Nonparametric Regression: Some Practical Topics and Splines

Paweł Polak

February 10 & 12, 2016

STAT W4413: Nonparametric Statistics - Lecture 8

Confidence Bands

- For notational simplicity I will use the $K_h(x - x_i)$ instead of $K(|x - x_i|/h)$.
- In every estimation problem (including also curve fitting!) a topic of major practical importance is the confidence interval.
- To understand some of the issues in estimating the confidence interval in the curve fitting problem we focus on the kernel regression.
- Most of these discussions can be extended to other curve fitting schemes. Unfortunately, we don't have enough time to cover other techniques (see the book by L. Wasserman for more information).

Confidence Bands

Consider the kernel regression:

$$\hat{g}_h(x) = \frac{\sum_{i=1}^n K_h(x - x_i) y_i}{\sum_{i=1}^n K_h(x - x_i)} = \frac{\sum_{i=1}^n K_h(x - x_i) g(x_i)}{\sum_{i=1}^n K_h(x - x_i)} + \frac{\sum_{i=1}^n K_h(x - x_i) \epsilon_i}{\sum_{i=1}^n K_h(x - x_i)}$$

Therefore,

$$\hat{g}_h(x) - g(x) = \underbrace{\left(\frac{\sum_{i=1}^n K_h(x - x_i) g(x_i)}{\sum_{i=1}^n K_h(x - x_i)} - g(x) \right)}_{\text{The term that led to bias}} + \underbrace{\frac{\sum_{i=1}^n K_h(x - x_i) \epsilon_i}{\sum_{i=1}^n K_h(x - x_i)}}_{\text{The term that led to variance}}$$

For a good confidence interval for $g(x)$ we need to calculate the first term, and also evaluate the distribution of the second term.

- If we assume that $x_1, x_2, \ldots, x_n$ are fixed (no random model), then the second term will be Gaussian and as we will see we can efficiently estimate its standard deviation.
- Calculating the term $\frac{\sum_{i=1}^n K_h(x - x_i) g(x_i)}{\sum_{i=1}^n K_h(x - x_i)} - g(x)$ on the other hand, is challenging since $g(x)$ is not known.

Confidence Bands

$$\frac{\sum_{i=1}^{n} K_h(x - x_i)g(x_i)}{\sum_{i=1}^{n} K_h(x - x_i)} - g(x)$$

There are two issues regarding this term that do not let us settle this problem:

1. In most practical situations this term is not negligible and is at the order of the other term. You will see this in the next homework.
2. In general settings, we don't have a good method for estimating this term.

Therefore, instead of reporting a confidence interval for g, we report it for the smoothed version of g, i.e.,

$$\frac{K_h(x - x_i)g(x_i)}{\sum K_h(x - x_i)}.$$

We look for a confidence interval of the form

$$[\hat{g}_h(x) - \alpha, \hat{g}_h(x) + \alpha]$$

for $\frac{\sum K_h(x-x_i)g(x_i)}{\sum K_h(x-x_i)}$ rather than $g(x)$.

But first we need to estimate $\hat{\sigma}$, i.e., the standard deviation of the noise...

Estimate of σ^2

Consider the following model

$$Y_i = g(x_i) + \epsilon_i, E(\epsilon_i) = 0, E(\epsilon_i^2) = \sigma^2.$$

We consider one more assumption on ϵ_i and that is $\epsilon_i \sim N(0, \sigma^2)$. Suppose that $X_1, X_2, \ldots, X_n$ are ordered meaning that $X_1 \leq X_2 \leq \ldots \leq X_n$.

Rice[1984] considered the following estimate:

$$\hat{\sigma}^2 = \frac{1}{2(n-1)} \sum_{i=1}^{n-1} (Y_{i+1} - Y_i)^2.$$

We need to characterize the mean and variance of this estimate. In particular, we can show that

- the mean is approximately σ^2, and
- the variance is inversely proportional to n which means that for large values of n it provides a very reliable estimate for $\hat{\sigma}^2$.

Estimate of σ^2

First consider the expected value of $\hat{\sigma}^2$:

$$\begin{aligned} E(\hat{\sigma}^2) &= \frac{1}{2(n-1)} \sum_{i=1}^{n} E(f(x_{i+1}) - f(x_i) + \epsilon_{i+1} - \epsilon_i)^2 \\ &= \sigma^2 + \underbrace{\frac{1}{2(n-1)} \sum_{i=1}^{n-1} (f(x_{i+1}) - f(x_i))^2}_{\text{This term is usually negligible. Why? Hence, the expected value of } \hat{\sigma}^2 \text{ is close to } \sigma^2} \end{aligned}$$

When the number of samples is large, we expect the second term to be negligible. Therefore, we expect the expected value of $\hat{\sigma}^2$ to be close to σ^2.

Estimate of σ^2

The next step is to evaluate the variance of this estimator.

$$\begin{aligned} E(\hat\sigma^2 - E\hat\sigma^2)^2 &= E(\hat\sigma^4) - (E(\hat\sigma^2))^2 \\ E(\hat\sigma^4) &= \frac{1}{4(n-1)^2} \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} E[(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2] \end{aligned}$$

In order to simplify this expression consider the following cases first:

- if $i = j$:

$$E(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2 \overset{(a)}{\approx} E[(\epsilon_{i+1}-\epsilon_i)^2(\epsilon_{i+1}-\epsilon_i)^2] \overset{(b)}{=} 3(2\sigma^2)^2$$

  To obtain (a) we assumed that $f(x_i + 1) \approx f(x_i)$, for (b) we used the Gaussianity of ϵ_i.

