MARS Basics

Multivariate Adaptive Regression Splines (MARS) is regression software developed by Jerry Friedman in the early 1990s. It uses continuous piece-wise linear splines to construct a regression model in a sequential manner. Variable selection, knot selection, and interaction inclusion are all done adaptively. Because variable transformation is done automatically, via the creation of splines, good models can be made without tremendous skill and effort.

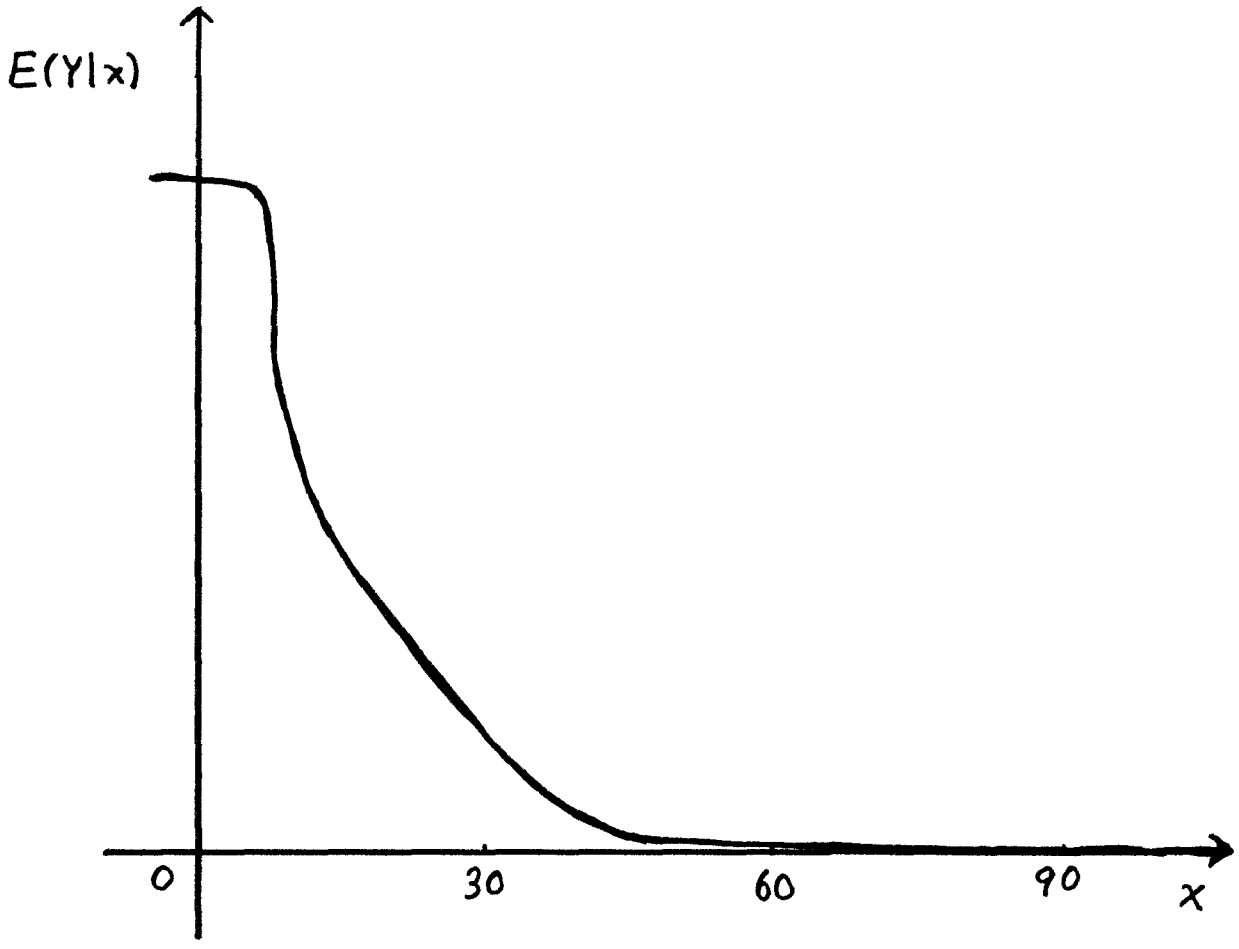
MARS Basis Functions

With just a single explanatory variable, x, one can fit a cont. piecewise linear spline regression model by preselecting knots, say at 30, 60, and 90, and then using OLS to estimate the coefficients for the basis functions

$$h_0(x) = 1,$$
$$h_1(x) = x,$$
$$h_2(x) = (x-30)_+,$$
$$h_3(x) = (x-60)_+,$$
& $$h_4(x) = (x-90)_+.$$

But if the true relationship between $E(Y|x)$ and x is as shown in the

sketch below, the fitted linear spline model cannot be expected to be very good, because the preselected knots are not favorable for providing a good fit. Having knots at 8, 12, and 44



should result in a much better fit.

Problems are encountered if one attempts to use the data to determine near optimal knot placement. While one might be able to do a decent job using simple graphics in the case of one explanatory variable, with two explanatory variables nonlinear relationships, interaction of the variables, and unfavorable data distributions (e.g., the (x_1, x_2) points not close to being in a grid pattern or uniformly distributed) all hinder attempts to use conditioning and graphics to select good knots, and beyond two variables the challenge can

increase severely. Additionally, there is the problem of overfitting, due to using too many knots per variable (and thus leading to too many paramters to estimate), when the sample size is relatively small.

