POLYMARS

Charles Kooperberg Martin O'Connor

December 12, 1997

1 Introduction

In this document we describe the POLYMARS procedure. POLYMARS is a variation on the Multiple Adaptive Regression Spline (MARS) procedure of Friedman (1991)[6], that was introduced by Kooperberg, Bose and Stone (1997)[7]. In POLYMARS a multiple regression function is approximated using linear splines and their tensor products. POLYMARS can take multiple responses and can be used for classification. The POLYMARS procedure has a substantial number of options that make it a convenient tool for modeling. POLYMARS is implemented to be applicable to large data sets.

2 Description of POLYMARS

The model

We assume that regression data is generated from a "true model"

$$Y = f(X_1, \ldots, X_p) + \epsilon.$$

In POLYMARS the function f is approximated using functions that depend on only one or two of the x_i. That is, we use the model

$$f(\mathbf{X}) = g_0 + \sum_{j_1} g_{j_1}(X_{j_1}) + \sum_{j_1<j_2} g_{j_1 j_2}(X_{j_1}, X_{j_2}) + \epsilon$$

Stone et. al (1997)[10] motivate such an approximation from the perspective of ANOVA decompositions in which higher order interactions are ignored.

POLYMARS uses linear splines and their tensor products to model the functions $g()$. A one dimensional linear spline can be written in the form

$$g(x) = b_{-1} + b_0 x + \sum_{k=1}^{K} b_k (x - t_k)_+,$$

where

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

and the knots t_k are in the range of X. Thus, the function g is in a linear space with the $K + 2$ basis functions

$$1, \quad x, \quad \text{and} \quad (x - t_k)_+, \quad k = 1, \ldots, K.$$

We model the $g_{j_1 j_2}$, that depend on two predictors, using tensor product splines:

$$g_{12}(x_1, x_2) = g_1(x_1) \times g_2(x_2).$$

Therefore, we use the model

$$\begin{aligned} g_0 &= \beta_0 \\ g_{j_1}(X_{j_1}) &= \sum_i \beta_i^{j_1} B_i^{j_1}(X_{j_1}), \\ g_{j_1 j_2}(X_{j_1}, X_{j_2}) &= \sum_i \beta_i^{j_1 j_2} B_i^{j_1 j_2}(X_{j_1}, X_{j_2}), \end{aligned}$$

where the B's are the spline basis functions discussed above and the β's are coefficients. We can now write the POLYMARS model as

$$f(X) = \sum_i \beta_i B_i(X),$$

where the basis functions can be of the form 1, x_i, $(x_i - t_k)_+$, $x_i x_j$, $(x_i - t_k)_+ x_j$, and $(x_i - t_k)_+ (x_j - t_l)_+$. The coefficients are fitted by the method of least squares with the coefficients, matrix $\hat{\beta}$, described by

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y. \tag{1}$$

In the rest of this document $\mathbf{X}$ (*design* matrix) is the matrix of basis functions at the predictor values and Y is the vector of responses.

Model fitting

The basic algorithm for modeling f is:

```
Initially the constant function g_0 in equation is fitted to the data.
do{
  decide which basis functions are candidates for addition
  decide which basis function can be added to the current model
  add the best candidate
  fit the model
  evaluate the model
  if the model is better than the best one, save it
  }until(maximum model size is reached or no candidates are left)
do{
```

```
decide which basis function can be removed from the model
remove the one that is the worst predictor
fit the model
evaluate the model
if the model is better than the best one, save it
}until(minimum model size)
```

At each stage the procedure must find which potential basis functions can be added to the model. This procedure continues until a maximum model size is reached. (The maximum size of the model is a user specified parameter, the default of which depends on the sample size.) Then stepwise deletion of basis functions is employed. The candidates for removal are those that are currently in the model that can be removed taking a set hierarchical constraints, that are described in the next section, into account.

