Classification and non-linear Regression Methods.
SMART, MARS and other algorithms

JAVIER CABRERA
Department of Statistics and Biostatistics.
Rutgers U

1 The SMART algorithm for projection pursuit regression.

The original idea which gave rise to the SMART algorithm (Smooth Multiple Additive Regression Technique) was proposed by Friedman et al.\(^{15}\), under the name of Projection Pursuit Regression. The problem is to estimate a nonlinear mapping \(f : \mathbb{R}^m \rightarrow \mathbb{R}^n\) from data \(y\) on \(\mathbb{R}^m\), and \(x\) on \(\mathbb{R}^n\) satisfying the equation:

\[
y = f(x) + \epsilon.
\]

It is based in the following expansion of \(f\): For any mapping \(f : \mathbb{R}^m \rightarrow \mathbb{R}^n\) were \(m \geq n\), for \(M\) sufficiently large we can approximate \(f(x)\) by the sum

\[
f(x) \approx \sum_{i=1}^{M} \beta_i f_i(\alpha_i x),
\]

where the \(\beta_i\)'s are vectors on \(\mathbb{R}^n\), the \(\alpha_i\)'s are unit vectors in \(\mathbb{R}^m\) and the \(f_i\)'s are functions from \(\mathbb{R}\) to \(\mathbb{R}\).

The properties of this approximation were studied by Diaconis et al.\(^{11}\), where it was found that for many functions –e.g. polynomials– the equality is also true, but for others –e.g. \(h(x_1, x_2) = e^{x_1 x_2}\) – the sum needs an infinite number of terms in order to be exact.

The SMART estimator of the function \(f\) takes the form

\[
\hat{f}(X) = \hat{X} + \sum_{i=1}^{M} \beta_i f_i(\alpha_i X),
\]

where \(X\), \(\beta_i\), and \(\alpha_i\) are all in \(\mathbb{R}^{k+1}\), and \(f_i : \mathbb{R} \rightarrow \mathbb{R}\) are smooth functions.

The estimator \(\hat{f}\) is obtained by the method of least squares, which in our case corresponds to minimizing the criteria

\[
L_2 = \sum_{i=1}^{N} |Y_i - \hat{f}(X_i)|^2
\]

over the three groups of parameters \(\{\beta_i\}_i^M\), \(\{\alpha_i\}_i^M\), and \(\{f_i\}_i^M\).

A more general form of this criteria is the following:

\[
L^*_2 = E|X_* - \hat{f}(X)|^2
\]

where \(X_*\) is the observed image of \(X\), and the operator \(E\)– expected value – is the weighted average

\[
Ez = \frac{\sum \omega_i z_i}{\sum \omega_i}
\]
The weights \( \omega_i \) are arbitrary and should reflect the variability of the error term, which sometimes is associated with the size of the jump. A reasonable value for \( \omega_i \) is \( \tilde{\sigma}(Y)^{-1} \), where \( \tilde{\sigma}(Y) \) is an estimate of the standard deviation of \( \varepsilon \) at \( Y \) or proportional to it. In many cases \( \tilde{\sigma}(Y) \) is proportional to the change of \( f(X) - Y \), such that a rough estimator estimator could be obtained.

The algorithm proposed by Friedman\(^6\) for the estimation of \( f \) consist of a global loop, and at each pass the following steps are performed:

**SMART ALGORITHM**

**STEP 1:** For each \( j \) obtain \( \beta_j \) that minimizes \( L_2 \) given that all the other parameters \( \{ \beta_i \}_{i \neq j}, \{ \alpha_i \}, \) and \( \{ \hat{f}_i \} \) are known. This is the linear regression estimator.

**STEP 2:** The same as step 1 but for \( \alpha_j \). For each \( j \) find \( \alpha_j \) that minimizes \( L_2 \) given that all the other parameters \( \{ \alpha_i \}_{i \neq j}, \{ \beta_i \}, \) and \( \{ \hat{f}_i \} \) are known. In order to minimize \( L_2 \) we may write

\[
Y_i - \hat{f}(X_i) = Y_i - \bar{X} - \sum_{i=1}^{M} \beta_i \hat{f}_i(\alpha'_i X_i)
\]

\[
= R_{ij} - \beta_j \hat{f}_j(\alpha'_j X_i),
\]

where

\[
R_{ij} = Y_i - \bar{X} - \sum_{i \neq j}^{M} \beta_i \hat{f}_i(\alpha'_i X_i)
\]

The resulting optimization criteria is

\[
L_2(\alpha_j) = \sum_{i=1}^{N} |R_{ij} - \beta_j \hat{f}_j(\alpha'_j X_i)|^2,
\]

and the \( \alpha_j \)'s are obtained by a Gauss-Newton algorithm.

