Local regression II

Patrick Breheny

October 27

Pointwise inference

- At any given target point x_0, $\hat{f}$ is a simple linear model
- Thus,

$$\mathrm{E}\hat{f}(x_0) = \sum_i l_i(x_0) f(x_i)$$
$$\mathrm{Var}\hat{f}(x_0) = \sigma^2 \sum_i l_i(x_0)^2,$$

  where $\sigma^2 = \mathrm{Var}(y|x)$
- One method of constructing pointwise confidence intervals, then, is via

$$\hat{f}(x_0) \pm z_{\alpha/2}\sigma\sqrt{\sum_i l_i(x_0)^2}$$

The bias problem

- Recall, as with splines, that this is technically an interval for $\bar{f} = \mathrm{E}(\hat{f})$, not for f
- Also, our derivation of a nearly unbiased estimator for σ^2 from the spline lecture is valid for any linear estimator:

$$\hat{\sigma}^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{n - 2\nu + \tilde{\nu}},$$

  where $\nu = \mathrm{tr}(\mathbf{S})$ and $\tilde{\nu} = \mathrm{tr}(\mathbf{S}'\mathbf{S})$
- As with the confidence interval problem, the bias problem is difficult to resolve, so typically we assume that it is small and essentially ignore it in practice

Local likelihood

- Suppose our outcome is binary; the notion of fitting local least squares models is somewhat bothersome
- A reasonable alternative would be to fit a local logistic regression model instead
- The principle is the same as loess, although instead of minimizing the residual sum of squares, we maximize the log likelihood of the logistic regression model, fitting a new pair of regression coefficients at each target point x_0

Local logistic regression

For local logistic regression:

$$(\hat{\alpha}, \hat{\beta}) = \text{arg max}_{\alpha,\beta} \sum_i K_\lambda(x_0, x_i) l(y_i, \hat{\pi}_i),$$

where the contribution of observation i to the likelihood is once again weighted by the kernel, and

$$\begin{aligned} \hat{\pi}_i &= \frac{e^{\alpha + x\beta}}{1 + e^{\alpha + x\beta}} \\ l(y_i, \hat{\pi}_i) &= y_i \log(\hat{\pi}_i) + (1 - y_i) \log(1 - \hat{\pi}_i) \end{aligned}$$

Fitting via IRLS

- Fitting of logistic regression models already proceeds according to an iteratively reweighted least squares (IRLS) algorithm, which easily incorporates local weighting
- The weight given to an observation i in a given iteration of the IRLS algorithm is then a product of the weight coming from the quadratic approximation to the likelihood and the weight coming from the kernel ($w_i = w_{1i}w_{2i}$)

Cross-validation

- Fitting may be straightforward, but cross-validation becomes somewhat trickier
- In particular, does it still make sense to cross-validate based on squared error?
- According to SAS, yes – SAS uses the exact same formulas for GLM cross-validation as it does for least squares
- Admittedly, however, this is a bit strange, as we are not fitting parameters in terms of this criterion

Cross-validation & deviance

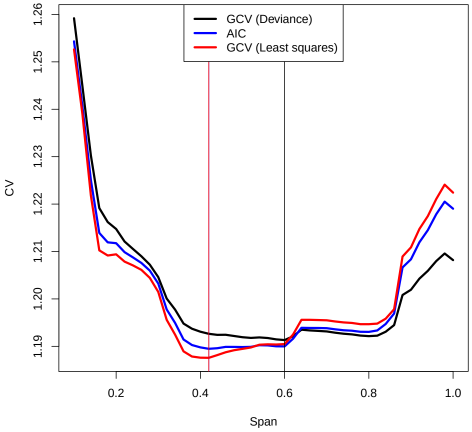
- A more natural criterion (used by `gam` and `mgcv`) is the deviance (-2 times the log-likelihood plus a constant) or average deviance:

$$CV = -\frac{2}{n}\sum_{i=1}^{n} l(y_i, \hat{\pi}_{(-i)})$$

- In either case, the many simplifications and closed forms that hold for cross-validation involving least squares regression and squared error loss do not hold anymore
- Nevertheless, GCV scores are still used based on the fact that, empirically, they seem to work:

$$GCV = \frac{1}{n}\frac{(-2)\sum_{i=1}^{n} l(y_i, \hat{\pi}_i)}{(1-\text{tr}(\mathbf{S})/n)^2}$$