As basis functions are added to the model the residual sum of squares gets smaller, no matter how unimportant the new basis function is. So, the residual sums of squares, while providing a good comparison for lack of fit between models of the same size, does not penalize for over-fitting. Instead we use a modified form of the cross-validation criterion originally proposed in Craven and Wahba[1] and modified in Friedman and Silverman[4] and Friedman[2].

$$GCV(\hat{f}_M) = \frac{\frac{1}{n}\sum_{i=1}^{N}[y_i - \hat{f}_M(x_i)]^2}{\left[1 - \frac{C(\hat{f}_M)}{n}\right]^2}, \tag{2}$$

where $C(\hat{f}_M)$ is a function proportional to the number of basis functions added, $C(\hat{f}_M) = d \times M$ Here d is a constant that penalizes for larger models, usually $3 \leq d \leq 5$. This criterion is evaluated at the end of each addition and deletion step and the best-model-so-far by this criterion is stored. In our implementation this constant d is referred to as `gcv` and can be specified by the user.

Which candidate functions considered for model selection

At the initial (addition) stage a constant model is fit. After that, the number of basis functions that are candidates for addition depends on the number of possible knots per predictor. With more than a few predictors and knots this can be a large number, so evaluating each possible candidate is the computationally most expensive chore of POLYMARS. To keep the procedure fast we limit the possible candidates we use at each addition step. We specifically take these rules, since this procedure will prefer univariate functions and linear functions, which will yield a better interpretable final model.

The candidate basis functions are:

- x_i, $i = 1, \ldots, p$;
- $(x_i - t_{ik})_+$ if x_i is already a basis function in the model;

- $x_i x_j$ if x_i and x_j are already basis functions in the model;
- $x_i(x_j - t_{jk})_+$, if $x_i x_j$ and $(x_j - t_{jk})_+$ are in the model;
- $(x_i - t_{ik})_+(x_j - t_{jk})_+$ if $x_i(x_j - t_{jk})$ and $x_j(x_i - t_{ik})$ are in model.

The number of candidate knots is a user specified parameter. Initially the procedure computes order statistics for each predictor and take a subset of these as knots. Typically we may consider 20 or so potential knots for a predictor. Kooperberg, Bose, and Stone (1997) [7] argue that by limiting the number of candidate knots, the resulting procedure will be an order of magnitude faster than MARS.

The first iteration would fit a linear basis function to one of the predictors and the second iteration would consider linear basis functions in one of the other predictors as well as basis functions with knots on the predictor already added. This creates a preference in the procedure for linear models over nonlinear ones, while interactions are only considered if they are between predictors that are already in the model.

For the deletion stage, the candidates for removal are the basis functions in the model that, when removed, yield a valid remaining model. Each of these basis functions is taken out in turn and the *RSS* for each reduced model is calculated. The model which results in the smallest increase in RSS becomes the new model.

3 Options of POLYMARS

Multiple Response Regression

When the response for each observation is a vector, we can use a multiple response version of POLYMARS. The model fitted uses the same basis functions for each response but has different coefficients. The coefficients are computed as in equation (1)

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

The usual expression for the RSS

$$RSS = Y^T Y - Y^T\mathbf{X}\hat{\beta},$$

now becomes

$$RSS = \mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{X}\hat{\beta}.$$

The responses, $\mathbf{Y}$ and the coefficients $\hat{\beta}$ are matrices.

Model selection using a test set

A test data set may be used to select the best overall model. The models fitted in the stepwise addition and deletion stages uses the original (training) data. The overall best model for these is then selected by how well it fits the test data by a lack of fit criteria, usually RSS.

Using POLYMARS as a classifier

As a multiple response regression procedure POLYMARS can be used as a classifier, see Kooperberg, Bose and Stone [7]. If we have a vector Y that can take on a finite number of discrete values (classes) it may be appropriate to fit POLYMARS model to predict the corresponding classes based on a vector of covariates $\mathbf{X}'$.

Rather than fitting a multiple classification model using a polychotomous likelihood, we use POLYMARS with multiple response regression. Here for each case $\mathbf{Y}$ is the binary vector of length C with one 1, corresponding to the correct class. A numerical response vector can be given with the argument `classify = T` and the procedure will convert this response to multiple binary columns (Y should contain only integers). If Y is a vector of characters is always considered a classification problem (`classify = T` need not be set).

For classification by POLYMARS with a training set and test set the best model is selected using a loss function with unit loss for misclassification on the test set, adjusted for model-size similar to how we deal with RSS in equation 2.

Weights can also be supplied for the training and test sets.

4 Using POLYMARS

4.1 Arguments to POLYMARS

See the help files for more details.

