Nonparametric Estimation

Uses

- Data visualization and exploration
- Estimating a function without knowing function structure

How? examples...

- Kernel Density Estimation,
- Histograms
- Local regression (lowess, loess)
- Smoothing
Why?

Besides tools for exploring data, can yield a deeper understand of the trade-offs at play with fitting a model to data.
Besides tools for exploring data, can yield a deeper understand of the **trade-offs** at play with fitting a model to data.
Conceptually: As a model produces less variance (i.e. by penalizing the beta coefficients), it increases bias.
Bias, Variance Tradeoff

Conceptually: As a model produces less variance (i.e. by penalizing the beta coefficients), it increases bias.

too much bias: underfit

too much variance: overfit
Formally,

\( \hat{g}_n \) -- estimator of true function, \( g \) (regression or density)

bias: \[ b(x) = \mathbb{E}(\hat{g}_n(x)) - g(x) \]

variance: \[ v(x) = \mathbb{V}(\hat{g}_n(x)) = \mathbb{E}((\hat{g}_n(x)) - \mathbb{E}(\hat{g}_n(x))^2) \]
Bias, Variance Tradeoff

Formally,

\( \hat{g}_n \) -- estimator of true function,  \( g \)  (regression or density)

Risk: \[ R(g, \hat{g}_n) = \int b^2(x)dx + \int v(x)dx \]

bias: \[ b(x) = \mathbb{E}(\hat{g}_n(x)) - g(x) \]

variance: \[ v(x) = \mathbb{V}(\hat{g}_n(x)) = \mathbb{E}((\hat{g}_n(x)) - \mathbb{E}(\hat{g}_n(x))^2) \]
Bias, Variance Tradeoff

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

\[
b(x) = \mathbb{E}(\hat{g}_n(x)) - g(x)
\]

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

**FIGURE 20.2.** The Bias-Variance trade-off. The bias increases and the variance decreases with the amount of smoothing. The optimal amount of smoothing, indicated by the vertical line, minimizes the risk = bias^2 + variance.

(Wasserman, 2005, AoS)
Bias, Variance Tradeoff

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

\[ b(x) = \mathbb{E}(\hat{g}_n(x)) - g(x) \]

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

\[ \text{Risk} = \text{Bias}^2 + \text{Variance} \]

**FIGURE 20.2.** The Bias-Variance trade-off. The bias increases and the variance decreases with the amount of smoothing. The optimal amount of smoothing, indicated by the vertical line, minimizes the risk = bias$^2$ + variance.

(Wasserman, 2005, AoS)
Bias, Variance Tradeoff

Risk = Bias^2 + Variance

Conceptually:
As a model produces less variance (i.e. by penalizing the beta coefficients), it increases bias.

(Wasserman, 2005, AoS)
Nonparametric Regression

Regression: \( Y_i = r(x_i) + \epsilon_i \)

Nadaraya-Watson kernel estimator:

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

where weights are given by \((K \text{ is a kernel})\):

\[
w_i(x) = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right)}
\]

(Wasserman)
Nonparametric Regression

Regression: \( Y_i = r(x_i) + \epsilon_i \)

Nadaraya-Watson kernel estimator:

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

where weights are given by (\( K \) is a kernel):

\[
w_i(x) = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x-x_i}{h}\right)}
\]
Nonparametric Regression

Regression: \( Y_i = r(x_i) + \epsilon_i \)

Nadaraya-Watson kernel estimator:

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

where weights are given by (\( K \) is a kernel):

\[
w_i(x) = \frac{K(\frac{x-x_i}{h})}{\sum_{j=1}^{n} K(\frac{x-x_i}{h})}
\]

(Hastie et al., 2009)
Nonparametric Regression

Nadaraya-Watson kernel estimator:

$$\sum_{j=1}^{n} K\left(\frac{x-x_i}{h}\right)$$

(Hastie et al., 2009)
Nonparametric Regression

Regression: \( Y_i = r(x_i) + \epsilon_i \)

Local Linear Regression:

\[ \hat{r}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0 \]

(Hastie et al., 2009)
Nonparametric Regression

Regression: $Y_i = r(x_i) + \epsilon_i$

Local Linear Regression:

$\hat{r}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0$

Local regression coefficients

$x_0$: a range of points around a given $x_i$;
e.g. the 30 nearest neighbors

(Hastie et al., 2009)
Nonparametric Regression

Regression:
\[ Y_i = r(x_i) + \epsilon_i \]

Local Linear Regression:
\[ \hat{r}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0 \]

Objective: Minimize weighted least squares:
\[ \sum_{i=1}^{n} K_\lambda(x_0, x_i)[y_i - \beta_0(x_0) - \beta_1(x_0)x_i]^2 \]

(Hastie et al., 2009)
Nonparametric Regression

Regression: \( Y_i = r(x_i) + \epsilon_i \)

Local Linear Regression:

\[ \hat{r}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0 \]

Objective: Minimize weighted least squares:

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

Full solution for local linear regression with normal equations for weighted least squares:

\[ \hat{r}(x_0) = b(x_0)^T (\beta^T W(x_0) \beta)^{-1} \beta^T W(x_0)y \]

(Hastie et al., 2009)
Nonparametric Regression

Regression:  \( Y_i = r(x_i) + \epsilon_i \)

Local Linear Regression:

\[ \hat{r}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0 \]

Objective: Minimize weighted least squares:

\[ \sum_{i=1}^{n} K_{\lambda}(x_0, x_i)[y_i - \hat{\beta}_0(x_0) - \hat{\beta}_1(x_0)x_i]^2 \]

Full solution for local linear regression with normal equations for weighted least squares:

\[ \hat{r}(x_0) = b(x_0)^T (\beta^T W(x_0) \beta)^{-1} \beta^T W(x_0)y \]

(Hastie et al., 2009)
Other Nonparametric Methods

Histogram

Kernel Density Estimation

The Bootstrap

```python
#compute the bootstrap:

iters = 5000  # number of iterations
all_means = []  # will store the sample mean per iteration
# run the simulation loops:
for i in range(iters):
    resample = np.random.choice(sample, size=n, replace=True)
    resample_mean = resample.mean()
    all_means.append(resample_mean)

# sort the resampled means from least to greatest:
sorted_means = sorted(all_means)
# pick the upper and lower values for 95% CI:
lower = sorted_means[int(0.025*iters)]
upper = sorted_means[-int(0.025*iters)]
print("95 CI based on the bootstrap: [%3.3f, %3.3f]" % (lower, upper))
```

95 CI based on the bootstrap: [19.239, 19.626]