To avoid the difficulties associated with trying to carefully place knots, one could opt to represent each explanatory variable in the model by a natural cubic spline having 3 to 5 knots based on empirical dist'n percentiles. But this can result in too many coefficients

to estimate, especially if one creates interaction terms using the splines. (Adding tensor product basis functions for each pair of explanatory variables, to account for two-way interactions, greatly increases the number of unknowns to be estimated.)

To combat the previously referred to problems, although MARS considers a lot of basis functions, to allow for great flexibility where needed, it carefully selects variables, nonlinearities, and interactions as it first constructs a

possibly quite complex model which should overfit the data, then prunes away weaker terms to create a sequence of models with declining complexity, and then finally uses cross-validation or a test sample to select the model from the sequence which (hopefully) has the proper degree of complexity. Overall, the process allows for great flexibility in fitting a sufficiently complex model which should avoid overfitting and give relatively accurate predictions.

Letting x_{ij} be the value of the $j^{\underline{th}}$ explanatory variable for the $i^{\underline{th}}$ case, letting M_j be the number of distinct values of the $j^{\underline{th}}$ variable in the training data, and letting $x_{(i)j}$ be the $i^{\underline{th}}$ distinct ordered value of the $j^{\underline{th}}$ variable so that we have

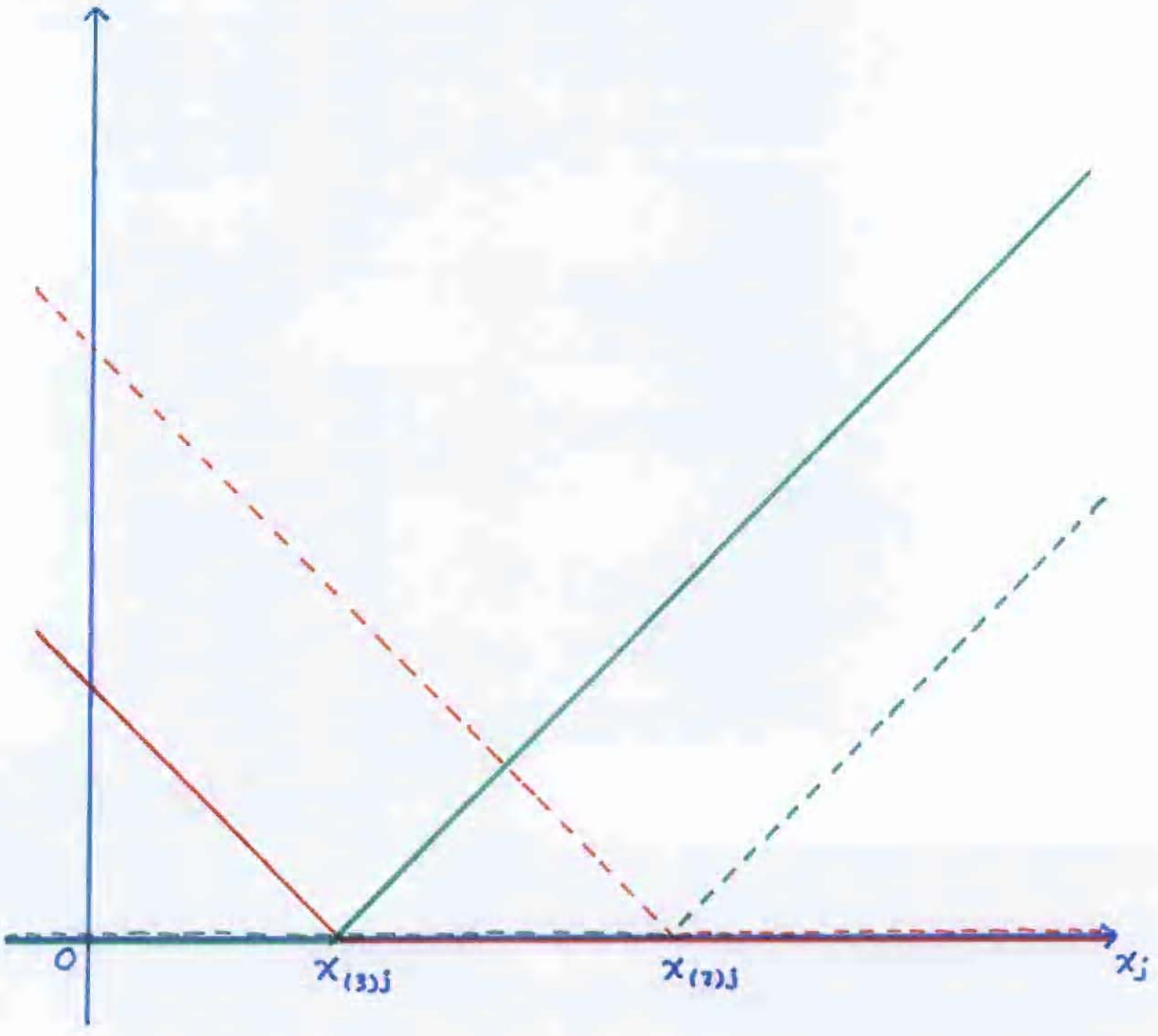
$$x_{(1)j} < x_{(2)j} < \cdots < x_{(M_j)j},$$

MARS considers the following $2M_j$ basis functions associated with the $j^{\underline{th}}$ explanatory variable:

$$(x_j - x_{(i)j})_+ \qquad (i = 1, 2, \ldots, M_j),$$

$$(x_{(i)j} - x_j)_+ \qquad (i = 1, 2, \ldots, M_j).$$

The sketch below shows four such *hockey stick* basis functions. Each function drawn in red is a *mirror-image* basis fn corresponding to one of the basis fns drawn in green.

