1. Introduction

A generalized additive model (GAM) is a GLM in which the linear predictor depends linearly on predictor variables and smooth functions of predictor variables. We will follow closely the implementation in [1]. Another good resource for GAM can be found in [2]

2. A Simple Linear Model

Consider n observations, x_i with response variable y_i where y_i is an observation on random variable Y_i. Let $u_i \equiv \mathrm{E}(Y_i)$. Assuming a linear relationship between the predictor variables and the response, the following relationship exists between x_i and Y_i as:

$$Y_i = u_i + \epsilon_i \text{ where } u_i = \beta_1 x_i + \beta_0$$

and β_0, β_1 are unknown parameters, ϵ_i are i.i.d zero mean variables with variances δ^2. We already know how to estimate β_0, β_1 using GLM. The matrix X that contains $\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \dots & \dots \end{bmatrix}$ is referred to as the model matrix.

3. A Simple Linear GAM Model

Using the same observations as in section 2, a linear GAM model can be:

$$Y_i = f(x_i) + \epsilon_i \text{ where } f(x_i) = \sum_{j=1}^{k} b_j(x_i)\,\beta_j + \beta_0$$

Again, $\beta = [\beta_0, \beta_1, \dots, b_k]$ is an unknown parameter vector which can also be estimated using GLM. This can be done by using $[b_1(x_i), b_2(x_i), \dots, b_K(x_i)]$ as the predictor variables instead of x_i. The model matrix X in this case will contain

$$\begin{bmatrix} b_1(x_1) & b_2(x_1) \dots & 1 \\ b_1(x_2) & b_2(x_2) \dots & 1 \\ \dots & \dots & \dots \end{bmatrix}$$

Here, we are basically estimating $f(x_i)$ using a set of basis functions $\{b_1(x_i), b_2(x_i), \dots, b_K(x_i)\}$ where k is the number of basis functions used. Note that, for each predictor variables, we get to decide the types of basis functions and the number of basis functions that we would like to use to best generate a GAM.

4. A Simple Piecewise Linear Basis Function

To understand the role of basis functions, we are going to use a linear tent function.

Using piecewise basis functions, we need to pay attention to the locations of the function's derivative discontinuities, that is by the locations at which the linear pieces join up. These locations are referred to as the knots and denoted by $\{x_i^* : j = 1, \dots, K\}$ and suppose that the knots are sorted meaning that $x_i^* > x_{i-1}^*$. Then for $j = 2, \dots, K-1$, we have basis function $b_j(x)$ defined as

$$b_j(x) = \begin{cases} (x - x_{j-1}^*) \Big/ (x_j^* - x_{j-1}^*), x_{j-1}^* \le x \le x_j^* \\ (x_{j+1}^* - x) \Big/ (x_{j+1}^* - x_j^*), x_j^* < x \le x_{j+1}^* \\ 0, \quad \text{otherwise} \end{cases}.$$

For $k = 1$, we have

$$b_1(x) = \begin{cases} (x_2^* - x) \Big/ (x_2^* - x_1^*), x \le x_2^* \\ 0, \quad \text{otherwise} \end{cases}.$$

For $k = K$, we have

$$b_K(x) = \begin{cases} (x - x_{K-1}^*) \Big/ (x_K^* - x_{K-1}^*), x > x_{K-1}^* \\ 0, \quad \text{otherwise} \end{cases}.$$

5. Using Piecewise Tent Function to Approximate One Predictor Variable

To illustrate how we can use the piecewise tent functions to approximate a predictor variable, let's use the following example for a predictor:

- Predictor value goes from 0.0 to 1.0;
- Set $K = 10$ to use 10 piecewise tent functions;
- The knots are located at 0, 1/9, 2/9, 3/9, …, 8/9, 1.

The basis function values are plotted in Figure 1. Note that there are 10 basis functions. The basis function values overlap with its neighbors from the left and the right except for the first and the last basis functions.

