MATH 829: Introduction to Data Mining and Analysis
Splines

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

March 16, 2016

Transforming data

- Very often the relationship between variables is not linear.
- We saw before that transformations of the features can be used.
- If $h_m : \mathbb{R}^p \to \mathbb{R}$, then we can use the model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X).$$

Transforming data

- Very often the relationship between variables is not linear.
- We saw before that transformations of the features can be used.
- If $h_m : \mathbb{R}^p \to \mathbb{R}$, then we can use the model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X).$$

Common transformations:

1. $h_m(X) = X_m$ (Usual linear regression).
2. $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ (Taylor polynomials).
3. $h_m(X) = \log(X_j), \sqrt{X_j}$.
4. $h_m(X) = I(L_m \leq X_k < U_m)$ (Indicator functions in some intervals).

Transforming data

- Very often the relationship between variables is not linear.
- We saw before that transformations of the features can be used.
- If $h_m : \mathbb{R}^p \to \mathbb{R}$, then we can use the model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X).$$

Common transformations:

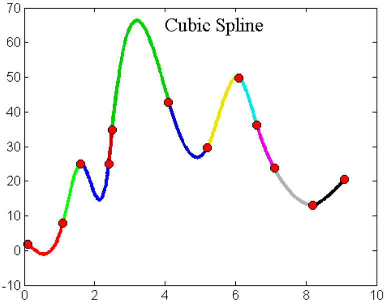
1. $h_m(X) = X_m$ (Usual linear regression).
2. $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ (Taylor polynomials).
3. $h_m(X) = \log(X_j), \sqrt{X_j}$.
4. $h_m(X) = I(L_m \leq X_k < U_m)$ (Indicator functions in some intervals).

Note:

- Need a large sample size to include many functions.
- Risk of over-fitting when including too many functions.

Splines

Splines are piecewise polynomials with a given number of continuous derivatives.



For example, *cubic* splines are degree 3 polynomials pasted together to get 2 continuous derivatives.

Splines (cont.)

More generally, given knots $t_0 < t_1 < \cdots < t_k$, a spline of degree n is a piecewise polynomial

$$S(x) := \begin{cases} S_0(x) & t_0 \leq x \leq t_1 \\ S_1(x) & t_1 \leq x \leq t_2 \\ \vdots & \\ S_{k-1}(x) & t_{k-1} \leq x \leq t_k \end{cases}$$

such that

Splines (cont.)

More generally, given knots $t_0 < t_1 < \cdots < t_k$, a spline of degree n is a piecewise polynomial

$$S(x) := \begin{cases} S_0(x) & t_0 \le x \le t_1 \\ S_1(x) & t_1 \le x \le t_2 \\ \vdots & \\ S_{k-1}(x) & t_{k-1} \le x \le t_k \end{cases}$$

such that

1. $S_i(x)$ is a polynomial of degree n.
2. $S(x)$ is $n-1$ times continuously differentiable.

Splines (cont.)

More generally, given knots $t_0 < t_1 < \cdots < t_k$, a spline of degree n is a piecewise polynomial

$$S(x) := \begin{cases} S_0(x) & t_0 \leq x \leq t_1 \\ S_1(x) & t_1 \leq x \leq t_2 \\ \vdots & \\ S_{k-1}(x) & t_{k-1} \leq x \leq t_k \end{cases}$$

such that

1. $S_i(x)$ is a polynomial of degree n.
2. $S(x)$ is $n-1$ times continuously differentiable.

- Most commonly used value is $n = 3$ (cubic splines).
- Said to be the smallest n for which it is impossible to detect the location of the knots by eye.

Splines (cont.)

More generally, given knots $t_0 < t_1 < \cdots < t_k$, a spline of degree n is a piecewise polynomial

$$S(x) := \begin{cases} S_0(x) & t_0 \le x \le t_1 \\ S_1(x) & t_1 \le x \le t_2 \\ \vdots & \\ S_{k-1}(x) & t_{k-1} \le x \le t_k \end{cases}$$

such that

1. $S_i(x)$ is a polynomial of degree n.
2. $S(x)$ is $n-1$ times continuously differentiable.