- `responses` and `predictors`

  The only required arguments needed for the POLYMARS function.

- `weights`

  Case-weights, nonnegative.

- `maxsize`

  The maximum number of basis functions that the model can grow to. This will not be reached if no more candidate basis functions can be fit to the model at any addition step with model-size less than `maxsize`. By default maxsize $= \min(6 \times n^{\frac{1}{3}}, \frac{n}{4}, 100)$ where n is the number of cases in the dataset.

- `no.interact`

  For interpretability it may be desired that certain interactions should not be allowed. A $2 \times m$ matrix of these disallowed interaction terms is given to the `no.interact` argument. Each row contains the indices of a pair of predictor variables that may not contain interaction terms in the model. The procedure doesn't normally support interactions involving categorical variable levels, see `factors` below for more about this.

- `additive`

  Setting `additive = T` requires that the procedure only considers additive models. No interaction term will be considered for the model.

- `startmodel`

  An initial model may be specified. This has three main purposes. It suggests a starting point in model selection for datasets where there are many possible predictors but a small subset may be considered more important than the others. Further it may specify basis functions that the user feels *must* be in the model. Using `startmodel`, basis functions can be specified to be in all models considered.

  Because of the order in which basis functions must be added for any one predictor the procedure may be adverse to fitting the first linear term to predictors with a highly curved relationship which is almost orthogonal to the predictor itself. In this case a linear basis function may improve the *RSS* of the model very little beyond the intercept although two basis functions would fit well. By introducing an initial linear term in the `startmodel` such predictors have a better chance of being in the final model and if they are inappropriate the procedure should remove them at the deletion stage.

  See the help files for the required format of `startmodel`.

- `knots`

  Crucial to the procedure in modeling non-linearity are the possible knots for the basis functions. Ideally every data point for a predictor should be considered as a knot location. However due to computational expense it is usually better to consider only a subset of the data points.

  See the help files for the possible formats of `startmodel`.

- `knot.space`

  As noted in the section on the `knots` argument, the knots are selected as evenly spaced ordered statistics of the original data points, constrained to be at most every third order statistic. This is a numerical stability safeguard as basis functions with knots too close in the model may cause the design matrix to be nearly singular and unstable. This can be changed with the `knot.space` argument.

- `factors`

  Categorical predictors are converted to dummy zero-one variables so a factor of k levels can have at most $k - 1$ of its levels in the model. The `factors` argument specifies which predictor variables are to be treated as categorical. All unique values in the data-set for that predictor are then treated as levels.

  The procedure doesn't support interactions involving categorical variables. This can be gotten around by some pre-processing of the data, converting categorical variables into dummy 0/1 variables not defined as categorical.

- `classify`

  For a classification problem the responses can be treated as classes instead of a continuous variable. If the response is a vector of characters the procedure will automatically perform

POLYMARS with classification. If the response is of integer form this is done when `classify = T` is given as an argument. See Section 3. If `ts.resp` and `ts.pred` are given as arguments, the rate of misclassification of the test set response will be used to select the overall model for the best model at each step. It is adjusted for model-size as RSS is. Candidate basis functions are still evaluated by RSS.

- `ts.resp` and `ts.pred`

  At each step a candidate is selected for addition or deletion and a new larger or smaller model is then selected. Normally a GCV score based on the original training dataset is computed, see Section 2, so as to compare the model selected at each step adjusting for model size. When `ts.resp` and `ts.pred` are arguments, the RSS of the test set is used to find the best overall model from the models produced at each addition and deletion stage.

- `ts.weights`

  Case-weights for the test set, nonnegative.

- `tolerance`

  We also take advantage of the fact that X^TX grows only by adding a row and column when we compute $(X^TX)^{-1}$ after one addition step. By storing the $(X^TX)^{-1}$ after each addition step the new $(X^TX)^{-1}$ from an X matrix with one more column can be computed without direct matrix inversion. Using a result from Rao[9] p.33 :- When A and D are symmetric matrices and all inverses exist

  $$\begin{pmatrix} A & B \\ B^T & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + FE^{-1}F^T & -FE^{-1} \\ -E^{-1}F^T & E^{-1} \end{pmatrix}$$

  where $E = D - B^TA^{-1}B$ and $F = A^{-1}B$.