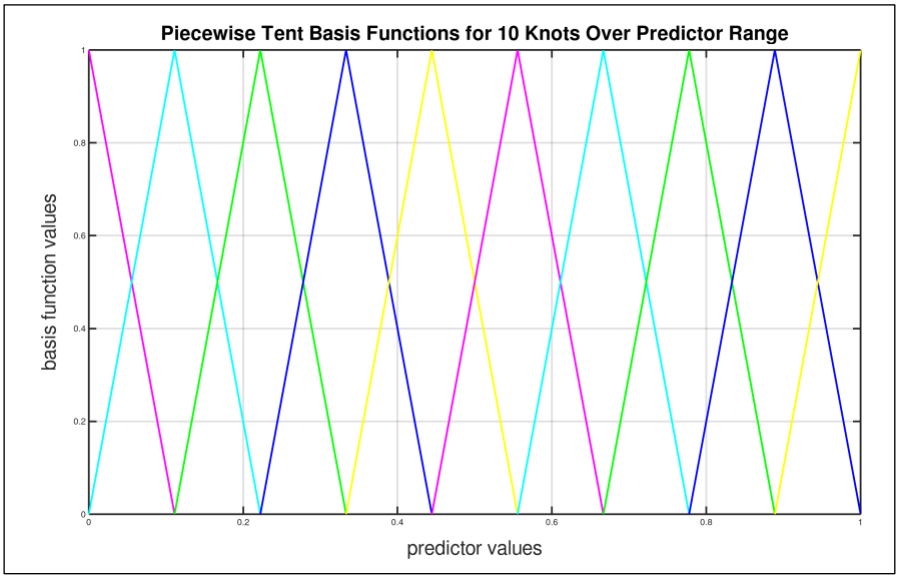

Figure 1. Piecewise tent basis functions

For simplicity, let's assume that we only have 21 predictor values uniformly spreading over the range from 0 to 1 with values 0, 0.05, 0.1, 0.15, …, 1.0. The next task is to express each x_j using the set of 10 basis function values. This means that for every value of x_j, we will obtain 10 values, each one corresponding to the contribution from one of the basis function.

For the predictor value at 0, the only basis function that contribute to its value is the first one. All the other basis function contributes 0 to the predictor value. Hence, for $x_j = 0$, the vector corresponding to all basis functions will have the following values: $\{1,0,0,0,0,0,0,0,0,0\}$ because the first basis function value is 1 at $x_j = 0$ (substitute $x = 0$ to the first basis function $b_1(x) = \left(\frac{1}{9} - x\right) \Big/ \left(\frac{2}{9} - \frac{1}{9}\right)$).

For predictor value 0.05, only the first and second basis functions contribute to its value while the other basis functions are 0 at 0.05. The value of the first basis function is 0.55 (substitute $x = 0.05$ to the first basis function $b_1(x) = \left(\frac{1}{9} - x\right) \Big/ \left(\frac{2}{9} - \frac{1}{9}\right)$). The value of the second basis function at 0.05 is 0.45 (substitute $x = 0.05$ to the second basis function $b_2(x) = x \Big/ \left(\frac{1}{9}\right)$).

Hence, for $x_j = 0.05$, the vector corresponding to all basis function is $\{0.55,0.45,0,0,0,0,0,0,0,0\}$.

I have calculated the expanded basis function vector for all predictor values and they can be found in following table.

x_j	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
0	1	0	0	0	0	0	0	0	0	0
0.05	0.55	0.45	0	0	0	0	0	0	0	0
0.1	0.1	0.9	0	0	0	0	0	0	0	0
0.15	0	0.65	0.35	0	0	0	0	0	0	0
0.2	0	0.2	0.8	0	0	0	0	0	0	0
0.25	0	0	0.75	0.25	0	0	0	0	0	0
0.3	0	0	0.3	0.7	0	0	0	0	0	0
0.35	0	0	0	0.85	0.15	0	0	0	0	0
0.4	0	0	0	0.4	0.6	0	0	0	0	0
0.45	0	0	0	0	0.95	0.05	0	0	0	0
0.5	0	0	0	0	0.5	0.5	0	0	0	0
0.55	0	0	0	0	0.05	0.95	0	0	0	0

0.6	0	0	0	0	0	0.6	0.4	0	0	0
0.65	0	0	0	0	0	0.15	0.85	0	0	0
0.7	0	0	0	0	0	0	0.7	0.3	0	0
0.75	0	0	0	0	0	0	0.25	0.75	0	0
0.8	0	0	0	0	0	0	0	0.8	0.2	0
0.85	0	0	0	0	0	0	0	0.35	0.65	0
0.9	0	0	0	0	0	0	0	0	0.9	0.1
0.95	0	0	0	0	0	0	0	0	0.45	0.55
1	0	0	0	0	0	0	0	0	0	1

6. Spline functions

It has been proven in [2] that the natural cubic splines are the smoothest interpolators. For a set of points $\{x_i, y_i : i = 1, \dots, n\}$ where $x_i \leq x_{i+1}$. The natural cubic spline, $g(x)$, interpolating these points, is a function made up of sections of cubic polynomial, one for each $[x_i, x_{i+1}]$. They are joined up together so that the whole spline is continuous to second derivative, while $g(x_i) = y_i$ and $g''(x_1) = g''(x_n) = 0$. To ensure smooth function, we can add a penalty function $J(f) = \int_{x_1}^{x_n} (f''(x))^2 dx$ to the actual objective function that we are trying to optimize. The rationality behind this penalty is that the second derivative of a function measures the gradient change. For functions that wriggles a lot, it will have a higher second derivative magnitude. For a straight line which does not wriggle at all, the second derivative is zero.

