

Adaptive Regression in SAS/IML

David Katz, David Katz Consulting, Ashland, Oregon

ABSTRACT

Adaptive Regression algorithms allow the data to select the form of a model in addition to estimating the parameters. Friedman's procedure exploits computational shortcuts in Adaptive Regression, obtaining the power of Neural Networks with a fraction of the resources. Proc IML allows us to explore these tools in a flexible environment, with exciting results. This paper describes an ongoing project that implements this approach. We review the algorithm, discuss the programming techniques, and give some examples of useful applications.

ADAPTIVE METHODS AND SPLINES

Standard Multiple Linear Regression searches for a model of the form $y = \sum_i w_i x_i$. Since the x_i can be transformed using any

a priori transformation, this includes such variants as polynomial regression, which are often informally referred to as nonlinear models; however the model is linear in the transformed variables, so the standard least squares methods still apply. A problem arises when there are many variables in the analysis. Searching through many possible transformations becomes impractical, as the number of parameters we need to estimate grows rapidly and outstrips the number of parameters which our data can estimate.

This has motivated the development of a class of models called Adaptive Dictionary Methods. These are of the form

$$y = \sum_i w_i g_i(\vec{x}) \quad (1)$$

where the g_i are nonlinear functions estimated from the data vector $\vec{x} = (x_1 \ x_2 \ x_3 \dots x_n)$. For example, each g_i could be defined as the moving average of x_i . Once the g_i have been defined, the w_i are then estimated by least squares. The g_i are often called features, reflecting the intuitive notion that they represent notable features of the data that can be recognized and then combined as in (1). An example of a useful feature is an interaction term.

Adaptive methods are useful when the main goal of the analysis is prediction rather than hypothesis testing. They avoid using the strong assumptions needed for Logistic regression or Multiple Regression, such as linearity. We choose a model which minimizes the prediction error when the model is applied to a new data sample. This is done with a holdout sample, or via cross-validation, or generalized cross-validation (see below). The main issues in Adaptive Dictionary Methods are

1) Selecting the form of the g_i . We need a suitable set of functions, i.e. functions that are flexible enough to fit the data, but can be estimated in a feasible manner. Generally these basis functions are parameterized, and we need a procedure for estimating the parameters.

2) The number of features g_i needs to be appropriate to the data we are fitting. This is similar to the familiar process of Stepwise variable selection. We add or delete basis functions (terms) and create new candidate models, and evaluate these models for the best balance of fit and stability. However, the theoretical basis is different, since here we are not making any assumption about the form of the underlying process that produced the data.

In his 1990 paper, Friedman proposed an Adaptive Dictionary method he called Multivariate Adaptive Regression Splines. Splines are piecewise polynomial functions with some added constraints to assure continuity. The most commonly used splines are one-dimensional piecewise cubic polynomials which are constrained to be continuous and have continuous first and second derivatives at the points where the pieces meet. These points are called knots.

Splines are useful because they are flexible; like polynomials of arbitrary degree, they can uniformly approximate any function (over a compact set) with similar continuity requirements. Unlike polynomials, splines are of limited degree, but add flexibility by using more knots and thus more pieces. They have the desirable characteristic that a sum of, for example, cubic splines are also cubic splines.

Splines can be generalized to dimensions >1 in a number of ways. Friedman used tensor product splines, which take the product of univariate splines s_j in distinct dimensions. That is,

$$g_i = \prod_j s_j \text{ where the } s_j \text{ are univariate splines from distinct}$$

dimensions. Combining this with (1) we have

$$y = \sum_i w_i \prod_j s_j(x) \quad (2)$$

as the form of the models. Friedman's procedure constructs these models with a procedure that is computationally efficient.

First observe that piecewise linear splines can easily be smoothed to cubic. This reduces the problem to finding models of form (2) with s_j now representing piecewise linear splines. In fact, for many applications, the linear spline representation is accurate enough, and is easier to compute.

Next, observe that linear splines have a convenient basis.

Consider the functions of the form $(x - a)_+$ and $(x - a)_-$, where

$$x_+ = \begin{cases} x & \text{when } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$x_- = \begin{cases} x & \text{when } x \leq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Piecewise linear functions can be represented by linear combinations of functions having this form. If these primitive basis

functions are denoted b_i , then it follows that (2) can be rewritten as

$$y = \sum_i w_i \prod_j b_j(x). \quad (3)$$

Each b_j can be characterized by its dimension (the input variable x_i used in its definition), by its sign orientation, and by its knot position.

To determine which basis functions to use, we start by selecting the best model that uses just a single pair of b_j . One of the pair will be oriented positively, and one negatively, and they will share the same knot. We find this pair by stepping through each variable, and each data point in our training sample provides a potential knot position projected onto that variable. Using least squares, we fit each possible model in this set, and select the one with the best fit, usually based on finding the lowest MSE.

