S&DS 365 / 665

Intermediate Machine Learning

Smoothing and Density Estimation

September 8

Yale

Version of 2025/09/08 at 09:23

Topics for today

- Recap of lasso
- Smoothing kernels
- Kernel density estimation
- Bias-variance decomposition and the curse of dimensionality
- Next up: Intro to Mercer kernels

Administrivia

- Quiz 1: Great job!
- Assn 1 posted on Wednesday
- Topics: Lasso, smoothing, Mercer kernels, LOOCV
- Recordings
- Questions?

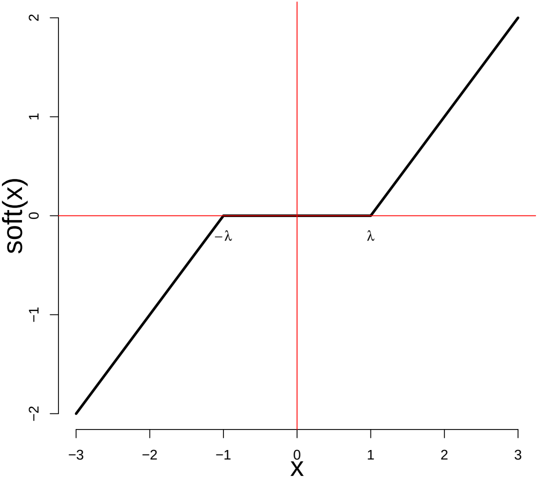
Reminders

- Notes posted to course page
  `http://interml.ydata123.org`
  - Notes on lasso optimization posted last week
- Readings from “Probabilistic Machine Learning: An Introduction”
  `https://probml.github.io/pml-book/book1.html`
- Also: “Probabilistic Machine Learning: Advanced Topics”
  `https://probml.github.io/pml-book/book2.html`

Recap: Lasso

- Lasso navigates bias-variance tradeoff by selecting subsets of predictor variables
- Replaces ℓ_2 norm of ridge regression by ℓ_1 norm
- Key is to combine sparsity with convexity
- Fundamental operation of lasso is *soft-thresholding*
- A scalable algorithm for computing the lasso estimator is *iterative soft thresholding*
  - iteratively compute a 1-dimensional lasso using soft-thresholding
  - cycle over the variables one at a time

ℓ_1 and soft thresholding



$$\text{Soft}_\lambda(X) \equiv \text{sign}(X)\ (|X| - \lambda)_+.$$

The lasso: Computing $\widehat{\beta}$

To minimize $\frac{1}{2n}\sum_{i=1}^{n}(Y_i - \beta^T X_i)^2 + \lambda\|\beta\|_1$ by *coordinate descent*:

- Standardize the predictor variables
- Set $\widehat{\beta} = (0, \ldots, 0)$ then iterate until converged:
- for $j = 1, \ldots, p$:
  - set $R_i = Y_i - \sum_{s\neq j}\widehat{\beta}_s X_{si}$
  - Set $\widehat{\beta}_j$ to be least squares fit of R_i's on X_j.
  - $\widehat{\beta}_j \leftarrow \mathrm{Soft}_\lambda(\widehat{\beta}_j)$
- Then use least squares $\widehat{\beta}$ on selected subset S.

The lasso

To find choose regularization/sparsity level λ:

1. Find $\widehat{\beta}(\lambda)$ and $\widehat{S}(\lambda)$ for each λ.
2. Compute $\widehat{R}(\lambda)$ for each λ using LOOCV.
3. Choose $\widehat{\lambda} = \arg\min_\lambda \widehat{R}(\lambda)$ to minimize estimated risk.
4. Let $\widehat{S} = \widehat{S}(\widehat{\lambda})$ be the selected variables.
5. Let $\widehat{\beta} = \widehat{\beta}(\widehat{\lambda})$ be the least squares estimator using only $\widehat{S}$.
6. Prediction: $\widehat{Y} = X^T \widehat{\beta}$.

Nonparametric Regression

Given $(X_1, Y_1), \ldots, (X_n, Y_n)$ predict Y from X.