GCV: Deviance vs LS



The various criteria are very similar, and often select the same smoothing parameter, but not always

Scatterplot smoothers and GAMs

- We will now discuss the implementation of local regression in SAS and `R`
- It is worth mentioning that both SAS (`PROC LOESS`) and `R` (`loess()`) have simpler interfaces for calculating loess fits for single variables where the outcome is normally distributed
- These are valuable procedures/functions, but we will focus on tools that are capable of fitting generalized additive models based on local regression

PROC GAM

- Fitting additive local models is straightforward in SAS, in the sense that you can use `PROC GAM` as with smoothing splines, only replacing `SPLINE` with `LOESS`:

```
PROC GAM DATA=bmd;
  MODEL Spnbmd = LOESS(Age) / METHOD=GCV;
RUN;
```

- Again, by default, SAS uses 4 degrees of freedom for each spline term; to have λ selected by GCV, specify `METHOD=GCV`:

PROC GAM: Modeling details

- The primary difference in using LOESS is that PROC GAM no longer splits the term into linear and nonlinear components (*i.e.*, it behaves more like mgcv did, where we get a single test of overall effect)
- The other noticeable difference is that the smoothing parameter for local regression is more interpretable

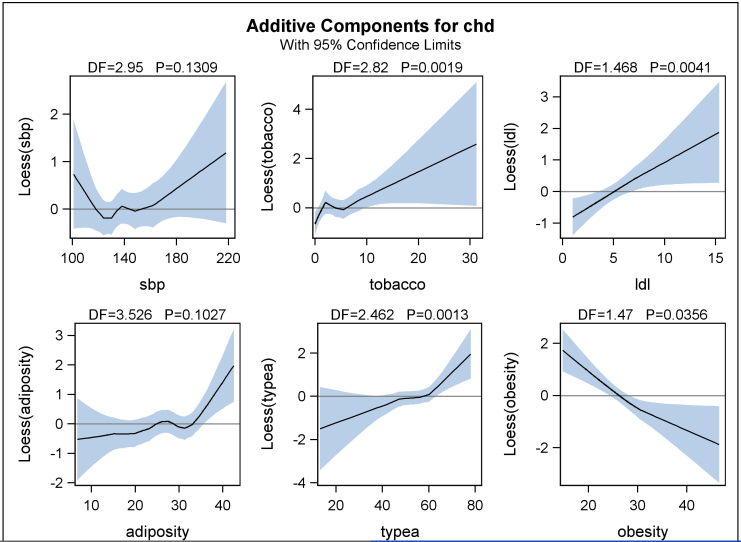
PROC GAM: Further options

- PROC GAM allows the usual choices of outcome distributions and any combination of parametric, spline, and loess terms
- For example, using the data from the CHD data set, we could submit:

