DATA 607 Statistical and Machine Learning
Session 2: Linear Smoothers

Matthew Greenberg

Department of Mathematics and Statistics
University of Calgary

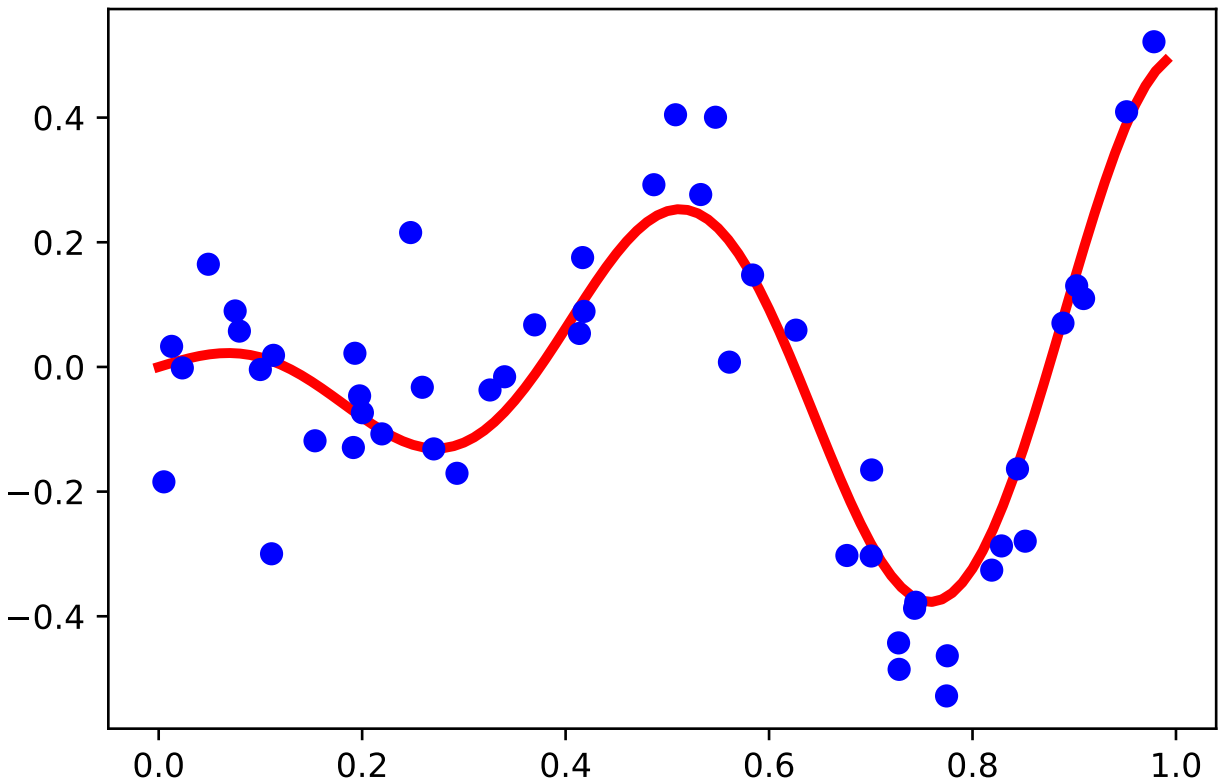
26.02.2020

This Evening's Agenda

1. Linear Smoothers
   - Definition
   - Example: Linear Regression
   - Example: k-Nearest Neighbors
   - Example: Sliding Window
2. Evaluating Smoothers and Tuning Parameters

Smoothers

Smoothers are regression models for fitting "smooth" curves to data.



Linear Smoothers

Dataset:

$$(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)$$

Regression Model:

$$Y_i = r(\mathbf{x}_i) + \varepsilon_i$$

Definition

A *linear smoother* is an estimator of r of the form

$$\widehat{r}(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) Y_i = \mathbf{w}(\mathbf{x}) \cdot \mathbf{Y}.$$

The $w_i(\mathbf{x})$ are called *weights*.

Linear smoother are so named because they are *linear functions of* $\mathbf{Y}$. Their graphs need not be lines!

Example 1: Linear Regression

Dataset:

$$(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n), \qquad \mathbf{x}_i \in \mathbb{R}^{1\times p}, \qquad Y_i \in \mathbb{R}$$

Regression Model:

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \varepsilon \\ &= \underbrace{\begin{pmatrix} 1 & \mathbf{x}_i \end{pmatrix}}_{1\times(p+1)} \boldsymbol{\beta} + \varepsilon, \end{aligned}$$

where

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix} \in \mathbb{R}^{(p+1)\times 1}.$$

Least-Squares Line:

$$\widehat{r}(\mathbf{x}) = \mathbf{x}\widehat{\beta}, \qquad (*)$$

$\widehat{\beta}$ is the least-squares solution of

$$X\beta = \mathbf{Y}, \qquad \text{where} \qquad X = \begin{pmatrix} 1 & \mathbf{x}_1 \\ \vdots & \vdots \\ 1 & \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{n\times(p+1)}.$$

Explicitly,

$$\widehat{\beta} = (X^T X)^{-1} X^T \mathbf{Y}.$$

Substituting into $(*)$,

$$\widehat{r}(\mathbf{x}) = \underbrace{(\mathbf{x}(X^T X)^{-1} X^T)}_{\mathbf{w}(\mathbf{x})} \mathbf{Y}.$$

Thus, $\widehat{r}$ is a linear smoother.

Coding Activity 1

1. Mock some data,

$$(x_0, y_0), \ \ldots, \ (x_{19}, y_{19}), \qquad x_i \in (0,1), \ y_i \in \mathbb{R},$$

suitable for linear regression:

   1. Choose β_0 and β_1 uniformly at random from the intervals $(0,1)$ the and $(-1,0)$, respectively.
   2. Choose $x_0, \ldots, x_{19}$ uniformly at random from the interval $(-1,1)$.
   3. Draw a random sample $\varepsilon_0, \ldots, \varepsilon_{19}$ from $N(0, 0.2)$.
   4. Set $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$.

2. Plot y versus x. It should look roughly linear.
3. Write a function that computes $\mathbf{w}(x)$ from x.
4. Plot the graph of $y = \widehat{r}(x)$.

