S&DS 365 / 665

**Intermediate Machine Learning**

# Lasso, Smoothing and Kernels

Monday, January 31

Yale

# Reminders

- OH posted to Canvas / EdD
- Reminder: Slides updated often; please refresh
- Assignment 1 next week
- Any questions?

# Topics for today

- Recap of lasso
- A simple algorithm for the lasso
- Nonparametric regression
- Smoothing methods
- Bias, variance, and curse of dimensionality

# Recall from last time

- For low dimensional (linear) prediction, we can use least squares.
- For high dimensional linear regression, we face a bias-variance tradeoff: omitting too many variables causes bias while including too many variables causes high variance.
- The key is to select a good subset of variables.
- The *lasso* ($\ell_1$-regularized least squares) is a fast way to select variables.
- If there are good, sparse linear predictors, lasso will work well.

# Regression

Given the training data $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ we want to construct $\widehat{m}$ to make

$$\text{prediction risk} = R(\widehat{m}) = \mathbb{E}(Y - \widehat{m}(X))^2$$

small. Here, $(X, Y)$ are a new pair.

**Key fact**: Bias-variance decomposition:

$$R(\widehat{m}) = \int \text{bias}^2(x)p(x)dx + \int \text{var}(x)p(x) + \sigma^2$$

where

$$\begin{aligned} \text{bias}(x) &= \mathbb{E}(\widehat{m}(x)) - m(x) \\ \text{var}(x) &= \text{Variance}(\widehat{m}(x)) \\ \sigma^2 &= \mathbb{E}(Y - m(X))^2 \end{aligned}$$

# Bias-Variance Tradeoff

More generally, we need to tradeoff approximation error against estimation error:

$$R(\widehat{f}) - R^* = \underbrace{R(\widehat{f}) - \inf_{f \in \mathcal{F}} R(f)}_{\text{estimation error}} + \underbrace{\inf_{f \in \mathcal{F}} R(f) - R^*}_{\text{approximation error}}$$

where $R^*$ is the smallest possible risk and $\inf_{f \in \mathcal{F}} R(f)$ is smallest possible risk using class of estimators $\mathcal{F}$.

- Approximation error is a generalization of squared bias
- Estimation error is a generalization of variance
- Decomposition holds more generally, even for classification

# Sparse Linear Regression

Ridge regression does not take advantage of sparsity.

Maybe only a small number of covariates are important predictors. How do we find them?

We could fit many submodels (with a small number of covariates) and choose the best one. This is called *model selection*.

The inaccuracy is

prediction error = bias$^2$ + variance + $\sigma^2$

Now the bias is the errors due to omitting important variables, and the variance is the error due to having to estimate many parameters.

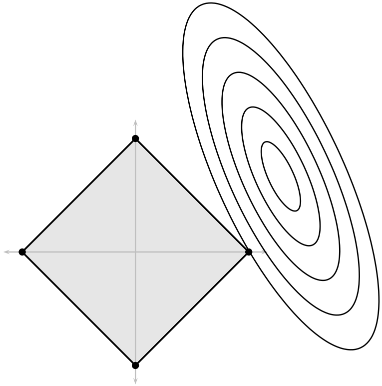
# Sparsity Meets Convexity

**Lasso regression**

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{2n} \sum_{i=1}^{n} (Y_i - \beta^T X_i)^2 + \lambda \|\beta\|_1$$

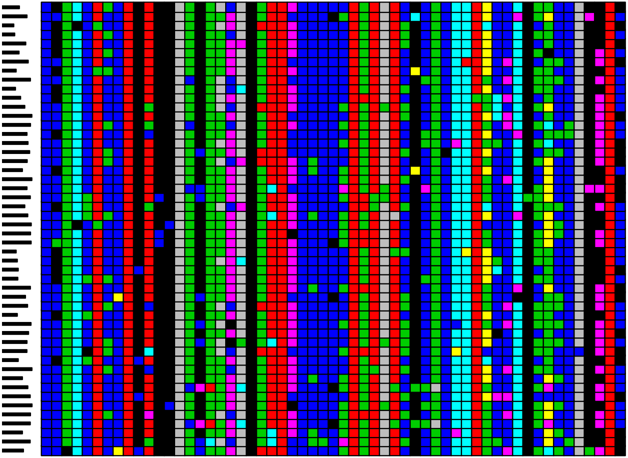
where $\|\beta\|_1 = \sum_j |\beta_j|$.

