DATA 607 Statistical and Machine Learning

Session 2: Linear Smoothers

Matthew Greenberg

Department of Mathematics and Statistics
University of Calgary

28.02.2020

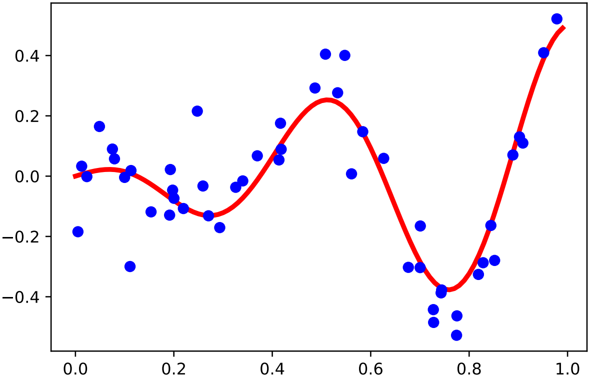
This Evening's Agenda

1. Regressors/Smoothers
2. Linear Smoothers
   - Definition
   - Example: Linear Regression
   - Example: k-Nearest Neighbors
   - Example: Sliding Window
3. Evaluating Smoothers and Tuning Parameters
   - Prediction Error
   - Train/Test Split
   - Cross-Validation

Regressors/Smoothers

Regression Models or **Regressors** are models for fitting curves to data.

Smoother is a synonym for regressor, typically used in the context of nonparametric models.



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:

$$\widehat{r}(\mathbf{x}) = \begin{pmatrix} 1 & \mathbf{x} \end{pmatrix} \widehat{\beta}, \qquad (*)$$

$\widehat{\beta}$ 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 $(*)$,

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

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

Example 2: k-Nearest Neighbor Smoother

Data:

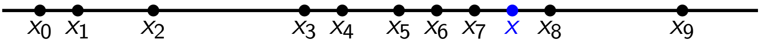
$$(\mathbf{x}_0, Y_0)\ \ldots,\ (\mathbf{x}_{n-1}, Y_{n-1})$$

Regression Model:

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

***k*-Nearest Neighbors:**

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



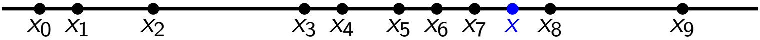
$$N_3(x) = \{x_6, x_7, x_8\}$$

Definition

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

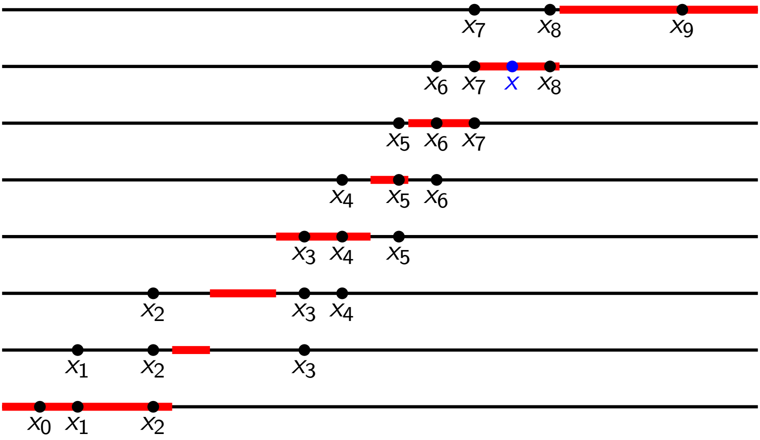
$$\widehat{r}_k(\mathbf{x}) = \text{average of the } Y_i \text{ for which } \mathbf{x}_i \in N_k(\mathbf{x})$$