Example 2: k-Nearest Neightbor Smoother

Data:

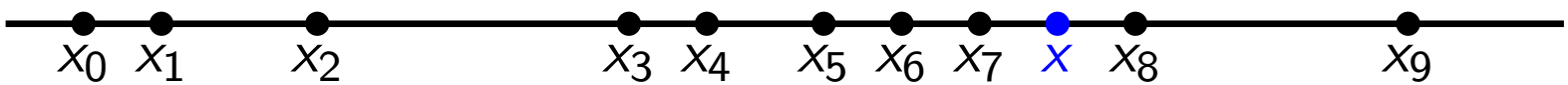
$$(\mathbf{x}_0, Y_0) \ \ldots, \ (\mathbf{x}_{n-1}, Y_{n-1})$$

Regression Model:

$$Y_i = r(\mathbf{x}_i) + \varepsilon_i$$

k-Nearest Neighbors:

$$N_k(\mathbf{x}) = \text{the } k \text{ elements of } \mathbf{x}_0, \ \ldots, \ \mathbf{x}_{n-1} \text{ closest to } \mathbf{x}$$



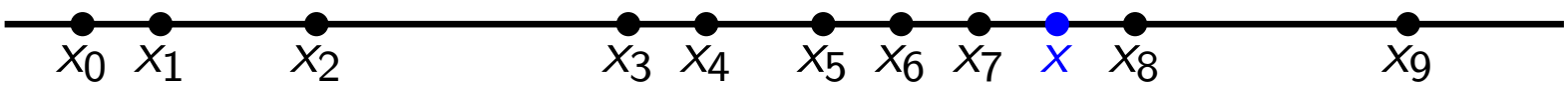
$$N_3(x) = \{x_6, x_7, x_8\}$$

Definition

The k-**nearest neighbor smoother** is the estimator of r is defined by

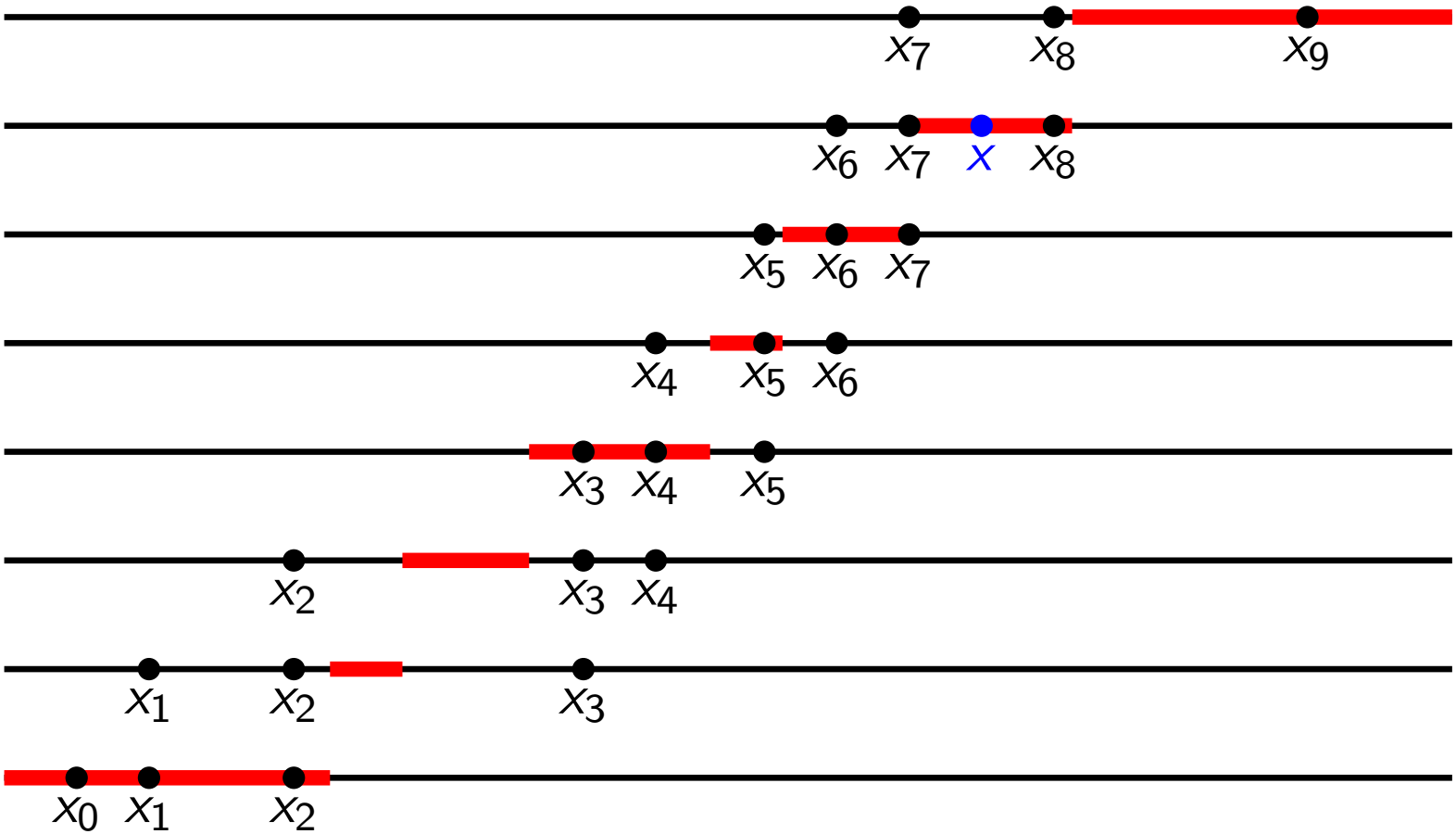
$$\widehat{r}_k(\mathbf{x}) = \text{average of the } Y_i \text{ for which } \mathbf{x}_i \in N_k(\mathbf{x})$$

$$= \sum_{i=0}^{n-1} w_i(\mathbf{x}) Y_i, \quad \text{where} \quad w_i(\mathbf{x}) = \begin{cases} \frac{1}{k} & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

