STATS 205: Introduction to Nonparametric Statistics

Lecturer: Tengyu Ma
Scribe: Tiffany Cheng

Lecture #2
October 4th, 2019

1 Review

1.1 Regressogram

Last lecture, we covered the regressogram, which involves dividing a range (a, b) into m equally spaced bins denoted by $B_1, B_2, ..., B_m$. Suppose that $a \leq x_i \leq b$ for all $i = 1, ..., n$, then we can define the regressogram estimator, $\hat{r}(x)$, as being:

$$\hat{r}(x) = \frac{1}{k_j} \sum_{i:x_i \in B_j} Y_i, \qquad \text{for } x \in B_j \tag{1}$$

where k_j is the number of points in B_j.

Written in words, the regressogram estimator is a step function created by averaging the Y_is in each bin.

1.2 Kernel Estimator (Nadayara-Watson)

Previously, we also discussed kernel regression estimators, which take a weighted average of the Y_i's, giving higher weight to those points near x.

Recall that we find $\hat{r}(x)$ by solving:

$$\hat{r}(x) = \underset{c}{\operatorname{argmin}} \sum_{j=1}^{n} w_j (Y_j - c)^2 = \frac{\sum_{j=1}^{n} w_j Y_j}{\sum_{j=1}^{n} w_j} \tag{2}$$

where $w_j = K(\frac{x - x_j}{h})$, K is a kernel, and h is a positive number called the bandwidth.

We also talked specifically about the **Nadayara-Watson kernel estimator**, which is defined as:

$$\hat{r}(x) = \sum_{i=1}^{n} l_i(x) Y_i, \qquad \text{where } l_i(x) = \frac{K(\frac{x - x_i}{h})}{\sum_{j=1}^{n} K(\frac{x - x_j}{h})} \tag{3}$$

1.3 Bias-Variance Tradeoff

Finally, we discussed how the *Mean Squared Error (MSE)* can be decomposed into bias and variance:

$$
\begin{aligned}
MSE = Risk &= bias^2 + variance \\
&= \mathbb{E}_{y_i}[\frac{1}{n}\sum_{i=1}^{n}(\hat{r}(x_i) - r(x_i))^2] \\
&= \frac{1}{n}\sum_{i=1}^{n}(E_{y_i}[(\hat{r}(x_i)] - r(x_i))^2 + \frac{1}{n}\sum_{i=1}^{n} Var(\hat{r}(x_i))
\end{aligned} \tag{4}
$$

Recall that the bias term is the error incurred from smoothing on clean data, while the variance stems from noise.

We also discussed how changes to bandwidth and the number of data points used affect bias and variance:

- A **bigger bandwidth** (i.e., h) $\rightarrow$ more continuity and smoothing assumptions. This leads to **larger bias and smaller variance** because you're taking into account more data points.
- In contrast, a **smaller bandwidth** $\rightarrow$ **smaller bias but larger variance**.
- If you fix the bandwidth but increase n, your bias will not change drastically. This is because bias does not depend on noise, but instead it depends on how smooth your function is. Therefore, **increasing** n $\rightarrow$ **the same bias and decreased variance**. However, since you have changed the balance between bias and variance (i.e., variance is smaller and less important now), you will want to re-balance these two terms by decreasing bandwidth.

2 Theorem for Rate

2.1 Risk/MSE of Kernel Estimator

Risk depends on our values for $x_1, ..., x_n$. During last week's lecture, we treated x as being fixed. However for the theory below, let's assume $x_1, ..., x_n \overset{IID}{\sim} D$, where D has density $f(x)$ and $n \rightarrow \infty$. Also, we define $\hat{r}_n$ as the estimator obtained using n samples and bandwidth h_n.

Let's say that we want to summarize risk over different values of x (as opposed to computing risk at a specific point x). We can accomplish this goal by using **integrated risk** or **integrated mean squared error**, which is defined as:

$$
R(\hat{r}_n, r) = \int (\hat{r}_n(x) - r(x))^2 dx. \tag{5}
$$

Recall that $r(x)$ is the ground truth and that it exists over the entire space.