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

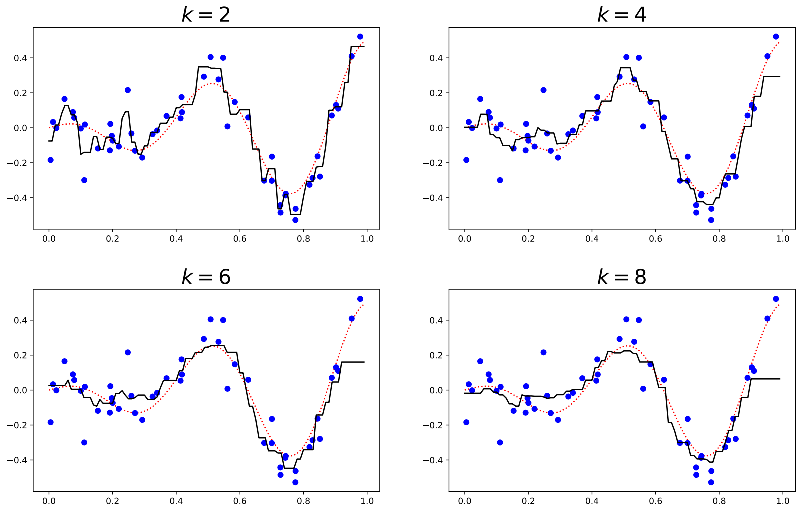


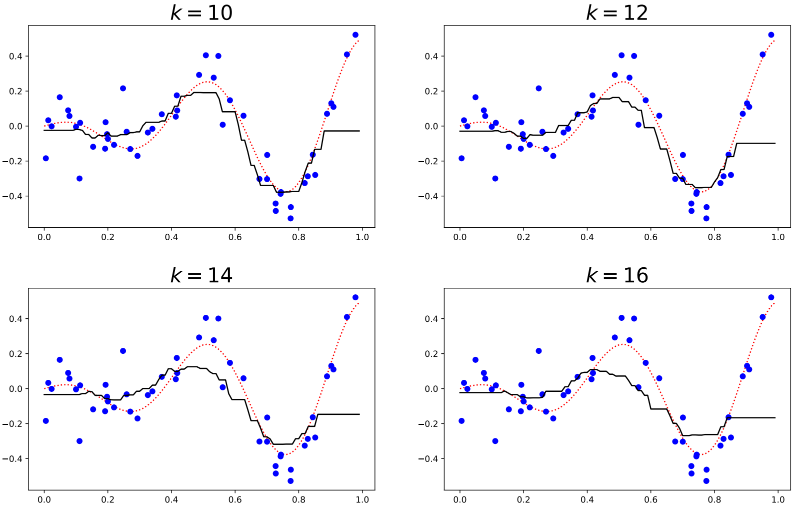
$$N_3(x) = \{x_6, x_7, x_8\}, \qquad \widehat{r}_3(x) = \frac{1}{3}(Y_6 + Y_7 + Y_8)$$

3-**Nearest Neighbors:**

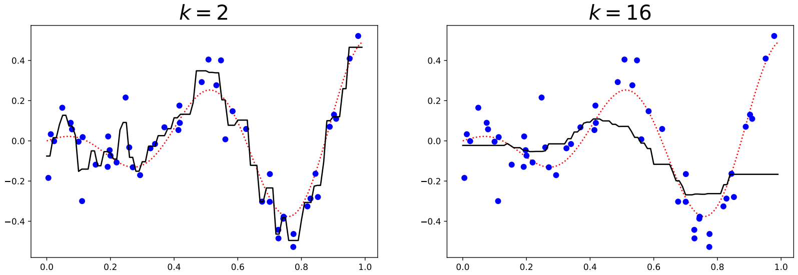


Sample Plots




$k = 10$
$k = 12$
$k = 14$
$k = 16$
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:**

$$\begin{aligned} N_h(\mathbf{x}) &= \text{the elements of } \mathbf{x}_0, \; \ldots, \; \mathbf{x}_{n-1} \text{ within a distance } h \text{ of } \mathbf{x} \\ &= \{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\| < h\} \end{aligned}$$