  The only inverses needed are A^{-1} which is the $(X^TX)^{-1}$ of the model produced by the previous iteration and D^{-1} which in our case is a scalar ($= \sum B_i(x_i)^2$ for the new basis fit). So the initial $(X^TX)^{-1}$ is a scalar and it is built on to get all the following $(X^TX)^{-1}$s. The B and D components are gotten from the candidate part of the $Y^TX \mid X^TX$ matrix.

  This matrix inversion must be done for each candidate at each step of the procedure and so benefits greatly from being fast.

  Normally inverting an $r \times r$ matrix requires $0(r^3)$ operations but here it is reduced to $0(r^2)$. So Y^TX and $(X^TX)^{-1}$'s main components are stored from iteration to iteration. The X matrix is also stored and updated for calculating inner products with new candidates.

  This procedure is reversed during the deletion stage of the algorithm.

  It has been found that the size of the element E which is a number in our case is important for stability. Basis functions that cause the design matrix to become unstable (nearly singular) produce either very large or very small values for E. To avoid instability of the numerical procedure, (computing $(X^TX)^{-1}$), candidate basis functions are rejected if they produce values of E less than the value of `tolerance` or greater than 1 divided by `tolerance`. The default of `tolerance` is 1.0×10^{-5}.

Note This implementation uses a matrix inversion function from the *Lapack* libraries. This function is called if there is an initial model specified and once per addition/deletion step to update the $(X^T X)^{-1}$ matrix.

- `verbose`

  When `verbose = T` is given as an arguement some information is printed during the model fitting. A `+` means an addition to the model, a `-` means a basis function is taken away. The model size is printed next and the basis function which is added or deleted.

4.2 Other functions

See the help files for details

- **plot.polymars**

  Plots 1 and 2 predictor graphs of a model returned from polymars.

- **predict.polymars**

  Computes fitted values of a model returned from polymars.

- **print.polymars, summary.polymars**

  Prints formatted output of a model returned from polymars.

4.3 Interpreting the returned model

- `mymodel<-polymars(theresponse,thepredictors)`

- The visible model
  Using `summary` or `print` (which defaults to `summary`) on an object returned from `polymars` prints out three components of the POLYMARS object "mymodel".

  – The **call** which produced the object.

  – An Splus data-frame, **fitting** which contains certain statistics about the model fitting routine. Each row represents one step in the fitting routine. The first column *0/1* has a 1 for an addition step and a zero for a deletion step. The second column, *size*, contains the resulting number of basis functions in the model after this step. Next there is a column *RSS* containing the residual sum of squares. For multiple response regression or classification there will be more than one column *RSS1*, *RSS2*..., one for each response or class.

  The last column contains the measure of fit which used to find the best overall model. It is normally *GCV* for generalized cross validation, see Section 2. But it can also be *T.S. RSS* for test set residual sum of squares or *T.S.M.C.* for test set misclassification, see Section 3.

– The POLYMARS **model** itself is printed as a data-frame, each row corresponding to a basis function. The first row corresponds to the intercept. The first four (or five) columns relate to the basis function and the last column (more than one for multiple response regression or classification) contains the coefficients. For classification coefficients see also *conversion* below.

The first column *pred1* contains the index of the first predictor of the basis function. Column *knot1* is a possible knot in this predictor. If this column is NA, the first component is linear. If any of the basis functions of the model are categorical then there will be a *level1* column. Column *pred2* is the possible second predictor involved (if it is NA the basis function only depends on one predictor). Column *knot2* contains the possible knot for the predictor *pred2*, and it is NA when this component is linear. For example the following model

```
  pred1 knot1 level1 pred2 knot2      coefs      SE
1     0    NA     NA     0    NA  59.123913  9.0277
2    13    NA     NA     0    NA  -5.508833  2.2838
3    13  6.29     NA     0    NA   4.834013  1.9678
4     4    NA      1     0    NA   9.486980  2.6050
```

has an intercept of 59.12 and predictor 13 has two terms in the model, a linear term with coefficient -5.51 and a term with a knot $(X13-6.29)_+$ with coefficient 4.83. One level of variable 4 is in the model (this was actually a 0/1 variable) with coefficient 9.49. Standard errors for the coefficients are also included.

