STAT 441: Lecture 19
Regularization

We want to find something replacing knot selection
But arrive to something with more general use

History

Whittaker (1923), “graduation” of actuarial mortality table
Given $y_1, y_2, \ldots, y_n$, find $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ such that

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum (\Delta^2 \hat{y}_i)^2 \looparrowright \min_{\hat{y}} !$$

Here $\lambda \geqslant 0$. Objective: to rid the original data of fluctuations

General functional fitting

Contrary to the textbook, we formulate the initial problem in a *functional*, that is, *infinite-dimensional* space. No splines yet.

Given $y_1, y_2, \dots, y_n$, and $x_1 < x_2 < \cdots < x_n$, find f such that

- $f(x_1), \dots, f(x_n)$ fit well $y_1, y_2, \dots, y_n$
- but at the same time, f is not too "wiggly", "rough"

How to do it? First, we have to propose some measure of "wiggliness". We may take some derivative f the fitted function, say, f', or f'', or f'''; then take its square or absolute value; and obtain a global measure via integration. For instance,

$$J(f) = \int (f''(x))^2 dx; \qquad \text{or } J(f) = \int |f''(x)| dx$$

Such $J(f)$ will be referred to as (roughness) *penalty*. To gain some insight about such a penalty, it is instructive to investigate for which f is $J(f) = 0$; for both examples of J given above, it means that f is linear, $f(x) = \alpha + \beta x$. (Apparently, such investigation is useful, but does not explain everything.)

We may still need to specify the domain of integration in the above examples - but we will see that sometimes it is not that essential.

Penalized fits

We seek a fit with guaranteed wiggliness

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 \looparrowright \min_f ! \qquad J(f) \leqslant \Lambda \qquad \text{(a tuning constant)}$$

Via Lagrange multiplier theory, this equivalent task is

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda J(f) \looparrowright \min_f !$$

Here, $\lambda > 0$ is another tuning constant, with unambiguous (but typically not explicit) relationship to Λ

Schoenberg (1964): smoothing splines

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx \looparrowright \min_f !$$

There are mathematical details here, which we omit. However: how come we can speak about splines?

Because the solutions are splines

The solution of the smoothing spline problem is a *natural* cubic spline, with knots at x_i

("Natural": continues outside of knots linearly)

Note: once $f(x_i)$ given, the solution is found by minimizing $J(f)$

That gives the linearity of f on $(-\infty, c]$ and $[d, \infty)$, where c is the minimum and d the maximum of the x_i. Now we can restrict the integration to $[c, d]$,

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int_c^d (f''(x))^2 dx \mapsto \min_f !$$

and use some further mathematics (or just integration by parts) to show that...

... the solution to the smoothing spline problem, exists within the class of natural cubic splines, with knots at x_i.

Finitary perspective

The original problem acted in the general functional spaces; now, however, we are in the finite dimensional space: all natural splines with given nodes (finite number of node, right?) can be written as linear combination of some (finite) basis functions

$$f(x) = \sum_j b_j g_j(x)$$

With some skill, we rewrite everything as a finite-dimensional problem in b_j - and in fact a quadratic one, as

- we are doing least-squares fitting) problem

- and the penalty has some square in it too, so it can be written as a quadratic form in b_j

So the original

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_c^d (f''(x))^2 dx \looparrowright \min_f !$$

becomes

$$(y - Lb)^\mathsf{T}(y - Lb) + \lambda b^\mathsf{T} G b \looparrowright \min_b !$$

The solution in fact solves the system $L^\mathsf{T} L + \lambda G b = L^\mathsf{T} y$

Remarks

The selection of the basis does not play a role, as long as the bases are equivalent (they generate the same linear spaces, any function that is a linear combination in one base, is a linear combination in another one)

Thus, we have something more general than just bases here...

Technical issue in this particular case: if x_i have duplicate values among them, we should take some care; there is no problem in the first, lack-of-fit part of the objective function, but the second, penalty part, should involve only "cleaned" x_i, with duplicates removed.

The greatest issue still ahead is that everything depends on *smoothing, regularization parameter* λ.

It is a *tuning* parameter: looking at the original formulation, we notice

- for large λ the penalty prevails: the fit is linear
- for $\lambda \to 0$ ($\lambda = 0$ won't fly!) the lack-of-fit prevails: the fit, if there are no duplicates in the x_i's, is just the spline interpolation of the data

So what do we have here?

Originally, we faced problem of selecting the *right* spline: how many knots, where to place them, ...

We somewhat mitigated the problem by the approach which

- put in many knots (in every x_i; do we need more?)
- introduced a reasonable criterion (penalty) to distinguish among various fits
- via regularization (fitting with penalty), we reduced the problem with many loose ends to a problem with just one loose end: λ

