Statistics 8330: Data Analysis III
MARS: Multivariate Adaptive Regression Splines and Bagging

Suggested Reading: James, Witten, Hastie, and Tibshirani (JWHT), 2013, Chapter 8.2.1

Supplemental Reading: Hastie, Tibshirani, and Friedman (HTF), 2009, Chapter 9.4

Christopher K. Wikle

University of Missouri
Department of Statistics

MARS

The multivariate adaptive regression splines (MARS) procedure is, as the name suggests, an adaptive procedure for regression that works well for settings where there are a large number of input variables. It can be viewed as either a generalization of stepwise linear regression or, a modification of the CART procedure that improves its performance in the regression setting.

MARS uses expansions in piecewise linear basis functions, similar to what we saw with regression splines previously. In particular, we consider basis functions of the form (note, the "+" means "positive part" with a knot at t):

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

and

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

MARS (cont.)

Consider the example functions $(x - 0.5)_+$ and $(0.5 - x)_+$ from HTF (2009):

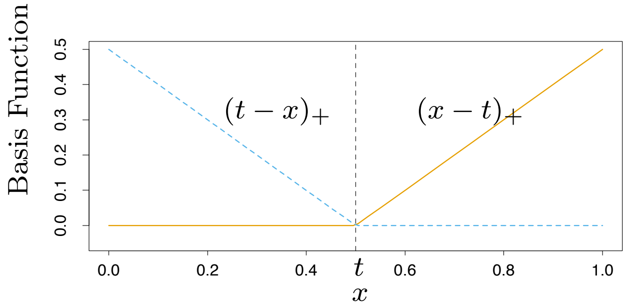

FIGURE 9.9. *The basis functions $(x - t)_+$ (solid orange) and $(t - x)_+$ (broken blue) used by MARS.*

MARS (cont.)

So, these are linear splines and the pair is called a *reflected pair*. The point is that we form such reflected pairs for each input X_j with knots at each observed value x_{ij} of that input variable. So, the collection of basis functions is

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

Thus, there can be as many as $2np$ basis functions (if all of the input values are distinct), which can be quite large.

We then consider a model of the form

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

where each $h_m(X)$ is a function in $\mathcal{C}$, or a product of two or more such functions. The model-building strategy is then like forward stepwise linear regression, but instead of using the original inputs, we are using these functions of the inputs.

MARS (cont.)

CKW

For a given choice of h_m, one estimates the β_m by standard least squares (i.e., minimizing the residual sum-of-squares). The challenge then is in the construction of the functions $h_m(x)$.

First, we choose the constant function $h_0(X) = 1$ and all of the functions in $\mathcal{C}$ are candidates. At each stage we consider as a new basis function pair all products of a function h_m in the existing model set $\mathcal{M}$ with one of the reflected pairs in $\mathcal{C}$, i.e., the

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

that produces the largest decrease in training error. These are then added to the model set $\mathcal{M}$.

MARS (cont.)

As an example, at the first stage consider adding to the model a function of the form $\beta_1(X_j - t)_+ + \beta_2(t - X_j)_+ : t \in \{x_{ij}\}$ (note that multiplication by the constant function produces the function itself).

Let's assume that for a particular data set the best choice is $\hat{\beta}_1(X_2 - x_{72})_+ + \hat{\beta}_2(x_{72} - X_2)_+$. We then add this to the set $\mathcal{M}$.

MARS (cont.)

At the next stage, we consider adding a pair of products of 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

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

For example, the third choice produces functions such as $(X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$, as illustrated below (HTF 2009).

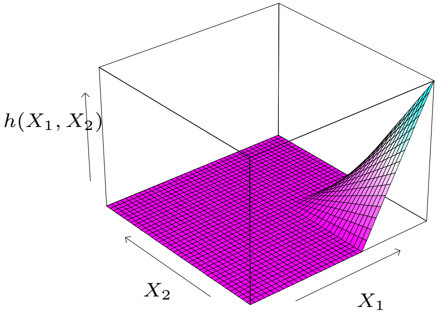

FIGURE 9.11. *The function $h(X_1, X_2) = (X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$, resulting from multiplication of two piecewise linear MARS basis functions.*

MARS (cont.)