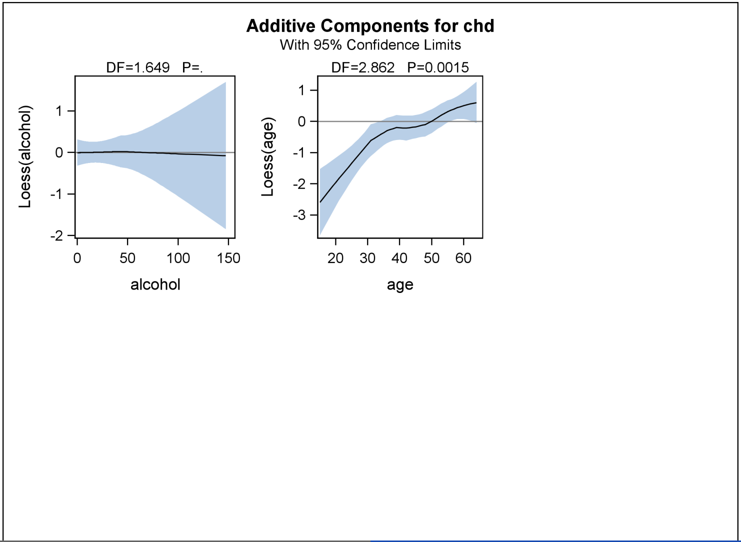
```
PROC GAM DATA=heart DESC;
  CLASS Famhist;
  MODEL Chd = PARAM(Famhist) SPLINE(Sbp) LOESS(Tobacco) /
                METHOD=GCV DIST=BIN;
RUN;
```

- Recall also that the smooth functions can be visualized using the ODS system by adding PLOTS=COMPONENTS(CLM ADDITIVE) to the PROC GAM statement

Heart study


Additive Components for chd
With 95% Confidence Limits
DF=2.95 P=0.1309
Loess(sbp)
sbp
DF=2.82 P=0.0019
Loess(tobacco)
tobacco
DF=1.468 P=0.0041
Loess(ldl)
ldl
DF=3.526 P=0.1027
Loess(adiposity)
adiposity
DF=2.462 P=0.0013
Loess(typea)
typea
DF=1.47 P=0.0356
Loess(obesity)
obesity

Heart study


Additive Components for chd
With 95% Confidence Limits
DF=1.649 P=.
Loess(alcohol)
1
0
-1
-2
0
50
100
150
alcohol
DF=2.862 P=0.0015
Loess(age)
1
0
-1
-2
-3
20
30
40
50
60
age

The `gam` package

- In `R`, on the other hand, the `mgcv` package is spline-specific; we must turn to a separate package to fit local regression models: `gam`
- A few comments up front:
  - The `gam` package is very similar to `PROC GAM` in that it allows parametric, spline, or loess terms, and is entirely based on backfitting
  - The main function in the `gam` package is `gam()`; note that there is also a `gam()` function in the `mgcv` package – you cannot load both packages during the same session and expect them both to work
  - If the `gam` package can incorporate splines, why even talk about `mgcv`? As we will see, one shortcoming of `gam` is that it does not provide automatic selection of smoothing parameters

The gam package

- The basic syntax of model fitting is as follows:

  ```
  fit <- gam(spnbmd~lo(age),data=bmd)
  ```

  where `lo` controls the local polynomial which is fit to the data
- Two important arguments to `lo` are `span`, which controls the fraction of points in the smoothing window, and `deg`, which controls the degree of the local fit:

  ```
  fit <- gam(spnbmd~lo(age,span=.8,deg=2),data=bmd)
  ```

- Default is linear fit with 50% of the points in the window

Selection of α

- As mentioned earlier, the `gam` package has one notable shortcoming: it does not provide automatic selection of the smoothing parameter
- The package does, however, provide both AIC and deviance to assist you in manually choosing a smoothing parameter:

```
AIC <- numeric(length(Span))
for (i in 1:length(Span))
  {
    fit <- gam(spnbmd~lo(age,span=Span[i]),data=bmd)
    AIC[i] <- fit$aic
  }
Span[which.min(AIC)]
```