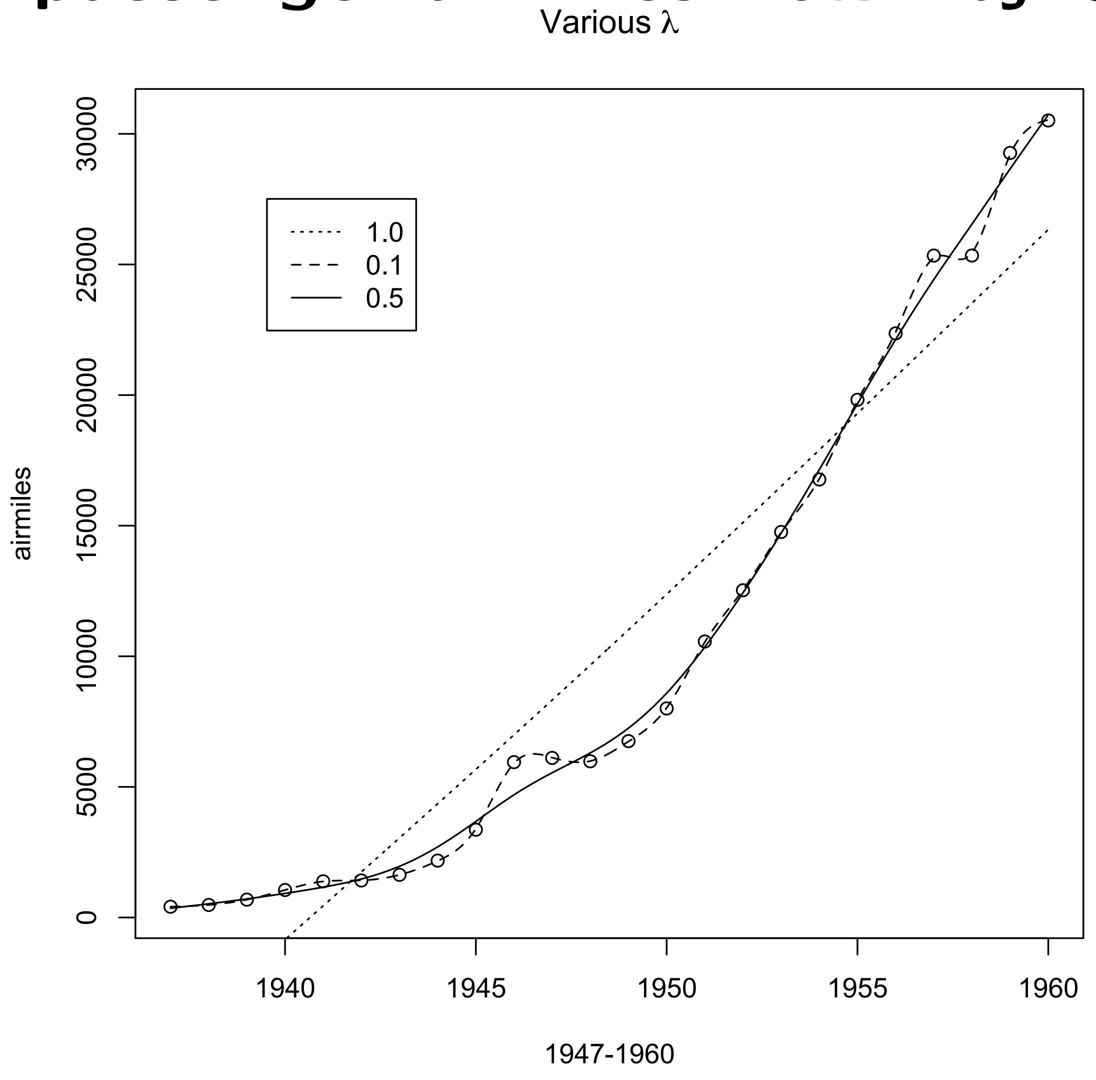
Turns out that this can have far-reaching generalizations:

- we design some class of flexible fits
- we assign some measure of complexity to those
- we reduce the problem to selecting one tuning parameter

Note also the analogy in tree-based methods: $R + \alpha$ size

- the lack-of-fit criterion here is R
- the complexity measure (penalty) is size
- only λ is named α

Revenue passenger airmiles flown by US airlines



A closer look at `help(smooth.spline)` reveals that `spar` shown in

```
> legend(locator(),lty=c(3,2,1),legend=c('1.0','0.1','0.5'))
```

is not λ, but a monotonous function of it, normed so that `spar` lies between 0 and 1.

Finesses of the R implementation I

```
> plot(1937:1960,airmiles,xlab='1947-1960')
> title(expression(paste("Various ",lambda)))
> xx=seq(1937,1960,len=400)
> smsp=smooth.spline(1937:1960,airmiles,spar=1)
> lines(xx,predict(smsp,xx)$y,lty=3)
> smsp
Call:
smooth.spline(x = 1937:1960, y = airmiles, spar = 1)
Smoothing Parameter  spar= 1  lambda= 0.9681153
Equivalent Degrees of Freedom (Df): 2.063613
Penalized Criterion: 197298405
GCV: 9840216
```

Finesses of the R implementation II

```
> smsp=smooth.spline(1937:1960,airmiles,spar=.1)
> lines(xx,predict(smsp,xx)$y,lty=2)
> smsp
Call:
smooth.spline(x = 1937:1960, y = airmiles, spar = 0.1)
Smoothing Parameter  spar= 0.1  lambda= 3.045644e-07
Equivalent Degrees of Freedom (Df): 22.97617
Penalized Criterion: 34624.87
GCV: 792768.6

> smsp=smooth.spline(1937:1960,airmiles,spar=.5)
> lines(xx,predict(smsp,xx)$y)
> smsp
Call:
smooth.spline(x = 1937:1960, y = airmiles, spar = 0.5)
Smoothing Parameter  spar= 0.5  lambda= 0.0002363563
Equivalent Degrees of Freedom (Df): 7.460656
Penalized Criterion: 6128442
GCV: 537681.1
```