A line such as

```
  pred1 knot1 pred2 knot2      coefs      SE

      2     0     3  0.28   12.45667  3.3382
```

corresponds to a basis function $X2 \times (X3 - 0.28)_+$ with coefficient 12.46.

- The invisible attributes
  - **model.size** contains the number of basis functions in the returned model.
  - **fitted** and **residuals** contain the fitted values and the residuals of the original dataset used to fit the model.
  - **responses** contains number of responses per case in original dataset.
  - **ranges.and.medians** is a $3 \times p$ matrix, where p is the number of predictors in the original dataset, each column corresponding to a predictor. Rows 1 and 2 contain the minimum and maximum values of the predictor and row 3 contains the median value of the predictor. These are computed from the original (training) dataset and are used in the plotting function.
  - **factor.matrix** is is a $r \times s$ matrix where r is the number of categorical predictor variables in the model and s is the maximum number of levels that any of these has +2. Each column represents a categorical variable. The first row contains the index of the predictor, the second row contains the number of levels it has and the remaining rows contain the value of each level (filled out to the end with NA's if necessary). This is used in the plotting function.

– **conversion** is a $c \times 2$ matrix where c is the number of classes or categories in the response when POLYMARS is used as a classifier. In classification a single response vector is split up into a multiple response matrix with each column corresponding to a class and each case having a 1 in the column corresponding to it's original response and all other columns zero.

  In the **conversion** matrix each row corresponds to one class. In the first column it's original class, as a character or number, is recorded. The second column holds the response number, or column index of the new response matrix. The coefficients of **model** are ordered according to this numbering. It is used for computing further fitted values (classes).

5 Examples

5.1 Boston housing data

The data for this example comes from Statlib, (`http://lib.stat.cmu.edu/`). The data-set contains variables on various criteria such as distance to urban amenities, pollution and crime which may effect house prices. The data is originally from Harrison and Rubinfeld [5].

The variables in this data set are:

1. per capita crime rate by town.
2. proportion of residential land zoned for lots over 25,000 sq.ft.
3. proportion of non-retail business acres per town.
4. Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. nitric oxides concentration (parts per 10 million)
6. average number of rooms per dwelling
7. proportion of owner-occupied units built prior to 1940
8. weighted distances to five Boston employment centers
9. index of accessibility to radial highways
10. full-value property-tax rate per \$10,000
11. pupil-teacher ratio by town
12. $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
13. % lower status of the population
14. Median value of owner-occupied homes in \$1000's

For this regression we left all variables untransformed. The response variable is x_{14}, the median value of owner-occupied homes in \$1000's. For demonstration we asked for 15 possible knots per predictor, some predictors will have less than this, for example (x_2 has 73% of it's cases equal to zero) so will have less knots. Variable 4, x_4, is specified as a categorical variable (this is a bit redundant as it only has two levels).

The procedure will use a test set and training set instead of the usual GCV parameter to select the model so we split the data into two groups.

For interpretation of the final model we may want to exclude certain possible interactions

such as a nitrous oxide/rooms per dwelling effect. In this example we picked two interactions that we didn't want to see in the model, those between x_5 and x_6 and between x_3 and x_{11}. For a realistic analysis there are other interactions which are just as uninterpretable.

The *Splus* code was

```
index<-sample(nrow(boston.dat),nrow(boston.dat)/2)

dni<-matrix(c(5,6,3,11),byrow=T,ncol=2)

boston.mars<-polymars(boston.dat[index,14], boston.dat[index,-14],
                      knots=15,factor=4, ts.resp=boston.dat[-index,14],
                      ts.pred=boston.dat[-index,-14], no.interact=dni)
```

The returned model was:

Basis function	Coefficient
Intercept	-115.92
x_1	-0.18
x_6	31.10
$(x_6 - 6.44)_+$	5.94
$(x_6 - 7.84)_+$	-9.37
x_8	-43.72
$(x_8 - 1.53)_+$	43.07
x_{11}	11.34
$(x_{11} - 17.80)_+$	-2.02
x_{13}	-1.21
$(x_{13} - 5.52)_+$	-1.98
$x_6 \times x_{11}$	-1.56
$x_8 \times x_{13}$	0.04
$x_{11} \times x_{13}$	-0.07

This model has a multiple R^2 of 0.93.

