

Linear Model in High Dimensions, I: Introduction and Implementation

Jesper Riis–Vestergaard Sørensen

University of Copenhagen, Department of Economics

October 6, 2022

Overview

Introduction

A High-Dimensional Framework

Least Squares in High Dimensions

Sparsity

Sparsity-Inducing Estimation

Penalty Selection

Python Implementation

Introduction

Linear Mean Regression Model

$$Y = X'\beta + \varepsilon = \sum_{j=1}^p \beta_j X_j + \varepsilon, \quad \mathbb{E}[\varepsilon \mid X] = 0.$$

Y : Dependent variable (scalar)

$X = (X_1, \dots, X_p)'$: Covariates

$\beta = (\beta_1, \dots, \beta_p)'$: Coefficients

ε : Noise/unobservables

p : Number of covariates

Classical Approach to Modeling and Sampling

Independent sampling n times in accordance with

$$Y = \sum_{j=1}^p \beta_j X_j + \varepsilon, \quad \mathbb{E}[\varepsilon \mid X] = 0.$$

Distribution (X, Y) thought of as fixed (typically).

From the (infinite) array

$$(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n), (X_{n+1}, Y_{n+1}), \dots$$

we have access to the ‘first’ n observations.

Data: $(X_1, Y_1), \dots, (X_n, Y_n) \stackrel{\text{indep.}}{\sim} (X, Y)$

Classical Approach to Estimation and Inference

Least squares (LS) estimator:

$$\hat{\beta}^{\text{LS}} := \left(\sum_{i=1}^n X_i X_i' \right)^{-1} \sum_{i=1}^n X_i Y_i.$$

LLN + CLT \Rightarrow asymptotic distribution:

$$\sqrt{n}(\hat{\beta}^{\text{LS}} - \beta) \xrightarrow{d} N(\mathbf{0}_{\textcolor{red}{p} \times \textcolor{red}{1}}, \mathbf{V}) \text{ as } n \rightarrow \infty,$$

where

$$\mathbf{V} := \mathbf{A}^{-1} \mathbf{B} \mathbf{A}^{-1},$$

$$\mathbf{A} := E[XX'],$$

$$\mathbf{B} := E[\varepsilon^2 XX'].$$

think the CDF $F_{\cdot}(X)$

(Review Q: What does \rightarrow_d mean?)

Asymptotics yield Approximations

View \rightarrow_d as $=_d$ (or \sim). Suggests approximation

$$\widehat{\beta}^{\text{LS}} \stackrel{d}{\approx} N(\beta, \mathbf{V}/n) \quad \text{for } n \text{ 'large.'}$$

Allows us to gauge uncertainty. For example,

$$\text{CI}_j(.95) := \left[\widehat{\beta}_j^{\text{LS}} \pm 1.96 \times \sqrt{V_{jj}/n} \right]$$

(With \mathbf{V} unknown, construct $\widehat{\mathbf{V}}$ consistent. Same idea.)

CIs **valid in asymptotic sense**,

$$P(\beta_j \in \text{CI}_j(.95)) \xrightarrow[\text{jacking up the sample size}]{\text{whatever the statistical significance level I evaluate by}} .95 \text{ as } n \rightarrow \infty.$$

Problem in High Dimensions

Implicit in ‘as $n \rightarrow \infty$ ’: p is fixed, so $p/n \rightarrow 0$.
ndim for \beta
or number of regressors

Abstraction makes sense for datasets where $p/n \approx 0$.

Approximation should be accurate w/ n ‘large’ and p ‘small.’
with a large n

With p/n nonnegligible? May be poor.

in modern datasets, not
always true

approx may be very poor

it just doesnt fit the
characteristics of the said
dataset

But p/n may be sizeable in applications.

How do we estimate β ? Do approximations? Construct CIs?

Example: What Determines Economic Growth?

Sala-i-Martin (1997), “I just ran two million regressions,” *AER*

Cross-country growth regression

$$\text{long-run GDP growth} = X'\beta + \varepsilon, \quad E[\varepsilon|X] = 0.$$

X : 62 country-level variables.

Including initial level of GDP. do initially poor countries catch up to initially rich countries?