**STEP 3:** For each \( j \) find \( \hat{f}_j \) that minimizes \( L_2 \) given that all the other parameters \( \{ \alpha_i \}, \{ \beta_i \}, \) and \( \{ \hat{f}_i \}_{i \neq j} \) are known. In order to minimize \( L_2 \) we take from Eqn. (4)

\[
\hat{f}_j(\alpha'_j X_i) = \frac{E(\beta'_j R_{ij})}{\beta'_j \beta_j}
\]

were the expected valued is conditioned on the value of \( \alpha'_j X_i \). The expected value can be calculated using a local average or a spline.

These three steps are repeated until the change is less than \( \epsilon \) in two consecutive iterations.

The only task left is to choose \( M \), the number of terms. In the examples that are shown here we found that fixing \( M \) to 5 to 20-50 terms gives satisfactory results, however certain amount of trial is necessary.

2. **Criteria for model selection**

In this section we will discuss the basic criterias for model selection for conventional statistical models, which could be also applied for the SMART and MARS models. The computation of these criterias involves the estimation of the map which is computationally very slow, and hence it is not feasible to do it for too many models.
The basic criteria for model selection was proposed by Akaike in 1970\(^1\) – see reference for details~–, it is known as AIC(Akaike Information Criteria). An improved version of AIC– AICC where the last C stands for Corrected– was proposed by Hurvich e.a.\(^20\).

We will use the basic model

$$y_i = f_\theta(x_i) + \varepsilon_i$$

where \(\theta\) is the vector of parameters, and the \(\varepsilon_i\)’s are i.i.d. gaussians with zero mean and variance \(\sigma^2\).

The Kullback-Liebler information criteria measures the distance from any model to the true model \(F\) under the true model and it is given by

$$\Delta(\theta, \sigma^2) = E_F\{-2\log l_{\theta, \sigma^2}(y)\}$$

where

$$l_{\theta, \sigma^2}(y) = (2\pi\sigma^2)^{-\frac{N}{2}} Exp\{-\frac{1}{2\sigma^2}(y - h(\theta))'(y - h(\theta))\}$$

is the likelihood function of the data. The values of \(\theta\) and \(\sigma^2\) that maximize the likelihood function are called maximum likelihood estimators(MLE) and are represented by \(\hat{\theta}\) and \(\hat{\sigma}^2\).

Then, the following approximation holds.

$$\Delta(\hat{\theta}, \sigma^2) = N\log(\hat{\sigma}^2) + N\frac{\hat{\sigma}^2_0}{\hat{\sigma}^2} + \{\frac{1}{2\sigma^2}(\mu - h(\hat{\theta}))'(\mu - h(\hat{\theta}))\}$$

where where \(\theta_0\) and \(\sigma^2_0\) are the values of the parameters under the true model, and \(\mu = h(\theta_0)\).

A second approximation yields the two criterias.

$$AIC = N(\log(\hat{\sigma}^2) + 1) + 2(m + 1),$$

and

$$AICC = AIC + \frac{2(m + 1)(m + 2)}{N - m - 2},$$

where \(m\) is the number of constrains of the model and \(N\) is the number of observations.

These criterias require the calculation of \(\hat{\sigma}^2\) which itself requires the estimation of the rule.

3. **Multivariate Additive Regression Splines(MARS)**

We have seen that the main criticism of the SMART algorithm for fitting models is that is expensive computationally. Another criticism that has appeared in the literature is that it does not work very well if the function we are estimating is not very smooth.

In this section we will study a new statistical methodology for nonparametric function estimation known as Multivariate Additive Regression Splines(MARS) which was introduced by Friedman\(^17\). As we will see here MARS does overcome the problems of the SMART algorithm.

The ideas of MARS come from the one dimensional case, where the method of splines is the most widely used method for function estimation. Consider the one dimensional model

$$y = f(x) + \varepsilon.$$ \hspace{1cm} (5)

The method of splines consists of dividing the domain of \(x\) in \(K + 1\) regions by the ordered points \(t_1, \ldots, t_K\), and consider the basis of functions \(
\{x^j\}_{j=1}^{K}, \{x - t_j\}_{j=1}^{K}\) which will be denote by \(\{B_j(x)\}_{j=1}^{K+q}\). Then the spline estimator of \(f\) is

$$\hat{f}(x) = \sum_{i=1}^{K+q} a_i B_i(x) ,$$
where the $a_i$'s are fitted by least squares.