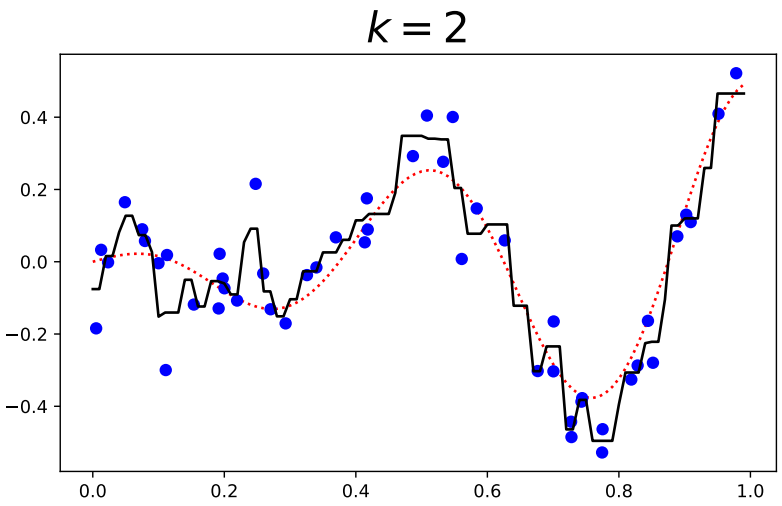


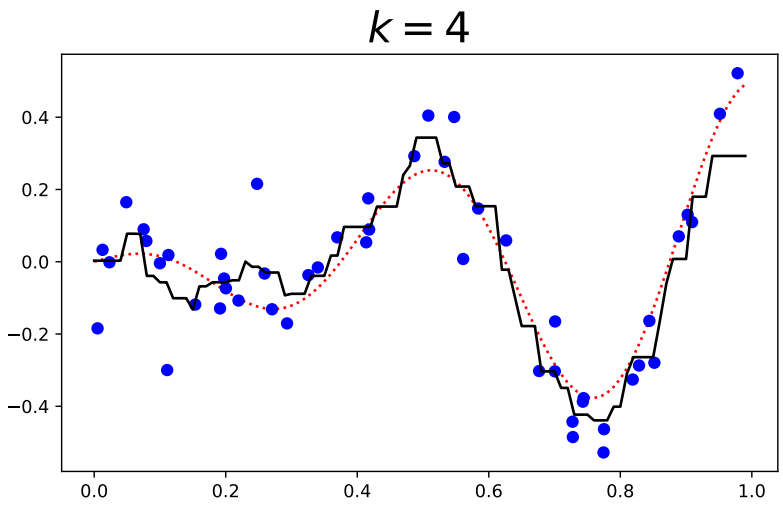
$$N_3(x) = \{x_6, x_7, x_8\}, \qquad \widehat{r}_3(x) = \frac{1}{3}(Y_6 + Y_7 + Y_8)$$

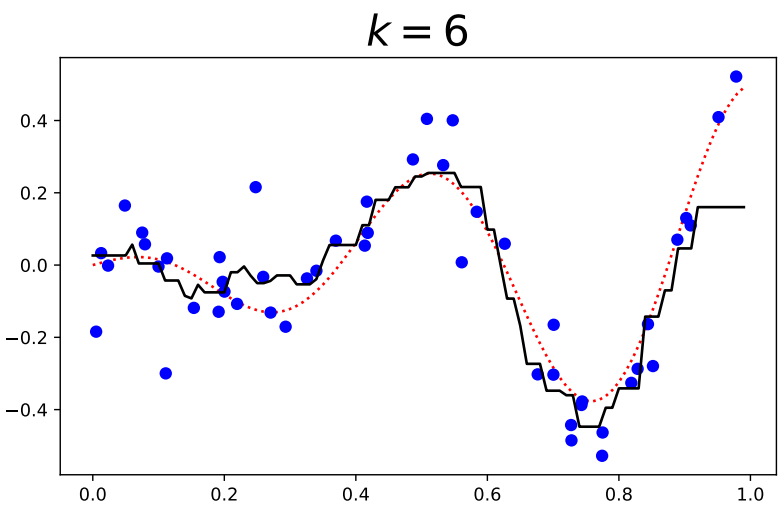
3-Nearest Neighbors:

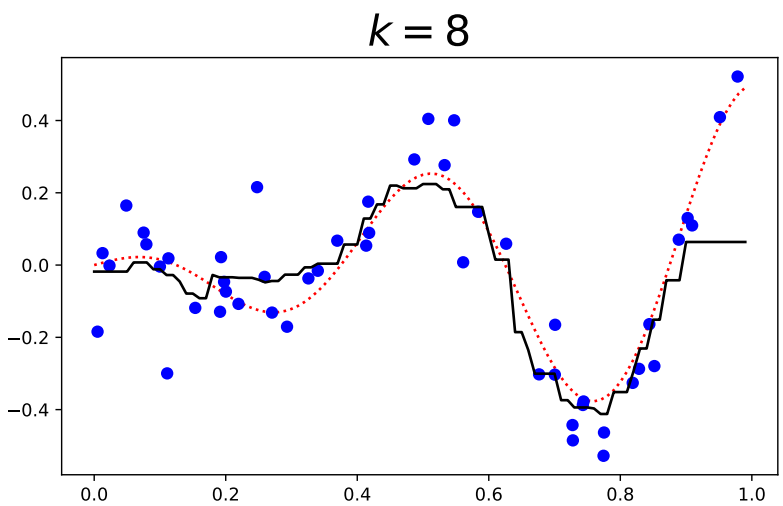


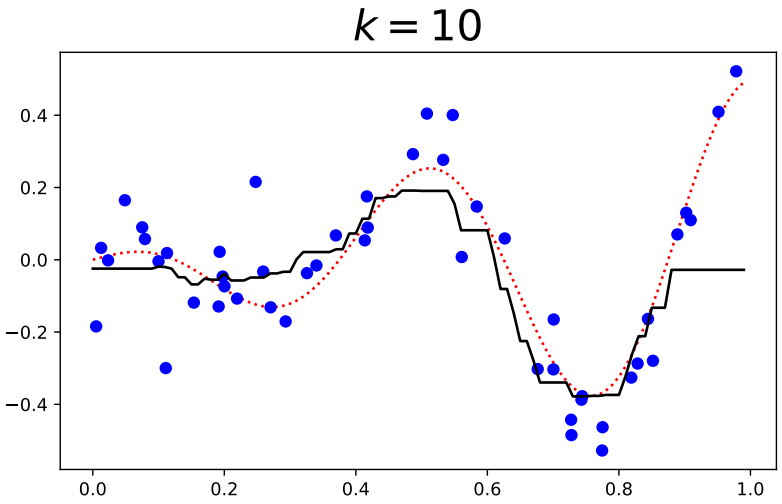
Sample Plots

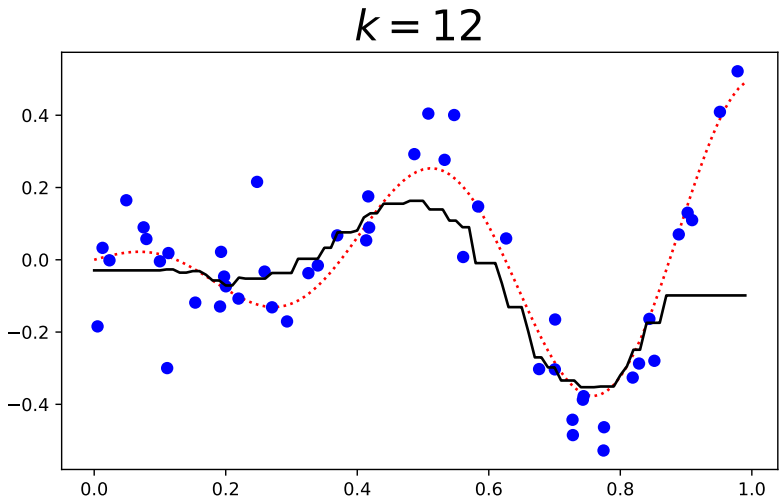


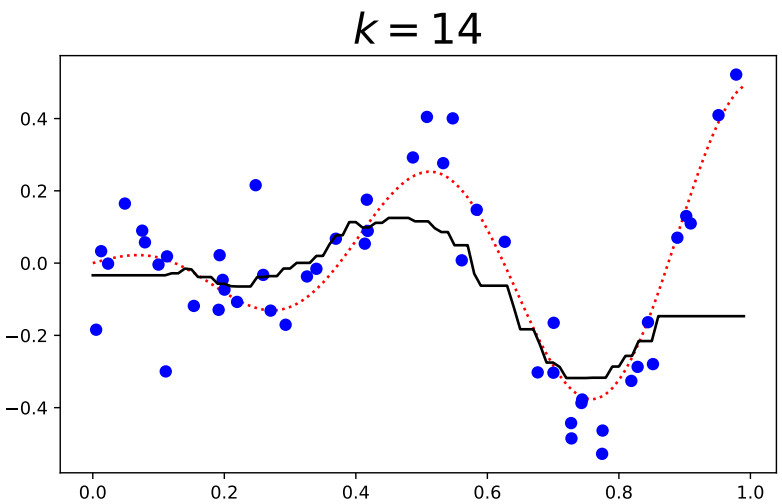


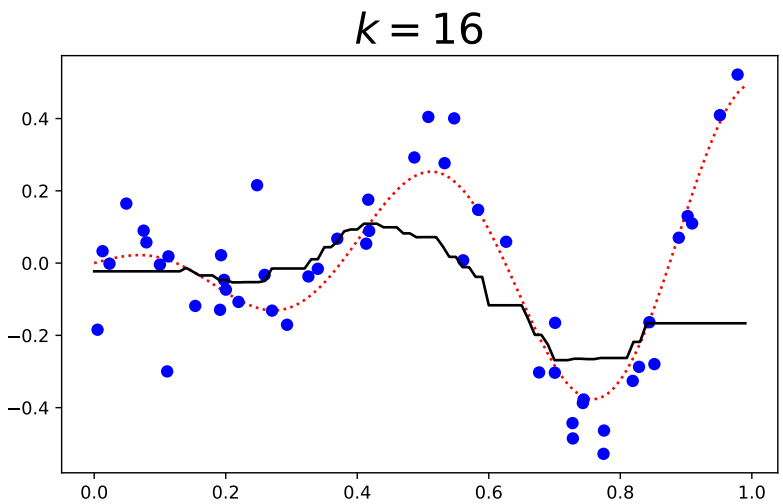




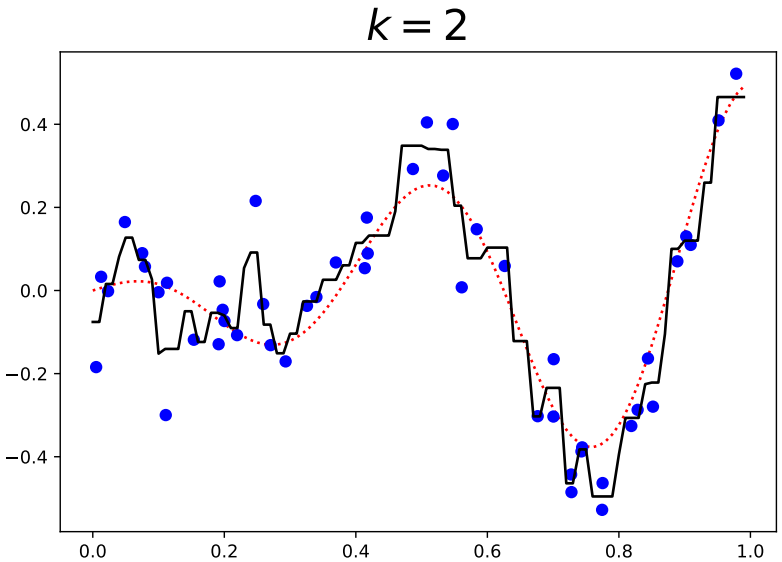

$k = 10$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

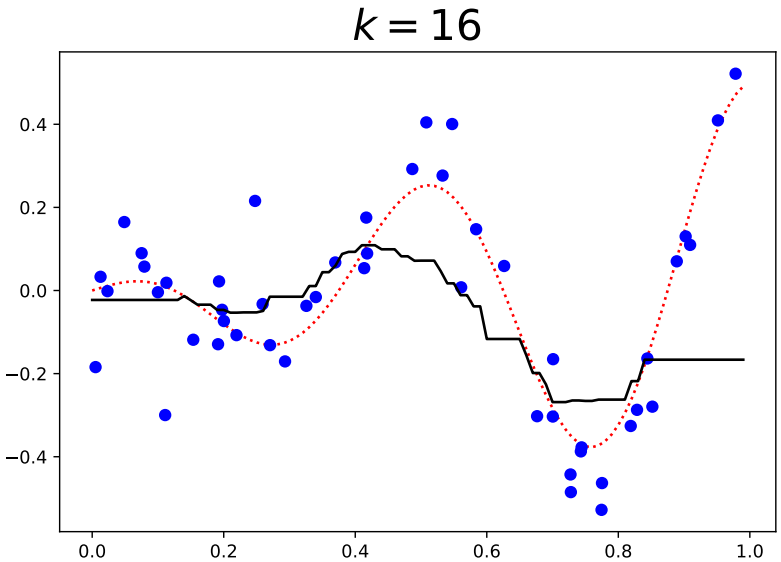

$k = 12$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0


$k = 14$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0


$k = 16$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

Overfitting and Underfitting





For small k, the graphs of $\widehat{r}(x)$ are too wiggly, fitting the noisy sample *too closely*. We've **overfit** (**undersmoothed**) the data.

For big k, the graphs of $\widehat{r}(x)$ don't fit data very well (they're too flat). We've **underfit** (**oversmoothed**) the data.

Is there a "right" k?

