Nonparametric (kernal) regression

Jacob M. Montgomery

2017

Nonparametric regression models

Where are we?

- We have done parametric regression
- MLE and Bayesian and (a little) frequentist

Where are we?

- We have done parametric regression
- MLE and Bayesian and (a little) frequentist
- Today we will look at estimating the relationship between x and y nonparametrically.
- But to get there, we need to establish a few more fundamentals about nonparametric inference.

Bias variance tradeoff

- Let's go back to imagining we only care about estimating a single variable.
- Let $g(x)$ be the true (unknown) density of x and $\hat{g}(x)$ be our estimate of the density.
- We want to estimate the density of x, such that we minimize the **risk** (essentially squared error loss integrated over x).

$$L(g, \hat{g}) = \int (g(u) - \hat{g}_n(u))^2 du$$

Bias variance tradeoff

- Let's go back to imagining we only care about estimating a single variable.
- Let $g(x)$ be the true (unknown) density of x and $\hat{g}(x)$ be our estimate of the density.
- We want to estimate the density of x, such that we minimize the **risk** (essentially squared error loss integrated over x).

$$L(g, \hat{g}) = \int (g(u) - \hat{g}_n(u))^2 du$$

$$R(g, \hat{g}) = E\left(L(g, \hat{g})\right)$$

Bias variance tradeoff

- Let's go back to imagining we only care about estimating a single variable.
- Let $g(x)$ be the true (unknown) density of x and $\hat{g}(x)$ be our estimate of the density.
- We want to estimate the density of x, such that we minimize the **risk** (essentially squared error loss integrated over x).

$$L(g, \hat{g}) = \int (g(u) - \hat{g}_n(u))^2 du$$

$$R(g, \hat{g}) = E\left(L(g, \hat{g})\right)$$

$$R(g, \hat{g}) = \int Bias^2 + \int Var$$

For some specific value of x:

$$b(x) = E(\hat{g}_n(x)) - g(x)$$

For some specific value of x:

$$b(x) = E(\hat{g}_n(x)) - g(x)$$

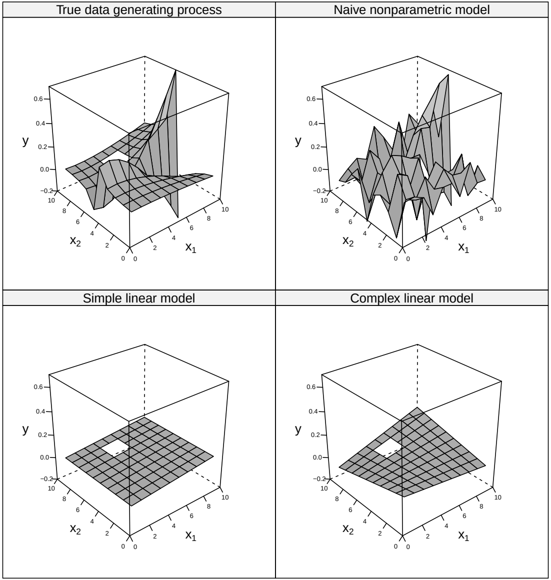
$$v(x) = V(\hat{g}(x)) = E\left([\hat{g}_n(x) - E(\hat{g}(x))]^2\right)$$

$$R(g, \hat{g}) = \int b^2(x)dx + \int v(x)dx$$

The signal and the noise

X_1 \ X_2	1	2	3	4	5	6	7	8	9	10
10	4	9	6	9	4	5	6	5	5	3
9	7	8	4	7	4	13	12	5	4	3
8	5	3	7	7		8	6	8	3	6
7	2	4	6	6	5	13	12	4	5	5
6	7	5	6	8	4	7	4	6	4	5
5	2	6	11	11	2	11	4	2	3	6
4	5	6	6	5	3	9	5	4	3	5
3	7	6	11	5	5	4	2	4	8	5
2	4	7	8	7	10	7	9	8	2	7
1	8	10	5	7	5	5	13	2	9	7

The signal and the noise



Understaning the bias/variance tradeoff

- When you have a simple model, the variance will always be low.
- When you have an extremely "accurate" model, you are going to have a lot of variability.