Assume only that $Y_i = m(X_i) + \epsilon_i$ where where $m(x)$ is a smooth function of x.

The most popular (classical) methods are *kernel methods*. However, there are two types of kernels:

1. Smoothing kernels
2. Penalization kernels (Mercer kernels)

Smoothing kernels involve local averaging.
Mercer kernels involve norms and regularization.

Smoothing Kernels

- Smoothing kernel estimator:

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^n Y_i \; K_h(X_i, x)}{\sum_{i=1}^n K_h(X_i, x)} = \sum_{i=1}^n w_i(x) Y_i$$

where $K_h(x, z)$ is a *kernel* such as

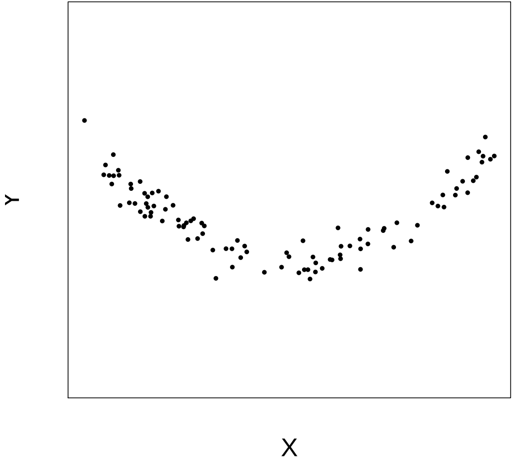
$$K_h(x, z) = \exp\left(-\frac{\|x - z\|^2}{2h^2}\right)$$

and $h > 0$ is called the *bandwidth*.

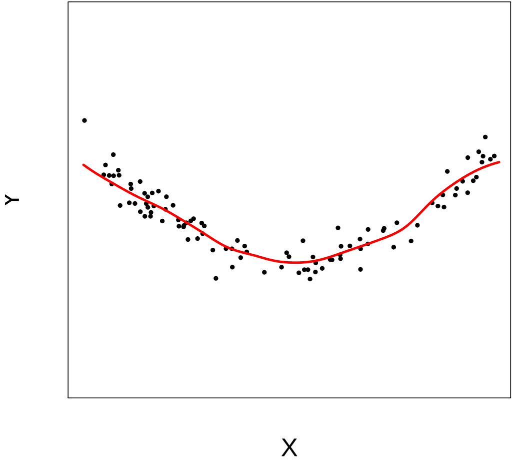
- $\widehat{m}_h(x)$ is just a local average of the Y_i's near x.

- The bandwidth h controls the bias-variance tradeoff:
Small h = large variance while *large h = large bias*.

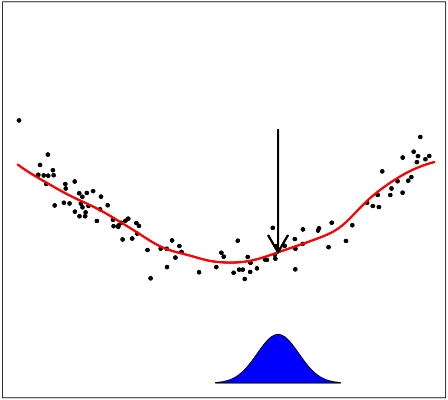
Example: Some Data – Plot of Y_i versus X_i



Example: $\widehat{m}(x)$



$\widehat{m}(x)$ is a local average

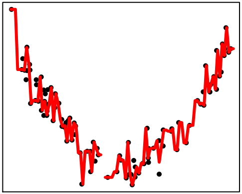


$\widehat{m}(x)$ is a local average

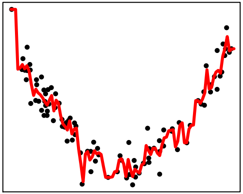
The estimator minimizes a weighted least squares criterion