Example 3: The Sliding Window Smoother

Data:

$$(\mathbf{x}_0, Y_0) \ \ldots, \ (\mathbf{x}_{n-1}, Y_{n-1})$$

Regression Model:

$$Y_i = r(\mathbf{x}_i) + \varepsilon_i$$

***h*-neighbors:**

$$N_h(\mathbf{x}) = \text{the elements of } \mathbf{x}_0, \ \ldots, \ \mathbf{x}_{n-1} \text{ within a distance } h \text{ of } \mathbf{x}$$

$$= \{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\| < h\}$$

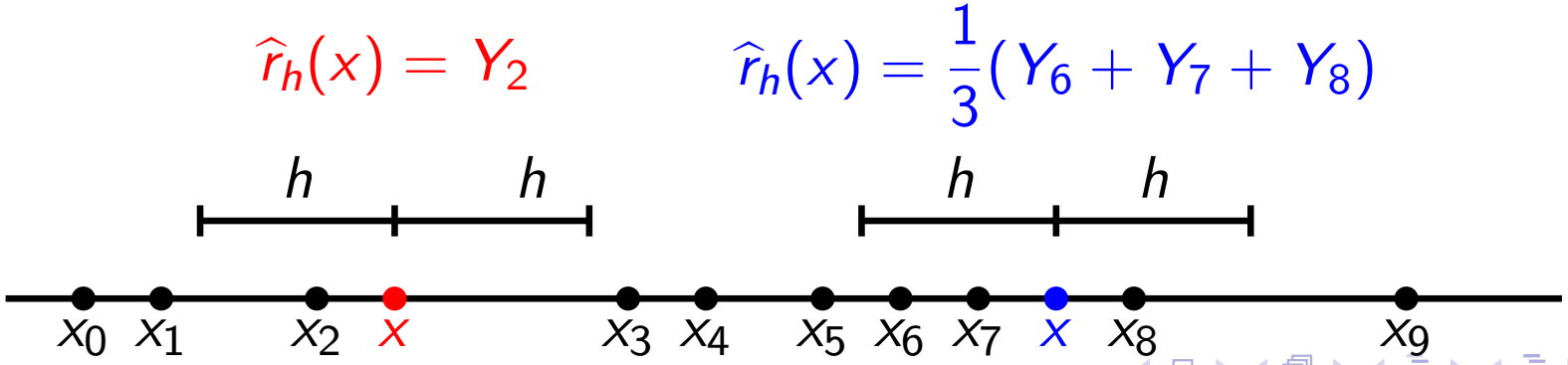
Definition

The **sliding window smoother with bandwidth** $h > 0$ is the estimator of r defined by

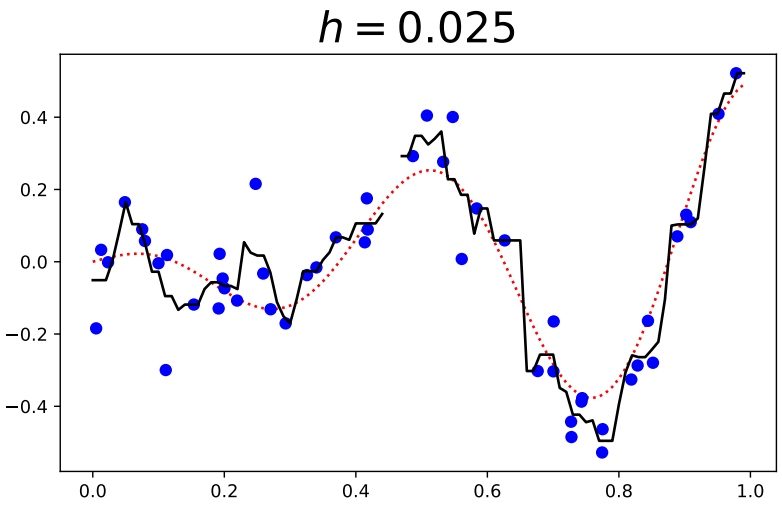
$$\widehat{r}_h(\mathbf{x}) = \text{average of the } Y_i \text{ for which } \|\mathbf{x} - \mathbf{x}_i\| < h$$

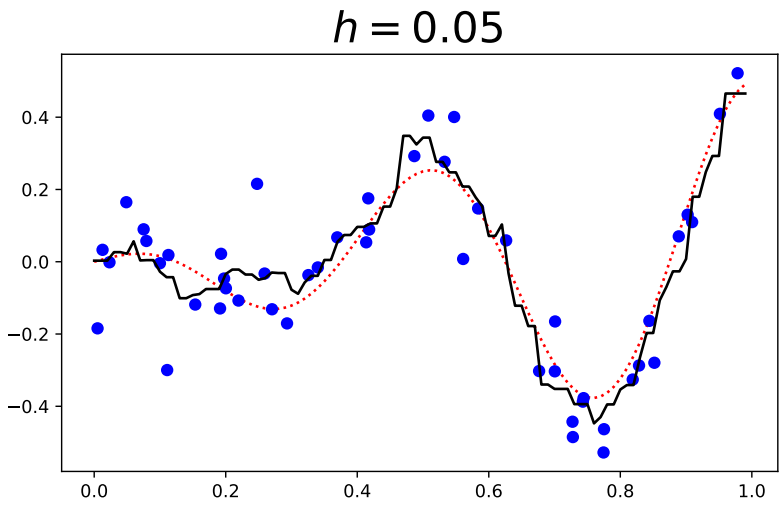
$$= \frac{\sum_{\|\mathbf{x}_i - \mathbf{x}\| < h} Y_i}{\sum_{\|\mathbf{x}_i - \mathbf{x}\| < h} 1}$$

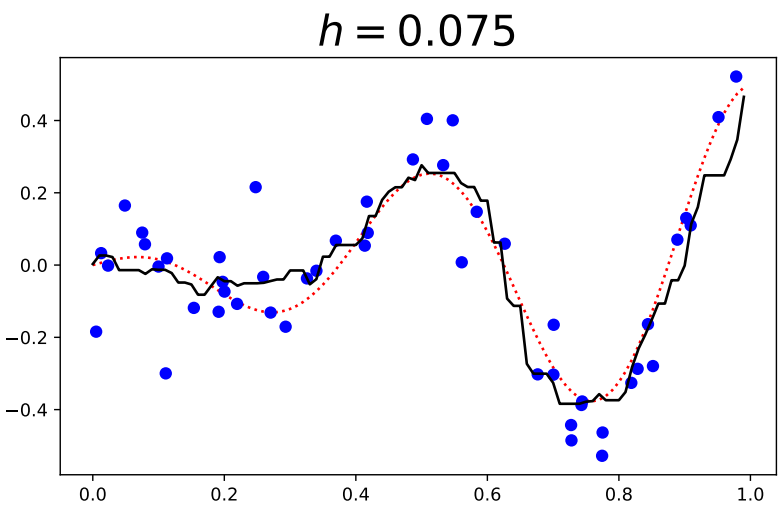
$$= \sum_{i=0}^{n-1} w_i(\mathbf{x}) Y_i, \quad \text{where} \quad w_i(\mathbf{x}) = \begin{cases} \dfrac{1}{\#N_h(\mathbf{x})} & \text{if } \mathbf{x}_i \in N_h(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