We then add additional basis function terms to the model, one pair at a time. In addition to searching the univariate basis functions, we also test "child" basis functions. These are built by using a basis function already in the model as one factor, and one of the b_j (not already a factor in the "parent") as another factor,

thus testing many tensor product basis functions $\prod b_j$ for possible inclusion in the model.

For example, suppose we have selected an initial pair of primitive basis functions for inclusion in the model:

$$b_1 = (x_2 - a_1)_+ \\ b_2 = (x_2 - a_1)_-$$

so that the initial model is estimated as $w_1 b_1 + w_2 b_2$. The initial step of the search has determined that this is the best model of this form of all choices of raw variables and knots in the search. In this example, we have chosen raw variable 2 and knot a_1 which

is a value of x_2 that appears in the data. When we search for another basis function, we evaluate models of the form (3) with 4 terms. The additional pair of terms could be another primitive like the first two, but we also test products such as

$(x_2 - a_1)_+ (x_1 - a_2)_+$. This is called a child of the first basis function. We are building up a tree of basis functions with the intercept term at the root of the tree. The greedy algorithm reduces the enormous search space of all possible tensor products to those which have one factor already in the model, which shows that is of interest for prediction. In this respect, the algorithm is similar to the CART and CHAID procedures, but it can produce better results when the data contain additive characteristics.

We are clearly performing a large number of multiple regressions to search all these possibilities. Friedman makes this feasible by developing formulae to avoid recomputing the sum of squares and cross products from scratch each time. As we move from one candidate knot to the next, we can update the SSCP matrix efficiently by only computing the change in this matrix. It is even

possible to simplify the matrix inversion step by using results from the previous knot.

This type of forward search is often called a greedy algorithm, in that it takes the best choice at each step. Sometimes this can lead us astray, as when an early choice leads us in a suboptimal direction. One method often used with greedy algorithms is to repeat the forward search beyond what seems necessary, and to follow it with a backwards approach where the model is simplified. We select the term which adds the least to the model and delete it. We can then select the best model found in the backwards search by using the Generalized Cross-Validation estimate of model fit. This technique adjusts the MSE to reflect the complexity of each model. This provides a brake on overfitting and estimates the MSE which would be expected via cross-validation. The model with the lowest Generalized Cross-Validation is selected as the best. Thus Generalized Cross-Validation holds a position analogous to the C_p statistic in multiple regression. Both of these are guides to the number of terms to include in the model.

IMPLEMENTATION

SAS/IML provides an excellent tool for exploring these ideas. Proc IML implements a matrix language distinct from the data step and macro languages of SAS, but well integrated with the rest of the SAS system. Tools are included for importing and exporting SAS datasets into the SAS/IML workspace. Within this workspace, they are manipulated using a command language closely related to matrix notation. Since SAS/IML can manipulate matrices easily, the basic algorithm translates readily to this language. IML provides operators for matrix multiplication and also for elementwise multiplication. There are built-in functions like Solve for linear systems, and even Trisolve for triangular systems, which turns out to be particularly relevant for Friedman's procedure.

Another useful feature is the ability to specify submatrices easily. This proved to be useful for the incremental formulae. In IML, let i and j be arrays of indices; then $A[i,j]$ denotes the appropriate submatrix of A .

Because SAS/IML is interpreted, it is easy to make changes and see the effect on the results. However, for the same reason, there are some challenges with the speed of the calculation. I addressed the speed issue by using a technique known as subsampling. When there are very large datasets, it is probably not essential to test every single point as a potential knot. Skipping points will still provide an excellent approximation in most cases. The number of points to skip becomes a parameter of the SAS/IML program.

Another challenge was the lack of arrays of matrices in SAS/IML. I needed to track lists of knots for each variable. The lists were of varying length, so it would have been wasteful of memory to allocate a single 2-dimensional matrix. The solution was to use the SAS/IML execute command. This enables you to execute an expression that you construct on the fly. In this case, I created a character array of matrix names called cutlistnames. These names would be used in an expression like:

```
Call execute(cutlistnames[jx] || "=cutlistptr3;");
```

This assigned the name at index jx the value in cutlistptr3 which in this case was an array of arbitrary length. Thus cutlistnames acted as a virtual array of arrays of potentially different lengths.