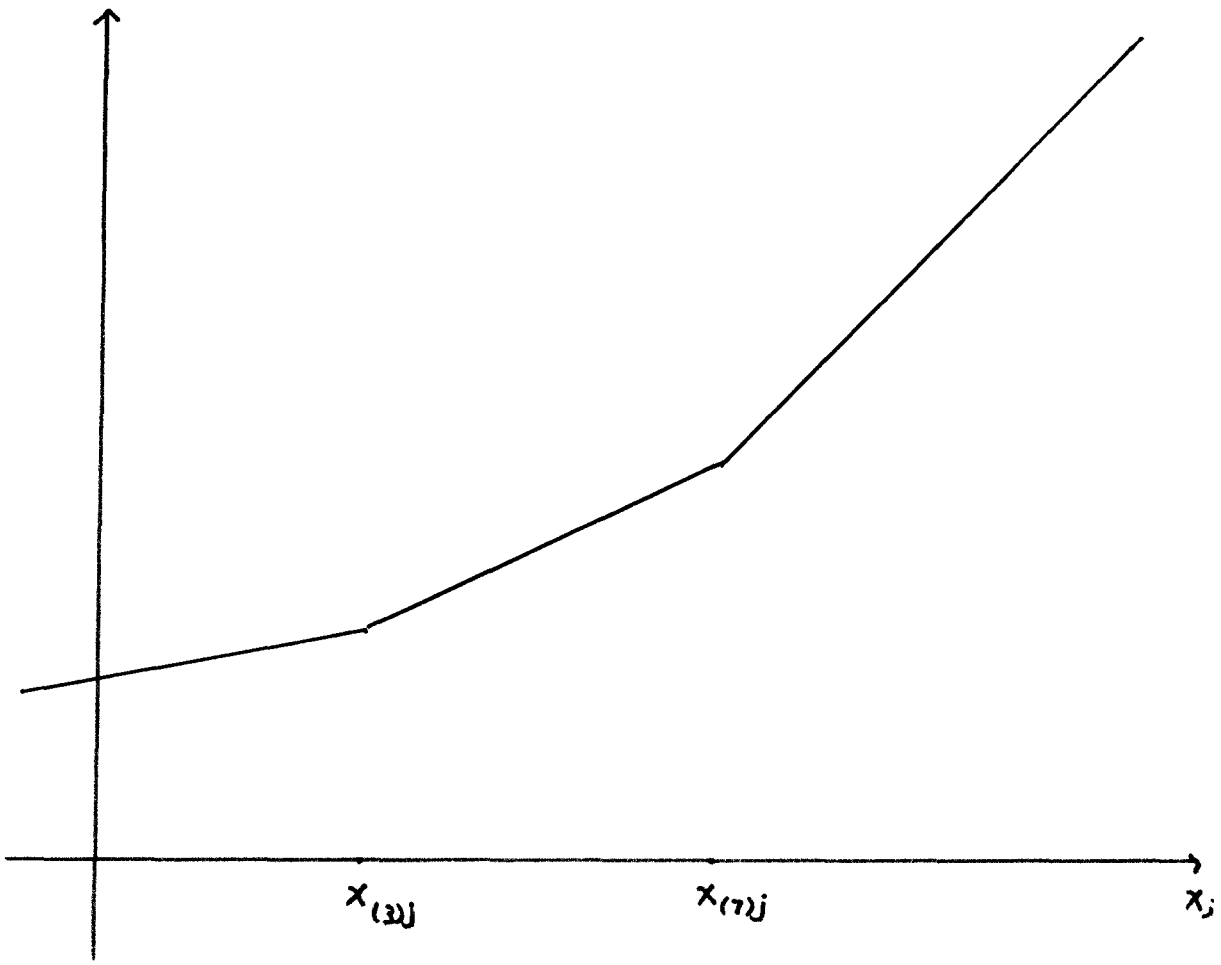


The set of $2M_j$ basis functions for x_j are not linearly independent. The spline sketched below is a linear combination of

$$1,\ (x_{(3)j} - x_j)_+,\ (x_j - x_{(33)j})_+,\ \&\ (x_j - x_{(7)j})_+,$$

but it is also a linear combination of

$$1,\ (x_j - x_{(3)j})_+,\ (x_{(7)j} - x_j)_+,\ \&\ (x_j - x_{(7)j})_+.$$



Constructing the Initial Model

MARS starts with just a constant in the model. Then it searches for the variable-knot combination which results in the greatest amount of improvement, as measured by the reduction in the sum of the squared errors, when the corresponding basis function pair (a primary fn and its mirror image) is added and a new model is fit using OLS.

