Nonparametric Statistics: Homework 6

蒋翌坤 20307100013

Solution to Problem 1

From the data, we have $n = 1425$. Using cross-validation method, we have,

$$\hat{J}(h) = \int \left(\hat{f}_n(x)\right)^2 dx - \frac{2}{n}\sum_{i=1}^{n} \hat{f_{(-i)}}(X_i)$$

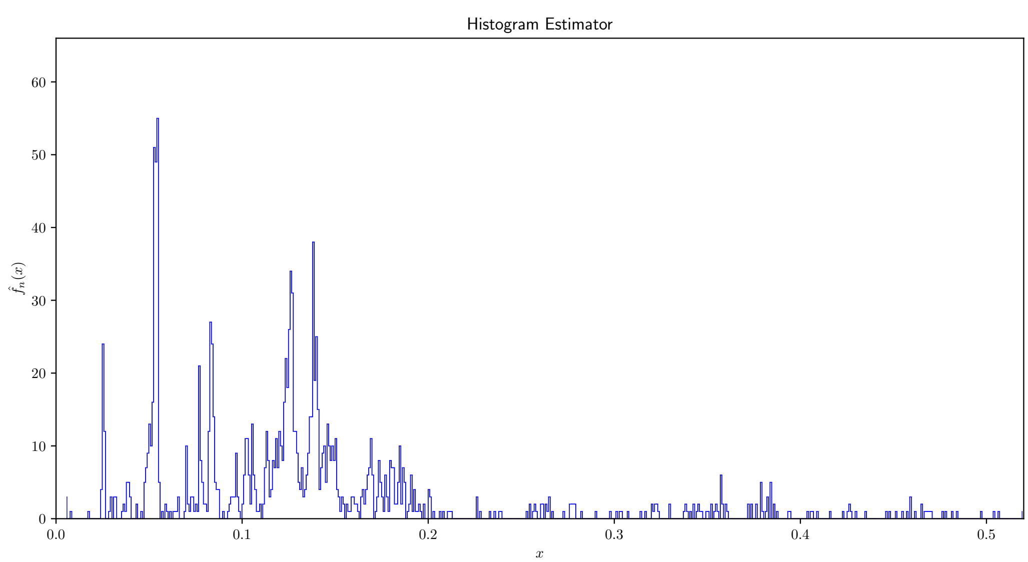
Histogram estimator

Here, $\hat{f}_n(x)$ is like the histogram of observations with $m = \frac{1}{h}$ bins. We also have,

$$\hat{J}(h) = \frac{2}{h(n-1)} - \frac{n+1}{h(n-1)}\sum_{j=1}^{m} \hat{p_j}^2$$

We can not deduce the optimal h from the formula, but, we can enumerate each m. In this way, we can find the minimum $\hat{J}(h)$ and the corresponding m. In this case, $\hat{J}(h)^* = -5.99$ and $m^* = \frac{1}{h^*} = 597$

We can therefore plot the graph of $\hat{f}_n(x)$,



Kernal estimator

Here, We use the Gaussian kernel: $K(x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{x^2}{2}\}$.