- Most commonly used value is $n = 3$ (cubic splines).
- Said to be the smallest n for which it is impossible to detect the location of the knots by eye.
- A *natural cubic spline* imposes the supplementary conditions that the spline is linear beyond the boundary knots.

Basis for cubic splines

Cubic splines basis: With 2 knots ξ_1, ξ_2:

$$h_1(X) = 1, \qquad h_3(X) = X^2, \qquad h_5(X) = (X - \xi_1)_+^3,$$
$$h_2(X) = X, \qquad h_4(X) = X^3, \qquad h_6(X) = (X - \xi_2)_+^3.$$

Basis for cubic splines

Cubic splines basis: With 2 knots ξ_1, ξ_2:

$$\begin{aligned} h_1(X) &= 1, & h_3(X) &= X^2, & h_5(X) &= (X-\xi_1)_+^3, \\ h_2(X) &= X, & h_4(X) &= X^3, & h_6(X) &= (X-\xi_2)_+^3. \end{aligned}$$

More generally, with M knots, add $(X-\xi_3)_+^3, \ldots, (X-\xi_M)_+^3$.

Basis for cubic splines

Cubic splines basis: With 2 knots ξ_1, ξ_2:

$$h_1(X) = 1, \qquad h_3(X) = X^2, \qquad h_5(X) = (X - \xi_1)_+^3,$$
$$h_2(X) = X, \qquad h_4(X) = X^3, \qquad h_6(X) = (X - \xi_2)_+^3.$$

More generally, with M knots, add $(X - \xi_3)_+^3, \ldots, (X - \xi_M)_+^3$.

Natural cubic splines basis: With M knots

$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2}(X) = d_k(X) - d_{M-1}(x),$$

where

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_M)_+^3}{\xi_M - \xi_k}.$$

Basis for cubic splines

Cubic splines basis: With 2 knots ξ_1, ξ_2:

$$h_1(X) = 1, \qquad h_3(X) = X^2, \qquad h_5(X) = (X - \xi_1)_+^3,$$
$$h_2(X) = X, \qquad h_4(X) = X^3, \qquad h_6(X) = (X - \xi_2)_+^3.$$

More generally, with M knots, add $(X - \xi_3)_+^3, \ldots, (X - \xi_M)_+^3$.

Natural cubic splines basis: With M knots

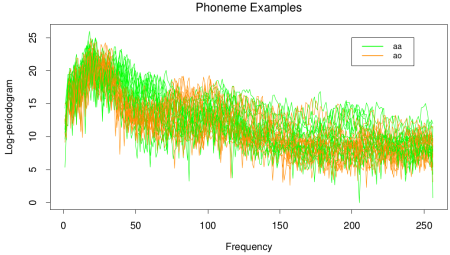
$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2}(X) = d_k(X) - d_{M-1}(x),$$

where

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_M)_+^3}{\xi_M - \xi_k}.$$

- Can include spline basis in linear regression.
- Not always obvious how to choose the knots.
- Natural splines can be used to avoid the erratic behavior of polynomials beyond the knots.

Example: Phoneme Recognition (ESL, Example 5.2.3)

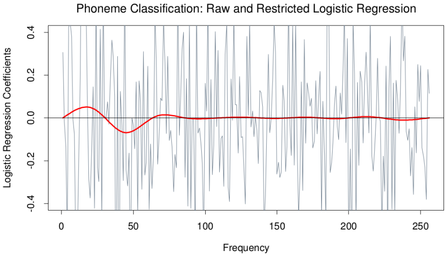


$X = X(f)$
$f =$ frequency.

$$\log \frac{P(aa|X)}{P(ao|X)} = \sum_{i=1}^{256} X(f_i)\beta_i = X^T\beta.$$

15 examples each of the phonemes "aa" and "ao" sampled from a total of 695 "aa"s and 1022 "ao"s.

Phoneme recognition (cont.)