# Sparsity: How corners create sparse estimators

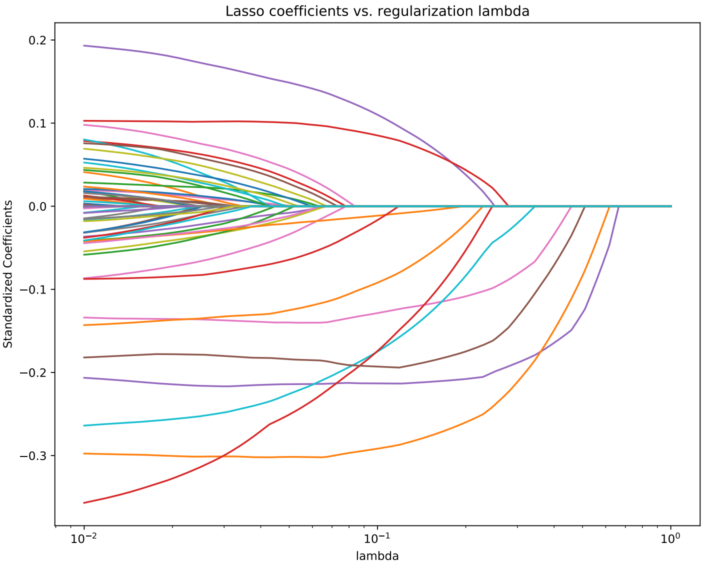


# The lasso: HIV example

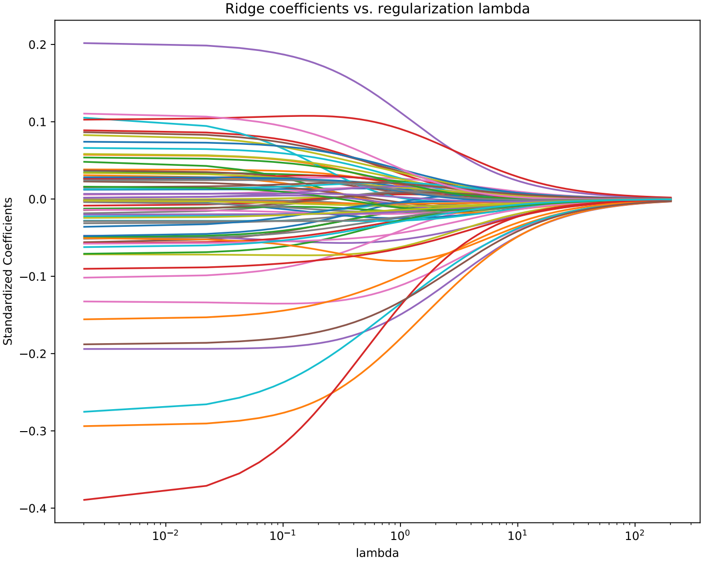
- $Y$ is resistance to HIV drug.
- $X_j$ = amino acid in position $j$ of the virus.
- $p = 99$, $n \approx 100$.



# The lasso: HIV example



# Contrast with ridge regression



# The lasso

- $\widehat{\beta}(\lambda)$ is called the lasso estimator. Selected set of variables is

$$\widehat{S}(\lambda) = \left\{ j : \ \widehat{\beta}_j(\lambda) \neq 0 \right\}.$$

- After you find $\widehat{S}(\lambda)$, you should re-fit the model by doing least squares on the sub-model $\widehat{S}(\lambda)$.

# Selecting $\lambda$

To choose $\lambda$ by risk estimation:

Re-fit the model with the non-zero coefficients. Then apply leave-one-out cross-validation:

$$\widehat{R}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_{(i)})^2 = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \widehat{Y}_i)^2}{(1 - H_{ii})^2} \approx \frac{1}{n}\frac{RSS}{\left(1 - \frac{s}{n}\right)^2}$$

where $RSS$ is residual sum of squares and $H$ is the hat matrix and $s = \|\widehat{\beta}\|_0 = \#\{j : \widehat{\beta}_j \neq 0\}$.