The main loop for the Adaptive Regression in SAS/IML looks something like this:

```

/* Step thru each term in the model, testing all children for better regression */
m=2; /* initialize term index */
do until(m>maxterms);
  lof=10**99; /* initialize for loop to high-value */
  found=0; /* any nonsingular reg done this m? */
  /* searching for lowest lack-of-fit criterion to add a term */
  do ibasis=1 to ncol(allterms); /* step thru all basis fns in model so far*/
    if degree(factor[ibasis,])<maxdeg
      then do j=1 to &varn; /* step thru new basis fns of low degree */
        window w2 group=main 'terms' m / 'parent' ibasis/ 'child ' j;
        display w2.main noinput;
        if factor[ibasis,j]='none' then do; /* skip if variable in parent */
          %prnt('work on variable ' j ' with parent basis ' ibasis);
          nuvalues=x[,j]; /* col vector of new variable values */
          %searchthruknots
          nextvariable:
        end; /* variable in parent */
      end; /* step through variables */
    end; /* step thru basis functions */
  if found then %addbestbasis;
  else m=maxterms+1; /* ends loop */
end; /* of main loop */

```

The first term is the intercept. The children of the intercept term are of degree 1. Children of other basis functions are of the next higher degree. The maximum degree searched is a parameter of the program run, as is the maximum number of basis functions. When this maximum is reached, the program proceeds to backwards selection. Friedman suggests that the final model should have no more than half the basis functions in the maximal model. If not, the procedure likely needs to be rerun with a higher value for maxterms.

The integration of the SAS Macro facility was another advantage of using SAS/IML. The %prnt macro is a simple tool for debugging. The verbose variable is a positional parameter which can be set to enable various levels of print messages. It defaults to verbose=1, which means the message only prints if the global verbose flag is 1 or greater. Macros like %searchthruknots and %addbestbasis simply make the code more readable, while avoiding the overhead of an interpreted function call to a SAS/IML module.

Here is the code for the %prnt macro.

```

%macro prnt(text,prntlvl=1);
  %if &verbose>=&prntlvl %then %do;
    print &text;
  %end;
%mend;

```

Another handy macro for SAS/IML debugging is given below. It was helpful where large matrices might be involved, so that the built-in print function would produce more information than needed.

```

%macro printdim(mat);
  sysrows=nrow(&mat);
  syscols=ncol(&mat);
  print "&mat" sysrows syscols;
%mend;

```

REPRESENTING THE TREE

As described above, the search for the best model involves building a tree of basis functions. Each basis function is a tensor product of the primitive splines. We represented each of these products as a pair of row vectors. One vector showed the knot locations, and the other showed the type of primitive, positive or negative. So for an analysis with 4 input variables we would have basis functions such as:

cut cut none none	.43 8 . .
-------------------	-----------

This is the basis function $(x_1 - .43)_+(x_2 - 8)_+$. Another example shows how the negatively oriented primitives are represented:

cut -cut none none	5 7.5 . .
--------------------	-----------

Represents $(x_1 - 5)_+(x_2 - 7.5)_-$.

By combining these rows, we obtained two matrices that represented the entire growing model. This representation made it possible to perform the search using SAS/IML commands.

ENHANCING THE ALGORITHM

There are some cases where a priori knowledge of the problem domain makes it possible to modify the algorithm to fit the analysis. In one case, I suspected that one variable might be interacting with any of the others, but it seemed very unlikely that these other variables would interact with each other. It was a simple matter to add a filter to the search for new basis functions. This insured a model of the form I required, while saving a great deal of calculation.

In many cases we have knowledge of the sign a given parameter should have. Because our code controls the details of the search, we can incorporate sign checks as required.

These modifications simply restrict the search space, and require no other modifications to the algorithm or the theory.

