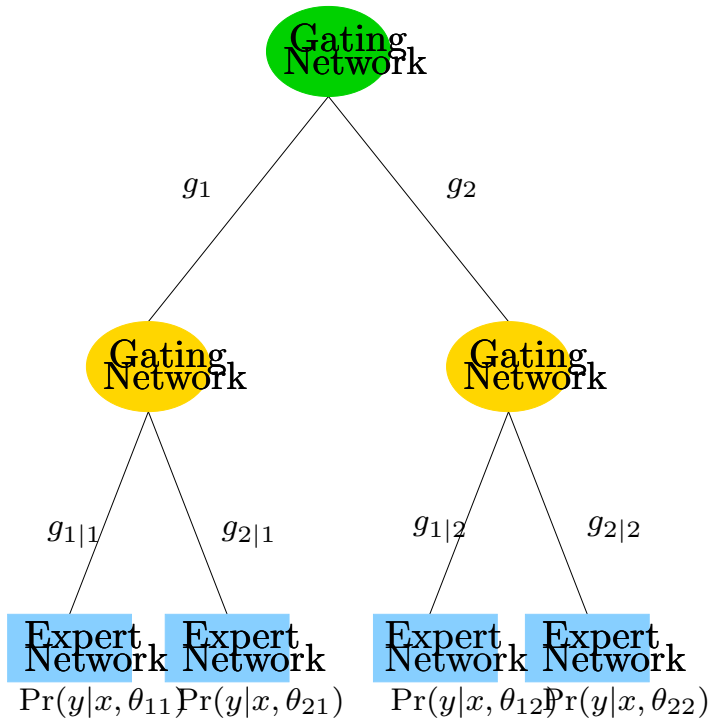

FIGURE 9.13. *A two-level hierarchical mixture of experts (HME) model.*

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 γ_j is a vector of unknown parameters.

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

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

At each expert (terminal node) we have a model for the response as

$$Y \sim \mathbb{P}(y|x, \theta_{j\ell})$$

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

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

Classification: The linear logistic regression model is used:

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

The parameters $\gamma_j, \gamma_{j\ell}, \theta_{j\ell}$ are then estimated using the EM approach.

[1] Trevor Hastie Gareth James, Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Number v. 6. Springer, 2013.

[2] Robert Tibshirani Trevor Hastie and Jerome Friedman. *The Elements of Stastical Learning: Data Mining, Inference, and Prediction*. Number v.2 in Springer Series in Statistics. Springer, 2009.