- if $j = i - 1$:

$$E(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2 \overset{(a)}{\approx} E[(\epsilon_{i+1}-\epsilon_i)^2(\epsilon_{i+1}-\epsilon_i)^2] \overset{(b)}{=} 3(2\sigma^2)^2$$

$$\begin{aligned} E(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2 &\approx E[(\epsilon_{i+1}-\epsilon_i)^2(\epsilon_i-\epsilon_{i-1})^2] \\ &= E[(\epsilon_{i+1}^2+\epsilon_i^2-2\epsilon_i\epsilon_{i+1})(\epsilon_i^2+\epsilon_{i-1}^2-2\epsilon_i\epsilon_{i-1})] \\ &= \sigma^2(2\sigma^2)+3\sigma^4+\sigma^4. \end{aligned}$$

- if $j = i + 1$:

$$E(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2 \approx 6\sigma^4.$$

- if $j \notin \{i, i-1, i+1\}$, then:

$$E(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2 \approx 4\sigma^4.$$

Estimate of σ^2

$$\begin{aligned}
E(\hat\sigma^4) &\approx \frac{1}{4(n-1)^2}\sum_{i=1}^{n-1}\sum_{j=i-1}^{i+1} E(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2 \\
&+ \frac{1}{4(n-1)^2}\sum_{i=1}^{n-1}\sum_{\substack{j=1\\ j\notin\{i,i-1,i+1\}}}^{n-1} E[(Y_{i+1}-Y_i)^2(Y_{j+1}-Y_j)^2] \\
&= \frac{1}{4(n-1)^2}\sum_{i=1}^{n} 12\sigma^4+6\sigma^4+6\sigma^4 \\
&+ \frac{1}{4(n-1)^2}\sum_{i=1}^{n-1}\sum_{\substack{j=1\\ j\notin\{i,i-1,i+1\}}}^{n-1} 4\sigma^4 = \frac{24\sigma^4(n-1)}{4(n-1)^2}+\frac{4\sigma^4(n-1)(n-4)}{(n-1)^2} \\
&= \frac{4\sigma^4(n-1)(n+2)}{4(n-1)^2} \approx \sigma^4+\frac{3\sigma^4}{n-1}.
\end{aligned}$$

Therefore

$$E(\hat\sigma^2-\sigma^2)^2 \approx \frac{3\sigma^4}{n-1}.$$

As we discussed before for large values of n the variance is quite small, and combined with the fact that the mean of $\hat\sigma^2$ is σ^2, provides a very good estimate.

Heteroscedastic nonparametric regression

- In all the discussions of so far we assume that $y_i = g(x_i) + \epsilon_i$, where $E(\sigma_i) = 0$, and $E(\epsilon_i^2) = \sigma^2$.
- But the main assumption is that the noise variance is independent of the value of x_i.
- In many practical situation, this is not correct. For those situations a more realistic model is to assume that $y_i = g(x_i) + \sigma(x_i)\epsilon_i$, where again $\mathbb{E}(\epsilon_i) = 0$, and $E(\epsilon_i^2) = 1$. This is called "heteroscedastic regression setting".
- Most of the discussions that we have had so far are going to work in this setting as well. Except for the estimation of "confidence band". We won't cover this in our course. But I strongly recommend that you read [6].

In short, if

$$y_i = g(x_i) + \sigma(x_i)\epsilon_i,$$

then

- estimate $g(x)$ with any nonparametric method to get an estimate $\widehat{g}(x)$;
- define $Z_i = \log\left(y_i - \widehat{g}(x_i)\right)^2$;
- regress the Z_i's on the x_i's (again using a nonparametric method) to get an estimate $\widehat{q}(x)$ of $\log \sigma^2(x)$; and
- let $\widehat{\sigma}^2(x) = \exp\left(\widehat{q}(x)\right)$.

Construction of confidence bands for any linear smoother

Suppose that our estimator can be written as

$$\begin{bmatrix} \hat{g}(x_1) \\ \hat{g}(x_2) \\ \vdots \\ \hat{g}(x_n) \end{bmatrix} = S \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad (1)$$

where S is an $n \times n$ matrix which does not depend on $y_1, y_2, \ldots, y_n$ (the kernel method, locally linear regression, spline and many other methods fall into this category). E.g. for the kernel smoother we have

$$\widehat{g}_h(x) = \sum_{i=1}^{n} \frac{K_h(|x - x_i|)}{\sum_{j=1}^{n} K_h(|x - x_j|)} y_i = \sum_{i=1}^{n} S_i(x) y_i, \text{ for any } x.$$

Then

$$E[\widehat{g}_h(x)] = \sum_{i=1}^{n} S_i(x) g(x_i),$$

ghat(x)~N()

and

$$\mathrm{Var}(\widehat{g}_h(x)) = \sigma^2 \|S(x)\|_2^2$$

One method of constructing pointwise confidence intervals is via

$$I(x) = \left[\widehat{g}_h(x) - z_{\alpha/2}\widehat{\sigma}\, ||S(x)||_2, \widehat{g}_h(x) + z_{\alpha/2}\widehat{\sigma}\, ||S(x)||_2\right].$$

The global confidence band for $E[\widehat{g}_h(x)]$ is

$$I(x) = \left[\widehat{g}_h(x) - c\widehat{\sigma}\, \|S(x)\|_2, \widehat{g}_h(x) + c\widehat{\sigma}\, \|S(x)\|_2\right].$$

Construction of confidence bands for any linear smoother

Construction of global confidence bands is more complicated. It requires the computation of the distribution of the maximum of a Gaussian process. This is a well studied problem but we will not go into the details.

