Linear Smoothers I

First examples and basic properties

DATA 607 — Session 1 — 25/02/2019

Session Plan

1. Welcome and introductions
2. Smoothing
3. Coding activity 1
   - Code a simple smoother.
   - Explore `sklearn`'s `Estimator` and `Regressor` APIs.
4. Linear smoothers: First examples
   - *k*-nearest neighbors
   - local averaging
5. Evaluating smoothers: Loss and risk
6. Coding activity 2
   - Work with `sklearn`'s built in regressors:
     - `KNeighborsRegressor`, `RadiusNeighborsRegressor`
   - Tune smoothing parameters to minimize average risk.

7. Bias-variance decomposition
8. CODING ACTIVITY 3
    - Verify the bias-variance decomposition.
9. Training error
10. Leave one out cross validation
11. CODING ACTIVITY 4
    - Tune smoothing parameter using leave one out cross validation.
    - `LeaveOneOut` and `split`
12. *K*-fold cross validation
13. CODING ACTIVITY 5
    - Tune smoothing parameter using *K*-fold cross validation.
    - `cross_val_score`
    - Verify results of CODING ACTIVITY 4

Linear smoothers

Dataset:

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

Suppose:

$$Y_i = r(\vec{x}_i) + \epsilon_i$$

Definition

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

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

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

Example 0: Linear Regression

Dataset:

$$(\vec{x}_1, Y_1), \ldots, (\vec{x}_n, Y_n) \in \mathbb{R}^p \times \mathbb{R}$$

Regression line:

$$\begin{aligned}\widehat{r}(\vec{x}) &= \vec{x} \cdot \vec{\beta} \\ &= \vec{x} \cdot \left((X^T X)^{-1} X^T \vec{Y}\right) \\ &= \left(X (X^T X)^{-T} \vec{x}\right) \cdot \vec{Y}\end{aligned}$$

This is a linear smoother with $\vec{w}(\vec{x}) = X(X^T X)^{-T} \vec{x}$.

Coding activity 1

- Write a nearest neighbor smoother, first as a function and then as a class implementing `sklearn`'s `Estimator` and `Regressor` APIs.

Example 1: k-nearest neighbors

$N_k(x) :=$ set of the k elements of $\{x_1, \ldots, x_n\}$ closest to x.

Definition

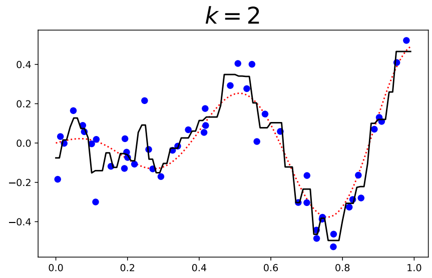
The *k-nearest neighbor smoother* is

$$\widehat{r}_k(\vec{x}) = \frac{1}{k} \sum_{i=1}^{n} \mathbf{1}_{N_k(\vec{x})}(\vec{x}_i) Y_i$$

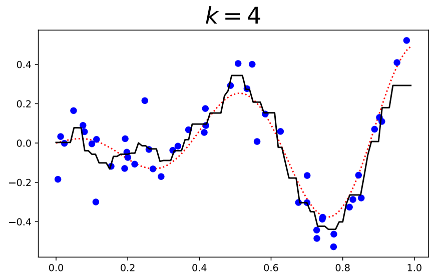
This is a linear smoother with $w_i(\vec{x}) = \frac{1}{k} \mathbf{1}_{N_k(\vec{x})}(\vec{x}_i)$.

Indicator function of $A \subseteq B$:

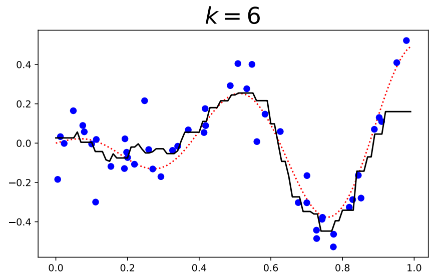
$$\mathbf{1}_A(b) := \begin{cases} 1 & \text{if } b \in A, \\ 0 & \text{otherwise.} \end{cases}$$



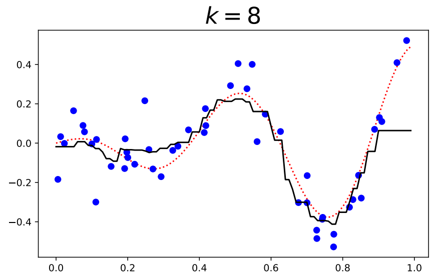
$k = 2$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0



