Problem 1

Consider the nonlinear regression model,

$$y_i = m(x_i) + \sigma\epsilon_i, \quad i = 1, 2, \ldots, n,$$

where $n = 100$ and the data (x_i, y_i) are provided in "HW10_data.csv".

(1) (10 pts)

Using the uniform kernel giving weights of the form

$$w_{i,h}(x) = \frac{\mathbf{1}(|x_i - x| < h)}{\sum_{i=1}^n \mathbf{1}(|x_i - x| < h)},$$

find the estimator of $m(0)$; i.e. $\widehat{m}(0)$, obtained by minimizing

$$\sum_{i=1}^n w_{i,h}(0)(y_i - \theta)^2$$

with $h = 0.1$.

Solution:

$$\widehat{m}(x) = \frac{\sum_{i=1}^n y_i w_{i,h}(x)}{\sum_{i=1}^n w_{i,h}(x)}$$

$$\widehat{m}(0) = \frac{\sum_{i=1}^n y_i w_{i,h}(0)}{\sum_{i=1}^n w_{i,h}(0)} = \sum_{i=1}^n y_i w_{i,h}(0) = -0.1111179$$

is a Nadaraya-Watson estimator.

In [11]:

```python
import pandas as pd
df = pd.read_csv("data.csv")
h = 0.1
x = 0

def w_i_h(x_i, x, h, all_xi):
    if abs(x_i - x) < h:
        denominator = sum([(1 if abs(x_i_el - x) < h else 0) for x_i_el in all_xi])
        return 1 / denominator
    else:
        return 0



# check that weights sum to one
sum([w_i_h(x_i=x_i, x=x, h=h, all_xi = df["x"]) for x_i in df["x"]])
```

Out[11]: 1.0000000000000002

In [18]:
```
m_hat_0 = sum([df.loc[i, "y"] * w_i_h(x_i=df.loc[i, "x"], x=x, h=h, all_xi = df["x"])
               for i in df.index])
m_hat_0
```

Out[18]: -0.11111796963636363

(2) (10 pts)

Write down an expression for and hence find the value of the bias of $\widehat{m}(0)$ if the true function $m(x)$ is linear with slope 1.2.

Solution

$$E\widehat{m}(0) - m(0) = \frac{\sum_{|x-x_i|<h}\left(m\left(x_i\right) - m(0)\right)}{\sum_{|x-x_i|<h} 1} = \frac{\sum_{|x-x_i|<h}\left(1.2x_i\right)}{\sum_{|x-x_i|<h} 1} = -0.0305$$

In [20]:
```
bias = sum([1.2* x_i for x_i in df["x"] if abs(x-x_i) < h]) /
       sum([1 for x_i in df["x"] if abs(x-x_i) < h])
bias
```

Out[20]: -0.030515620690909087

(3) (10 pts)

What is the variance of $\widehat{m}(0)$? Leave your answer in terms of the unknown σ^2.

Solution

$$Var(\widehat{m}(0)) = \sigma^2 \sum_{i=1}^{n} w_{i,h}^2(0) = 0.0909\sigma^2$$

In [21]:
```
sum([w_i_h(x_i=x_i, x=x, h=h, all_xi = df["x"]) ** 2 for x_i in df["x"]])
```

Out[21]: 0.09090909090909091

(4) (10 pts)

Without actually calculating the variance, write down an expression for estimating σ^2.

Solution

$$\widehat{\sigma}^2 = \sum_{i=1}^{100} \frac{\left(\mathrm{y}_i - \hat{m}\left(\mathrm{x}_i\right)\right)^2}{(\mathrm{n}-\mathrm{q})}$$

(5) (10 pts)

If the true value of σ is known to be 0.1, find the 95% confidence interval for the predictive value of y at $x = 0$.

$$Var(\widehat{m}(0)) = 0.0909\sigma^2 = 0.0909 * (0.1^2) = 0.000909$$

$$\sqrt{Var(\widehat{m}(0))} = 0.030$$

$$\widehat{m}(0) \pm c_{\alpha/2}\sqrt{\mathrm{Var}\{\widehat{m}(0)\}} = -0.111 \pm 1.96 * 0.030$$

$$CI = (-0.698, 0.478)$$

In [29]:
```
0.0909 * (0.1) ** 2
```

Out[29]: 0.0009090000000000001

In [28]:
```
-0.11 - 1.96*0.3
```

Out[28]: -0.698

In [27]:
```
-0.11 + 1.96*0.3
```

Out[27]: 0.478

Problem 2

Consider the nonlinear regression model, with $n = 100$ and data (x_i, y_i) provided in "HW10_data.csv" (the same dataset as Problem 1),

$$y_i = m(x_i) + \sigma\epsilon_i, \quad x_i \in (-1, 1), \quad i = 1, \dots, n,$$

and the aim is to split the range of $x \in (-1, 1)$ into two intervals; the start of a regression tree. That is, we are looking for the estimator

$$\widehat{m}(x) = \begin{cases} m_1 & x \leq r \\ m_2 & x > r \end{cases}$$

(1) (10 pts)