5.2 A simple function of ten variables

This example is taken from Friedman's MARS [3] paper which as an example illustrates the ability of the procedure to find structure is data while ignoring noise. The data is created using the function

$$f(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - \frac{1}{2})^2 + 10x_4 + 5x_5 + \epsilon$$

in a $n = 10$ dimensional hypercube using $N = 100$ points. Covariates were drawn from a uniform distribution and ϵ is from a standard normal distribution. Figure 1 shows the original function without noise for variables x_1, x_2 and x_3. $f(\mathbf{x})$ is linear in x_4 and x_5.

This example with it's highly curved relationships is not an ideal setting for POLYMARS with it's piecewise linear splines. This is particularly true for x_3 as our procedure insists that a linear basis function is the first term fit to any variable. A linear term will not improve the fit any more than just the constant intercept term so the model building procedure will be averse

to adding any x_3 terms. For this reason we specify x_3 to be in the startmodel.

The model with *gcv* equal to 3.0 and the default number of knots, $knots = 20$.

The *Splus* code was

```
mars2 <- polymars(responses = yy, predictors = cbind(x1, x2, x3, x4, x5, x6, x7,
x8, x9, x10), gcv = 3, startmodel = c(3, NA, 0, 0))
```

The returned model was

Basis function	Coefficient
Intercept	-2.21
x_1	13.83
x_2	13.18
$(1-0.71)_+$	-23.50
$(2-0.55)_+$	-12.16
x_3	-10.18
$(3-0.45)_+$	17.94
x_4	9.37
x_5	4.69

Figure 2 shows the fitted functions for x_1, x_2 and x_3. The coefficients of x_4 and x_5 are quite close to that of the original function. The procedure works quite well in picking up the main structure present in the data without picking up any of the 5 noise covariates.

The introduction of an initial linear term to detect *U-shaped* relationships is something that would be explored in exploratory data analysis.

Calling POLYMARS with a linear term for all 10 variables in `startmodel` may be also a reasonable thing to do. The maximum size the model should grow to (`maxsize`), was set to 35 to compensate for these initial basis functions being already in the model (by default the maximum size is 25 for this size of a dataset, see section 4). The model produced was very similar to the one above with one extra basis function, x_6. This had a coefficient of -2.43. It did pick up one of the noise variables but regressing the x variables on y using *Splus* function `lm` fits a coefficient of -3.35 on x_6 with a t-value of -3.37 with 89 degrees of freedom, the most "significant" of the noise variables.

This example was original chosen to demonstrate the capabilities of a different procedure, but this piecewise linear $MARS$ implementation works quite well.

5.3 Phoneme data

The source of this data is the Center for Spoken Language Understanding in Portland, Oregon. A full description of the data can be found in Kooperberg and Stone [8]. The data used in this example is a subset of the original dataset consisting of a three class response, corresponding to three phonemes; the vowels in b*ee*t, b*e*t and b*ou*ght. The predictor set consists of 81 variables. These variables were obtained by processing the audible spectrum of the utterance that produced each phoneme, to produce perceptual linear predictive (PLP) features. The dataset consists of a training set of 14,735 cases and a test set of 10,167 cases. We selected a random subset of 1000 cases from the training set and 300 from the test set although this implementation has also be used on the full set.

The model produced contained 17 of the predictor variables summarized as follows:

Intercept	
Linear variables	9
Variables with 1 knot	5
Variables with 2 knots	1
Variables with 3 knots	1
Interactions with 2 linear terms	12
Total number of basis functions (with intercept)	39

Fitted values were obtained from this model for the remaining 13,736 cases. The model had a misclassification rate of 6% on this data.

The *Splus* code was

```
index1<-sample(1:14736,1000)
index2<-sample(1:10617,300)

phoneme.mars<-polymars(phontr.dat2[index1,1],phontr.dat2[index1,-1],
ts.resp=phonts.dat[index2,1],ts.pred=phonts.dat[index2,-1],
knots=30,classify=T)

predictions<-predict.polymars(phoneme.mars,phontr.dat[-index1,-1],classify=T)
```

Appendix: Outline of the program

The basic program flow

1. **Mesh computed**

   If the possible knots for each predictor are not given as an argument knots are calculated for each predictor. They as selected as evenly spaced order statistics.