$k = 4$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

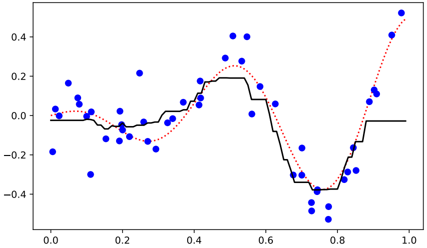


$k = 6$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

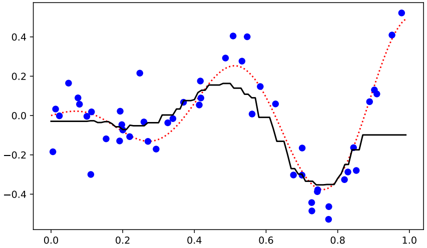


$k = 8$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

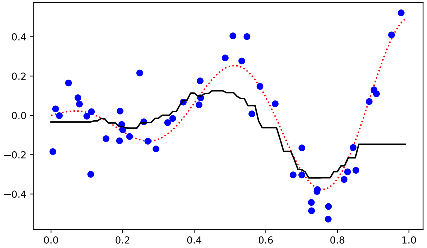
$k = 10$



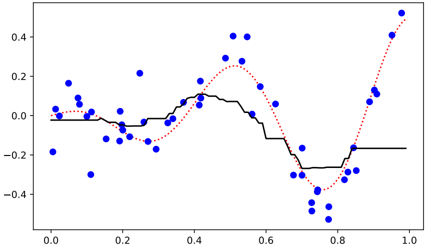
$k = 12$



$k = 14$



$k = 16$



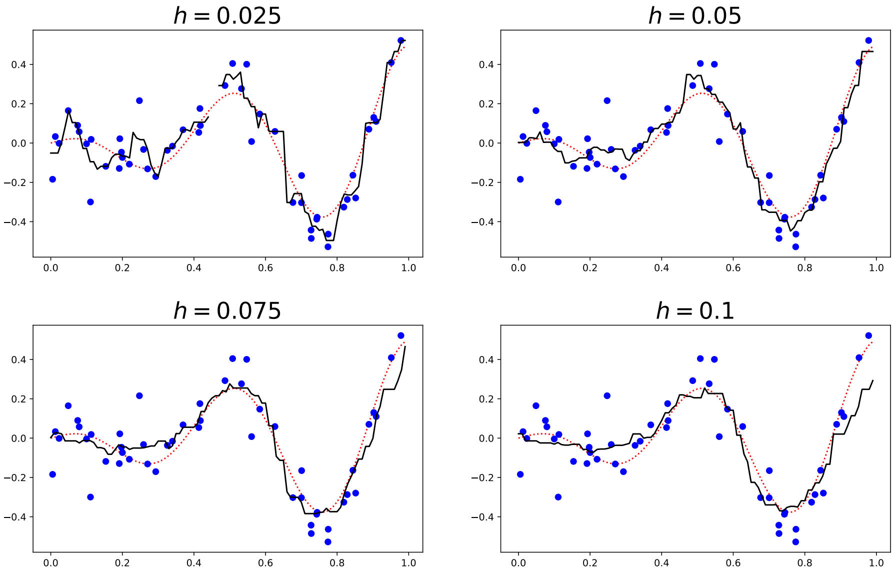
Example 2: Local averaging

Definition

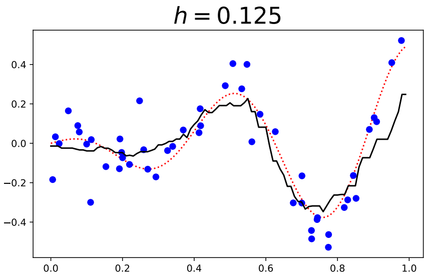
The *local average smoother* with *bandwidth* $h > 0$ is

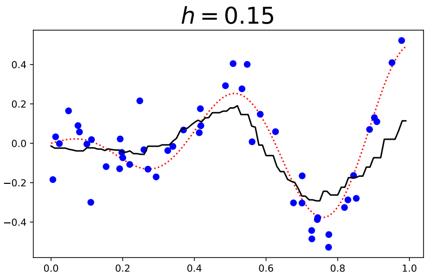
$$\widehat{r}_h(\vec{x}) = \frac{\sum_{\|\vec{x}_i - \vec{x}\| < h} Y_i}{\sum_{\|\vec{x}_i - \vec{x}\| < h} 1}$$

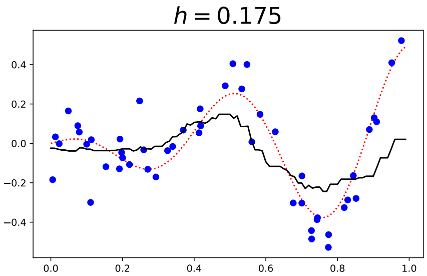
In words: $r_h(\vec{x})$ is the average of the Y_i for which $\vec{x}_i$ lies a distance less than h from $\vec{x}$.