Based on the above assumptions and definitions, we introduce:

Theorem - *the risk (using integrated squared error loss) of the Nadaraya Watson kernel estimator is*:

$$R(\hat{r}_n, r) = \frac{h_n^4}{2}\left(\int x^2 K(x)dx\right)^2 \int \left(r''(x) + 2r'(x)\frac{f'(x)}{f(x)}\right)^2 dx + \sigma^2 \frac{\int K^2(x)dx}{nh_n}\int \frac{1}{f(x)}dx \; + \; o(nh_n^{-1}) \; + \; o(h_n^4) \quad (6)$$

as $h_n \to 0$ and $nh_n \to \infty$.

Note that the first term in (6) is squared bias while the second term is variance.

Let's take a closer look at the components of the bias and variance terms and interpret how they contribute to the bias and variance:

Interpreting the Components of Bias

- $\frac{h_n^4}{2} \to$ this tells us that bias depends on the bandwidth, h. As bandwidth decreases, bias decreases too.
- $\int x^2 K(x)dx \to$ this term captures the flatness of the kernel. Note that a flatter kernel will have a larger bias because it averages over points further away from x.
- $r''(x) \to$ how much this term contributes to the overall bias depends on $r(x)$. If r(x) is constant (i.e., a flat line), then $r''(x) = 0$ and it won't contribute to the bias term. However, if we have a function like the sine function, then $r''(x)$ will be larger and the bias will be larger.
- $2r'(x)\frac{f'(x)}{f(x)} = 2r'(x)(logf(x))' \to$ we call this term the **design bias** since it depends on the design (i.e., the distribution of the x_i's). In other words, the bias is sensitive to the position of the x_i's. A subset of design bias is **boundary bias**, which is the high bias that kernel estimators have near boundaries. This is a motivating reason for using local linear smoothing, which we will discuss later in lecture.

Interpreting the Components of Variance

- $\sigma^2 \to$ variance depends on σ. Recall that $\sigma^2 = Var(\xi_i)$.
- $\frac{1}{nh_n} \to$ the larger the bandwidth, h_n, the smaller the variance term.
- **Choice of bandwidth:** The variance term provides guidance on how to choose a bandwidth. If we rewrite the theorem above (i.e., equation 6) so that the non-bandwidth terms are abstracted away into constants, we get:

$$R(\hat{r}_n, r) = h_n^4 c_1 + \frac{c_2}{nh_n} \; + \; \text{lower order term} \quad (7)$$

Let us then find the **optimal bandwidth**, h_n^*, by differentiating (6) and setting the result to 0. We get:

$$\begin{aligned} h_n^* &= \underset{h_n}{\text{argmin}}\ h_n^4 c_1 + \frac{c_2}{n h_n} \\ &= (\frac{c_2}{4c_1 n})^{1/5} = (\frac{c_3}{n})^{1/5} \\ &= O(n^{-1/5}) \end{aligned} \tag{8}$$

If we plug h_* back into (6), we find that the risk decreases at rate $O(n^{-4/5})$. Note that in most parametric methods, excess risk (defined as our risk - the optimal risk) decreases to 0 at a rate of $\frac{1}{n}$. However, non-parametric methods have a slower rate of $n^{-4/5}$.

2.2 Linear Smoothers

Earlier we mentioned that the boundary bias in kernel estimators is a motivation for considering local linear smoothing. Before we dive into that topic, we need to first introduce terminology regarding linear smoothers.

Definition: An estimator $\hat{r}$ of r is a **linear smoother** if, for each x, there exists a vector $l(x) = (l_1(x), ..., l_n(x))$ such that:

$$\hat{r}(x) = \sum_{i=1}^{n} l_i(x) Y_i \tag{9}$$

where l can depend on $x_1, ..., x_n$ but not on $Y_1, ..., Y_n$.