	Raw	Regularized
Training error	0.080	0.185
Test error	0.255	0.158

Logistic regression coefficients, and smoothed version with natural cubic splines.

$$\beta(f) = \sum_{i=1}^{M} h_m(f)\theta_m = \mathbf{H}\theta,$$

where $\mathbf{H}$ is a $p \times M$ matrix of spline functions.
Now, note that

$$X^T\beta = X^T\mathbf{H}\theta.$$

Letting $x^* = \mathbf{H}^T x$, we can therefore fit the logistic regression on the *smoothed* inputs.

Preprocessing data

- In the previous example, we fitted a logistic regression to transformed inputs.
- Non-linear transformations are very useful for *preprocessing* data.
- Powerful method for improving the performance of a learning algorithm.

Smoothing splines

- Splines can be very useful.
- Problem: How to choose the knots in an *optimal* way?

Smoothing splines avoid this problem.

Smoothing splines

- Splines can be very useful.
- Problem: How to choose the knots in an *optimal* way?

Smoothing splines avoid this problem.
Smoothing splines: Find a function $f \in C^2$ the minimizes

$$\mathrm{RSS}(f, \lambda) := \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int f''(t)^2 \, dt \qquad (\lambda > 0).$$

Smoothing splines

- Splines can be very useful.
- Problem: How to choose the knots in an *optimal* way?

Smoothing splines avoid this problem.
Smoothing splines: Find a function $f \in C^2$ the minimizes

$$\mathrm{RSS}(f, \lambda) := \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int f''(t)^2 \, dt \qquad (\lambda > 0).$$

- First term controls closeness to data.
- Second term controls curvature of the function.

Smoothing splines

- Splines can be very useful.
- Problem: How to choose the knots in an *optimal* way?

Smoothing splines avoid this problem.
Smoothing splines: Find a function $f \in C^2$ the minimizes

$$\mathrm{RSS}(f, \lambda) := \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int f''(t)^2 \, dt \qquad (\lambda > 0).$$

- First term controls closeness to data.
- Second term controls curvature of the function.

Note:

- If $\lambda = 0$: any function that interpolates the data works.
- As $\lambda = \infty$: least squares fit.

Smoothing splines (cont)

- To compute a smoothing spline, we need to optimize on an infinite dimensional space of functions.

Smoothing splines (cont)

- To compute a smoothing spline, we need to optimize on an infinite dimensional space of functions.
- Remarkably, it can be shown that the problem has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the x_i , $i = 1, \ldots, N$. (See next homework).

Smoothing splines (cont)

- To compute a smoothing spline, we need to optimize on an infinite dimensional space of functions.
- Remarkably, it can be shown that the problem has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the x_i , $i = 1, \ldots, N$. (See next homework).
- The penalty term translates to a penalty on the spline coefficients, which are shrunk some of the way toward the linear fit.

Nonparametric logistic regression

Consider the logistic regression problem with a binary output.

$$\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = f(x).$$

Equivalently,

$$P(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}}.$$

Nonparametric logistic regression

Consider the logistic regression problem with a binary output.

$$\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = f(x).$$

Equivalently,

$$P(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}}.$$

Before, we used a linear model for f, and chose the coefficients using maximum likelihood.

Nonparametric logistic regression

Consider the logistic regression problem with a binary output.

$$\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = f(x).$$

Equivalently,

$$P(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}}.$$

Before, we used a linear model for f, and chose the coefficients using maximum likelihood.

Consider the *penalized* log-likelihood criterion:

$$\begin{aligned} l(f;\lambda) &= \sum_{i=1}^{n} [y_i \log p(x_i) + (1-y_i)\log(1-p(x_i))] - \frac{1}{2}\lambda \int f''(t)\ dt \\ &= \sum_{i=1}^{n} [y_i f(x_i) - \log(1+e^{f(x_i)})] - \frac{1}{2}\lambda \int f''(t)\ dt. \end{aligned}$$

One can show that the optimal f is a natural spline with knots at the unique x_is (see ESL for more details).