The global confidence band is given by

$$I(x) = \left[\widehat{g}_h(x) - c\widehat{\sigma}\left\|S(x)\right\|_2, \widehat{g}_h(x) + c\widehat{\sigma}\left\|S(x)\right\|_2\right],$$

where c solves

$$2(1-\Phi(c)) + \frac{\kappa_0}{\pi} e^{-c^2/2} = \alpha,$$

with

$$\kappa_0 = \int_a^b \left\|T'(x)\right\|_2 \mathrm{d}x \approx \left(\frac{b-a}{h}\right) \frac{\left\|K_h'\left(|x-x_i|\right)\right\|_2}{\left\|K_h\left(|x-x_i|\right)\right\|_2},$$

where the last approximation is valid for equally spaced grid of x_i's.
Instead we will talk about how to obtain confidence bands using the *locfit* package in R

The loess function in R

In R, local linear regression is implemented through the loess function, which uses a formula interface similar to that of other regression functions:

- fit = loess($y \sim x$, $data$, $span = 0.3$, $degree = 3$)
- where span is the smoothing parameter which controls the bias-variance tradeof
- degree: this lets you specify local constant regression (degree=0), local linear regression (degree=1), or local polynomial fits (degree=2)
- only the tricube kernel is implemented, $K(u) = \frac{70}{81}\left(1 - |u|^3\right)^3 \mathbb{I}_{\{|u| \leq 1\}}$
- span refers to the proportion of the observations x_i within its compact support
- the kernel in loess is adaptive: specifying $span = 0.2$ means that the bandwidth of the kernel at x_0 is made just wide enough to include 20% of the x_i values

Local linear models tend to be biased in regions of high curvature, a phenomenon referred to as

"trimming the hills and filling in the valleys".

Higher-order local polynomials correct for this bias, but at the expense of increased variability.

The conventional wisdom on the subject of local linear vs. local quadratic fitting says that:

- Local linear fits tend to be superior at the boundaries
- Local quadratic fits tend to be superior in the interior
- Local fitting to higher order polynomials is possible in principle, but rarely necessary in practice

The locfit function in R

The basic syntax of model fitting is as follows:

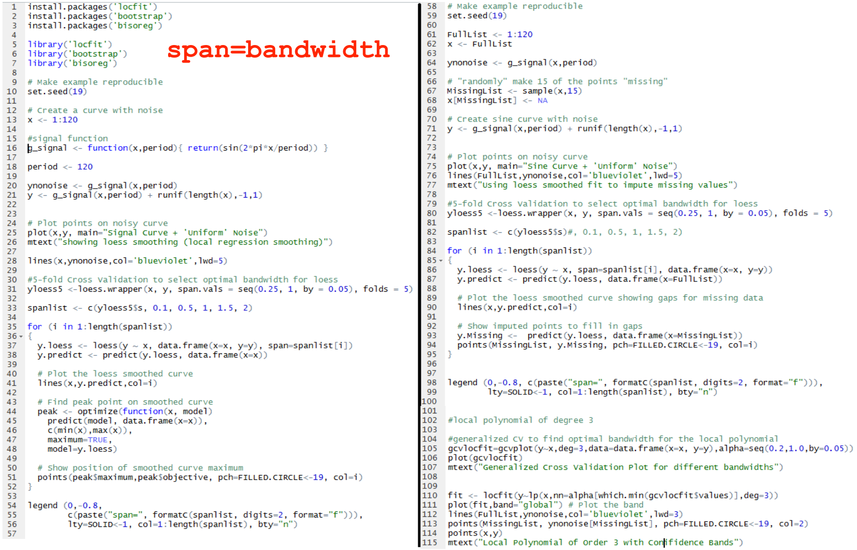
- fit $=$ locfit$(y \sim lp(x, nn = .7, deg = 2))$
- where *lp* controls the local polynomial which is fit to the data
- Just like loess, there is a *nn* parameter (analogous to span), which adaptively determines the bandwidth.
- There is also a *deg* parameter, which controls the degree of the local polynomial (like loess, the default is 2)

The locfit package is very well-developed, and we cannot possibly cover all of its features here.

The two very important features are its ability to construct pointwise and simultaneous confidence bands

$plot(fit, band = "global")$ plots the band.

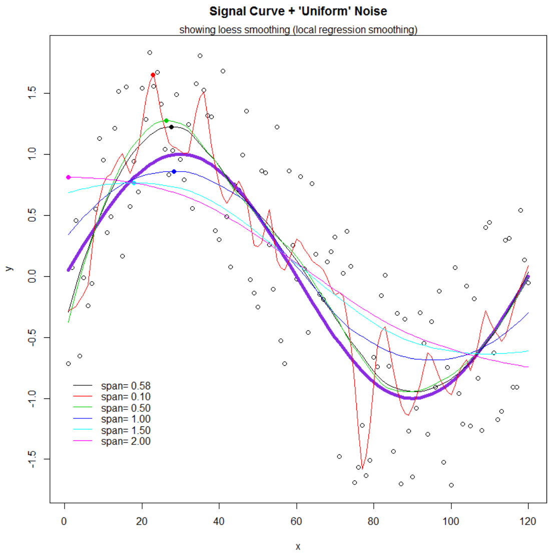
Example in R: LOESS and Local Polynomial