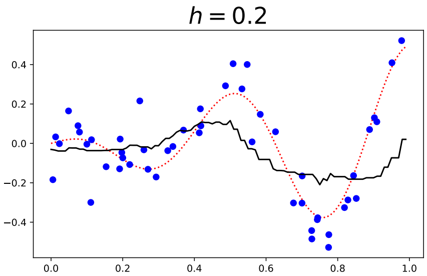


$h = 0.025$
$h = 0.05$
$h = 0.075$
$h = 0.1$
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0


$h = 0.125$
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 k or h, the graphs of $\widehat{r}(x)$ are too wiggly. We've *overfit* or *undersmoothed* the data.

For big k or h, the graphs of $\widehat{r}(x)$ data very well (they're too flat). We say we've *underfit* or *oversmoothed* the data.

How do we choose k or h optimally?

Evaluating smoothers: Loss and risk

Definitions

1. *Squared error loss at x:*

$$L(\widehat{r}(x), r(x)) := \big(\widehat{r}(x) - r(x)\big)^2$$

   - $L(\widehat{r}(x), r(x))$ is a random variable as $\widehat{r}(x)$ is.

2. *Mean squared error (MSE)* or *risk* at x:

$$R(\widehat{r}(x), r(x)) := \mathbb{E}\big[L\big(\widehat{r}(x), r(x)\big)\big] \approx \frac{1}{n}\sum_{i=1}^{n} L\big(\widehat{r}(x_i), r(x_i)\big)$$

3. *Average MSE* or *average risk*:

$$R(\widehat{r}, r) := \int R(\widehat{r}(x), r(x))\, dx \approx \frac{1}{n}\sum_{i=1}^{n} R(\widehat{r}(x_i), r(x_i))$$

Choosing the smoothing parameter

Obviously, smoothers with smaller average risk are preferred.

So, choose the smoothing parameter to *minimize average risk*:

$$k := \operatorname{argmin}_k R(\widehat{r}_k, r), \qquad h := \operatorname{argmin}_h R(\widehat{r}_h, r)$$

But there's a problem...

We don't know $r(x)$!

Loss at x: $$L(\widehat{r}(x), r(x)) = \big(\widehat{r}(x) - r(x)\big)^2$$

Risk at at x: $$R(\widehat{r}(x), r(x)) = \mathbb{E}\big[L\big(\widehat{r}(x), r(x)\big)\big]$$

Average risk at at x: $$R(\widehat{r}, r) = \int R(\widehat{r}(x), r(x))\,dx$$

These expressions all involve $r(x)$, which we typically don't know!

We can't compute these expressions; they also need to be estimated.

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

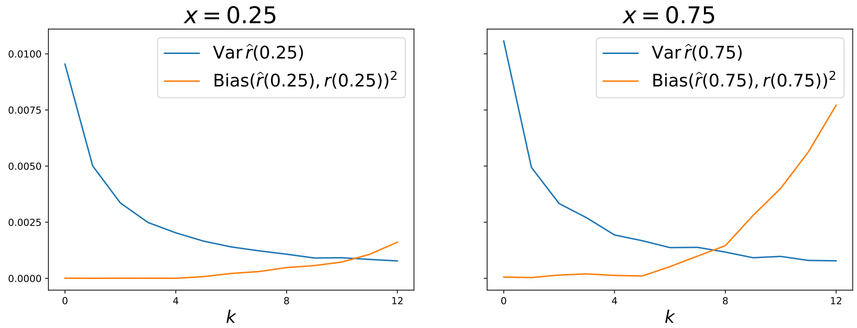
$$\mathrm{Bias}(\widehat{r}(x), r(x)) := \mathbb{E}[\widehat{r}(x)] - r(x)$$

Bias-variance decomposition

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

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

$\mathrm{Bias}(\widehat{r}(x), r(x))$ 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 $\{1, \dots, n\}$ into K *folds*, $I_1, \dots, I_K$, 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.

Binary Classification

Regression:

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

Binary classification:

$$Y_i = r(x_i) \in \{0, 1\}$$

Want $\widehat{r}$ to be $\{0, 1\}$-valued, too.

If $w_i(x) \geq 0$ and $\sum_i w_i(x) = 1$ for all x,

$$\widehat{r}(x) = d\left(\sum_{i=1}^{n} w_i(x) Y_i\right) \in \{0, 1\}$$

for any *decision function* $d : [0, 1] \to \{0, 1\}$.

Most common: $d(x) = \mathbf{1}_{[1/2,1]}(x)$ or $d(x) = \mathbf{1}_{[\alpha,1]}(x)$.

Loss and risk

$$L(\widehat{r}(x), r(x)) = |\widehat{r}(x) - r(x)| = \begin{cases} 1 & \text{if } \widehat{r}(x) = r(x), \\ 0 & \text{if } \widehat{r}(x) \neq r(x) \end{cases}$$

$$R(\widehat{r}(x), r(x)) = \mathbb{E}[L(\widehat{r}(x), r(x))]$$

$$R(\widehat{r}, r) = \int R(\widehat{r}(x), r(x))\, dx$$