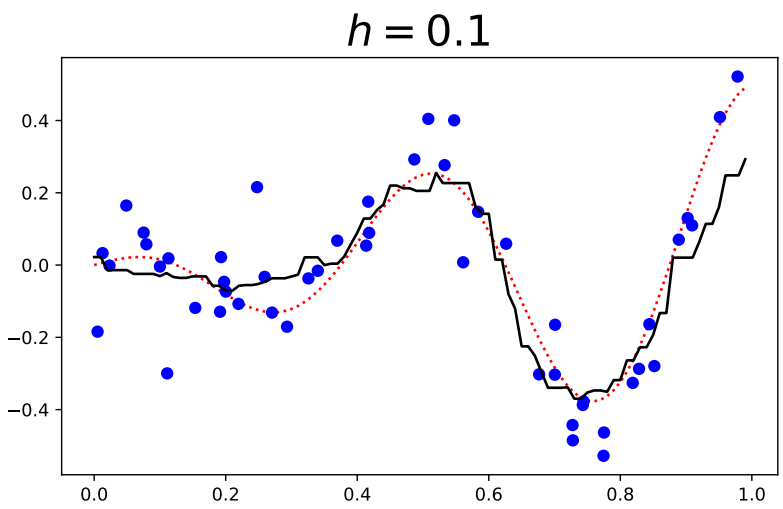


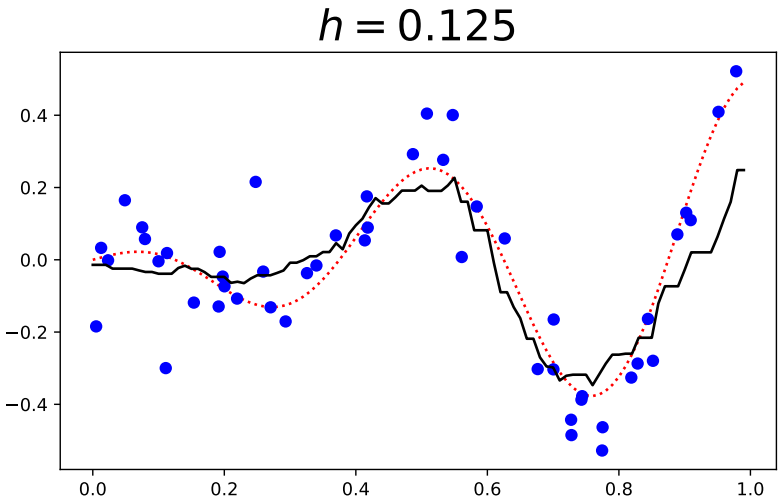
Sample Plots

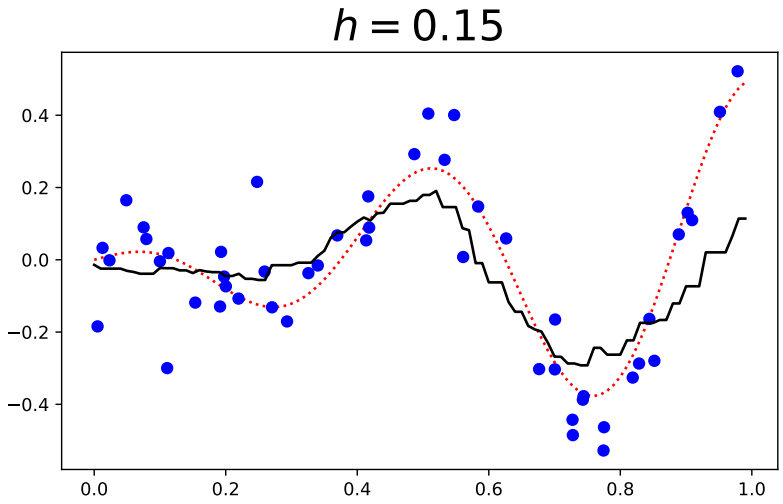


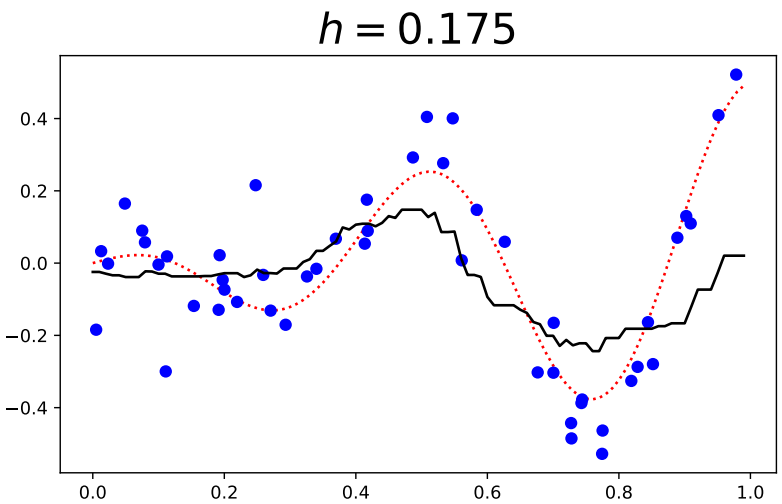


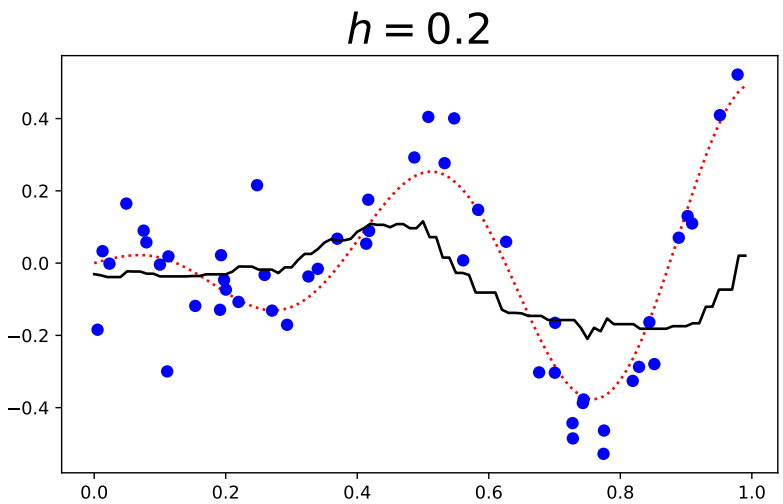




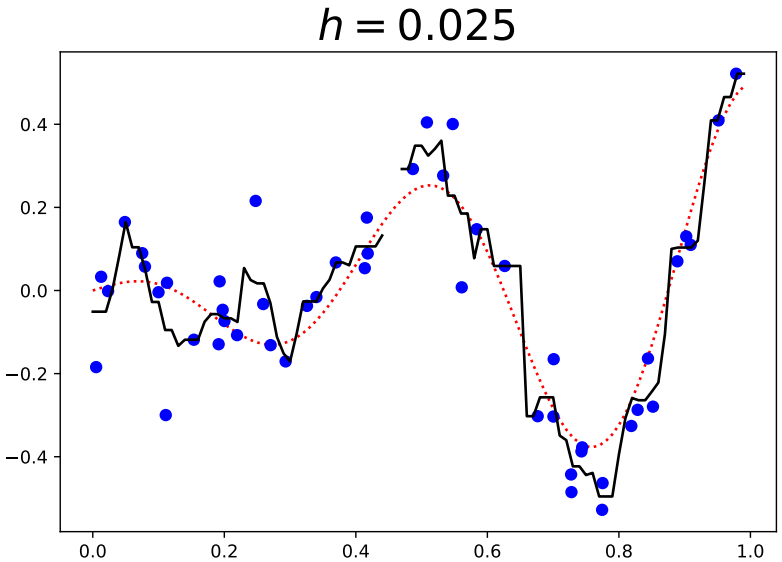

$h = 0.125$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

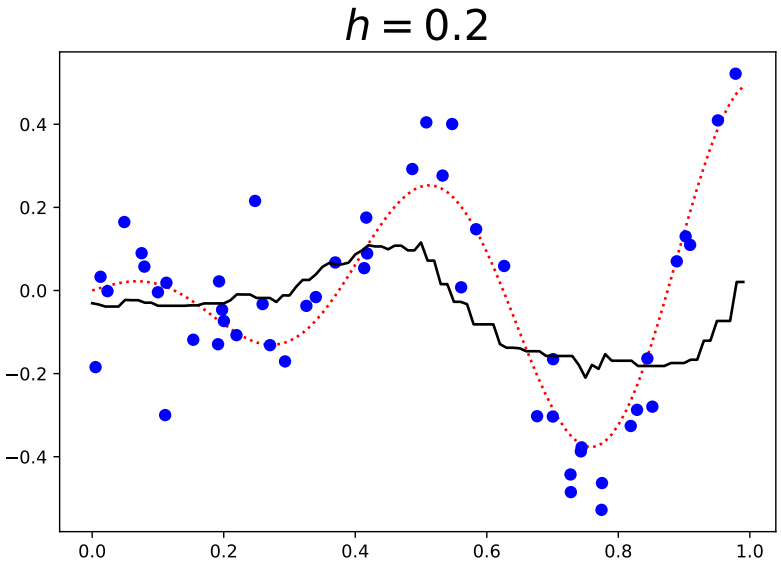

$h = 0.15$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0


$h = 0.175$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0


$h = 0.2$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

Overfitting and Underfitting





For small h, the graphs of $\widehat{r}(x)$ are too wiggly, fitting the noisy sample *too closely*. We've **overfit** (**undersmoothed**) the data.

For large h, the graphs of $\widehat{r}(x)$ don't fit data very well (they're too flat). We've **underfit** (**oversmoothed**) the data.

Is there a "right" h?

Evaluating Smoothers: Predictive Accuracy

We evaluate a smoother based on the accuracy of its predictions.

Let $\widehat{r}_{\mathcal{D}}$ be a smoother trained on the data set $\mathcal{D}$.

Assume we are in possession of a **test set**,

$$\mathcal{D}_{\text{test}} = \{(\mathbf{x}_1, Y_1), \ \ldots, \ (\mathbf{x}_n, Y_n)\},$$

i.e., a random sample drawn from the **same distribution** as — but **independently** of — $\mathcal{D}$.

The **expected prediction error** of $\widehat{r}_{\mathcal{D}}$ is approximately the average error on $\mathcal{D}_{\text{test}}$:

$$\text{test error} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \widehat{r}_{\mathcal{D}}(\mathbf{x}_i))^2$$

Tuning Model Parameters Using a Test Set

Given a test set, $\mathcal{D}_{\text{test}}$, choose your parameters (k for k-nearest neighbors, h for sliding window) to **minimize**

$$\text{test error} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{r}_{\mathcal{D}}(\mathbf{x}_i))^2,$$

our proxy for the expected prediction error.

Compute the test error for a range of values of k or h, and choose the parameter value giving the smallest test error.

Coding activity 2

But, we can work in an artificial situation where we *do* know $r(x)$:

- Choose:
  - a function, $r(x)$
  - $x_1, \ldots, x_n$
  - a fine partition, P, of an interval containing the x_i
  - k (or h)
- for $k = 1, \ldots, K$
  - for $j = 0, \ldots, J-1$:
    - Generate $y_i^{(j)}$, $1 \leq i \leq n$, by sampling from $N(r_k(x_i), \sigma^2)$.
    - Compute the losses $L(\widehat{r}_k^{(j)}(x), r(x))$ at each x in P.
  - Average the losses over j to get the risks, $R(\widehat{r}_k(x), r(x))$.
  - Average the risks over x to get the average risk, $R(\widehat{r}_k, r)$.
- Plot $R(\widehat{r}_k, r)$ vs k and choose k.

The bias-variance decomposition

Definition

The *bias of* $\widehat{r}(x)$, as an estimator of $r(x)$, is

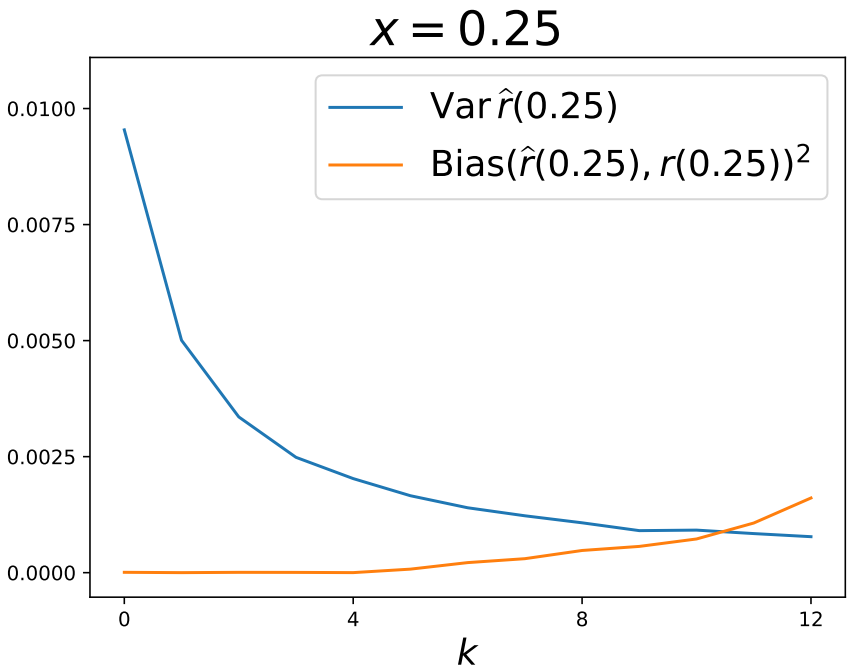
$$\text{Bias}(\widehat{r}(x), r(x)) := \mathbb{E}[\widehat{r}(x)] - r(x)$$