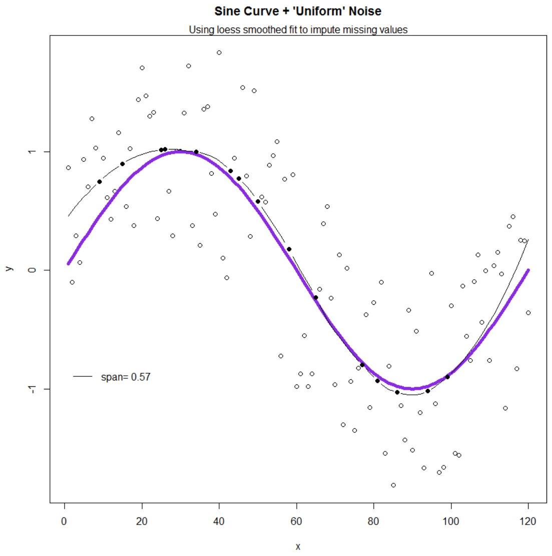
span=bandwidth

```r
install.packages('locfit')
install.packages('bootstrap')
install.packages('bisoreg')

library('locfit')
library('bootstrap')
library('bisoreg')

# Make example reproducible
set.seed(19)

# Create a curve with noise
x <- 1:120

#signal function
g_signal <- function(x,period){ return(sin(2*pi*x/period)) }

period <- 120

ynonoise <- g_signal(x,period)
y <- g_signal(x,period) + runif(length(x),-1,1)


# Plot points on noisy curve
plot(x,y, main="Signal Curve + 'Uniform' Noise")
mtext("showing loess smoothing (local regression smoothing)")

lines(x,ynonoise,col='blueviolet',lwd=5)

#5-fold Cross Validation to select optimal bandwidth for loess
yloess5 <-loess.wrapper(x, y, span.vals = seq(0.25, 1, by = 0.05), folds = 5)

spanlist <- c(yloess5$s, 0.1, 0.5, 1, 1.5, 2)

for (i in 1:length(spanlist))
{
  y.loess <- loess(y ~ x, data.frame(x=x, y=y), span=spanlist[i])
  y.predict <- predict(y.loess, data.frame(x=x))

  # Plot the loess smoothed curve
  lines(x,y.predict,col=i)

  # Find peak point on smoothed curve
  peak <- optimize(function(x, model)
    predict(model, data.frame(x=x)),
    c(min(x),max(x)),
    maximum=TRUE,
    model=y.loess)

  # Show position of smoothed curve maximum
  points(peak$maximum,peak$objective, pch=FILLED.CIRCLE<-19, col=i)
}

legend (0,-0.8,
        c(paste("span=", formatC(spanlist, digits=2, format="f"))),
        lty=SOLID<-1, col=1:length(spanlist), bty="n")

# Make example reproducible
set.seed(19)

FullList <- 1:120
x <- FullList

ynonoise <- g_signal(x,period)

# "randomly" make 15 of the points "missing"
MissingList <- sample(x,15)
x[MissingList] <- NA

# Create sine curve with noise
y <- g_signal(x,period) + runif(length(x),-1,1)


# Plot points on noisy curve
plot(x,y, main="Sine Curve + 'Uniform' Noise")
lines(FullList,ynonoise,col='blueviolet',lwd=5)
mtext("Using loess smoothed fit to impute missing values")

#5-fold Cross Validation to select optimal bandwidth for loess
yloess5 <-loess.wrapper(x, y, span.vals = seq(0.25, 1, by = 0.05), folds = 5)

spanlist <- c(yloess5$s)#, 0.1, 0.5, 1, 1.5, 2)

for (i in 1:length(spanlist))
{
  y.loess <- loess(y ~ x, span=spanlist[i], data.frame(x=x, y=y))
  y.predict <- predict(y.loess, data.frame(x=FullList))

  # Plot the loess smoothed curve showing gaps for missing data
  lines(x,y.predict,col=i)

  # Show imputed points to fill in gaps
  y.Missing <-  predict(y.loess, data.frame(x=MissingList))
  points(MissingList, y.Missing, pch=FILLED.CIRCLE<-19, col=i)
}


legend (0,-0.8, c(paste("span=", formatC(spanlist, digits=2, format="f"))),
        lty=SOLID<-1, col=1:length(spanlist), bty="n")


#local polynomial of degree 3

#generalized CV to find optimal bandwidth for the local polynomial
gcvlocfit=gcvplot(y~x,deg=3,data=data.frame(x=x, y=y),alpha=seq(0.2,1.0,by=0.05))
plot(gcvlocfit)
mtext("Generalized Cross Validation Plot for different bandwidths")


fit <- locfit(y~lp(x,nn=alpha[which.min(gcvlocfit$values)],deg=3))
plot(fit,band="global") # Plot the band
lines(FullList,ynonoise,col='blueviolet',lwd=3)
points(MissingList, ynonoise[MissingList], pch=FILLED.CIRCLE<-19, col=2)
points(x,y)
mtext("Local Polynomial of Order 3 with Confidence Bands")
```

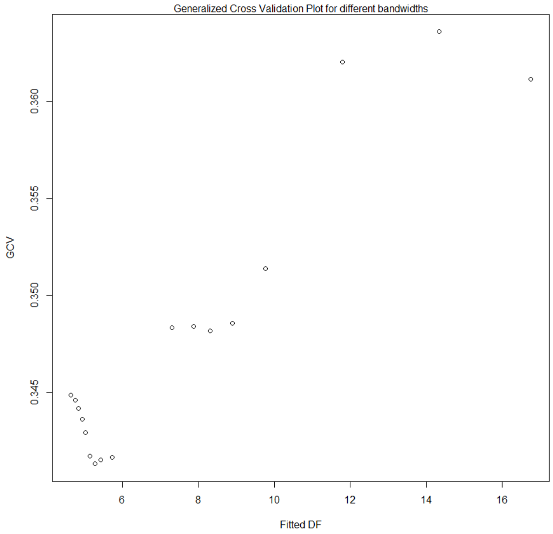
Example in R: LOESS fit



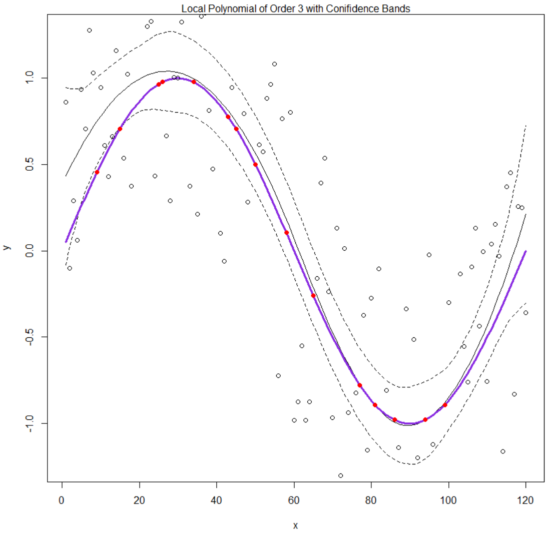
Example in R: LOESS for missing values



Example in R: General Cross Validation for locfit



Example in R: locfit - local polynomial of degree 3



Outlier detection

- In many practical situations, a small fraction of the data does not follow the model $y_i = g(x_i) + \epsilon_i$.
- This is usually due to mistakes in collecting data, personal mistakes, or not having a complete control over the experiment.
- These samples are called outliers.
- Every good regression algorithm should be robust to the outliers.
- In case of curve fitting, a major task is to detect the outlier samples before applying our regression techniques.
- We will not have enough time to cover this topic in class. Those of you who are interested in this topic and other practical issues of curve fitting problem are encourage to read [6].