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

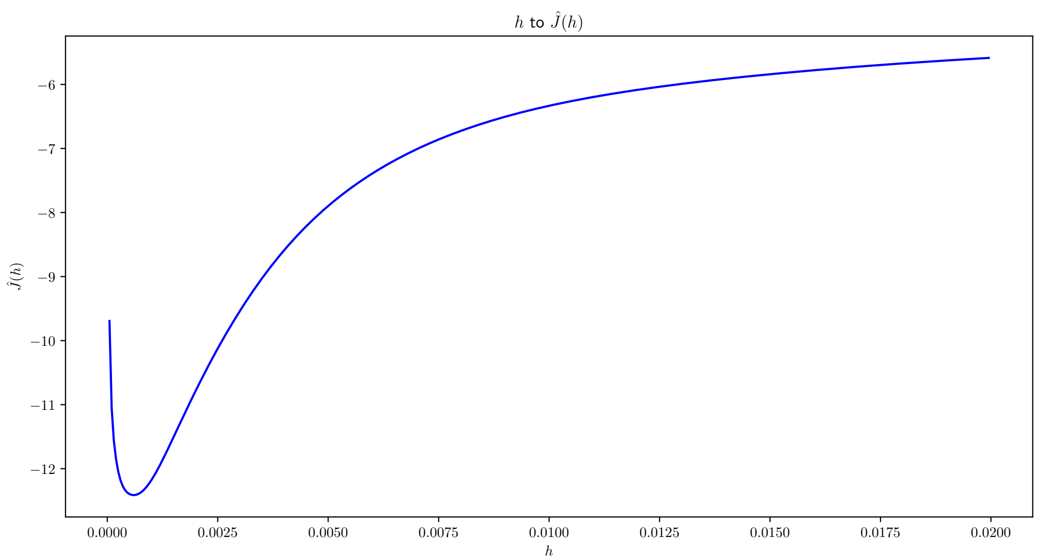
Then,

$$\begin{aligned}
\hat{J}(h) &= \frac{1}{hn^2} \sum_i \sum_j K^*\Big(\frac{X_i - X_j}{h}\Big) + \frac{2}{nh} K(0) + O(\frac{1}{n^2}) \\
&= \frac{1}{hn^2} \sum_i \sum_j \Big[K^{(2)}\Big(\frac{X_i - X_j}{h}\Big) - 2K\Big(\frac{X_i - X_j}{h}\Big)\Big] + \frac{2}{nh\sqrt{2\pi}} + O(\frac{1}{n^2})
\end{aligned}$$

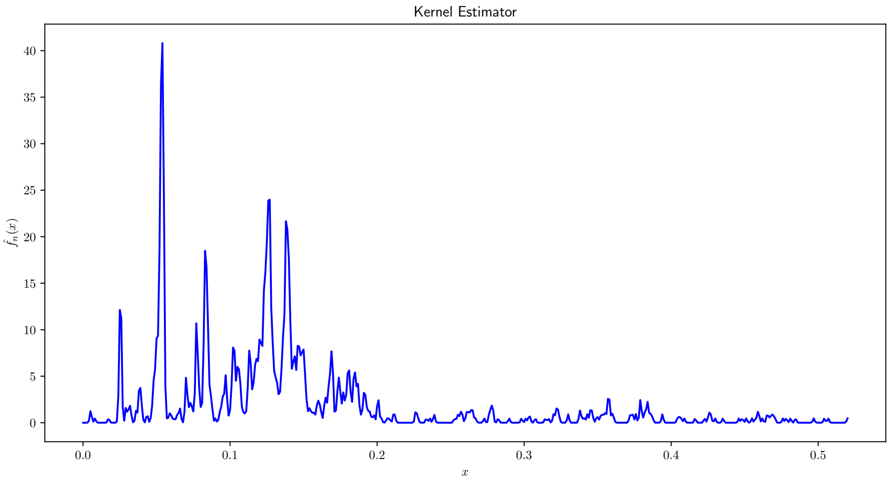
where,

$$K^{(2)}(x) = \frac{1}{2\sqrt{2\pi}} \exp\Big\{-\frac{x^2}{8}\Big\}$$

We can therefore plot the graph of $\hat{J}(h)$ with respect to h,



From the graph, we can find the minimum $\hat{J}(h)$ and the corresponding m. In this case, $\hat{J}(h)^* \approx -12.41$ and $h^* \approx 0.0006$, We can therefore plot the graph of $\hat{f}_n(x)$ using $h^* = 0.006$,



Normal reference rule

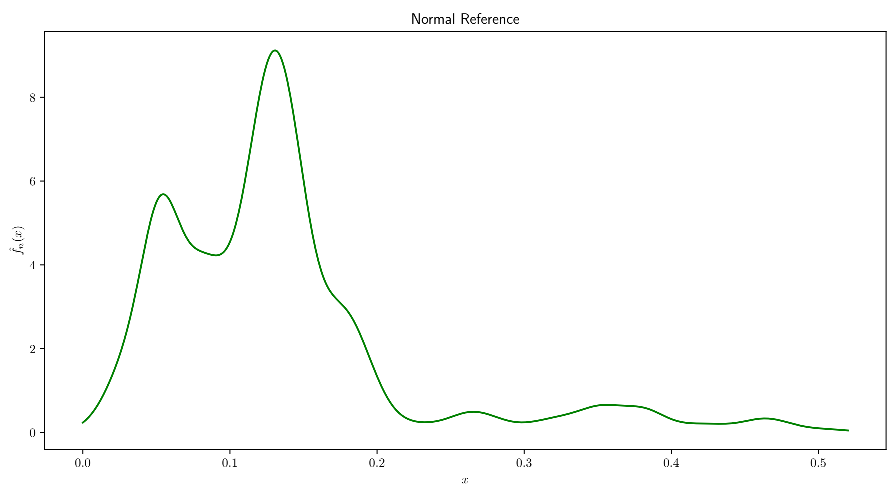
We can also apply Normal reference rule to pick a bandwidth for the kernel,

$$\hat{\sigma} = \min\left\{s, \frac{\text{Sample Interquantile range}}{1.34}\right\} = \min\left\{0.089, 0.053\right\} = 0.053$$