For example, with just the basis fn BF0 = 1 in the model, upon using

OLS to fit the model, we have that $\hat{y}_i = \bar{y}$ $(i = 1, \ldots, n)$, and the initial sum of squared errors is $\sum_{i=1}^{n} (y_i - \bar{y})^2$. (Here the y_i are the observed response variable values in the training set.) Now suppose that after a search MARS identifies $(x_2 - x_{(12)2})_+$ and $(x_{(12)2} - x_2)_+$ as the best basis fn pair to add. OLS is used to estimate the coefficients for the model constructed from

$BF0 = 1$, $BF1 = (x_2 - x_{(12)2})_+$, & $BF2 = (x_{(12)2} - x_2)_+$,

the fitted coefficients are used to obtain the fitted values,

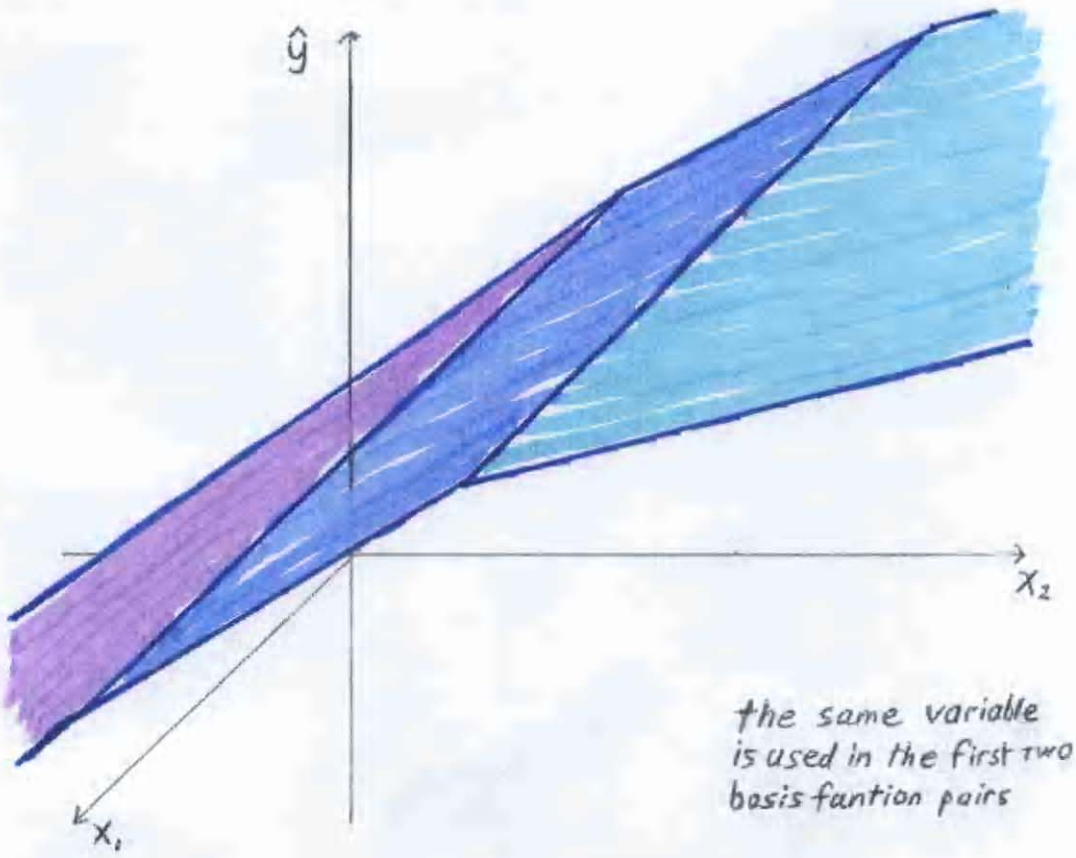
$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 BF1(\vec{x}_i) + \hat{b}_2 BF2(\vec{x}_i),$$

and the fitted values are used to obtain a new sum of squared errors, $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. The basis f'n pair, $(x_2 - x_{(11)2})_+$ and $(x_{(11)2} - x_2)_+$, was chosen because no other variable-knot combination produces a smaller sum of squared errors.