The estimators we have used are based on squared error loss. This is an easy loss function to use but the resulting estimator is potentially not robust to outliers.
In robust regression, we choose $\boldsymbol{\beta}$ to minimize

$$\sum_{i=1}^{n} K_h\left(|x_i - x|\right) \rho\left(y_i - \beta_0 - \beta_1 x_i\right), \text{ where}$$

- $\rho(t) = t^2$ gets us back the squared error loss
- More robust is Huber's function defined by $\rho'(t) = \max(-c, \min(c, t))$, where c is the tuning constant, a common choice is $c = 4.685$.

Smoothing Spline

Outline:

- Smoothing splines is a technique to automatically tune the number and location of the knots in splines.
- We start with the definition of cubic splines and natural cubic splines.
- Then we will show that natural cubic spline is a solution of an interesting optimization problem.
- We employ this optimization problem to design a new algorithm for fitting splines and tune their parameters.

From piecewise polynomial regression to cubic spline

Recall the piecewise polynomial regression:

- We break the interval into several smaller sub-intervals, on which the function g can be approximated well with a polynomial.
- Then, we fit a polynomial to the data in each subinterval.
- The procedure is exhibited in Figure below.
- We also stated that piecewise polynomial estimation is a special form of the basis expansion.
- The points that are the intersection of consecutive intervals are called knots.
- For instance, ζ_1 and ζ_2 are the knots of the piecewise polynomial function that is shown in Figure below.

Two main issues that moved us away from the piecewise polynomial regression were:

1. Our final estimate of the function, $\hat{g}$, is not necessarily smooth (even if the function g is smooth). In fact, it may be discontinuous (or non-differentiable) at the knots.
2. It is not clear how to set the number of knots and their locations.

From piecewise polynomial regression to cubic spline

Assume that the knots are given by an oracle and our task is to fit a good piecewise polynomial with knots at the given locations.

- Can we fit a piecewise polynomial function that is continuous, or differentiable of certain order? Yes, we can!
- Consider the example shown in Figure below.
  - As is clear from this figure, the fitted function has the form of $\hat{g}(x) = P_1(x)\mathbb{I}(0 \leq x < \zeta_1) + P_2(x)\mathbb{I}(\zeta_1 \leq x < \zeta_2) + P_3(x)\mathbb{I}(\zeta_2 \leq x)$.
  - As is also clear from this figure the fitted curve is not continuous at the knots.
- Can we fix the discontinuities at the knots?
- Theoretically speaking it seems very simple.
- For instance, we can say that we are only interested in piecewise polynomial functions that are continuous at the knots, i.e.,

$$P_1(\zeta_1) = P_2(\zeta_1),\ P_2(\zeta_2) = P_3(\zeta_2). \tag{2}$$

- We can also ensure that the function is differentiable or two times differentiable by imposing

$$P_1'(\zeta_1) = P_2'(\zeta_1),\ P_2'(\zeta_2) = P_3'(\zeta_2) \tag{3}$$

and

$$P_1''(\zeta_1) = P_2''(\zeta_1),\ P_2''(\zeta_2) = P_3''(\zeta_2), \tag{4}$$

respectively.

From piecewise polynomial regression to cubic spline

However, the main challenge is to figure out a way to fit such functions, piecewise polynomials with certain smoothness constraints, to the given data points $\{(x_i, y_i)\}_{i=1}^n$.

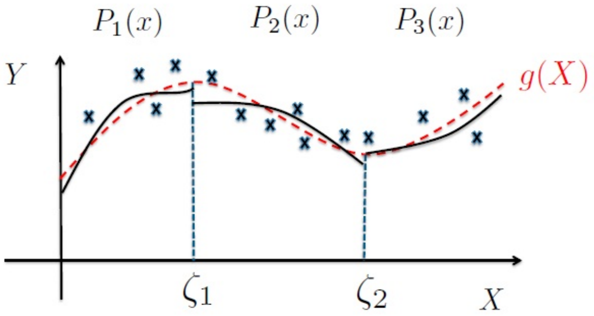


Figure : Piecewise polynomial curve fitting. The red dashed-curve shows the actual curve we would like to estimate. The black curve show our estimate of the red curve by using piecewise polynomial function. As is clear from the figure, we have two knots (ζ_1, ζ_2) in this example. The three polynomials that are fitted are called $P_1(x)$, $P_2(x)$, and $P_3(x)$. Therefore, our final estimate can be written as $\hat{g}(x) = P_1(x)\mathbb{I}(0 \le x < \zeta_1) + P_2(x)\mathbb{I}(\zeta_1 \le x < \zeta_2) + P_3(x)\mathbb{I}(\zeta_2 \le x)$.

From piecewise polynomial regression to cubic spline

It turns out that this problem can be casted as a special case of basis expansion problem and hence can be solved by the general method. Here is a "not" very accurate version of the theorem that enables us to convert this problem into a basis expansion problem.

Theorem (Optional)

Let $h(x)$ be a piecewise polynomial function with knots at $\zeta_1, \zeta_2, \ldots, \zeta_k$ and polynomial of degree at most M. Also, assume that the function satisfies certain smoothness constraints of the forms specified in (2), (3), and (4), or their generalizations. Then, there exist a finite number of functions $b_1(x), b_2(x), \ldots, b_\ell(x)$ such that

1. *Every $b_i(x)$ satisfies all the smoothness constraints.*
2. $h(x) = \sum_{i=1}^{\ell} \beta_i b_i(x)$,

where all β_js are numbers in $\mathbb{R}$ and their values depend on the function h.Furthermore, any linear combination of the form $\sum_j \alpha_j b_j(x)$ will be a piecewise polynomial function with the required smoothness properties.