2. **Initial model fit**

   If an initial model is an argument to polymars, it is checked for consistency to the addition of terms ordering and then fit. Any knots involved are added to the mesh. If no initial model is specified an intercept is fit.

3. **Addition iteration**

   (a) All possible (new) candidates are found and stored.

   (b) The best of these candidates is found

   (c) The best of candidate is added to the model and removed from the candidates. If no candidate can be fit, addition stops.

   (d) If largest model size specified has been reached, addition to the model stops.

4. **Deletion iteration**

   (a) From all possible candidates for removal from the model, the best one is calculated

(b) The best candidate for removal is taken out of the model

(c) If the smallest possible size is reached the deletion stage stops

5. **Best model is reported**

References

[1] Craven, P., and Wabha, G. (1979), Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation, *Numerical Mathematics*, **31**, 317–403.

[2] Friedman, J.H. (1988), Fitting functions to noisy data in high dimensions. In *Computing Science and Statistics: Proceedings of the Twentieth Symposium on the Interface*, (E.J. Wegman, D.T. Gantz and J.J.Miller, eds.) 13–43. American Statistical Association, Alexandra, Virginia.

[3] Friedman, J.H. (1991), "Multivariate Adaptive Regression Splines (with discussion)," *The Annals of Statistics*, **19**, 1–141.

[4] Friedman, J.H., and Silverman, B.W. (1989), "Flexible parsimonious smoothing and additive modeling," *Technometrics*, **31**, 3–39.

[5] Harrison, D. and Rubinfeld, D.L. (1978), "Hedonic prices and the demand for clean air," *J. Environ. Economics & Management*, **19**, 81–102.

[6] Hastie, T.J., and Tibshirani, R.J. (1994), *Generalized Additive Models*, Chapman & Hall.

[7] Kooperberg, C., Bose, S., and Stone C.J. (1997), "Polychotomous Regression", *Journal of the American Statistical Association*, **92**, 117–127.

[8] Kooperberg, C., and Stone C.J. (1997), "Using the Stochastic Gradient Method to fit Polychotomous Regression Models", *University of Washington, Department of Statistics, Technical Report* 319.

[9] Rao, C. R. (1989), *Linear Statistical Inference and Its Applications*(2^{nd} ed.), Wiley & Sons.

[10] Stone, C. J., Hansen, H., Kooperberg, C., and Truong, Y. K. (1997), "The use of polynomial splines and their tensor products in extended linear modeling (with discussion)," *The Annals of Statistics*, to appear.

Figure 1: Original function without noise

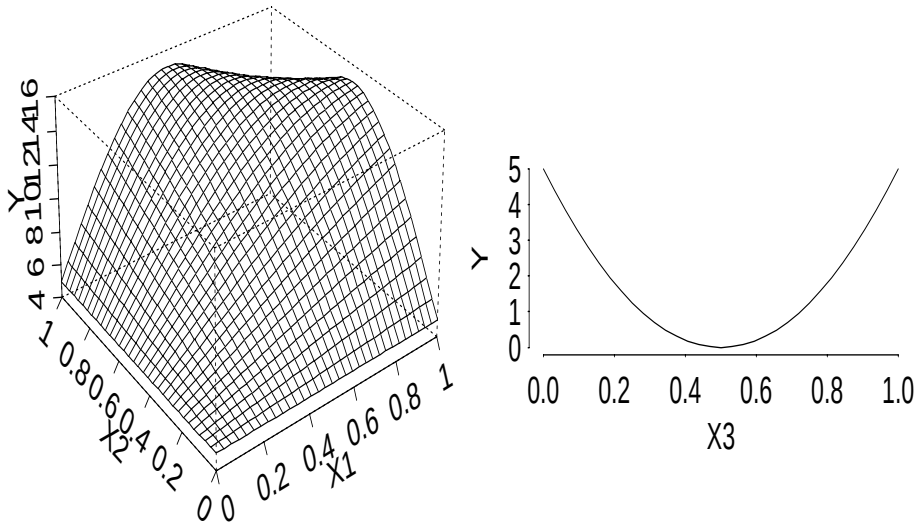

Figure 2: POLYMARS fit of the noisy data