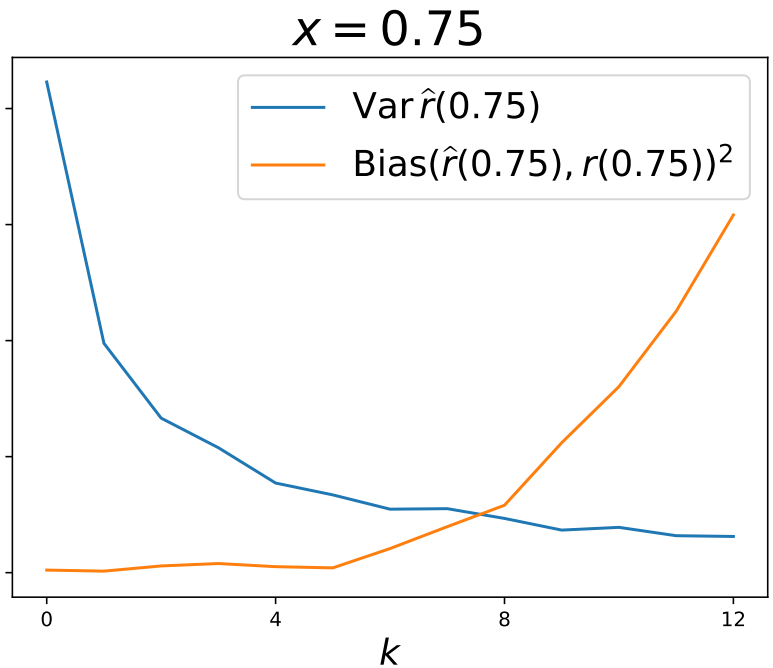
Bias-variance decomposition

$$R(\widehat{r}(x), r(x)) = \text{Var}\,\widehat{r}(x) + \text{Bias}(\widehat{r}(x), r(x))^2$$

$\text{Var}\,\widehat{r}(x)$ big / graph of $\widehat{r}$ wiggly / overfit / undersmoothed

$\text{Bias}(\widehat{r}(x), r(x))$ big / graph of $\widehat{r}$ too flat / underfit / oversmoothed





Variance decreases as k increases.

Bias increases as k increases.

Define:

$$\operatorname{Var} \widehat{r} := \int R(\widehat{r}(x), r(x))\, dx, \qquad \operatorname{Bias}(\widehat{r}, r)^2 := \int \operatorname{Bias}(\widehat{r}(x), r(x))^2\, dx$$

Integrate the bias-variance decomposition:

$$R(\widehat{r}, r) = \operatorname{Var} \widehat{r} + \operatorname{Bias}(\widehat{r}, r)^2$$

Since r is unknown, these quantities can only be estimated.

Coding activity 3

- For our running synthetic example, plot

$$R(\widehat{r}_k, r), \ \operatorname{Var} \widehat{r}_k, \ \text{and} \ \operatorname{Bias}(\widehat{r}_k, r)^2$$

  versus k on the same axes.
- Verify the integrated bias-variance decomposition using your computed $R(\widehat{r}_k, r)$, $\operatorname{Var} \widehat{r}_k$, and $\operatorname{Bias}(\widehat{r}_k, r)^2$.

Reuse as much of your code from CODING ACTIVITY 2 as possible.

Training error

Definition

The *empirical risk* or *training error* is

$$\frac{1}{n}\sum_{i=1}^{n} L(\widehat{r}(x_i), Y_i).$$

Empirical risk is not good estimator of R: We trained $\widehat{r}$ on the (x_i, Y_i), making $L(\widehat{r}(x_i), Y_i)$ biased downwards.

Leave one out cross-validation

Definition

The *leave one out cross validation (LOOCV) score* is

$$\widehat{R}(h) := \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{r}_h^{(-i)}(x_i)\right)^2,$$

where $\widehat{r}_h^{(-i)}$ is computed using the subdataset of the original one by removing the point (x_i, Y_i).

$\widehat{R}(h)$ is typically a good estimator of the average risk of $\widehat{r}_h$:

$$\widehat{R}(h) \approx R(\widehat{r}_h, r) + \sigma^2, \quad \text{where} \quad \sigma^2 = \operatorname{Var} Y_i.$$

We choose our smoothing parameter to be the one that minimizes $\widehat{R}(h)$:

$$h := \operatorname*{argmin}_h \widehat{R}(h)$$

Coding activity 4

For our running example, plot $\widehat{R}(k)$ versus k. Compute the $\widehat{R}(k)$ using `sklearn`'s `LeaveOneOut` class and the generator returned by its `split` method.

Which k that minimizes $\widehat{R}(k)$? Compare with the results of coding activity 2.

For this optimal value of k, plot $\widehat{r}_k^{(-i)}(x_i)$ versus x_i and $r(x)$ versus x on the same axes.

Remark: `LeaveOneOut` is fairly low-level; `sklearn` provides more convenient ways to tune hyperparameters using cross validation. Sometimes, though, it's necessary to work with the lower-level constructs.

K-fold cross validation

Partition $\{1, \ldots, n\}$ into K *folds*, $I_1, \ldots, I_K$, of roughly equal size. Let $\widehat{r}_h^{(-I_j)}$ be the smoother computed using the subdataset of the original one obtained by removing (x_i, Y_i), for $i \in I_j$.

Definition

The *K-fold cross validation score is*

$$\widehat{R}_K(h) := \sum_{j=1}^{K} \frac{1}{|I_j|} \sum_{i \in I_j} \left(Y_i - r_h^{(-I_j)}(x_i)\right)^2$$

We can use $\widehat{R}_K$ in place of $\widehat{R}$ to select a smoothing parameter. In practice, K is usually 5 (`sklearn`'s default) or 10.

Coding activity 5

Find the values of k that minimnize the 3-, 5-, and 10-fold cross validation scores. Compute these scores using `cross_val_score` from `sklearn.model_selection`.
Confirm your results from Coding activity 4 by performing LOOCV as n-fold cross validation, n being the size of the dataset.