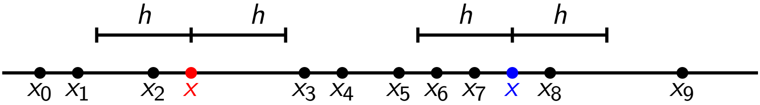
Definition

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

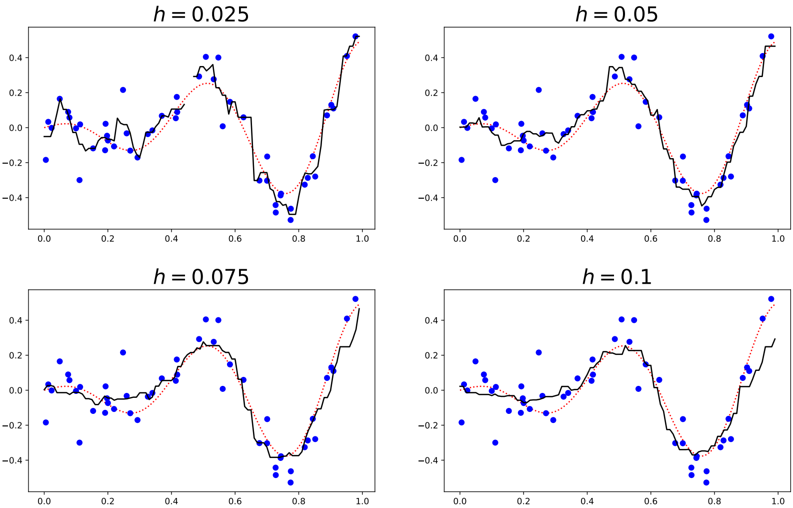
$$\widehat{r}_h(\mathbf{x}) = \text{average of the } Y_i \text{ for which } \|\mathbf{x} - \mathbf{x}_i\| < h$$

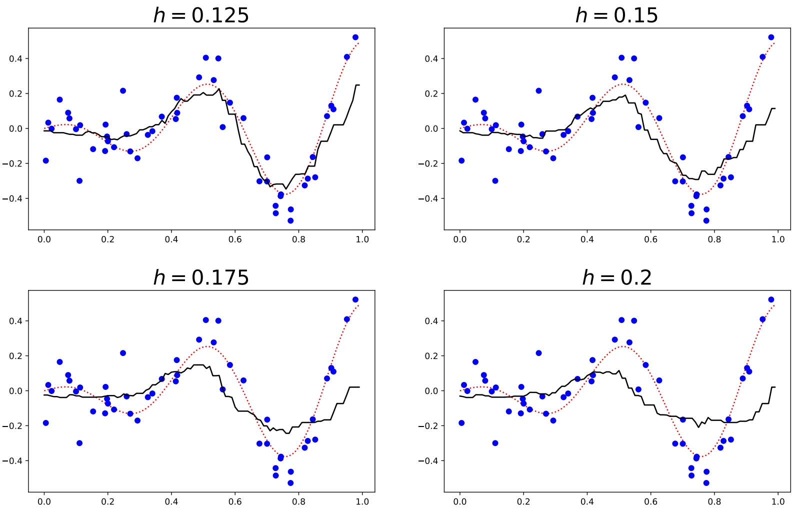
$$= \sum_{i=0}^{n-1} w_i(\mathbf{x}) Y_i, \quad \text{where} \quad w_i(\mathbf{x}) = \begin{cases} \dfrac{1}{\#N_h(\mathbf{x})} & \text{if } \mathbf{x}_i \in N_h(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

$$\widehat{r}_h(x) = Y_2 \qquad \widehat{r}_h(x) = \frac{1}{3}(Y_6 + Y_7 + Y_8)$$