Choose $\widehat{\lambda}$ to minimize $\widehat{R}(\lambda)$.

# The lasso

The complete steps are:

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

# An algorithm for the lasso: Derived in steps

We'll derive a simple algorithm for computing the lasso solution in steps.

I'll do the first step in detail. The next steps only require simple calculations that I'll leave to you.

# An algorithm for the lasso: Step 1

First consider minimizing

$$\frac{1}{2}(y - \beta)^2 + \lambda|\beta|$$

where $y$ is a single number.

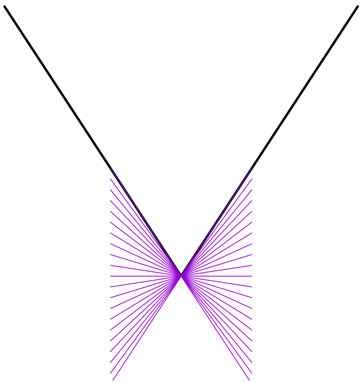
Taking the derivative and setting to zero, we get

$$\beta - y + \lambda v = 0$$

where

$$v \quad \begin{cases} = \text{sign}(\beta) & \text{if } |\beta| > 0 \\ \in [-1, 1] & \text{if } \beta = 0. \end{cases}$$

# Subdifferential for $|\cdot|$



The set of vectors $v$ pass through the tip at 0 and have slope between -1 and 1.

# An algorithm for the lasso: Step 1

Solution can be written as

$$\widehat{\beta} = \begin{cases} y - \lambda & \text{if } \beta > 0 \\ y + \lambda & \text{if } \beta < 0 \\ y - \lambda\left(\frac{y}{\lambda}\right) & \text{if } \beta = 0. \end{cases}$$

Equivalently:

$$\widehat{\beta} = \begin{cases} y - \lambda & \text{if } y > \lambda \\ y + \lambda & \text{if } y < -\lambda \\ 0 & \text{if } |y| \leq \lambda. \end{cases}$$

# An algorithm for the lasso: Step 1

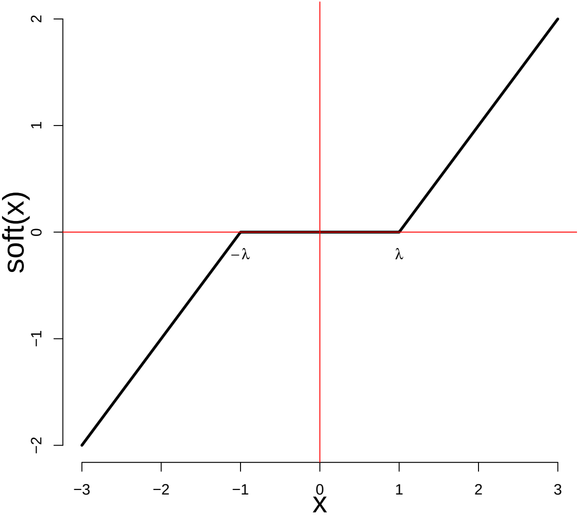
Equivalently:

$$\widehat{\beta} = \begin{cases} y - \lambda & \text{if } y > \lambda \\ y + \lambda & \text{if } y < -\lambda \\ 0 & \text{if } |y| \le \lambda. \end{cases}$$

**Soft thresholding**

$$\begin{aligned} \widehat{\beta} &= \mathrm{Soft}_\lambda(y) \\ &\equiv \mathrm{sign(y)}\,(|y| - \lambda)_+ = \left(1 - \frac{\lambda}{|y|}\right)_+ y \end{aligned}$$

# Soft thresholding



# An algorithm for the lasso: Step 2

Next consider minimizing

$$\frac{1}{2}(y - x\beta)^2 + \lambda|\beta|$$

where $y$ and $x$ are a single numbers.

Exercise: Show that

$$\widehat{\beta} = \text{Soft}_{\frac{\lambda}{x^2}}\left(\frac{xy}{x^2}\right)$$

# An algorithm for the lasso: Step 3

Now consider minimizing

$$\frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i\beta)^2 + \lambda|\beta|$$

for data $(x_1, y_1), \ldots, (x_n, y_n)$ with $p = 1$.