Note that the functions $b_1(x), b_2(x), \ldots, b_\ell(x)$ are independent of the choice of h and only depend on the knots, the maximum degree of polynomials M, and the smoothness constraints. To help you understand this theorem, let us consider a few examples and prove this result in those special cases.

From piecewise polynomial regression to cubic spline

Example

Let $h(x)$ be a piecewise linear function with only one knot and a continuity constraint at the knot. Prove that this function can be written as a linear combination of three bases $b_1(x) = 1$, $b_2(x) = x$, and $b_3(x) = (x - \zeta_1)\mathbb{I}(x \geq \zeta_1)$. Furthermore, show that every function of the form $\gamma_0 b_1(x) + \gamma_1 b_2(x) + \gamma_3 b_3(x)$ is a continuous piecewise linear function.

From piecewise polynomial regression to cubic spline

Proof.

Before we start the proof, note that all the basis functions b_1, b_2 and b_3 can be regarded as continuous piecewise linear functions with one knot at ζ_1.
We only prove the first part and leave the second part as an exercise for you. Let h be a piecewise linear function with the continuity constraint. This means that the function can be written as

$$h(x) = (\alpha_0 + \alpha_1 x)\mathbb{I}(x \leq \zeta_1) + (\beta_0 + \beta_1 x)\mathbb{I}(x > \zeta_1).$$

Also the continuity constraint implies that $\alpha_0 + \alpha_1\zeta_1 = \beta_0 + \beta_1\zeta_1$. We now start simplifying h.

$$\begin{aligned} h(x) &= (\alpha_0 + \alpha_1 x)\mathbb{I}(x \leq \zeta_1) + (\alpha_0 + \alpha_1 x)\mathbb{I}(x > \zeta_1) - (\alpha_0 + \alpha_1 x)\mathbb{I}(x > \zeta_1) + (\beta_0 + \beta_1 x)\mathbb{I}(x > \zeta_1) \\ &= \alpha_0 + \alpha_1 x + (\beta_0 - \alpha_0 + (\beta_1 - \alpha_1)x)\mathbb{I}(x > \zeta_1) \overset{a}{=} \alpha_0 + \alpha_1 x + ((-\beta_1 + \alpha_1)\zeta_1 + (\beta_1 - \alpha_1)x)\mathbb{I}(x > \zeta_1) \\ &= \alpha_0 + \alpha_1 x + (\beta_1 - \alpha_1)(x - \zeta_1)\mathbb{I}(x > \zeta_1). \end{aligned} \tag{5}$$

Equality (a) is due to the continuity constraint that implies $\alpha_0 + \alpha_1\zeta_1 = \beta_0 + \beta_1\zeta_1$. As is clear from the last expression h is written as a linear combination of $b_1(x)$, $b_2(x)$, and $b_3(x)$. □

From piecewise polynomial regression to cubic spline

There are two important instances of the piecewise polynomial regression with smoothness constraint that are very popular in curve fitting. One is called *cubic spline* and the other one is called *natural cubic spline*:

Definition (Cubic Spline)

Cubic spline is a piecewise polynomial function with the following properties

1. Degree of each polynomial is at most three.
2. The function is continuous, and two times differentiable.

Definition (Natural Cubic Spline)

Natural cubic spline is a cubic spline with an extra constraint that the boundary pieces are linear. In fact all the polynomial pieces have degree 3 except the first one and the last one that have degree 1.

From piecewise polynomial regression to cubic spline

- Let us emphasize again that both of these functions can be written as a linear combination of certain bases and
- hence in case we would like to fit them by using the maximum likelihood principle (under Gaussianity of the noise), we can use the basis expansion formulation.

In summary, by imposing smoothness constraints at the knots we could address the first problem of piecewise polynomial regression, which was the non smoothness of the the final estimate.

What about the second problem: the problem of selecting the knots?

Intuitive discussion of knots

- Suppose that we would like to use piecewise polynomial function with smoothness constraint to estimate a smooth function g from its noisy observation.
- Once the number of knots and their locations are given, we can fit the model using the general method.
- However, the main challenge is to decide how many knots to consider and how to set their locations.
- Let's start with a simple example:
  - Suppose that, once we set the number of knots to M, we set their locations to $\zeta_1 = \frac{1}{M+1}, \zeta_2 = \frac{2}{M+1}, \ldots, \zeta_m = \frac{M}{M+1}$.
  - We will discuss if this is a good choice or not later.
  - But, at this point we would like to discuss the impact of the number of knots under this simple scenario.
  - Our claim is that as we increase the number of knots in this situation, we increase the variance of the estimator and decrease the bias. Can you intuitively argue why this is true?

Intuitive discussion of knots

- In general, we do not consider equispaced knots, since in most cases they are not optimal.
- However, for many strategies of setting the knots we see the same trade-off; as the number of knots increases the bias decreases and the variance increases.
- In order to find the optimal value of M, we can use any of the model selection techniques which we discussed including cross validation.

Intuitive discussion of knots

- A more difficult issue is to set the knots' locations (which can have very important effect on the performance of the final estimate).
- First suppose that we only have one knot. How should we set its location?
- This can be considered as a model selection problem and we can optimally set it by CV.
    - We consider several different values for ζ_1.
    - Fit a piecewise polynomial function to the data and "calculate" the error of the resulting estimate. The value of ζ_1 that leads to the smallest error (obtained from cross validation) is considered as the best location for the knot.
- Can we extend this approach to more knots? Theoretically, yes. If for example, we have two knots, we consider several different values for each knot.
- Using a nested loop we evaluate the performance of the estimate for any combination of ζ_1 and ζ_2 and choose the pair that gives us the minimum error.
- However, as the number of knots increases the computational complexity of this approach grows rapidly and make this approach computationally infeasible.
- It turns out that, if we would like to solve the problem of allocation knots optimally, there is no better way that can significantly reduce the computational complexity.
- There are several heuristic methods to address this issue. But we do not discuss them here.
- Instead, we will talk about "smoothing spline" which is an indirect method of doing this.

