Statistics 452: Statistical Learning and Prediction

Chapter 7, Part 3: Smoothing Splines, Local Regression

Brad McNeney

2017-10-31

Smoothing Splines

Smoothing Splines Overview

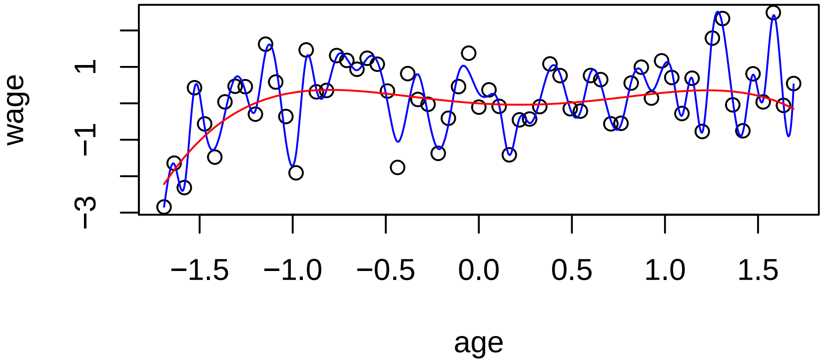
- Smoothing splines are an alternative approach to devising a smooth curve.
- Though the derivation is different, we end up with a natural cubic spline, with knots at each observed data point.
  - The estimated coefficients of the basis functions using least squares would interpolate the data points.
  - To avoid over-fitting, a shrinkage penalty is added to the RSS. The penalty is comprised of a tuning parameter times a penalty term.

Smoothness

- We seek a *smooth* function $g(x)$ that fits the data well.
  - Want $RSS = \sum_{i=1}^{n}(y_i - g(x_i))^2$ small.
- By smooth we mean not too "rough" (wiggly)
  - If RSS is the objective function, the curve g will interpolate the data and will be very rough.
  - Illustrate by fitting a natural cubic spline to simulated data on "age" and "wage"

```
set.seed(1)
age <- scale(18:80); betas <- c(2,-2,2,-2)
f <- poly(age,degree=4)%*% betas
wage <- f + rnorm(length(age))
sfit0 <- smooth.spline(age,wage,spar=0)
newage<-seq(from=min(age),to=max(age),length=1000)
pwage0 <- predict(sfit0,newage)
```

```r
plot(age,wage)
lines(pwage0$x,pwage0$y,col="blue") # fitted curve
lines(age,f,col="red") # true curve
```



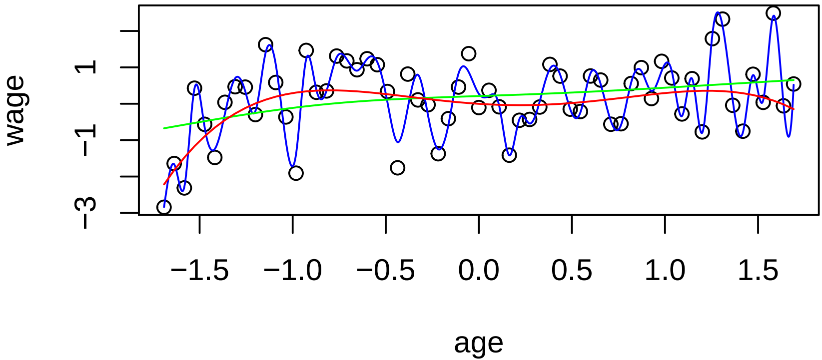
Penalized Objective Function

- The criterion function to minimimize is

$$\sum_{i=1}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt. \tag{1}$$

- Like ridge regression or the lasso, the objective function is of the form RSS + penalty, and we have a tuning, or smoothing parameter λ, but the penalty function is now $\int g''(t)dt$.
  - $g''(t)$ is the curvature of g at t.
  - $\int g''(t)^2 dt$ is a measure of the total curvature.
- It can be shown that the minimizer of (1) is a natural cubic spline with knots at the unique observed data points, with the coefficients of the basis functions *shrunken* towards zero.
  - The degree of shrinkage is controlled by λ.

```r
sfit1 <- smooth.spline(age,wage,spar=1) # const * log(lambda) = 1
pwage1 <- predict(sfit1,newage)
plot(age,wage)
lines(pwage0$x,pwage0$y,col="blue") # fitted curve with lambda=0
lines(pwage1$x,pwage1$y,col="green")
lines(age,f,col="red") # true curve
```



Choosing the Smoothing Parameter

- Could use cross validation to select λ.
  - How many folds?
- It turns out there is a very simple formula for the CV estimate of test error when using leave-out-one CV (LOOCV):

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{g}_\lambda(x_i)^{(-i)})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{S_\lambda\}_{ii}}\right)^2 \quad (2)$$

where

- $\hat{g}_\lambda(x_i)^{(-i)}$ is the fitted value for x_i when (x_i, y_i) is held out of the training set,
- $\hat{g}_\lambda(x_i)$ is the fitted value for x_i when all of the data are used to fit g,
- $\{S_\lambda\}_{ii}$ is the ith diagonal entry of the "smoother matrix".