Exercise: Show that

$$\widehat{\beta} = \mathrm{Soft}_{\lambda_x}\left(\widehat{\beta}_{OLS}\right)$$

where $\beta_{OLS}$ is the least squares estimator and $\lambda_x = \frac{\lambda}{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$.

# An algorithm for the lasso: Step 4

Finally, consider minimizing

$$\frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda|\beta|$$

for data $(x_1, y_1), \ldots, (x_n, y_n)$ with $p \geq 1$. Apply previous algorithm to estimate $\beta_1$, holding other coefficients fixed.

Exercise: Show that

$$\widehat{\beta}_1 = \text{Soft}_{\lambda_1}\left(\widehat{\beta}_{OLS}\right)$$

where $\beta_{OLS}$ is the least squares estimator for data $(x_{i1}, r_i)$ with $r_i = y_i - \sum_{j\neq 1} x_{ij}\beta_j$. and $\lambda_1 = \frac{\lambda}{\frac{1}{n}\sum_{i=1}^{n} x_{i1}^2}$.

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

To minimize $\frac{1}{2n}\sum_i (y_i - \beta^T x_i)^2 + \lambda\|\beta\|_1$:

**Lasso by coordinate descent**

- Set $\widehat{\beta} = (0, \ldots, 0)$, then iterate until convergence:
- 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 $x_j$.
  - $\widehat{\beta}_j \leftarrow \mathrm{Soft}_{\lambda_j}(\widehat{\beta}_j)$ where $\lambda_j = \frac{\lambda}{\frac{1}{n}\sum_i x_{ij}^2}$.
- Then use least squares $\widehat{\beta}$ on selected subset $\widehat{S}(\lambda)$.

# Next up

Nonparameteric regression by smoothing

# 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 methods are *kernel methods*. However, there are two types of kernels:

1. Smoothing kernels
2. Mercer kernels

Smoothing kernels involve local averaging.
Mercer kernels involve regularization.

**Smoothing kernel estimator**

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

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

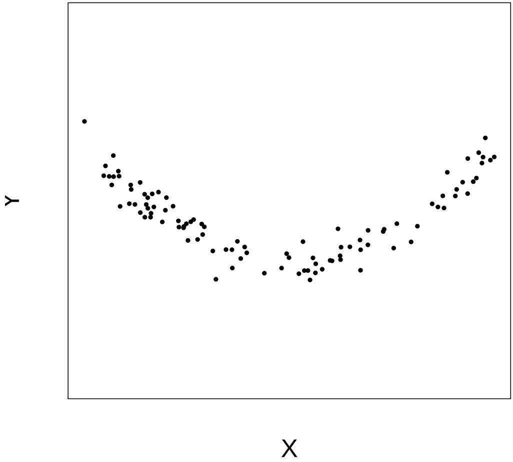
$$K_h(x, z) = \exp\left(-\frac{\|x - z\|^2}{2h^2}\right)$$

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

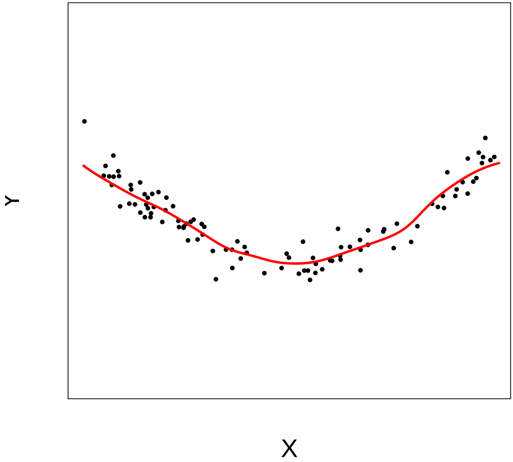
- $\widehat{m}_h(x)$ is just a local average of the $Y_i$'s near $x$.