Find the optimal choice of r which minimizes the sum of square errors. Show your working and algorithm.

$$T(r) = \sum_{x_i \leq r} (y_i - \widehat{m}_1)^2 + \sum_{x_i > r} (y_i - \widehat{m}_2)^2$$

In [94]:

```
import pandas as pd
df = pd.read_csv("data.csv")

df = df.sort_values("x")
df["split_points"] = (df["x"] + df["x"].shift(-1)) / 2
df["sum_of_square_errors"] = None

for index, row in df.iterrows():
    split_point = row.split_points
    if pd.isna(split_point):
        continue
    left_part = (df.loc[df.x <= split_point, "y"] -
                 df.loc[df.x <= split_point, "y"].mean()) ** 2
    right_part = (df.loc[df.x > split_point, "y"] -
                  df.loc[df.x > split_point, "y"].mean()) ** 2
    df.loc[index, "sum_of_square_errors"] = left_part.sum() + right_part.sum()
df = df.sort_values("sum_of_square_errors")
best_split = df.iloc[0]["split_points"]
df
```

Out[94]:

	y	x	split_points	sum_of_square_errors
92	0.014877	0.104154	0.119498	11.188598
39	-0.182545	-0.008309	0.013323	11.285947
76	0.013892	0.070919	0.078534	11.421178
10	0.106374	0.086149	0.095152	11.464895
13	0.169866	0.034954	0.052937	11.609307
...	...	...	...	...
94	0.939073	0.870908	0.904040	66.487117
8	0.498787	0.937172	0.945644	66.699354
50	0.780020	0.954115	0.961564	67.150652
83	0.385694	0.969013	0.970523	67.263005
32	0.433632	0.972033	NaN	None

100 rows × 4 columns

(2) (10 pts)

What is your result from (1)? State your optimal choice of r and $\hat{m}(x)$ function.

$$\widehat{m}(x) = \begin{cases} -0.6377 & x \leq 0.11949 \\ 0.86671 & x > 0.11949 \end{cases}$$

In [95]:

```
best_split
```

Out[95]:

```
0.119497607
```

In [98]:

```
df.loc[df.x <= best_split, "y"].mean()
```

Out[98]:

```
-0.6377018859259259
```

In [99]:
```
df.loc[df.x > best_split, "y"].mean()
```

Out[99]:
```
0.8667163272608694
```

(3) (5 pts)

What is the predictive value for the y corresponding to $x = 0.5$.

Since 0.5 > 0.11949, predictive value for the y is 0.86671

(4) (15 pts)

Using the non-parametric bootstrap, find an estimator for the variance of $\widehat{m}(0.5)$. Provide all your working.

In [96]:
```
import numpy as np
m_hat_list = []

for _ in range(200):
    bootstrap = df.iloc[:,:2].sample(n=df.shape[0], replace=True).sort_values("x")
    bootstrap["split_points"] = (bootstrap["x"] + bootstrap["x"].shift(-1)) / 2
    bootstrap["sum_of_square_errors"] = None

    for index, row in bootstrap.iterrows():
        split_point = row.split_points
        if pd.isna(split_point):
            continue
        left_part = (bootstrap.loc[bootstrap.x <= split_point, "y"] -
                     bootstrap.loc[bootstrap.x <= split_point, "y"].mean()) ** 2
        right_part = (bootstrap.loc[bootstrap.x > split_point, "y"] -
                      bootstrap.loc[bootstrap.x > split_point, "y"].mean()) ** 2
        bootstrap.loc[index, "sum_of_square_errors"] = left_part.sum() + right_part.sum()

    bootstrap = bootstrap.sort_values("sum_of_square_errors")
    best_split = bootstrap.iloc[0]["split_points"]
    if 0.5 <= best_split:
        m_hat = bootstrap.loc[bootstrap.x <= best_split, "y"].mean()
    elif 0.5 > best_split:
        m_hat = bootstrap.loc[bootstrap.x > best_split, "y"].mean()
    m_hat_list.append(m_hat)
m_hat_variance = np.var(m_hat_list)
```

In [97]:
```
m_hat_variance
```

Out[97]:
```
0.004076547533452653
```

$\widehat{m}(0.5) = 0.00407$

(5) (10 pts)

Using the answer to part (4), construct a 95% confidence interval for $m(0.5)$.

In [100…
```
m_hat_variance ** (0.5)
```

Out[100…
```
0.06384784674092504
```

$$\widehat{m}(0.5) \pm c_{\alpha/2}\sqrt{\mathrm{Var}\{\widehat{m}(0.5)\}} = 0.86671 \pm 1.96 * 0.0638$$

$$CI = (0.7416, 0.9917)$$

In [103…

```
0.86671 - 1.96 * 0.0638
```

Out[103… 0.741662

In [101…

```
0.86671 + 1.96 * 0.0638
```

Out[101… 0.9917579999999999