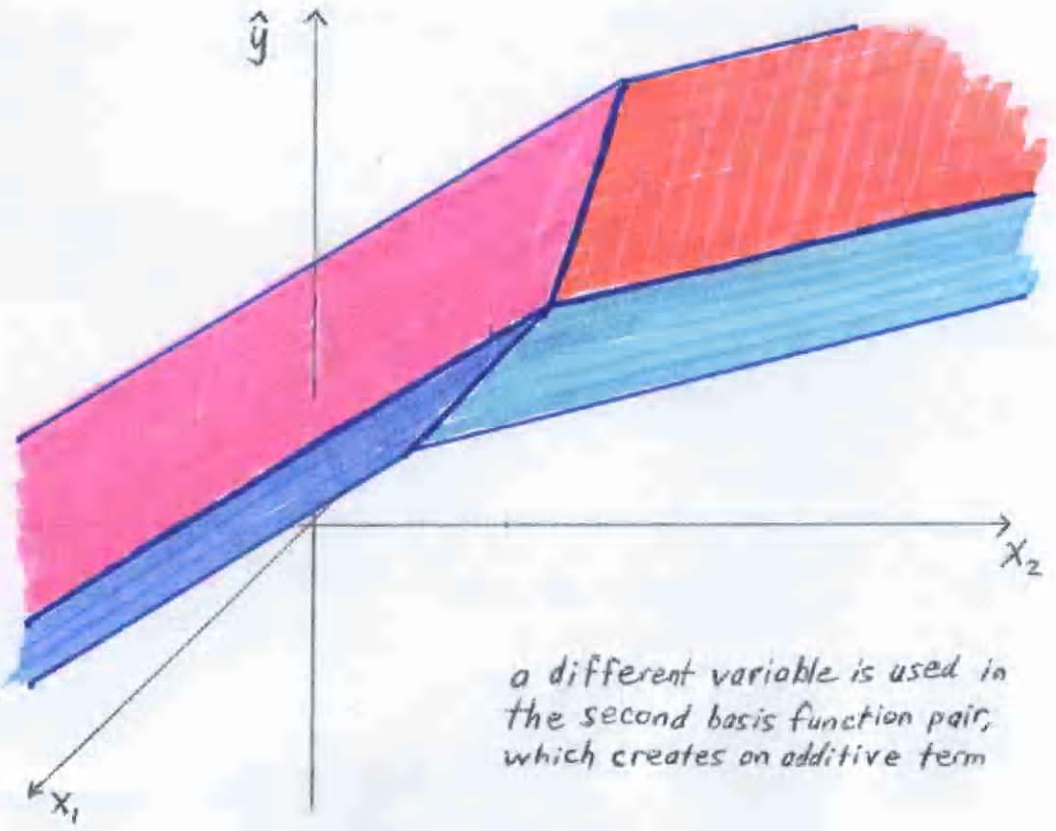
Once the first basis f'n pair has been added, things get a bit more complicated. Basically, there are three possibilities for what will happen next. One possibility is that another basis f'n pair, involving the same variable, but a different knot location, as the first basis f'n pair, will be added.

If so, then the resulting fitted response surface will have two bends, parallel to one another, and the value of the fitted response will depend on just a single variable. (Note: The two new basis fns will be called BF3 (the primary hockey stick fn) and BF4 (its mirror image). Only three of the four basis fns associated with the variable are needed, and eventually one of the two mirror image basis fns will be dropped.)

Another possibility is that a basis fn pair, involving a *different* variable, is



chosen. This results in an additive model. For example, there can be an effect on the response due to x_2 — below a certain value, say 3.3, $\hat{y}$ increases as x_2 inc. w/ one slope, say 0.85, and above that value (3.3), $\hat{y}$ inc. as x_2 inc. w/ a different slope, say 0.37, and the effect on $\hat{y}$ due to x_2



does not also depend on the value of x_1 ... there is no interaction effect. There can also be an effect on the response due to x_1 which is similar to the effect on $\hat{y}$ due to x_2 in that there are two different slopes. The effect due to x_1 is additive — the nature of the effect due

to x_1, depends only on the value of x_1, ... it does not additionally depend on the value of x_2. Overall, with such an additive effect, we have that

$$\hat{y} = \hat{b}_0 + \hat{g}_1(x_1) + \hat{g}_2(x_2).$$

The third possibility is for the second pair of basis fns to enter the model to form interaction terms, as opposed to creating an additive effect. This can only occur if the user overrides the default, which is to not allow interactions and instead create an additive model.

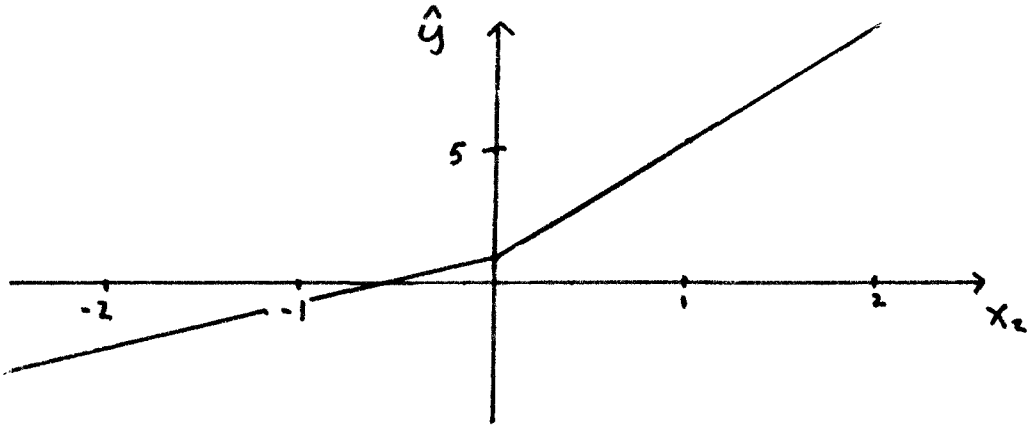
To explain how interaction terms are created, let me work with a specific example. Suppose that the first basis pair to enter the model involves x_2, with the knot being at 0. So we have

$$BF1(\vec{x}) = (x_2)_+ \;\&\; BF2(\vec{x}) = (-x_2)_+.$$

Further, suppose that the fitted model at this stage is

$$1.5 + 4\,BF1(\vec{x}) - 3.2\,BF2(\vec{x})$$

$$= 1.5 + 4(x_2)_+ - 3.2(-x_2)_+.$$



Now suppose that the next pair of basis fns added are

$$BF3(\vec{x}) = BF1(\vec{x}) * (x_1)_+ = (x_2)_+(x_1)_+$$

&

$$BF4(\vec{x}) = BF1(\vec{x}) * (-x_1)_+ = (x_2)_+(-x_1)_+.$$

That is, BF1 is multiplied by each member of a basis fn pair, involving x_1 with a knot (conveniently, to make this example less complicated) at 0, to create a new pair of basis functions. It should be noted that $(x_1)_+$ and $(-x_1)_+$ do not enter the model as basis functions. (We have two "interaction" terms involving x_1 without having any "main effect" terms involving x_1.)