$$\widehat{m}(x) = \arg\min_c \sum_{i=1}^{n} w_i(x)(y_i - c)^2$$

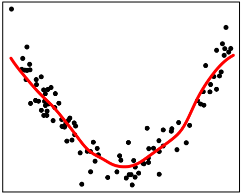
Effect of the bandwidth h



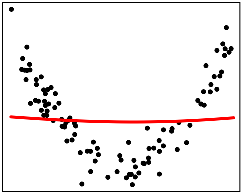
very small bandwidth



small bandwidth



medium bandwidth



large bandwidth

Smoothing Kernels

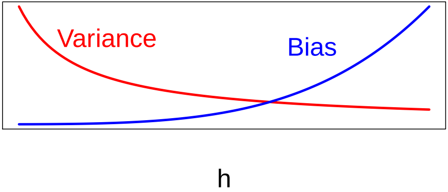
Risk $= \mathbb{E}(Y - \widehat{m}_h(X))^2 = \text{bias}^2 + \text{variance} + \sigma^2$.

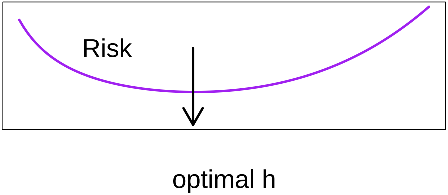
$\sigma^2 = \mathbb{E}(Y - m(X))^2$ is the unavoidable prediction error.

small h: low bias, high variance (undersmoothing)

large h: high bias, low variance (oversmoothing)

Risk Versus Bandwidth





The kernel shape doesn’t really matter

Let’s go to the notebook

Estimating the Risk: Cross-Validation

To choose h we need to estimate the risk $R(h)$. We can estimate the risk by using *cross-validation*.

1. Omit (X_i, Y_i) to get $\widehat{m}_{h,(i)}$, then predict: $\widehat{Y}_{(i)} = \widehat{m}_{h,(i)}(X_i)$.
2. Repeat this for all observations.
3. The cross-validation estimate of risk is:

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_{(i)})^2.$$

Shortcut formula: Whenever $\widehat{Y} = LY$ we can use the shortcut

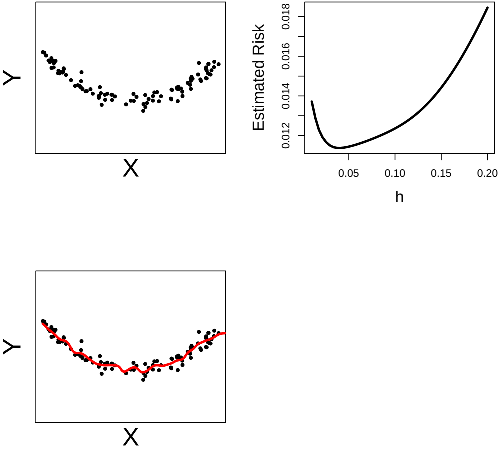
$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \widehat{Y}_i}{1 - L_{ii}}\right)^2.$$

In this case $L_{ii} = K_h(X_i, X_i)/\sum_t K_h(X_i, X_t)$.

Summary so far

1. Compute $\widehat{m}_h$ for each h
2. Estimate the risk $\widehat{R}(h)$ using LOOCV
3. Choose bandwidth $\widehat{h}$ to minimize $\widehat{R}(h)$
4. Let $\widehat{m}(x) = \widehat{m}_{\widehat{h}}(x)$

Example



The curse of dimensionality

The method is easily applied in high dimensions — but it doesn't work well.

- The squared bias scales as h^4 and the variance scales as $\frac{1}{nh^p}$
- As a result, the risk goes down no faster than $n^{-4/(4+p)}$
- Suppose we want to make this small, of size ϵ—how many data points do we need?

$$n \geq \left(\frac{1}{\epsilon}\right)^{1+p/4}$$

- Grows exponentially with dimension—*the curse of dimensionality*

We'll derive this later in the class.

Kernel density estimation

To estimate a density, use the same idea behind kernel smoothing:

$$\begin{aligned}\widehat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} K_h(X_i, x) \\ &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h}K\left(\frac{X_i - x}{h}\right)\end{aligned}$$

We require that $\int K(u)\,du = 1$ and $K \geq 0$ is symmetric around zero (an even function).