- For a **regressogram**, $l_j(x) = \mathbb{1}\{x, x_j \text{ are in the same window}\}$.
- For a **kernel**, $l_j(x) = K(\frac{x_j - x}{h}) / \sum_{i=1}^{n} K(\frac{x_i - x}{h})$.

Let's define our vector of fitted values to be:

$$r = (\hat{r}(x_1), ..., \hat{r}(x_n))^T \tag{10}$$

where vector $Y = (Y_1, ..., Y_n)^T$. Then it follows that:

$$r = LY \tag{11}$$

where L is a n x n matrix whose entry $L_{ij} = l_j(x_i)$. Therefore, the entries of the i^{th} row indicate the weights given to each Y_i when forming the estimate $\hat{r}(x_i)$.

Definition: The matrix L is called a **smoothing matrix** for linear smoothers.

2.3 Local Linear Smoothing

Recall that the kernel method assumes a locally constant function: $\sum_{i=1}^{n} w_i (Y_i - c)^2$.

Now let's assume a locally linear model instead. If we fix x, let's approximate $r(u)$ as:

$$r(u) \approx P_x(u;\ a) = a_1(u - x) + a_0 \tag{12}$$

We want to find $\hat{a}_1$ and $\hat{a}_0$ such that

$$\begin{aligned} \hat{a}_1, \hat{a}_0 &= \underset{a_1, a_0}{\operatorname{argmin}} \sum_{j=1}^{n} w_j (Y_j - (a_1(x_j - x) + a_0))^2 \\ &= \underset{a_1, a_0}{\operatorname{argmin}} \sum_{j=1}^{n} w_j (Y_j - P_x(x_j, a)^2 \end{aligned} \tag{13}$$

Define $u_j = x_j - x$. If we solve for the argmin above, we get:

$$a_0 = \frac{\sum_{j=1}^{n} b_j Y_j}{\sum_{j=1}^{n} b_j} \tag{14}$$

where $b_j = w_j \left(\sum_{i=1}^{n} w_j u_i^2 - u_j \sum_{i=1}^{n} w_i u_i \right)$ and $w_j = K(\frac{x_j - x}{h})$.

$$\hat{r}(x) = a_0 = \sum_{i=1}^{n} l_i(x) Y_i \tag{15}$$

where $l_i(x) = \frac{b_i(x)}{\sum_{j=1}^{n} b_j(x)}$.

Therefore, $\hat{r}(x)$ is still a linear smoother because it is linear in Y.

2.4 Local Polynomial Smoothing

Let's also consider the more complex polynomial scenario. Again, we fix x and our goal is to find $\hat{r}(x)$.

We approximate $r(u)$ locally with:

$$r(u) \approx P_x(u;\ a) = a_0 + a_1(u - x) + ... + \frac{a_p}{p!}(u - x)^p \tag{16}$$

The estimation is accomplished by finding a value of $a = (a_0, ..., a_p)^T$, say, $\hat{a} = (\hat{a}_0, ..., \hat{a}_p)^T$, which minimizes the locally weighted sum of squares

$$\begin{aligned} (\hat{a}_0, ..., \hat{a}_p) &= \underset{a_0, ..., a_p}{\operatorname{argmin}} \sum_{j=1}^{n} w_j \left(Y_j - P_x(x; a) \right)^2 \\ &= \underset{a_0, ..., a_p}{\operatorname{argmin}} \sum_{j=1}^{n} w_j \left(Y_j - a_0 - a_1(x_j - x), ..., -\frac{a_p}{p!}(x_j - x)^p \right)^2 \\ &= \underset{a_0, ..., a_p}{\operatorname{argmin}} \sum_{j=1}^{n} w_j \left(Y_j - a^T z_j \right)^2 \end{aligned} \tag{17}$$

where $z_j = (1,\ x_j - x, ...,\ \frac{(x_j - x)^p}{p!})^T$

Similar to local linear smoothing, we find that our estimate, $\hat{r}(x)$ is a linear smoother.