CKW

Consider a visual example of MARS model building (HTF 2009):

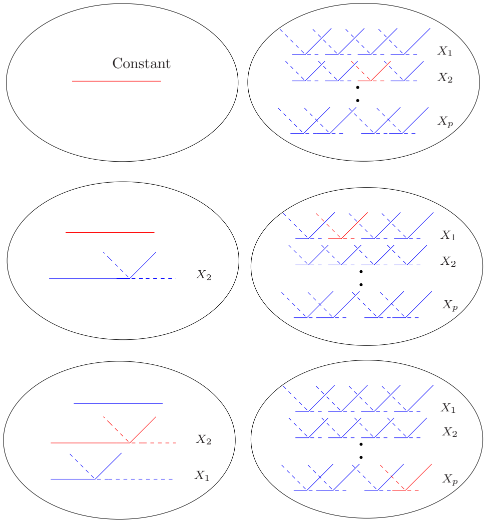

FIGURE 9.10. *Schematic of the MARS forward model-building procedure. On the left are the basis functions currently in the model: initially, this is the constant function $h(X) = 1$. On the right are all candidate basis functions to be considered in building the model. These are pairs of piecewise linear 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 all products of a candidate pair with a basis function in the model. The product that decreases the residual error the most is added into the current model. Above we illustrate the first three steps of the procedure, with the selected functions shown in red.*

MARS (cont.)

This procedure continues until some pre-specified number of terms is included in the model. This is generally a very large model that typically overfits the data. For this reason, a backward deletion procedure is usually applied. That is, the term that leads to the smallest increase in residual squared error is deleted at each stage. Thus, we have an estimated "best" model $\hat{f}_\lambda$ of each size (number of terms) λ.

We would like to use cross-validation to choose the best model but it is computationally problematic. Instead, we use *generalized cross-validation*, which is defined as

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

where $M(\lambda)$ is the effective number of parameters in the model (i.e., accounting for number of terms in the models plus the number of parameters used in selecting the optimal position of the knots).

MARS (cont.)

Typically, one uses $M(\lambda) = r + cK$ where r is the number of linearly independent basis functions in the model and K is the number of knots selected in the forward process, and $c = 3$ (typically) or $c = 2$ if the model is restricted to be additive.

Note that a major advantage of these piecewise linear basis functions is that they are "local" (i.e., non-zero in only a small portion of the feature space, even when multiplied together), which facilitates model fitting in high dimensions. There is also a significant computational advantage in that it only takes on the order of n operations to try every knot in each step of the process (for each of the $j = 1, \ldots, p$ variables).

Note that it is also beneficial that if higher order products are in the model, it is necessarily the case that the lower order components are also in the model. This need not be the case in reality, but is typically good modeling practice.

MARS (cont.)

CKW

HTF (2009) - GCV Example:

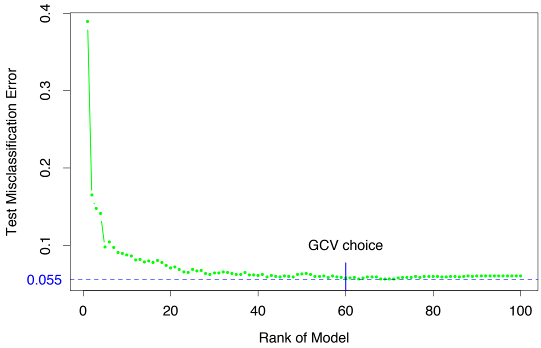

FIGURE 9.12. *Spam data: test error misclassification rate for the MARS procedure, as a function of the rank (number of independent basis functions) in the model.*

MARS (cont.)