6.1. Cubic regression splines

Following the implementation in [1], we have implemented the cubic regression splines for a single predictor variable. This approach defines the splines in terms of its values at the knots. Next, we define a cubic spline function, $f(x)$, with k knots, $x_1, x_2, \dots, x_k$. Let $\beta_j = f(x_j)$ and $\delta_j = f''(x_j) = \frac{d^2 f(x_j)}{d^2 x}$. The splines can be written as

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\delta_j + c_j^+(x)\delta_{j+1} \text{ for } x_j \leq x \leq x_{j+1}$$

where

- $a_j^-(x) = (x_{j+1} - x)/h_j$, $a_j^+(x) = (x - x_j)/h_j$
- $c_j^-(x) = \left[\frac{(x_{j+1}-x)^3}{h_j} - h_j(x_{j+1} - x)\right]/6$, $c_j^+(x) = \left[\frac{(x-x_j)^3}{h_j} - h_j(x - x_j)\right]/6$

Note that in order to ensure smooth fitting functions at the knots, the spline must be continuous to second derivative, at the x_j, and should have zero second derivative at x_1 and x_k. It can be shown that $B\delta^- = D\beta$ (I have proved this and will add it to Appendix later) where

- $\delta^- = (\delta_2, \delta_3, \dots, \delta_{k-1})^T$ since $\delta_1 = \delta_k = 0$;

- $B = \begin{cases} B_{i,i} = \frac{(h_i + h_{i+1})}{3}, i = 1 \dots k-2 \\ B_{i,i+1} = \frac{h_{j+1}}{6}, i = 1 \dots k-3 \\ B_{i+1,i} = \frac{h_{j+1}}{6}, i = 1 \dots k-3 \end{cases}$;
- $D = \begin{cases} D_{i,i} = \frac{1}{h_i}, i = 1 \dots k-2 \\ D_{i,i+1} = 1/h_i - 1/h_{j+1}, i = 1 \dots k-2 \\ D_{i,i+2} = 1/h_{j+1}, i = 1 \dots k-2 \end{cases}$;

Let $BinvD = B^{-1}D$ and let $F = \begin{bmatrix} 0 \\ BinvD \\ 0 \end{bmatrix}$, the spline can be rewritten entirely in terms of β as

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)F_j\beta + c_j^+(x)F_{j+1}\beta \text{ for } x_j \le x \le x_{j+1}$$

which can be re-written as $f(x_i) = \sum_{j=1}^{k} b_j(x_i)\,\beta_j + \beta_0$ where $b_j(x_i)$ are the basis functions and $\beta_0, \beta_1, \dots, \beta_k$ are the unknown parameters which can be estimated using GLM. In addition, the penalty term added to the final objective function can be derived to be:

$$\int_{x_1}^{x_k} (f''(x))^2 dx = \beta^T D^T B^{-1} D\beta = \beta^T D^T BinvD\beta = \beta^T S_{orig}\beta$$

where $S_{orig} = D^T B^{-1} D$. However, in R, more scaling is performed on S_{orig} as follows:

- $maXX = \max(abs(row\ sum\ of\ model\ matrix\ X\ excluding\ column\ of\ ones))$;
- $maS = \max\left(abs\left(column\ sum\ of\ penalty\ matrix\ S_{orig}\right)\right)/maXX$;
- final penalty matrix $S = S_{orig}/\,maS$.

For linear regression models, the final objective function to minimize is

$$\sum_{i=1}^{n}\left(y_i - \left(\sum_{j=1}^{k} b_j(x_i)\,\beta_j + \beta_0\right)\right)^2 + \lambda\beta^T S\beta$$

Note that λ will be another parameter for the user to choose using gridsearch. In future release, we may use cross-validation to automatically choose λ.

Hence, at this point, we can call our GLM. However, we still need to add the contribution of the penalty term to the gradient and hessian calculation.