This places a “bump function” around each data point, and averages them (a mixture model)

Kernel density estimation

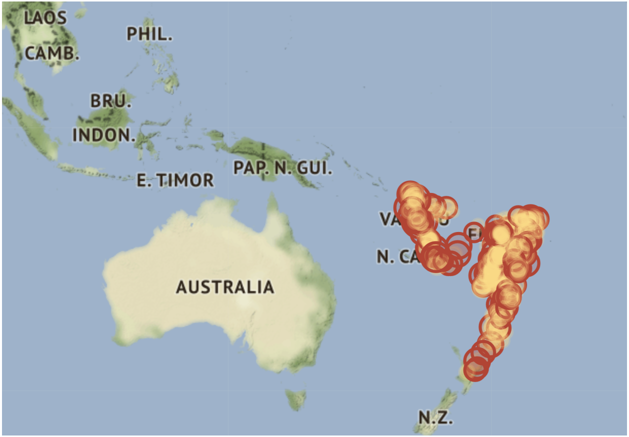
In p dimensions:

$$\begin{aligned}\widehat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} K_h(X_i, x) \\ &= \frac{1}{n\,h^p}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)\end{aligned}$$

We require that $\int K(u)\,du = 1$ and K is symmetric around zero.

This places a “bump function” around each data point, and averages them (a mixture model)

KDE demo: Fiji earthquakes



Kernel density estimation

The bias-variance tradeoff:

$$\mathrm{bias}^2(x) \approx h^4$$
$$\mathrm{var}(x) \approx \frac{1}{n\,h^p}$$

Note that the variance scales according to the expected number of data points in a cube of side length h in p-dimensions.

We'll go through the calculation of this on the board. Notes are posted to `http://interml.ydata123.org`

Back to regression

Using a kernel density estimator, the “plug-in” regression estimate gives us back the kernel smoother:

$$\widehat{m}(x) = \int y\,\widehat{f}(y \mid x)\,dy$$

$$= \frac{\int y\,\widehat{f}(x, y)\,dy}{\widehat{f}(x)}$$

$$= \frac{\sum_i Y_i K_h(X_i, x)}{\sum_i K_h(X_i, x)}$$

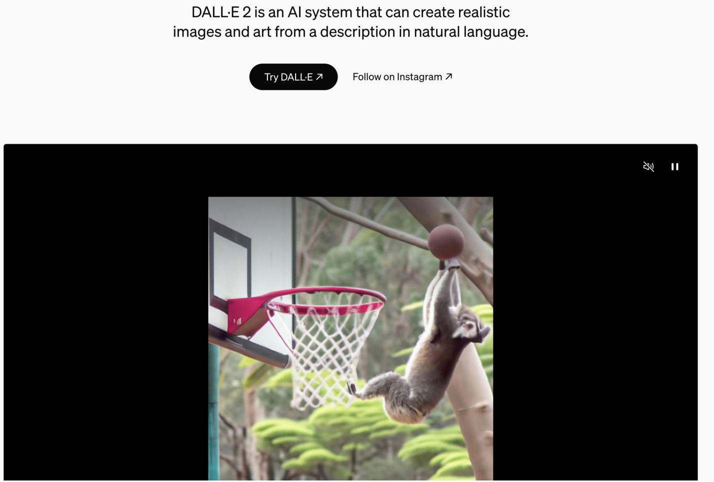
Generative models

- A density estimate is a *generative model*
- We can sample from the density to “generate” a new data point
- What is an algorithm for sampling from the estimated distribution?

Generative models

1. Sample an index i uniformly from 1 to n
2. Sample a point x from a Gaussian with mean X_i and variance h^2

Generative models



Generative models

As we'll see later in the course, Transformers can be naturally seen as a form of kernel smoothing and kernel density estimation.

Summary

- Smoothing methods compute local averages, weighting points by a kernel
- Shape of the kernel doesn't matter (much)
- KDE places a density around each data point, and averages
- The curse of dimensionality limits use of both approaches to low dimensions