The Smoother Matrix

- The smoother matrix S_λ is the matrix that turns the response y into the smooth $\hat{g}_\lambda$; i.e., $\hat{g}_\lambda(x) = S_\lambda y$, where x is the vector of explanatory variable values and $\hat{g}_\lambda(x)$ is the vector of $\hat{g}_\lambda(x_i)$ values.
- In linear regression, the smoother matrix is called the hat matrix and is denoted H.
  - H turns the response y into $\hat{y}$; i.e., $\hat{y} = Hy$,
  - The sum of the diagonal elements of H is the number of regression coefficients $p + 1$.
  - The diagonal elements are the hat values, or leverages h_i.
- Thus the $CV_{(n)}$ estimate for the smoothing spline is analogous to least squares, with the diagonal elements of H replaced by those of S_λ.

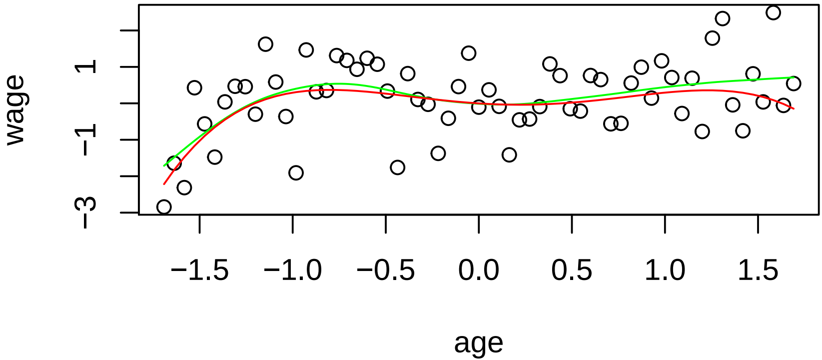
Effective Degrees of Freedom

- In linear regression, the model df $p + 1$ can be shown to equal the sum of the diagonal elements of H.
- By analogy, the sum of the diagonal elements of the smoother matrix is refered to as the *effective degrees of freedom* and is denoted df_λ.
  - One can show that as λ ranges from 0 to ∞, df_λ ranges from n to 2.
  - Can treat df_λ as the smoothing parameter and select its value by CV.

```r
sfitCV <- smooth.spline(age,wage,cv=TRUE)
sfitCV$df
```

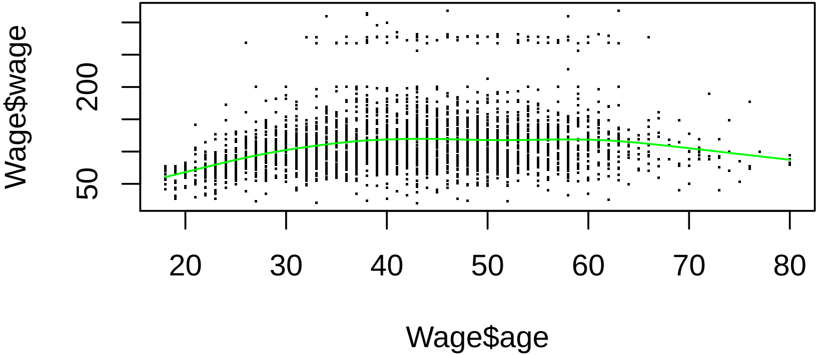
```
## [1] 5.769452
```

```r
pwageCV <- predict(sfitCV,newage)
plot(age,wage)
lines(pwageCV$x,pwageCV$y,col="green")
lines(age,f,col="red") # true curve
```



Example

```r
library(ISLR); data(Wage)
plot(Wage$age,Wage$wage,pch=".")
sfit <- smooth.spline(Wage$age,Wage$wage,cv=TRUE)
pwage <- predict(sfit)
lines(sfit,col="green") # can plot output of smooth.spline directly
```



Local Regression

Local Regression

- A variation on KNN: instead of fitting a constant over a neighborhood, fit a weighted regression.
  - weighted means points in the neighborhood closest to the point of prediction are up-weighted.
  - the regression could be constant, linear or quadratic.
- The neighborhood size is refered to as the "span" s.