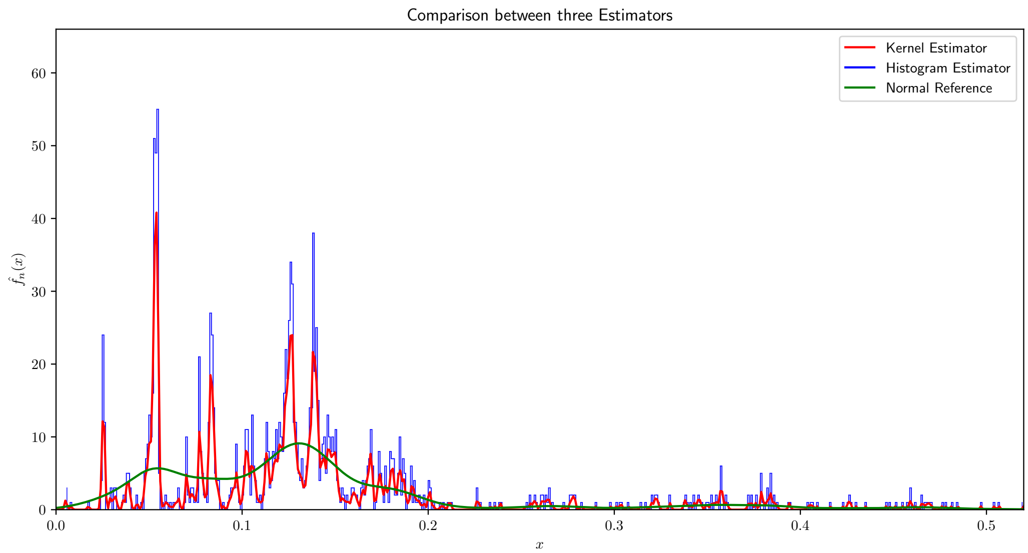
$$h_n = \frac{1.06\hat{\sigma}}{n^{1/5}} = 0.0131$$

Here, h_n is very different from the optimal h^* we get from enumeration.

We can therefore plot the graph of $\hat{f}_n(x)$ using $h_n = 0.0131$,



We can combine three graphs together, and get,



Appendix: Code

Python code and output used to solve the problem can also be found at: https://thisiskunmeng.github.io/nonparametric/hw6.html

The following provides Python code I use to complete this homework:

```
import numpy as np
import pandas as pd
import requests
from io import StringIO
import matplotlib.pyplot as plt
data = requests.get("https://www.stat.cmu.edu/~larry/all-of-nonpar/=data/galaxy.dat", verify=False)
df = pd.read_csv(StringIO(data.text), sep="\s+", header=None, names=["ra", "declination", "redshift"])
redshift = df["redshift"]
print(len(redshift))
```

```
1425
```

Histogram estimator

```
def get_jh_hist(m, n, dat):
    yj = np.histogram(dat, bins=m)[0]
    pj = yj / n
    first = 2 * m / (n - 1)
    second = (n + 1) * m / (n - 1)
    third = np.sum(pj ** 2)
    return first - second * third

def get_jh_hist_all(dat):
    jh = []
    for i in range(len(dat)):
        jh.append(get_jh_hist(i + 1, len(dat), dat))
    return jh

jh_all = get_jh_hist_all(redshift)
print(np.min(jh_all))
m_star = np.argmin(jh_all) + 1
print(m_star)
```

```
-5.989204025439116
597
```

```
plt.rcParams["text.usetex"] = True
fig, ax = plt.subplots(figsize=(12, 6))
height = np.histogram(redshift, bins=m_star)[0] / len(redshift)
x = (np.histogram(redshift, bins=m_star)[1])
```

```python

for i in range(1, m_star-1):
    ax.add_line(plt.Line2D([x[i], x[i+1]], [height[i], height[i]], color='blue', linewidth=0.5))
    ax.add_line(plt.Line2D([x[i], x[i]], [height[i - 1], height[i]], color='blue', linewidth=0.5))
    ax.add_line(plt.Line2D([x[i+1], x[i+1]], [height[i], height[i + 1]], color='blue', linewidth=0.5))
ax.set_xlim(0, np.max(redshift))
ax.set_ylim(0, 1.2 * np.max(height))
ax.set_xlabel(r"$x$")
ax.set_ylabel(r"$\hat{f_n}(x)$")
ax.set_title(r"Histogram Estimator")
plt.savefig("./fig/histogram_estimator.pdf")
```