The generalization of this method to dimensions greater than one is conceptually trivial, and it works in two dimensions, but for dimensions greater than two is computationally impractical in most situations.

In the remaining of this section we will describe the algorithm for recursive partitioning which is the basis for MARS, we will describe the MARS algorithm, and we will show examples.

Recursive partitioning is a method of estimating $f$, which consists of fitting a step function by sequentially partitioning the domain of $f$ into regions and assigning a value to each region. It was introduced as part of the methodology of classification and regression trees (CART\textsuperscript{5}). The estimator of $f$ is

$$
\hat{f}(x) = \sum_{i=1}^{m} a_i B_i(x),
$$

where the $B_i$ are indicator functions over sets that make a partition of $a$ the domain of $f$. The criteria that will be optimized is

$$
L_2(\hat{f}) = \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2.
$$

The parameters $\{a_i\}$ are estimated by least squares. The algorithm goes as follows

**MARS ALGORITHM**

**STEP 1:** Set $m = 2$ and find the component of $x$, namely $x^*$, a fixed value $z_1$ and the basis functions

$$
B_1(x) = H(-(x^* - z_1))
$$

$$
B_2(x) = H(+ (x^* - z_1))
$$

that minimize $L_2(\hat{f})$. $H(x)$ is the heavy side function which takes value one for non-negative argument and zero otherwise.

**STEP i:** Set $m = i + 1$ and find the component of $x$, namely $x^*$, $z_i$, and replace the basis function $B_j$ by

$$
H(+ (x^* - z_i)) \ B_j(x)
$$

and add the basis function

$$
B_{i+1}(x) = H(- (y^* - z_i))
$$

that minimize $L_2(\hat{f})$.

Figure 13 shows two graphs of a simple recursive partitioning. The left one side graph shows the tree representation of the recursive partitioning and the right hand side graph shows the actual partitioning of the domain of dimension 2.

The MARS algorithm is essentially the same as recursive partitioning but there are a few modifications.

(i) Replace the twos-sided basis functions $H(+ -(x^* - z*))$ by two-sided truncated power basis functions

$$
[+ -(x^* - z*)]^q +
$$
where the $[x]$ gives $x$ for non-negative values of $x$ and zero otherwise.

(ii) Intermediate functions are not replaced.

(iii) Products are restricted to factors with different variables.

If we are fitting a map in $k$ dimensions we apply the algorithm to each of the coordinates. This is problematic because MARS uses a large amount of memory for each fit. Work in progress will introduce a modification to the algorithm that would allow us to fit all the responses at the same time.

The most important advantages of MARS over SMART are first of all that will give better result when the true map is not very smooth, and in addition the computing time used by MARS is much less than the one taken by SMART.

3. Dinamical System. Comparison of orbits fitted with MARS and SMART

The algorithm stops after $m$ steps or after certain stopping rule is satisfied and the final coefficients are obtained by least squares. The stopping rule proposed by Friedman is based on cross-validation. We refer to the work of Friedman for a more detail discussion of the algorithm and examples in other areas. De Vaux e.a did a comparison of MARS with Neural Nets in several examples for very different applications, and for each of them MARS always seemed to give better results.

Figure 14 shows the orbits generated from the rule fitted with MARS to the Lorenz time-series. The one on the left was obtained from the original time-series, while the one on the right was obtained from a new time-series created by adding observational error from a uniform distribution $(\frac{-1}{2}, \frac{1}{2})$ to the Lorenz series. In both examples, the three dimensional embedding with delays 0, 1, and 2 was selected for the reconstruction of the attractor.

The graph on the left of Figure 14 shows an orbit very similar to the original data that was shown in Figure 6.1. This is an indication of an excellent reconstruction. However the graph on the right of Figure 14 shows the effect produced by moderately large noise to the MARS estimator. It is possible that using more data, and tuning up the estimation better we would be able to improve the estimator but it is also possible that MARS needs to be modified in order to account for errors in both sides of the model equation, namely errors and variables.

Figure 15 shows the orbits generated with MARS, for the Couette Flow series. The delays that were selected for the reconstruction were 2, 5, 8, 12, 18, 24, and 29. The orbit produced by MARS is a very simplified version of the original orbit, although it preserves the main feature of the orbit.

Figure 16 shows the corresponding orbit for the EEG series. Delays 6, 12, 18, 24, 30, 33, and 35 were used for the reconstruction of the attractor.

The orbit produced by MARS is a very simplified version of the original orbit because of the large noise component of the time-series. If we compare Figure 16 with the the graph on the right of Figure 14 we see some similarities, and perhaps this is also the effect of not accounting for errors and variables.
References.