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

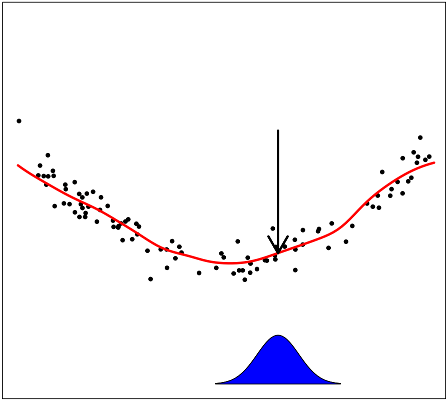
# Example: Some Data – Plot of $Y_i$ versus $X_i$



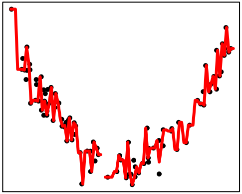
# Example: $\widehat{m}(x)$



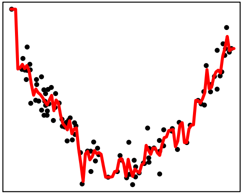
# $\widehat{m}(x)$ is a local average



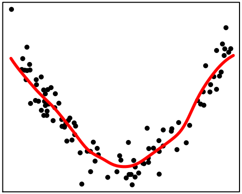
# Effect of the bandwidth $h$



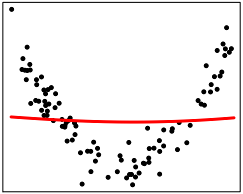
very small bandwidth



small bandwidth



medium bandwidth



large bandwidth

Let’s go to the notebook

# Smoothing Kernels

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

Under mild assumptions on the distribution of the data:

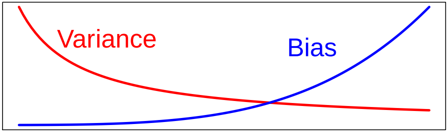
$$\text{bias}^2 \approx h^4$$
$$\text{variance} \approx \frac{1}{nh^p}$$

where $p$ = dimension of $X$.

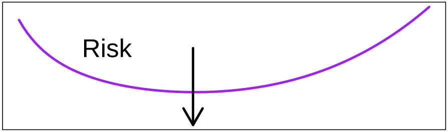
$\sigma^2 = \mathbb{E}(Y - m(X))^2$ is the unavoidable prediction error.

*small h*: low bias, high variance (undersmoothing)
*large h*: high bias, low variance (oversmoothing)

# Risk Versus Bandwidth




h




optimal h

# 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*:

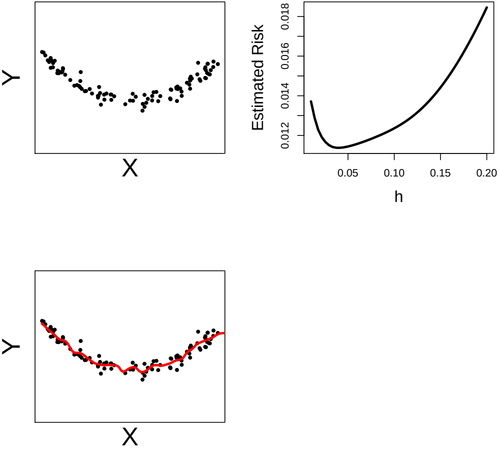
$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \widehat{Y}_i}{1 - L_{ii}}\right)^2$$

where $L_{ii} = K_h(X_i, X_i) / \sum_{i'=1}^{n} K_h(X_i, X_{i'})$.

# Summary so far

1. Compute $\widehat{m}_h$ for each $h$.
2. Estimate the risk $\widehat{R}(h)$.
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 $n^{-4/(4+p)}$ (Exercise)
- Suppose we want to make this small, of size $\epsilon$—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*

# Additive models

A compromise is to use *additive models* of the form

$$\widehat{m}(x) = \widehat{m}_1(x_1) + \widehat{m}_2(x_2) + \cdots \widehat{m}_p(x_p)$$

- Each function $\widehat{m}_j(x_j)$ is estimated by smoothing, holding the other functions fixed
- Soft thresholding can be used in high dimensions, leading to a generalization of the lasso

---

https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2009.00718.x

# Summary for today

- The lasso can be computed by iterative soft thresholding
- Smoothing methods compute local averages, weighting points by a kernel
- The curse of dimensionality limits use of smoothing methods to low dimensions