Smoothing spline

- We start with a totally new formulation of the problem of curve fitting that seems to have nothing in common with splines at first.
- But at the end of this lecture we will establish the connection of this new framework with spline and then we see how we can use splines efficiently without worrying about the location or number of knots.
- Let $\{(x_i, y_i)\}_{i=1}^n$ denote our data points, where $y_i = g(x_i) + \epsilon_i$. Let $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$.
- Under this model, the first idea that comes to mind for estimating g is the maximum likelihood approach.
- The maximum likelihood estimate is given by

$$\hat{g} = arg \max_h f(y_1, y_2, \ldots, y_h; h) = arg \max_h \frac{1}{(2\pi)^{\frac{n}{2}} \sigma^n} e^{-\sum_{i=1}^n (y_i - h(x_i))^2 / 2\sigma^2} = arg \max_h \sum_{i=1}^n (y_i - h(x_i))^2. \quad (6)$$

Is this a good estimate? Apparently no.

Smoothing spline

- There are many functions h, that satisfy $y_i = h(x_i)$ for every single i.
- Therefore, this optimization problem does not have a unique solution.
- This says that unless we have some restriction on the functions g that we would like to estimate, we cannot hope to estimate them well by maximum likelihood.
- One of the main restrictions that we have considered in this course is the smoothness of g.
- For instance, we discussed the MSE of kernel regression under the assumption that g is differentiable and its derivative is bounded.
- Here, we assume that g is two times differentiable and $\boxed{\int g''(t)^2 dt \leq C}$ where C is some constant in $\mathbb{R}$.
- Furthermore, note that we are now imposing a more global notion of smoothness.
- The function might be less smooth in some regions and more smooth in other regions.
- This is different from $|g''(t)| \leq C$ that ensures locally the function is smooth.
- With this constraint on the function g we can rewrite the maximum likelihood estimation in the following way:

$$\hat{g} = \arg \min_{\{h: \int h''(t)^2 dt \leq C\}} \sum_{i=1}^{n} (y_i - h(x_i))^2. \tag{7}$$

Smoothing spline

- Using some basic results in convex optimization one can show that for every value of C, there exists a value of λ for which the solution of (7) is the same as the solution of the following optimization problem:

$$\boxed{\hat{g} = arg\,\min_h \sum_{i=1}^{n} (y_i - h(x_i))^2 + \lambda \int h''(t)^2 dt.} \tag{8}$$

penalize if deviates from linearity

Smoothing spline: properties

The first interesting property of this optimization problem is that *under very general conditions, such as $\lambda \neq 0$, (8) has a unique minimizer*. It is also straightforward to see what the solution looks like in the two extreme cases of $\lambda = 0$ and $\lambda = \infty$.

(i) For $\lambda = 0, \hat{g}$ interpolates the data, i.e., $y_i = \hat{g}(x_i)$ for every i. Do you see why?

(ii) For $\lambda = \infty$, $\hat{g}''(t) = 0$. Therefore, $\hat{g}(x) = \alpha_0 + \alpha_1 x$.

Try to explain why each statement is true.

Smoothing spline: properties

We will discuss the impact of λ on the estimate and its bias and variance later. First we will describe how we can solve the optimization problem of (8) to obtain $\hat{g}$. The following theorem that you will prove in the next Homework plays a pivotal role in characterizing the solution of (8).

Theorem (Green and Silverman 1994)

Suppose that $n \geq 2$. Then the solution of (8) has the form of a natural cubic spline with knots at $x_1, x_2, \ldots, x_n$.

Smoothing spline: properties

As is clear, this theorem does not care about the value of λ. In fact, it claims that no matter what λ is, the solution will have the form of a natural cubic spline with knots at $x_1, x_2, \ldots, x_n$. Suppose that $N_1(x), N_2(x), \ldots, N_\ell(x)$ are the basis functions for the natural cubic spline. Then according to Theorem 2, $\hat{g}$ can be written as

$$\hat{g}(x) = \beta_1 N_1(x) + \beta_2 N_2(x) + \ldots + \beta_\ell N_\ell(x).$$

It is important to note that the functions $N_1(x), N_2(x), \ldots, N_\ell(x)$ are the same for different values of λ. However, $\beta_1, \ldots, \beta_\ell$ may change as λ changes.

Smoothing spline: properties

Therefore, the next question we would like to address is on how to choose $\beta_1, \beta_2, \ldots, \beta_\ell$ such that $\hat{g}(x) = \beta_1 N_1(x) + \beta_2 N_2(x) + \ldots + \beta_\ell N_\ell(x)$ minimizes (8).

If we plug $h(x) = \beta_1 N_1(x) + \beta_2 N_2(x) + \ldots + \beta_\ell N_\ell(x)$ in (8) the optimization problem can be casted as

$$\min_{\beta_1,\ldots,\beta_\ell} \sum_{j=1}^{n} (y_j - \beta_1 N_1(x_j) + \beta_2 N_2(x_j) + \ldots + \beta_\ell N_\ell(x_j))^2 + \lambda \int ((\beta_1 N_1(x) + \beta_2 N_2(x) + \ldots + \beta_\ell N_\ell(x))'')^2 dx$$

Smoothing spline: properties

By taking the derivative with respect to βs and setting them to zero one can obtain the optimal solution of the optimization problem.