Note that there is a limitation in the construction of the basis vectors in that each input can occur at most one time in a product. Thus, higher order powers of an input are not allowed. Such powers can produce unstable fits, particularly at the boundaries.

Sometimes, one restricts the products to be of a certain order (e.g., only second-degree or third-degree).

Note that MARS can be extended to classification settings (e.g., setting the response variable to 0/1) but this is sort of artificial. There have been specific MARS-like algorithms designed for classification.

MARS (cont.)

Although not necessarily obvious, there is a close relationship between the MARS and CART procedures. In particular, to mimic CART, the MARS procedure makes the following changes:

- Replace the piecewise linear basis functions by step functions; e.g., $I(x - t > 0)$ and $I(x - t \leq 0)$.
- When a model term is involved in a multiplication by a candidate term, it gets replaced by the interaction, and hence is not available for further interactions.

It can be shown that with these changes the MARS forward procedure is the same as the CART tree-growing algorithm. That is, multiplying a step function by a pair of reflected step functions is equivalent to splitting a node at the step. The second restriction says that a node may not be split more than once (leading to the binary tree structure). This illustrates that MARS is more powerful in that it does allow for additive structures because it is not limited to the binary tree structure.

Bagging

CKW

We can use the notions of bootstrapping to improve statistical learning methods such as the tree methods described previously. The decision trees (and MARS models) typically suffer from *high variance*. That is, if we split the training data into two samples at random and fit both halves, the results can be quite different. If the procedure has *low variance* it will yield similar results if applied repeatedly to distinct data sets (e.g., as in linear regression when n is larger relative to p).

Bootstrap aggregation or **bagging** is a general-purpose procedure for reducing variance in a statistical learning procedure.

The basic idea is that *averaging reduces variance*, which is something we have seen over and over again in statistics. So, a natural approach to reduce variation is to take many training sets and build a separate prediction model using each of those training sets, and average them together.

Bagging (cont.)

Specifically, say we calculate $\hat{f}^1(x)$, $\hat{f}^2(x)$, …, $\hat{f}^B(x)$, using B separate training sets, and average them

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

In practice, we do not have access to multiple training sets so use bootstrap samples from our (single) training set. That is, we generate B bootstrap samples (sampling randomly with replacement). In this case, $\hat{f}_{bag}(x)$ is the result of averaging the fits from these B bootstrap samples. This procedure is called *bagging*.

When using bagging for regression trees we typically construct B regression trees and average the resulting predictions, but the trees are grown large with no pruning. So, each tree has high variance but low bias and averaging then reduces the variance. Often, 100 samples is sufficient to reduce the variance appreciably.

Bagging (cont.)

Bagging can also be used for classification. In that case, the most popular approach is to simply record the class predicted by each of the B trees for a given test observation and then choose the category that occurs the most often.

One can also easily estimate the error associated with a bagged model without having to perform cross-validation. The observations in the not used in the bootstrap samples that fit the bagged tree are called the *out-of-bag (OOB)* observations. We can predict the response for the ith observation using each of the trees in which the observation was OOB. We can then average these to get a single prediction and, perhaps more importantly, we can then get the OOB MSE (or classification error).

This OOB error is a valid estimate of the test error for the bagged model because the response for each observation is predicted using only the trees that were fit without using that observation.

Bagging and Interpretability

CKW

Although bagging typically improves the utility of a tree-based model for prediction (relative to a single tree), it comes at the cost of reduced interpretability. That is, when we average over many different trees, the variable splits (or basis functions included) can change substantially, preventing interpretation. Thus, it isn't necessarily clear which variables are important.

One way around this is to keep tract of the total amount that the residual sum of squares is decreased due to splits over a given predictor averaged over all B trees (in the case of regression), or add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees (in the case of classification). This gives some measure of variable importance in the tree.