Estimating risk

- We know that the risk is equal to:

$$L(g, \hat{g}) = \int (g(x) - \hat{g}(x))^2 dx$$

$$= \int g^2(x) dx + \int \hat{g}^2(x) dx - 2 \int \hat{g}(x) g(x) dx$$

- To choose the amount of smoothing we want, we need to minimize this.

Estimating risk

- We know that the risk is equal to:

$$L(g, \hat{g}) = \int (g(x) - \hat{g}(x))^2 dx$$

$$= \int g^2(x)dx + \int \hat{g}^2(x)dx - 2\int \hat{g}(x)g(x)dx$$

- To choose the amount of smoothing we want, we need to minimize this. But note that the first term does not depend on my handling of the data.
- So, letting p be our smoothing parameters, we need to minimize:

$$J(p) = \int \hat{g}^2(x)dx - 2\int \hat{g}(x)g(x)dx$$

LOO CV estimator of risk

- We can (sometimes) calculate the first part fairly explicitly.
- But we are going to estimate the second part using "leave-one-out" cross validation.
- For some smoothing parameter s, we then get:

$$\hat{J}(s) = \int (\hat{g}(x))^2 \, dx - \frac{2}{n} \sum_{i=1}^{n} \hat{g}_{(-i)}(X)$$

where $\hat{g}_{(-i)}(X)$ is the estimator obtained after removing the i^{th} observation.

Kernal density estimation

- Let's start off by defining a Gaussian kernal

$$K(x) = (2\pi)^{-1/2} e^{-x/2}$$

- But it could be lot's of kinds of kernals for the following method.
- Then for each potential value of x, we estimate the density as

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x - X_i}{h}\right)$$

- Choice of kernal is often not that important.
- Choice of h is *very* important.

Class exercise

- Take the data points below and the bandwidth, $h = 1$, and create a kernal density plot

$$-2, -1, -1, 0, .5, .75, 1.1, 2.5$$

- Try this for different values of h

Kernal regression

- We can also create confidence intervals as follows.
- At each potential value x, we calculate the lower ($l(x)$) and upper ($u(x)$) bounds.
  - $l(x) = \hat{f}(x) - q\ se(x)$
  - $u(x) = \hat{f}(x) + q\ se(x)$
- $se(x) = \frac{s(x)}{\sqrt{n}}$
- $$s^2(x) = \frac{1}{n-1} \sum \left(Y_i(x) - \bar{Y}(x) \right)^2$$
- $$Y_i(x) = \frac{1}{h} K \left(\frac{x - X_i}{h} \right)$$

- $$q = \Phi\left(\frac{1+(1-\alpha)^{1/m}}{2}\right)$$
- $m = \frac{b-a}{\omega}$
- For the gaussian, it will be $\omega = 3h$
- I will ask you to calculate this in the problem set

Getting where we were going

The general problem in a regression framework remains figuring out how to estimate the model:

$$\hat{r}(x) = \sum_{i=1}^{n} w_i(x) Y_i$$

- $K()$ is a kernal and
- $$w_i(x) = \frac{K\left(\frac{x-X_i}{h}\right)}{\sum_{j=1}^{n} K(\frac{x-X_j}{h})}$$

Getting where we were going

The general problem in a regression framework remains figuring out how to estimate the model:

$$\hat{r}(x) = \sum_{i=1}^{n} w_i(x) Y_i$$

- $K()$ is a kernal and
- $$w_i(x) = \frac{K\left(\frac{x - X_i}{h}\right)}{\sum_{j=1}^{n} K(\frac{x - X_j}{h})}$$
- Conceptually this is:

$$r(x) = E(Y|X = x) = \frac{\int y f(x, y) dy}{\int f(x, y) dy}$$

- So this is "non-parametric," but as a practical matter we still need to choose h.