Testing

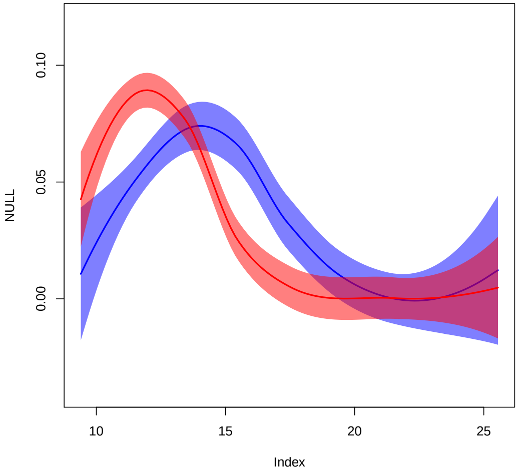
- summary.gam provides tests of nonlinearity, but not of overall effect (or of linear terms)
- To obtain them, you need to carry out the LRT manually:

```
fit <- gam(chd~lo(sbp)+lo(tobacco)+famhist,data=heart,family="binom
summary(fit)
fit0 <- gam(chd~lo(sbp)+famhist,data=heart,family="binomial")
anova(fit0,fit)
fit0 <- gam(chd~lo(sbp)+lo(tobacco),data=heart,family="binomial")
anova(fit0,fit)
```

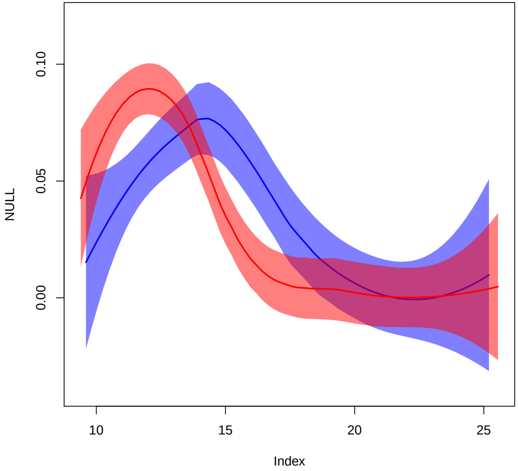
Global confidence bands

- `R` has another package for local regression models called `locfit`
- One interesting feature it provides is the computation of simultaneous confidence bands
- Note the distinction: pointwise confidence intervals have $(1-\alpha)\%$ coverage only at a given x, while simultaneous confidence bands have $(1-\alpha)\%$ coverage for containing the entire function f (technically $\bar{f}$) over all x
- The details are fairly complicated and involve Gaussian processes, but it is worth looking at an example

Pointwise CIs for the bone mineral density data



Simultaneous CIs for the bone mineral density data



Multivariate kernels

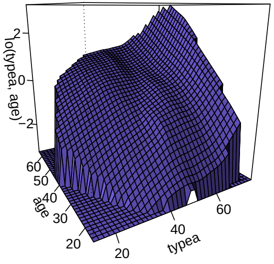
- It is straightforward to extend the idea of local regression to multiple dimensions – all that is needed is to define a multivariate kernel:

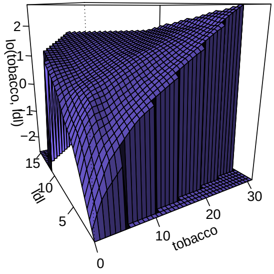
$$K_\lambda(\mathbf{x}_i, \mathbf{x}_0) = K\left(\frac{\|\mathbf{x}_i - \mathbf{x}_0\|}{\lambda}\right),$$

  where $\|\mathbf{a}\| = \sqrt{\sum_i a_i^2}$ is the Euclidean distance
- This can be further generalized by allowing different bandwidths in each dimension:

$$K_\lambda(\mathbf{x}_i, \mathbf{x}_0) = \prod_{j=1}^{p} K\left(\frac{x_{ij} - x_{0j}}{\lambda_j}\right)$$

Examples


lo(typea, age)
2
0
−2
60
50
40
age
30
20
20
40
typea
60


lo(tobacco, ldl)
2
1
0
−1
−2
15
10
ldl
5
0
10
tobacco
20
30

The curse of dimensionality?

- Although one can easily write down high-dimensional kernels, as we have remarked previously, the statistical properties of the estimator tend to become poor as p grows larger
- With adaptive kernel widths, however, the picture is not quite so clear cut
- Indeed, one can fit a model to the CHD data set with an eight-dimensional kernel encompassing all the continuous terms, and this model has a lower AIC than the additive model