Kernal estimator

```python
def kernel_function(xx):
    return 1 / np.sqrt(2 * np.pi) * np.exp(-xx ** 2 / 2)

def kernel_function_2(xx):
    return 1 / (2 * np.sqrt(2 * np.pi)) * np.exp(-xx ** 2 / 8)

def hat_f(xx, hx, dat):
    return 1 / len(dat) * np.sum(kernel_function((xx - dat) / hx) / hx)

datx, daty = np.meshgrid(redshift, redshift)
datx_x = datx - daty

def jh_kernel(hx, dat):
    n = len(dat)
    the_sum = np.sum(kernel_function_2(datx_x / hx) - 2 * kernel_function(datx_x / hx))
    the_jh = 1 / (hx * n ** 2) * the_sum + 2 / (hx * n * np.sqrt(2 * np.pi))
    return the_jh

step = 0.00005
hn = np.arange(step, 0.02, step)
jh = []
for i in hn:
    jh.append(jh_kernel(i, redshift))
jh = np.array(jh)

print(np.min(jh))
print(np.argmin(jh))
print((np.argmin(jh) + 1) * step)
```

```
-12.412517973369557
11
0.0006000000000000001
```

```python
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(hn, jh, color="blue")
ax.set_xlabel(r"$h$")
ax.set_ylabel(r"$\hat{J}(h)$")
```

```
ax.set_title(r"$h$ to $\hat{J}(h)$")
plt.savefig("./fig/h_to_jh.pdf")
```

```
h_star = (np.argmin(jh) + 1) * step
fig, ax = plt.subplots(figsize=(12, 6))
x = np.arange(0, np.max(redshift), 0.001)
y = []
for i in x:
    y.append(hat_f(i, h_star, redshift))
ax.plot(x, y, color="blue")
ax.set_xlabel(r"$x$")
ax.set_ylabel(r"$\hat{f_n}(x)$")
ax.set_title(r"Kernel Estimator")
plt.savefig("./fig/kernel_estimator.pdf")
```

```
# Comparison between histogram estimator and kernel estimator
fig, ax = plt.subplots(figsize=(12, 6))
height = np.histogram(redshift, bins=m_star)[0]
x = (np.histogram(redshift, bins=m_star)[1])
x_k = np.arange(0, np.max(redshift), 0.001)
y_k = []
for i in x_k:
    y_k.append(hat_f(i, h_star, redshift))
for i in range(1, m_star-1):
    ax.add_line(plt.Line2D([x[i], x[i+1]], [height[i], height[i]], color='blue', linewidth=0.5))
    ax.add_line(plt.Line2D([x[i], x[i]], [height[i - 1], height[i]], color='blue', linewidth=0.5))
    ax.add_line(plt.Line2D([x[i+1], x[i+1]], [height[i], height[i + 1]], color='blue', linewidth=0.5))
ax.plot(x_k, y_k, color="red", label="Kernel Estimator")
ax.plot(0,0, color='blue', label="Histogram Estimator")
ax.set_xlim(0, np.max(redshift))
ax.set_ylim(0, 1.2 * np.max(height))
ax.legend()
ax.set_xlabel(r"$x$")
ax.set_ylabel(r"$\hat{f_n}(x)$")
ax.set_title(r"Comparison between Histogram Estimator and Kernel Estimator")
plt.savefig("./fig/histogram_and_kernel_estimator.pdf")
```

Normal reference rule

```
s = np.std(redshift)
iqr = np.quantile(redshift, 0.75) - np.quantile(redshift, 0.25)
h = 1.06 * np.min([s, iqr / 1.34]) * len(redshift) ** (-1 / 5)
print(s, iqr / 1.34, h)
```

```
0.0889070103989908 0.05297985074626865 0.0131417627598199
```

```python
fig, ax = plt.subplots(figsize=(12, 6))
x_n = np.arange(0, np.max(redshift), 0.001)
y_n = []
for i in x_n:
    y_n.append(hat_f(i, h_norm, redshift))
ax.plot(x_n, y_n, color="green")
ax.set_xlabel(r"$x$")
ax.set_ylabel(r"$\hat{f_n}(x)$")
ax.set_title(r"Normal Reference")
plt.savefig("./fig/normal_reference.pdf")
```

```python
# Comparisons
fig, ax = plt.subplots(figsize=(12, 6))
for i in range(1, m_star-1):
    ax.add_line(plt.Line2D([x[i], x[i+1]], [height[i], height[i]], color='blue', linewidth=0.5))
    ax.add_line(plt.Line2D([x[i], x[i]], [height[i - 1], height[i]], color='blue', linewidth=0.5))
    ax.add_line(plt.Line2D([x[i+1], x[i+1]], [height[i], height[i + 1]], color='blue', linewidth=0.5))
ax.plot(x_k, y_k, color="red", label="Kernel Estimator")
ax.plot(0,0, color='blue', label="Histogram Estimator")
ax.plot(x_n, y_n, color="green", label="Normal Reference")
ax.set_xlim(0, np.max(redshift))
ax.set_ylim(0, 1.2 * np.max(height))
ax.legend()
ax.set_xlabel(r"$x$")
ax.set_ylabel(r"$\hat{f_n}(x)$")
ax.set_title(r"Comparison between three Estimators")
plt.savefig("./fig/normal_reference_comp.pdf")
```