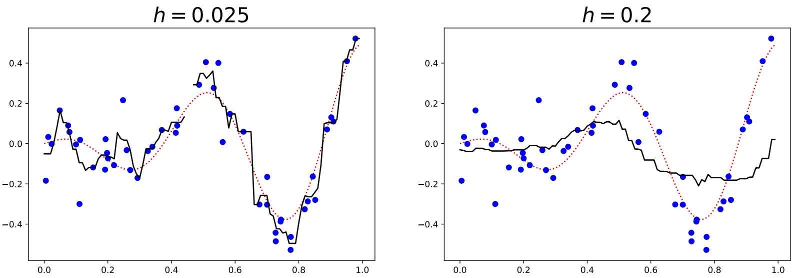


Sample Plots




$h = 0.125$
$h = 0.15$
$h = 0.175$
$h = 0.2$
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?

Prediction Error

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

Let $\widehat{r}_{\mathcal{D}}$ be a regressor 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 $\mathcal{D}_{\text{test}}$:

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

Train/Test Split

Split you data set $\mathcal{D}$ into two disjoint subsets: A larger **training set**, $\mathcal{D}_{\text{train}}$, and a smaller **testing set**, $\mathcal{D}_{\text{test}}$.

Train your model on $\mathcal{D}_{\text{train}}$, the use $\mathcal{D}_{\text{test}}$ to estimate its prediction error.

Training error typically **underestimates** prediction error!

Tuning Model Parameters Using a Test Set

Given a test set,

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

choose your parameters (k for k-nearest neighbors, h for sliding window) to **minimize**

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

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.

K-Fold Cross Validation

Data set:

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

When n is small, it may be undesirable to set aside a test set for tuning parameters. **Cross-validation** offers an alternative.

Partition $\{1, \ldots, n\}$ into K **folds**, $I_1, \ldots, I_K$, of roughly equal size:

$$\mathcal{D} = I_1 \cup I_2 \cup \cdots \cup I_K, \qquad \# I_j \approx \frac{n}{K}$$

We get K train/test splits:

$$\text{training set} = \mathcal{D}_h^{-j} = \{(\mathbf{x}_i, Y_i) : i \notin I_j\}, \qquad \text{size } \frac{K-1}{K} n$$

$$\text{testing set} = \mathcal{D}_h^{j} = \{(\mathbf{x}_i, Y_i) : i \in I_j\} \qquad \text{size } \frac{1}{K} n$$

Compute $\widehat{r}_h^{-j}$ by training your model on $\mathcal{D}_h^{-j}$.

Estimate the prediction error of $\widehat{r}_h^{-j}$ on the test set $\mathcal{D}_h^j$:

$$E_{h,j} = \text{error on } \mathcal{D}_h^j = \sum_{i \in I_j} \left(Y_i - \widehat{r}_h^{-j}(\mathbf{x}_i)\right)^2$$

Compute the weighted average of these K measurements, giving an overall error measurement for the parameter value, h:

Definition

The K-**Fold Cross Validation Score** is

$$E_h = \sum_{j=1}^{K} \frac{\# I_j}{n} \sum_{i \in I_j} \left(Y_i - r_h^{(-I_j)}(x_i)\right)^2.$$

In practice, K is usually 5 (`sklearn`'s default) or 10 or n.

Leave-One-Out Cross-Validation (LOOCV)

Leave-One-Out Cross-Validation is another name for n-fold cross-validation.

In this case, the folds and testing sets are singletons,

$$I_1 = \{1\}, \ldots, I_n = \{n\}$$

$$\mathcal{D}_h^1 = \{(\mathbf{x}_1, Y_1)\}, \ldots, \mathcal{D}_h^n = \{(\mathbf{x}_n, Y_n)\},$$

while the training sets, $\mathcal{D}_h^{-j}$, have size $n - 1$.

Definition

The **Leave-One-Out Cross Validation Score** is

$$E_h = \frac{1}{n} \sum_{j=1}^{n} \sum_{i \neq j} \left(Y_i - r_h^{(-j)}(\mathbf{x}_i)\right)^2.$$