- So this is "non-parametric," but as a practical matter we still need to choose h.
- We can choose the bandwidth (or any other tuning parameter) by minimizing the cv score:

$$\hat{J}(h) = \sum_{i=1}^{n} (Y_i - \hat{r}_{-i}(x_i))^2$$

- So this is “non-parametric,” but as a practical matter we still need to choose h.
- We can choose the bandwidth (or any other tuning parameter) by minimizing the cv score:

$$\hat{J}(h) = \sum_{i=1}^{n} (Y_i - \hat{r}_{-i}(x_i))^2$$

- This can be approximated directly as:

$$\hat{J}(h) = \sum_{i=1}^{n} (Y_i - \hat{r}(x_i))^2 \frac{1}{\left(1 - \frac{K(0)}{\sum_{j=1}^{n} K(\frac{x_i - x_j}{h})}\right)^2}$$

Confidence bounds

- When in doubt, we can alway bootstrap
- For time series data (where Y is ordered), we can first estimate:

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

Confidence bounds

- When in doubt, we can alway bootstrap
- For time series data (where Y is ordered), we can first estimate:

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

- $\hat{se}(x)$ is then

$$\hat{\sigma} \sqrt{\sum_{i=1}^{n} w_i^2(x)}$$

- $$q = \Phi^{-1}\left(\frac{1 + (1-\alpha)^{1/m}}{2}\right)$$

- $$m = \frac{b-a}{\omega}$$

Local polynomial regression

- OK. So now that we have this in hand, let's take it one step further and look at local polynomial regression.

Local polynomial regression

- OK. So now that we have this in hand, let's take it one step further and look at local polynomial regression.
- Remember that we can approximate some function $f(x)$ when it is close to x_0 as using a Taylor series as:

$$f(x) \approx f(x_0) + \frac{f^{(1)}(x_0)(x - x_0)^1}{1!} + \frac{f^{(2)}(x_0)(x - x_0)^2}{2!} + \ldots + \frac{f^{(p)}(x_0)(x - x_0)^p}{p!}$$

$$= \beta_0 + \beta_1(x - x_0) + \beta_2(x - x_0)^2 + \ldots + \beta_p(x - x_0)^p$$

- To estimate this, we focus on the neighborhood right around x_0
- We minimize with respect to $\sum(Y_i - \hat{r}(x))^2$:

$$\sum_{i=1}^{n} [Y_i - \beta_0 - \beta_1(x_i - x_0) - \ldots - \beta_p(x_i - x_0)^p]^2 \, K\left(\frac{x_i - x_0}{h}\right)$$

- In essence, we are going to conduct a weighted least squares regression where the weights are determined by the proximity of observation x_i and x_0.
- First, define:

$$\mathbf{X}_{x_0} = \begin{pmatrix} 1 & (x_1 - x_0) & \dots (x_1 - x_0)^p \\ 1 & (x_2 - x_0) & \dots (x_2 - x_0)^p \\ \vdots & & \vdots \\ 1 & (x_n - x_0) & \dots (x_n - x_0)^p \end{pmatrix}$$

- Second, we define weights based on some kernal:

$$\mathbf{X}_{x_0} = \begin{pmatrix} K((x_1 - x_0)/h) & 0 & \dots & 0 \\ 0 & K((x_2 - x_0)/h) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & K((x_n - x_0)/h) \end{pmatrix}$$

- Then we need to minimize:

$$(\mathbf{Y} - \mathbf{X}_{x_0}\beta)'\mathbf{W}_{\mathbf{x_0}}(\mathbf{Y} - \mathbf{X}_{x_0}\beta)$$

- Leads to the usual weighted least squares estimator:

$$\hat{\beta} = (\mathbf{X}'_{x_0}\mathbf{W}_{\mathbf{x_0}}\mathbf{X}_{x_0})^{-1}(\mathbf{X}_{x_0}\mathbf{W}_{\mathbf{x_0}}\mathbf{Y})$$

- Finally, for any given value of x_0, we can approsimate the function as:

$$\hat{f}(x_0) = \mathbf{e}'_1(\mathbf{X}'_{x_0}\mathbf{W}_{\mathbf{x_0}}\mathbf{X}_{x_0})^{-1}(\mathbf{X}_{x_0}\mathbf{W}_{\mathbf{x_0}}\mathbf{Y})$$

where $\mathbf{e}'$ is a vector of length $p + 1$ (degree of polynomials) and the first element is 1 and the rest zeros.

- When $p = 0$, this is just the kernal regression we discussed earlier.