Local Linear Regression Algorithm

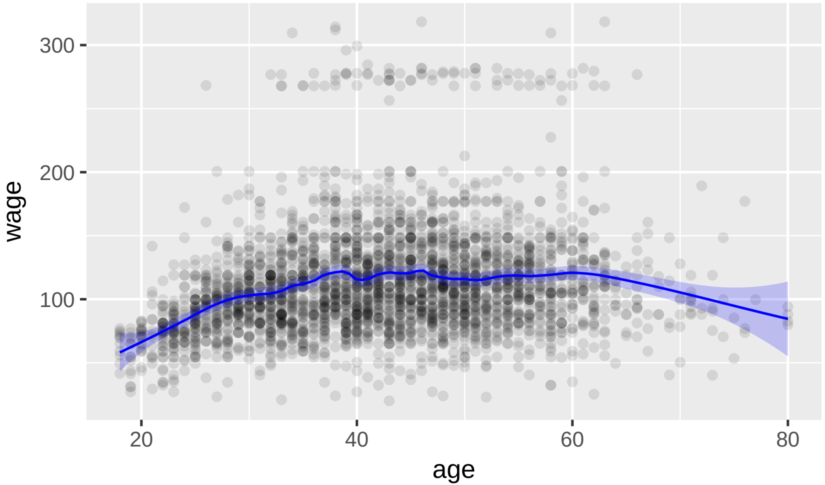
- Select a span s and a weight function $K(x_i, x_0)$.
- For $X = x_0$:
  1. Extract the nearest $s * n$ points to x_0 to train the local regression.
  2. Assign weight $K_{0i} = K(x_i, x_0)$ to each neighbor.
  3. Fit a weighted least-squares regression; that is, find the $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K(x_i, x_0)(y_i - [\beta_0 + \beta_1 x_i])^2$$

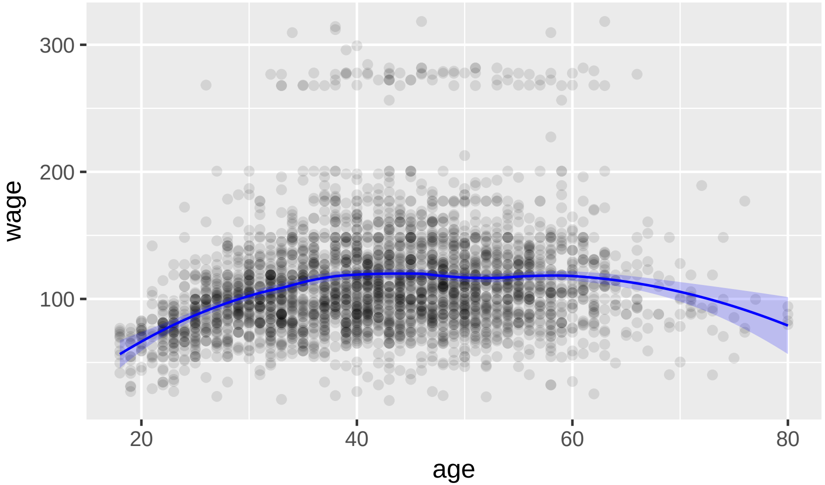
  4. $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

```r
plotfit <- function(fit,dat,newdat){
  preds <- data.frame(newdat,
          predict(fit,newdata=newdat,se=TRUE))
  preds <- mutate(preds,
                  lwr = preds$fit-2*preds$se,upr = preds$fit+2*preds$se)
  ggplot(dat,aes(x=age,y=wage)) + geom_point(alpha=0.1) +
    geom_ribbon(aes(x=age,y=fit,ymin=lwr,ymax=upr),
                data=preds,fill="blue",alpha=.2) +
    geom_line(aes(y=fit),data=preds,color="blue")
}
```

```r
library(ggplot2); library(dplyr)
newdat <- data.frame(age=seq(from=min(Wage$age),to=max(Wage$age),length=100))
fit2 <- loess(wage~age,span=0.2,data=Wage)
plotfit(fit2,Wage,newdat)
```



```
fit5 <- loess(wage~age,span=0.5,data=Wage)
plotfit(fit5,Wage,newdat)
```



Extensions to Higher Dimension

- For $p > 1$ explanatory variables, we can generalize local regression.
  - Choose neighborhoods based on $X_1, \ldots, X_p$.
  - Fit a multiple linear regression.
- However, it has been observed that local regression performs poorly for $p > 3$ because there are few points close to x_0 (recall homework 2).