At this point a model is fit using the basis functions

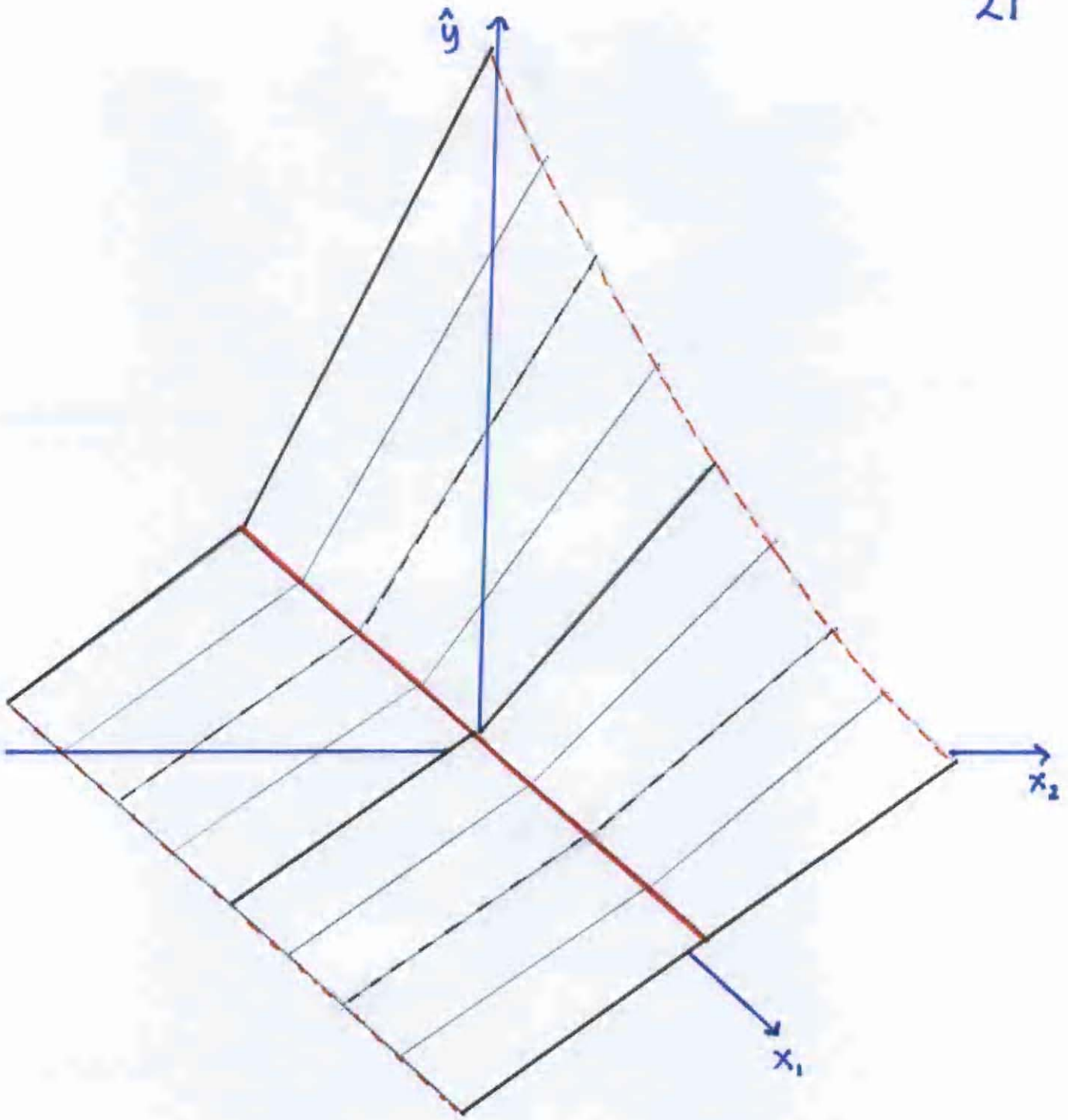
$$BF0(\vec{x}) = 1,$$
$$BF1(\vec{x}) = (x_2)_+,$$
$$BF2(\vec{x}) = (-x_2)_+,$$
$$BF3(\vec{x}) = (x_2)_+(x_1)_+,$$
& $$BF4(\vec{x}) = (x_2)_+(-x_1)_+.$$