EXPERIENCES USING THE ALGORITHM

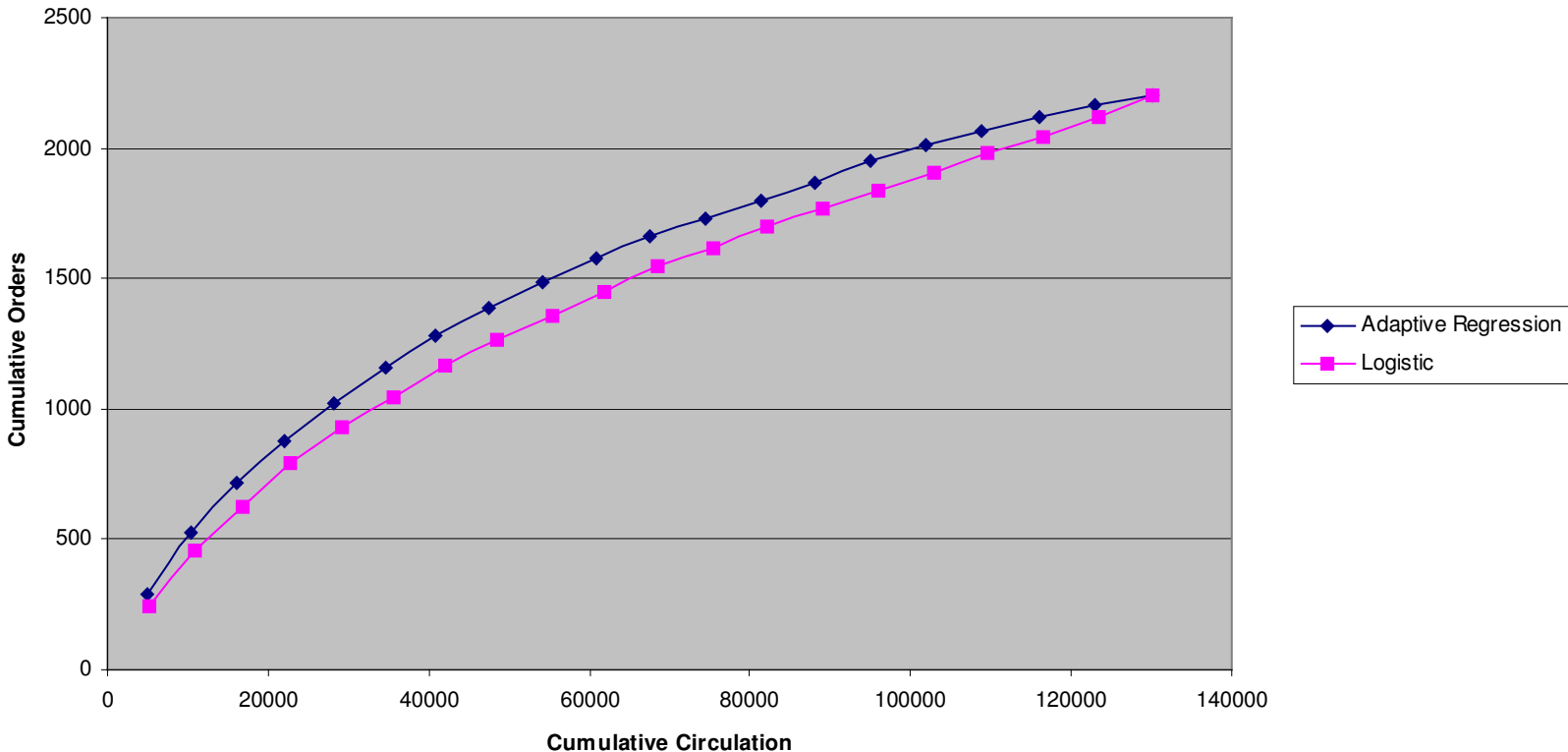
DIRECT MODELING ZIP MODEL

The ability of Adaptive Regression to fit nonlinear functions and interactions helped particularly in a recent project for a catalog company. They were looking for zip codes which responded better than expected to their rented lists. An obvious approach would be to look at the response rate by zip codes from their existing mailings and rank the zip codes by the actual results. Assuming a sufficient sample size, this would provide an excellent ranking. However, many of the zip codes had insufficient history, so the estimated response rate would not be stable. In these cases, we would naturally fall back on other things we know about these zip codes, i.e. zip demographics or proximity to a retail store. The best results would be obtained by using a combination of these approaches, weighting the actual observed response more for zip codes with more history. An Adaptive Regression model of degree 2 found the expected interactions with the count of historical data, and provided a better ranking of zip codes on the validation sample which had been held out for this purpose.

IMPROVING ON LOGISTIC REGRESSION

In a pilot project for a large mailer, we compared the results of Logistic Regression with those from Adaptive Regression. The data included a regression score developed using internal purchase history and a file of external behavior from a large data aggregator. Our goal was to develop a combined score. Preliminary explorations suggested that there was interaction between the internal score and the external data; namely, the external data told us more, and so raised the total score more, when the internal score was low. A straightforward logistic run showed model inadequacy – the logits were far from linear. An adaptive regression with maxdeg=2 performed much better. The adaptive regression algorithm automatically found the interactions we expected. The gains chart below compares the results. We held out a validation sample of data which was not used in the analysis; the results for this validation sample are reported in order to avoid an evaluation based on overfitted models. The gains chart is a display of the cumulative responses versus the cumulative catalogs mailed, mailing the best first according to each model. It is closely related to the ROC curve. A higher curve indicates a model with more discrimination.

Gains Chart - Validation Sample



BIBLIOGRAPHY

Cherkassky and Mulier, *Learning from Data*, Wiley 1998

Friedman, Jerome, Multivariate Adaptive Regression Splines, SLAC PUB-4960, 1990

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the author at:

David Katz
David Katz Consulting
1257 Siskiyou Blvd. #106
Ashland, Oregon 97520
(541) 482-4147
Email: david@davidkatzconsulting.com
Web: www.davidkatzconsulting.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.