- When $p = 0$, this is just the kernal regression we discussed earlier.
- When $p = 1$ it is a local "linear" (why?) regression. and we get:

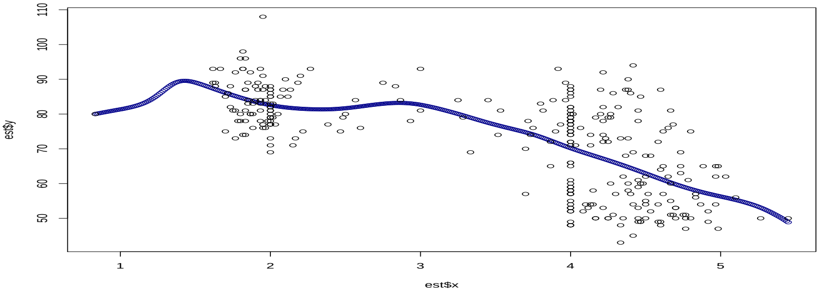
$$\hat{f}(x_0) = \frac{1}{n} \sum_{i=1}^{n} \frac{(\hat{s}_2 - \hat{s}_1)(x_i - x_0)K((x_i - x_0)/h)Y_i}{\hat{s}_2\hat{s}_0 - \hat{s}_1^2}$$

- where
$$\hat{s}_j = \frac{\sum_{i=1}^{n}(x_i - x_0)^j K((x_i - x_0)/h)}{n}$$

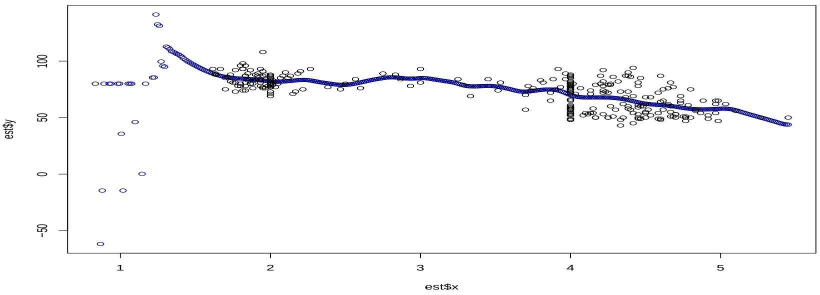
```r
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

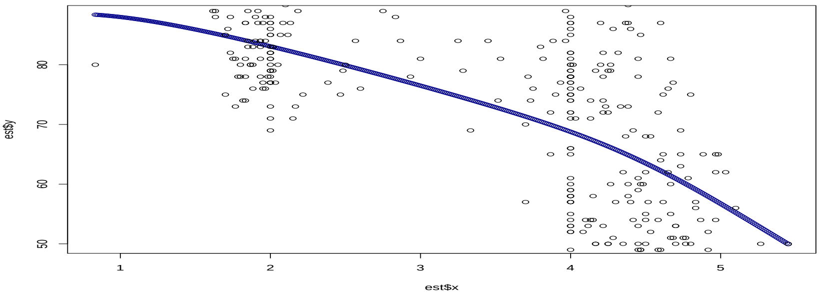
```r
data(geyser, package="MASS")
est<-locpoly(x=geyser$duration, y=geyser$waiting, bandwidth=0.25)
plot(est, ylim=c(min(geyser$waiting), max(geyser$waiting)), col="darkblue")
points(x=geyser$duration, y=geyser$waiting)
```



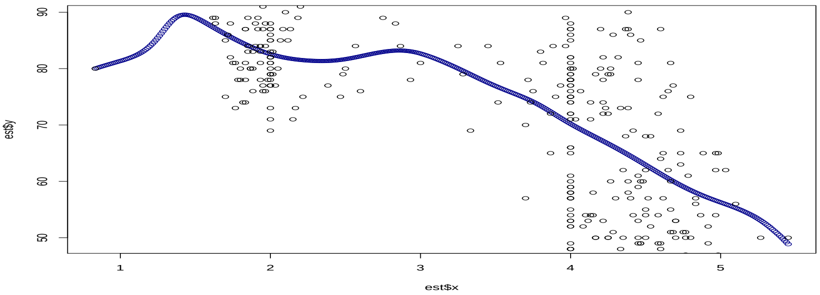
```r
library(KernSmooth)
data(geyser, package="MASS")
est<-locpoly(x=geyser$duration, y=geyser$waiting, bandwidth=0.1)
plot(est, col="darkblue")
points(x=geyser$duration, y=geyser$waiting)
```



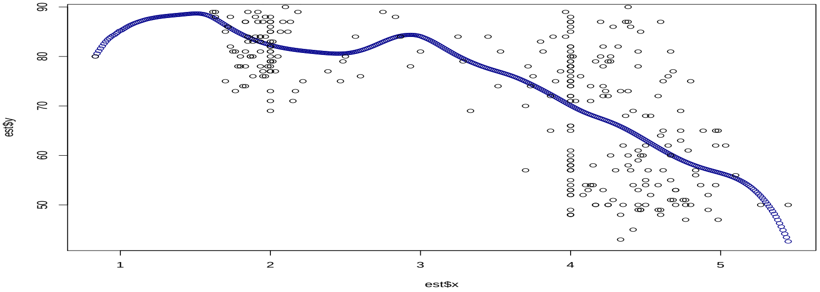
```r
est<-locpoly(x=geyser$duration, y=geyser$waiting, bandwidth=1)
plot(est, col="darkblue")
points(x=geyser$duration, y=geyser$waiting)
```



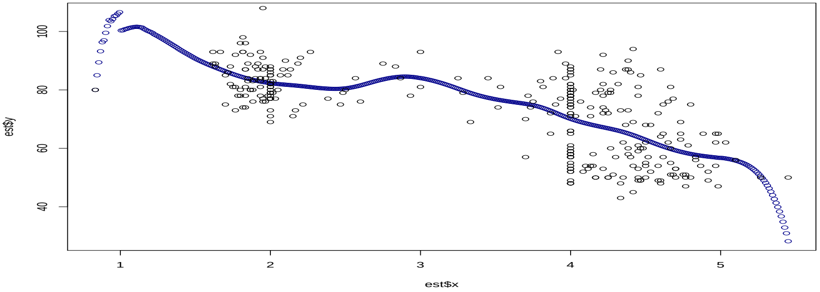
```r
est<-locpoly(x=geyser$duration, y=geyser$waiting, bandwidth=.25, degree=1)
plot(est, col="darkblue")
points(x=geyser$duration, y=geyser$waiting)
```