If the fitted model is

$$1.25\,BF0 + 5\,BF1 - 3BF2 - BF3 + 2\,BF4$$

$$= 1.25 + 5(x_2)_+ - 3(-x_2)_+ - (x_2)_+(x_1)_+ + 2(x_2)_+(-x_1)_+,$$

then the estimated response surface is as shown on the next page. Note that $\hat{y}$ depends on x_1 only if $x_2 > 0$. Also note that, unlike what CART produces, the response surface fit by MARS is everywhere continuous.



a different variable is used in the second basis function pair, which is involved in an interaction

One way to describe the model building process is that at each stage MARS selects a variable and a knot to create a basis fn pair of the form

$$(x_j - x_{(i)j})_+ \quad \& \quad (x_{(i)j} - x_j)_+,$$

and each member of this basis fn pair is multiplied by a basis fn already in the model, say BFk, to form the next two basis fns to add to the model,

$$BFk * (x_j - x_{(i)j})_+ \quad \& \quad BFk * (x_{(i)j} - x_j)_+.$$

To create another "additive layer" the multiplying basis fn has to be BF0 = 1. Otherwise, a pair of interaction terms of some sort is added to the model. In all

cases, the basis fn pair and the multiplying basis fn are the ones which will produce the greatest amount of improvement.

MARS adds pairs of new basis fns to the model in this way until a complex model is created which overfits the data. (It's good to first grow a model at least twice the size of the model which is ultimately selected.) Then MARS removes one basis fn at a time to create a sequence of increasingly pruned models, other than BF0, at each step removing the basis fn, which results in the smallest increase in the

sum of the squared residuals when it is removed. This pruning process continues until all basis fns (except BFO) have been removed, and one is left with a sequence of increasingly smaller models, ranging from one which is too complex and overfits the data to one which makes use of none of the explanatory variables and gives the same prediction, $\bar{y}$, for all values of $\vec{x}$.

The final model is chosen from the sequence of models on the basis of a generalized cross-validation (GCV)

measure. Letting $\hat{f}_p$ denote the fitted model, from the sequence of models under consideration, constructed from p basis fns, the GCV measure for this model is

$$\frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}_p(x_i)]^2 \Big/ \left(1 - \frac{e_c}{n}\right)^2,$$

where c is the cost, or effective df, for each basis fn. (Note: The GCV measure does not actually involve cross-validation.)

With ordinary regression, c = 1 — 1 df is associated with each term in the linear model. But since MARS does an intensive search to select variables and

knots during the model growing stage, c needs to be set to a value larger than 1 in order for the GCV measure to work correctly, so that the model producing the smallest GCV value is the one which actually predicts best. Based on some empirical evidence, Friedmon initially suggested that c should be between 2 and 5. But more recent results suggest that when the number of variables and/or the sample size is very large, c should be much larger than 5.

Rather than take a wild guess as to what value c should be, it is better to use a test sample or cross-validation to select the final model from the sequence of different-sized models which MARS created. With a test sample things are simple — we can forget about GCV and just use the test sample to estimate the MSPE for each model in the sequence, and we then pick the one having the smallest estimated MSPE.

To use cross-validation, we can grow 10 sequences of models, leaving out a

a different one tenth of the data each time. We then consider a number of different values of c, and use the GCV measure to determine the "best" model for each value of c. With a 10-fold c-v, we have 10 models corresponding to one value of c, 10 models corresponding to another value of c, and so on. The left out portions of the data can be used to estimate the MSPE for each model, and upon averaging the results, the value of c which gives the smallest (average) MSPE can be determined. This value of c, along w/ the GCV meas., can be used to select the best model from the sequence

of models originally created from 100% of the training data.

linear entry of variables

MARS enters variables linearly into a model by using a basis fn with a basis fn for that variable having a knot at the minimum observed value for the variable, and not also adding the mirror image basis fn which would give a possibility for a change of slope at the minimum value — but with no data below the value, there would be no good reason to change

the slope, since there would be no good way to fit such a mirror image basis f'n. For prediction purposes, the variable would be in the model linearly over the range of values used to fit the model (and also beyond the maximum observed value for the variable), but if one extrapolates below the minimum observed value, the variable would not influence the prediction at all, as opposed to having the observed linear trend assumed to continue. (Note: If a variable is in the model with a primary basis fn and its mirror image basis fn,

then extrapolation would be done under the assumption of a linear trend on both sides of the knot.)

handling of categorical predictors

In classical modeling, a categorical explanatory variable is typically expanded into a set of dummy variables to represent the different observed outcomes of the variable. (If the variable is binary, just one dummy variable is needed.) For example, if a variable, say *department*, has observed outcomes A, B, C, and D, then one can create variables x_1, x_2,

and x_3 such that

$$x_{i1} = \begin{cases} 1, & \text{if department for } i\text{th case is } A, \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{i2} = \begin{cases} 1, & \text{if department for } i\text{th case is } B, \\ 0, & \text{otherwise,} \end{cases}$$

&

$$x_{i3} = \begin{cases} 1, & \text{if department for } i\text{th case is } C, \\ 0, & \text{otherwise.} \end{cases}$$

Then if the fitted model is

$$b_0 + b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i3} + \text{terms involving other variables},$$

we would have

$$\text{contribution due to department} = \begin{cases} b_0 + b_1, & \text{if department is } A, \\ b_0 + b_2, & \text{if department is } B, \\ b_0 + b_3, & \text{if department is } C, \\ b_0, & \text{if department is } D. \end{cases}$$