In fact in the homework you will prove the following:

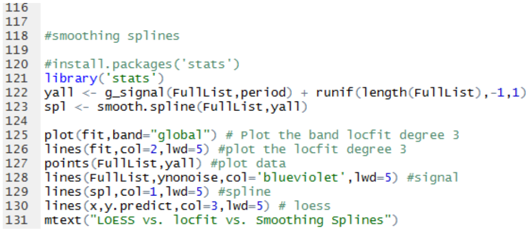
Define two matrices $\mathbf{N}$ as $\mathbf{N}_{ij} = N_j(x_i)$ and Ω with $\Omega_{ij} = \int N_i''(t)N_j''(t)dt$. Furthermore, define $\mathbf{y} = [y_1, \ldots, y_n]^T$ and $\hat{\beta} = [\hat{\beta}_1, \ldots, \hat{\beta}_\ell]^T$. Then

$$\boxed{\hat{\beta} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega)^{-1}\mathbf{N}^T\mathbf{y}.}$$

Just Like Ridge Reg

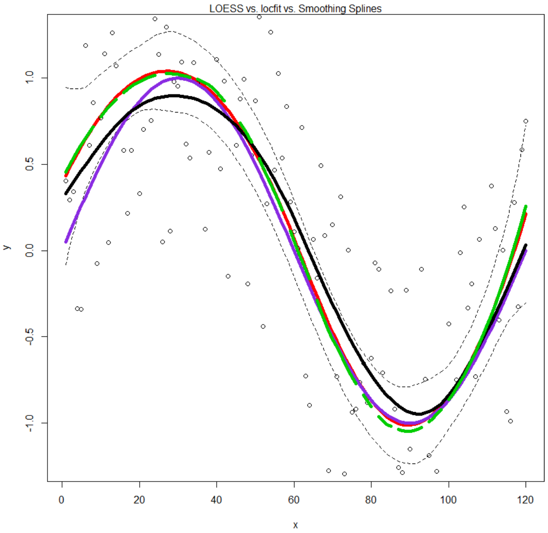
Therefore, as you can see, finding the smoothing spline is as easy as solving the basis expansion problem.

Example in R: LOESS vs. locfit vs. Smoothing Splines



```


#smoothing splines

#install.packages('stats')
library('stats')
yall <- g_signal(FullList,period) + runif(length(FullList),-1,1)
spl <- smooth.spline(FullList,yall)

plot(fit,band="global") # Plot the band locfit degree 3
lines(fit,col=2,lwd=5) #plot the locfit degree 3
points(FullList,yall) #plot data
lines(FullList,ynonoise,col='blueviolet',lwd=5) #signal
lines(spl,col=1,lwd=5) #spline
lines(x,y.predict,col=3,lwd=5) # loess
mtext("LOESS vs. locfit vs. Smoothing Splines")
```

Example in R: LOESS vs. locfit vs. Smoothing Splines



Bias, variance trade-off

- It seems that the solution of the smoothing spline, which is a natural cubic spline with knots at every single data point, must suffer from overfitting.
- Since we have too many knots (One knot for every data point).
- This in fact is true when λ is small.
- However, the free parameter λ can control the bias and variance of the smoothing spline.
- This will be more clear if we look at the other extreme. As we discussed before, the limiting solution as $\lambda \to \infty$ is a linear function.
- Hence, this solution usually suffers from a high bias.
- In general as λ increases the bias increases and the variance decreases.
- Again there is a sweet spot for λ that gives us the minimum error.
- To estimate the optimal value of λ we can use any of the model selection techniques, including the cross validation.

We can even theoretically analyze the bias and variance of the estimator. The interested readers may refer to the book by Hastie,Tibshirani, and Friedman.

Spline in $\mathbb{R}^p$: Thin-plate spline

So far we have seen smoothing spline in the case where we have only one predictor variable x.

In many situations we have more than one predictor. The data is in the form of $\{(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(p)}), y_i)\}_{i=1}^n$, where $y_i = g(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(p)}) + \epsilon_i$.

Can we extend the smoothing spline to such situations? Theoretically speaking - yes. Again we would like to minimize

$$\min_h \sum_{i=1}^n (y_i - h(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(p)}))^2 + \lambda J[h], \tag{9}$$

where J[h] is a certain measure of smoothness. For instance, for $p = 2$ we have

$$J[h] = \int\int \left(\frac{\partial^2 h(x^{(1)}, x^{(2)})}{\partial (x^{(1)})^2} + \frac{\partial^2 h(x^{(1)}, x^{(2)})}{\partial (x^{(2)})^2} + \frac{\partial^2 h(x^{(1)}, x^{(2)})}{\partial x^{(1)} \partial x^{(2)}} \right)^2 dx_1 dx_2.$$

This optimization is called *thin plate spline*.

Spline in $\mathbb{R}^p$: Thin-plate spline

- The optimization problem that is defined in (8) can also be solved and it has a nice solution.
- The only problem is that the computational complexity of solving the above optimization problem grows as n^3 as the number of samples increases.
- There are several heuristic methods that help reduce the computational complexity of these algorithms.
- For further information on the thin plate spline you may refer to the book by Hastie, Tibhsirani, Friedman and the reference therein.

Splines versus locally polynomial regression

- The smoothness constraint that we imposed in (8) is global, i.e., we are concerned with the overall behavior of the function, rather than its behavior at a single point.
- At certain points the second derivative can be large on small intervals and $\int h''(t)dt$ be still small.
- This type of smoothness constraint leads to a solution that employs every single data point.
- This is different from locally polynomial regression in which the estimate at point x only depends on the data points in the neighborhood of x.
- Based on this discussion we can make the following conclusions:

(i) Smoothing spline can lead to more accurate solutions specially when we do not have many data points. This is both because the conditions it assumes on the function is weaker and also because it can use all the data points efficiently in its estimate at point x.

(ii) The computational complexity of smoothing spline is usually much higher than that of locally polynomial regression. That is why it is not used very often for dimensions higher than 2.

References

- T. Hastie, R. Tibshirani, and J. Friedman, The elements of statistical learning.
- H. Akaike, "A new look at the statistical model identification," IEEE Trans. Automatic control, Dec. 1974.
- E. Lehman, J. Romano, "Testing statistical hypothesis," Springer.
- J. Ye, "On measuring and correcting the effects of data mining and model selection," J. Amer. Stat. Assoc. , vol. 93, 1998.
- B. Efron, "Covariance penalties and cross validation," J. Amer. Stat. Assoc. , vol. 99, 2004.
- J. Friedman and W. Stuetzle, "Smoothing of scatter plots".