6.2. Future Spline Functions

7. General GAM

In a general GAM, using the GLM jargon, the link function can be constructed using a mixture of predictor variables and smooth functions of predictor variables as follows:

$$g(u_i) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_m x_{mi} + \sum_{j=1}^{k_1} b_j^1(x_{li})\beta_{m+j} + \cdots + \sum_{j=1}^{k_q} b_j^q(x_{li})\beta_{m+k_1+\cdots+k_{q-1}+j}$$

This is the GAM we implemented in H2O. However, with multiple predictor variables in any form, we need to resolve the identifiability problems by adding identifiability constraints.

7.1. Identifiability Constraints

Consider GAM with multiple predictor smooth functions like the following:

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \epsilon_i$$

The model now contains more than one function introduces an identifiability problem: f_1 and f_2 are each only estimable to within an additive constant. This is due to the fact that $f_1(x_i) + f_2(v_i) = (f_1(x_i) + C) + (f_2(v_i) - C)$. Hence, identifiability constraints have to be imposed on the model before fitting to avoid the identifiability problem. The following sum-to-zero constraints are implemented in H2O:

$$\sum_{i=1}^{n} f_p(x_i) = 0 = 1^T f_p$$

where 1 is a column vector of 1 and f_p is the column vector containing $f_p(x_1), \ldots, f_p(x_n)$. To apply the sum-to-zero constraints, a Householder transform is used. Refer to [1] for details. This transform is applied to each basis function of any predictor column we choose on its own.

7.2. Sum-to-zero constraints implementation

Let X be the model matrix that contain the basis functions of one predictor variable, the sum-to-zero constraints required that

$$1^T f_p = 0 = 1^T X\beta$$

where β contains the coefficients relating to the basis functions of that particular predictor column. The idea is to create a $k\ by\ (k-1)$ matrix Z such that $\beta = Z\beta_z$, then $1^T X\beta = 0$ for any β_z. To see how this works, let's go through the following derivations:

- With Z, we are looking at $0 = 1^T X\beta = 1^T XZ\beta_z$;
- Let $C = 1^T X$, then the QR decomposition of $C^T = U\begin{bmatrix} P \\ 0 \end{bmatrix}$ where C^T is of size $k \times 1$, U is of size $k \times k$, P is of size 1×1;
- U is an orthogonal matrix and let's divide it into two parts as $U = [D \quad Z]$ where D is of size $k \times 1$ and Z is of size $k \times (k-1)$;
- Substitute everything back to $1^T XZ\beta_z = [P^T \quad 0]\begin{bmatrix} D^T \\ Z^T \end{bmatrix} Z\beta_z = [P^T \quad 0]\begin{bmatrix} D^T Z\beta_z \\ Z^T Z\beta_z \end{bmatrix} = P^T D^T Z\beta_z + 0Z^T Z\beta_z$=0 since $D^T Z = 0$.

7.3. Generating the Z matrix

One Householder reflection is used to generate the Z matrix. To create the Z matrix, we need to calculate the QR decomposition of $C^T = X^T 1$. Since C^T is of size $k \times 1$, the application of one Householder reflection will generate $HC^T = \begin{bmatrix} R \\ 0 \end{bmatrix}$ where R is of size 1×1. This implies that $H = Q^T = Q$ since HouseHolder reflection matrix is symmetrical. Hence computing XZ is equivalent to computing XH and dropping the first column.

7.4. Generating the HouseHolder reflection matrix H

Let $\bar{x} = X^T 1$ and $\bar{x}' = \begin{bmatrix} \alpha\|\bar{x}\| \\ 0 \end{bmatrix}$ $\alpha = \begin{cases} -1 \ if \ x[0] > 0 \\ 1 \ otherwise \end{cases}$, then $H = (I - 2\frac{uu^T}{(u^T u)})$ and $u = \bar{x} - \bar{x}'$.
The α is introduced for floating-point arithmetic.

7.5. Estimation of GAM coefficients with identifiability constraints

The following procedure is used to estimate the GAM coefficients:

- Generating Z matrix for each predictor column that uses smooth functions;
- Generate new model matrix for each predictor column smooth function as $X_Z = XZ$, new penalty function $\beta_Z^T Z^T S Z \beta_Z$.
- Call GLM using model matrix X_Z, penalty function $\beta_Z^T Z^T S Z \beta_Z$ to get coefficient estimates of β_Z;
- Convert β_Z to β using $\beta = Z\beta_Z$ and performing scoring with β and the original model matrix X.

References

[1]. Simon N. Wood, Generalized Additive Models An Introduction with R, Texts in Statistical Science, CRC Press, Second Edition.

[2].T.J. Hastie, R.J. Tibshirani, Generalized Additive Models, Chapman and Hall, First Edition, 1990.