Splines and basis expansion (week 3?)

31 Jan 2023

Table of contents

linear basis expansion

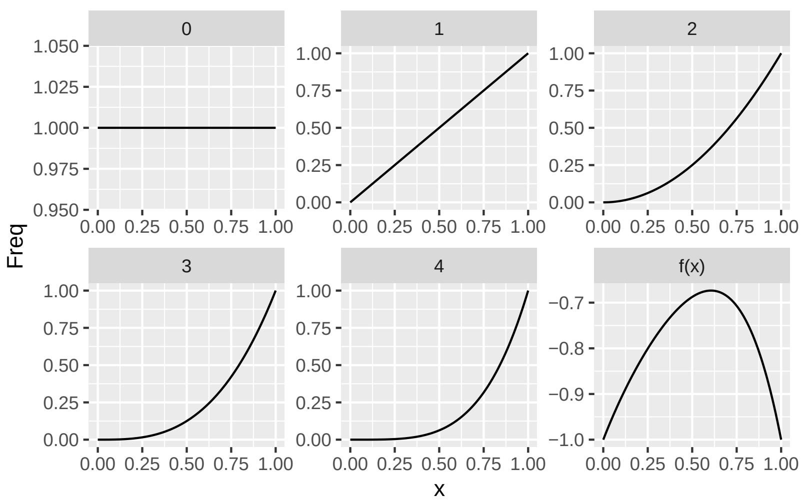
- transformations of various kinds
- quadratic expansion
- nonlinear transformations
- indicator variables

Select or regularize from the expanded set.

polynomial basis

- polynomial basis: $y_i = \sum_{j=0}^{n} \beta_j x_i^j$

```r
library(ggplot2)
x <- seq(0, 1, length = 101)
n <- 4
y <- sapply(0:n, \(j) x^j)
beta <- c(-1, 1, -1, 1, -1)
y <- cbind(y, fx = y %*% beta)
dimnames(y) <- list(x = x, j = c(0:n, "f(x)"))
yy <- as.data.frame(as.table(y))
yy$x <- as.numeric(as.character(yy$x))
ggplot(yy, aes(x, Freq)) + geom_line() + facet_wrap(~j, scale = "free")
```



piecewise polynomial bases

- constant, linear, continuous
- basis functions
- translate from x_i to columns of $\mathbf{X}$

splines

- **piecewise** polynomials with continuity/smoothness constraints
- very useful for function approximation
- convert a single numeric predictor into a flexible basis
- efficient

- with multiple predictors, consider **additive models**
- handle interactions (multidim smooth surfaces) *if reasonably low-dimensional*: tensor products etc.

spline terminology

- **knots**: breakpoints (boundary, interior)
- order-M (ESL): continuous derivatives up to order $M-2$ (cubic, $M=4$)
- typically $M=1$, 2, 4
- number of knots = df (degrees of freedom) -1 -intercept

truncated power basis

- $X^0 \dots X^n$
- remaining columns are $(x-\xi_\ell)+^{M-1}$ where ℓ are the *interior knots*

B-spline basis

- splines of a given order with *minimal support* (i.e., local)
- basis functions defined by recursion (not pretty)
- convenient for regression splines (see below)

natural cubic splines

- linear constraints beyond boundary knots (so 2d and 3d derivatives are 0 at the boundaries)

```r
library(splines)
bb <- bs(1:20, df = 5)
attributes(bb)[c("degree", "knots", "Boundary.knots")]
```

```
$degree
[1] 3

$knots
```

```
33.33333% 66.66667% 
 7.333333 13.666667 

$Boundary.knots
[1]  1 20
```

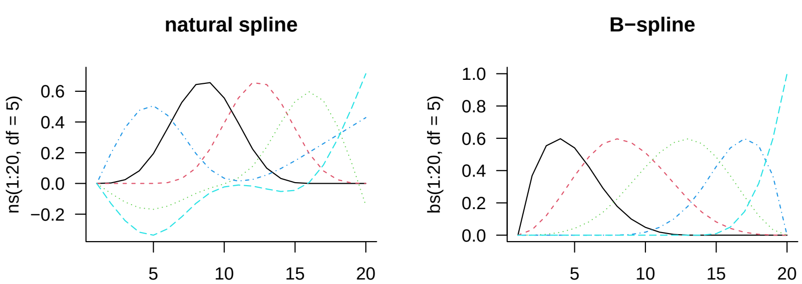
```
nn <- ns(1:20, df = 7)
attributes(nn)[c("degree", "knots", "Boundary.knots")]
```

```
$degree
[1] 3

$knots
14.28571% 28.57143% 42.85714% 57.14286% 71.42857% 85.71429% 
 3.714286  6.428571  9.142857 11.857143 14.571429 17.285714 

$Boundary.knots
[1]  1 20
```

```
par(mfrow = c(1,2),las =1, bty ="l")
matplot(ns(1:20, df = 5), type = "l", main = "natural spline")
matplot(bs(1:20, df = 5), type = "l", main = "B-spline")
```



smoothing splines

- as many knots as data points
- plus squared-second-derivative ("wiggliness") penalty

$$\text{RSS} + \lambda \int (f''(t))^2 \, dt$$

* defined on an infinite-dimensional space * minimizer is a natural cubic spline with knots at x_i

$$(\mathbf{y} - \mathbf{N}\theta)^\top (\mathbf{y} - \mathbf{N}\theta) + \lambda \theta^\top \Omega_N \theta$$

with $\{\Omega_N\}_{jk} = \int N_j''(t) N_k''(t) \, dt$ $$ **generalized** ridge regression: penalize by $\lambda\Omega_N$ rather than λI * same data augmentation methods as before except that now we use $\sqrt{\lambda}C$ where C is a matrix square root of Ω_N