```r
est<-locpoly(x=geyser$duration, y=geyser$waiting, bandwidth=.25, degree=2)
plot(est, col="darkblue")
points(x=geyser$duration, y=geyser$waiting)
```



```
est<-locpoly(x=geyser$duration, y=geyser$waiting, bandwidth=.25, degree=3)
plot(est, col="darkblue")
points(x=geyser$duration, y=geyser$waiting)
```



Your problem set

- You will be looking and using “kernal regularized least squares” in your problem set.

Your problem set

- You will be looking and using "kernal regularized least squares" in your problem set.
- With more than one covariate, we can set up the Gaussian kernal as:

$$k(x_j, k_i) = \exp\left(\frac{\|x_j - x_i\|^2}{\sigma^2}\right)$$

Your problem set

- You will be looking and using "kernal regularized least squares" in your problem set.
- With more than one covariate, we can set up the Gaussian kernal as:

$$k(x_j, k_i) = \exp\left(\frac{\|x_j - x_i\|^2}{\sigma^2}\right)$$

- $$f(x) = \sum_{i=1}^{N} c_i k(x, x_i)$$

- $$y = K_c = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \ldots k(x_1, x_N) \\ k(x_2, x_1) & & \\ \vdots & & \\ k(x_N, x_1) & \ldots & k(x_N, x_N) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

- In practice, this model is going to wildly over-fit the data. So we add a “regularization” parameter that penalizes complexity.
- $$\operatorname{argmin} \sum_i \left(\hat{f}(x_i) - y_i\right) + \lambda \|\hat{f}\|^2$$
- $$c^* = \underset{c \in R^D}{\operatorname{argmin}} (y - Kc)'(y - Kc) + \lambda c' K c$$