2.5 Theorem [Fan '92]

Written informally, the **theorem** says:

1. The asymptotic **variance** of local linear regression/smoothing is the same as the kernel estimator:
$$\sigma^2 \frac{\int K^2(x)dx}{nh_n} \int \frac{1}{f(x)} dx \tag{18}$$

2. Meanwhile, the **bias** of local linear regression is:
$$\frac{h_n^4}{2} \left(\int x^2 K(x)dx \right)^2 \int r''(x)^2 dx \tag{19}$$

   Recall that the N-W kernel estimator has the bias:
$$\frac{h_n^4}{2} \left(\int x^2 K(x)dx \right)^2 \int \left(r''(x) + 2r'(x)\frac{f'(x)}{f(x)} \right)^2 dx \tag{20}$$

   so the only difference is that we don't have the second term in the second integral anymore.

The above theorem also extends more generally to local polynomial regressions of order p. Note that having an odd p reduces design bias and boundary bias without increasing variance.

2.6 How do we selection the best h and p empirically?

Our challenge here is that we only have one draw of $Y_1, ..., Y_n$. Therefore, how do we estimate MSE? Recall that:

$$MSE = \mathbb{E}_Y \left[\frac{1}{n} \sum_{i=1}^{n} (\hat{r}(x_i) - r(x_i))^2 \right] \tag{21}$$

We cannot use the training error because when our bandwidth $h = 0$, our training error $= 0$. Recall that the **training error** is defined as:

$$\frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{r}(x_i))^2 \tag{22}$$

Note that:

$$\begin{gathered} \text{Predictive Risk} \neq \text{MSE} \\ \mathbb{E}_Y \left[\frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{r}(x_i))^2 \right] \neq \mathbb{E}_Y \left[\frac{1}{n} \sum_{i=1}^{n} (\hat{r}(x_i) - r(x_i))^2 \right] \end{gathered} \tag{23}$$

This is because on the left side $\hat{r}(x)$ depends on the Ys while on the right side $\hat{r}(x)$ does not depend on $r(x)$.

Instead, what we do is set aside a **validation set**. We split $(x_1, y_1), ..., (x_n, y_n)$ into separate training and validation sets using a random permutation: $(i_1, ..., i_n)$. For example, say

that we use $(x_{i1}, y_{i1}), ..., (x_{im}, y_{im})$ for training, and we use $(x_{im+1}, y_{im+1}), ..., (x_{in}, y_{in})$ for validation.
Deciding what percentage of the data to put in the training set vs. validation set can be a bit arbitrary. However, commonly used splits are $m = \frac{9}{10}n$ and $m = \frac{4}{5}n$.

3 Cross-Validation

Cross-validation in the non-parametric scenario is very similar to its counterpart in the parametric world.

3.1 Leave-One-Out Estimate

The **leave-one-out cross-validation score** is defined by:

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{r}_{(-i)}(x_i)\right)^2 \tag{24}$$

where $\hat{r}_{(-i)}$ is the estimator obtained from the data set with (x_i, y_i) removed.

Recall the formula for a linear smoother:

$$\hat{r}(x) = \sum_{j=1}^{n} l_j(x)Y_j \tag{25}$$

In the case of **LOOCV**, we have:

$$\hat{r}_{(-i)}(x_i) \triangleq \frac{\sum_{j\neq i} l_j(x_i)Y_j}{\sum_{j\neq i} l_j(x_i)} \tag{26}$$

3.2 Theorem

If $\hat{r}$ is a linear smoother, then $\hat{r}(x) = \sum l_j(x)Y_j$. Let's also assume that the weights sum to 1, i.e., $\sum_{j=1}^{n} l_j(x) = 1$.

In this case, the LOOCV score, $\hat{R}(h)$ can be written as:

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{r}(x_i)}{1 - l_i(x_i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{r}(x_i)}{1 - L_{ii}}\right)^2 \tag{27}$$

where $L_{ii} = l_i(x_i)$ is the i^{th} diagonal element of the smoothing matrix, L.

This is nice because we only need to use the data set once and don't need to recompute the estimator after dropping out each observation.