MARS does things a bit differently. It would

search for the best way to divide the observed outcomes of *department* into two groups, so that if a single dummy variable were created to reflect this split, it would result in the greatest decrease in the sum of the squared residuals when added to the model. This dummy variable would be represented by a basis fn, which is an indicator function. For example, we could have

$$BFd(\vec{x}) = \begin{cases} 1, & \text{if department is } A \text{ or } C, \\ 0, & \text{if department is } B \text{ or } D. \end{cases}$$

At some later stage in the model growing, MARS could create another basis fn

involving department, for example

$$BFk(\vec{x}) = \begin{cases} 1, & \text{if department is } C, \\ 0, & \text{if department is } A, B, \text{ or } D. \end{cases}$$

If in the end the fitted model is

$$b_0 BF0 + b_d BFd + b_k BFk + \begin{matrix}\text{terms involving} \\ \text{other variables}\end{matrix},$$

we would have

$$\begin{matrix}\text{contribution} \\ \text{due to} \\ \text{department}\end{matrix} = \begin{cases} b_0 + b_d + b_k, & \text{if department is } C, \\ b_0 + b_d, & \text{if department is } A, \\ b_0, & \text{if department is } B \text{ or } D. \end{cases}$$

In its handling of categorical variables, MARS breaks "the rules according to some", but since it carefully searches for the best model, and carefully selects the final model to achieve high prediction accuracy, it can

truly be said that typically the penalty will be paid by those who steadfastly obey "the rules." (MARS can treat a categorical variable in the traditional manner. If it does not do so, then it is because it appears that by not doing so, greater accuracy can be obtained.) Finally, it should be noted that when a basis f'n is created involving a categorical explanatory variable, a complementary basis f'n is also created (similar to the way that a primary hockey stick f'n and its mirror image f'n enter as a pair). But in the end, unnecessary basis f'ns are dropped.

handling of missing values

For each explanatory variable that has its value missing for one or more cases in the data set, MARS will automatically create a missing value indicator variable, and also a value present indicator variable. For example, suppose that x is a variable which has a missing value for some cases. Letting

$$(x = .) = \begin{cases} 1, & \text{if } x \text{ is missing,} \\ 0, & \text{if } x \text{ is not missing,} \end{cases}$$

&

$$(x > .) = \begin{cases} 1, & \text{if } x \text{ is not missing,} \\ 0, & \text{if } x \text{ is missing,} \end{cases}$$

the basis fns included in a final model might include

$$BF5 = (x - 21)_+ * (x > .),$$
$$BF6 = (21 - x)_+ * (x > .),$$
$$BF7 = (v - 0.5)_+ * (x = .),$$
&
$$BF8 = (0.5 - v)_+ * (x = .),$$

where v is a variable which is serving as a surrogate for x when x is missing. Any basis fn which adjusts the predicted response for x will have to include $(x > .)$ as a factor. Such factors are used even if one requests a model which does not allow interactions. (Note: $(x = .)$ doesn't have to be used if $(x > .)$ is used, but I suspect that if x is an important variable with more than a small number of missing values, that some surrogate variable terms will be used.)

Beyond the Defaults

maximum number of basis functions to allow
The default setting is 15, but typically this limit should be set much higher (but if you go overboard, it may take a long time to get results). In the end, you want the initial model to have at least twice as many basis fns as the final model. If you have already built a CART model, then it is recommended that you allow twice as many basis fns as there are terminal nodes in the optimal CART tree.

maximum degree of interactions

The default setting is 1, which prevents interactions and produces an additive model. While MAR's output is much easier to interpret with this setting, prediction accuracy may suffer. Using a setting of 2 tends to keep the model interpretation somewhat reasonable, but to help prevent you from missing something important, it is recommended that MARS be allowed to grow a model with the interaction level set to the depth of a satisfactory CART tree. (With a high level allowed, you might want to penalize added variables.)

penalty on added variables

The default is no penalty — a basis f'n pair involving a variable not already in the model will be chosen even if it is only a tad better than a basis f'n pair involving only variables which are already in the model. If some of the explanatory variables are highly correlated, then an easier to interpret model may be obtained, with little sacrifice of accuracy, if the penalty on added variables is increased, which will favor reusing variables already in the model over adding new variables.

forbidding transformations

For a variety of reasons, one may want to prevent certain variables from being transformed — this will have the variables put into the model in a linear manner, if they are used at all. If transformations are forbidden on all variables, MARs will produce a variation of a stepwise regression — only you can have cross-validation be used to select the final model from a sequence of different sized models, and this should be better than what stepwise routines usually do.

Other Tidbits

using MARS to construct a classifier

If the response variable is specified to be binary, and the threshold is set at 0.5, then MARS can produce a decent classifier provided that the training data can be viewed as a random sample from the same dist'n that will produce cases to be classified. (If this isn't the situation, then one might be able to get good results by adjusting the threshold appropriately.) The predicted values can be roughly viewed as being estimated probabilities for a "success" (a 1).

obtaining predictions from a MARS model

In addition to building a model to gain insight about a phenomenon, one might want to use the model to get predicted values of the response variable for new cases of explanatory variable values. There are fairly simple instructions for doing this on pp. 71-72 of the MARS User Guide.