Only 200ish countries...

Only 90 complete observations

$$p/n \approx \frac{2/3}{\quad}$$

'jacking up n ' -> can't generate more countries...

limit to how small p/n can actually be in this context

Example: Text Regression

Wu (2018), “Gendered Language on the Econ Job Market Rumors Forum,” (AER P&P)

Q: Are men and women portrayed differently?

Using anonymous discussions on EJMR.

Text regression

$$\text{Post discusses Female} = X'\beta + \varepsilon, \quad E[\varepsilon|X] = 0.$$

$n \approx 300,000$ posts

X : Word counts of 10,000 most frequent words.

no way, that this converges in distribution to a normal distribution

(Lots to improve here...)

A High-Dimensional Framework

High-Dimensional Linear Regression Model

Linear model (as before):

$$Y = X'\beta + \varepsilon = \sum_{j=1}^p \beta_j X_j + \varepsilon, \quad \mathbb{E}[\varepsilon \mid X] = 0.$$

Data (as before): $(X_1, Y_1), \dots, (X_n, Y_n) \stackrel{\text{indep.}}{\sim} (X, Y)$

NEW: To allow p ‘large,’ let $p = p_n$ and $p/n \rightarrow \text{const.} \in (0, 1]$

p/n greater than zero, even
in the asymptotic sense

(Later: $p > n$ or even $p/n \rightarrow \infty$ allowed.)

Approximations should take ‘large p ’ into account.

Sampling in High Dimensions

$$Y_i = \sum_{j=1}^{p_n} \beta_j X_{ij} + \varepsilon, \quad \mathbb{E}[\varepsilon_i \mid X_i] = 0.$$

If p depends on n , so must (X, Y) distribution.

$\Rightarrow (X_i, Y_i)$ should be (further) indexed by n .

Sampling is now from the array of arrays

$$\begin{array}{ll} (X_{1,1}, Y_{1,1}), (X_{1,2}, Y_{1,2}), (X_{1,3}, Y_{1,3}), \dots & (n = 1) \\ (X_{2,1}, Y_{2,1}), (X_{2,2}, Y_{2,2}), (X_{2,3}, Y_{2,3}), \dots & (n = 2) \\ (X_{3,1}, Y_{3,1}), (X_{3,2}, Y_{3,2}), (X_{3,3}, Y_{3,3}), \dots & (n = 3) \\ \vdots & \end{array}$$

We suppress the array subscript throughout.

Least Squares in High Dimensions

Predictive Behavior of OLS with Large p , I

Suppose our *goal* is to *predict* (\hat{Y}_i) *outcome* (Y_i) .

LS predictor: $\hat{Y}_i^{\text{LS}} := X_i' \hat{\beta}^{\text{LS}}$.

Optimal predictor: $Y_i^* := E[Y_i | X_i] = X_i' \beta$.

Measure performance by (expected average square) *prediction error*

$$E \left[\frac{1}{n} \sum_{i=1}^n \left(\hat{Y}_i^{\text{LS}} - Y_i^* \right)^2 \right] = E \left[\frac{1}{n} \sum_{i=1}^n \left(X_i' \hat{\beta}^{\text{LS}} - X_i' \beta \right)^2 \right].$$

How does OLS perform?

Predictive Behavior of OLS with Large p , II

Lemma

Suppose ε independent of X and $\varepsilon \sim N(0, \sigma^2)$. Then

$$E \left[\frac{1}{n} \sum_{i=1}^n \left(X_i' \hat{\beta} - X_i' \beta \right)^2 \right] = \frac{\sigma^2 p}{n}.$$

With p sizeable, $p/n \nrightarrow 0$ as $n \rightarrow \infty$.

Take-away: p large \implies OLS prediction poor.

Proof: See video.

Estimation Behavior of OLS with Large p

What if interest lies in *estimation* (of β)?

How does OLS perform?

Consider special case of orthonormal design,

$$\frac{1}{n} \sum_{i=1}^n X_i X' = \mathbf{I}_p.$$

Expected squared estimation error?

$$E \left[\sum_{j=1}^p \left(\hat{\beta}_j^{\text{LS}} - \beta_j \right)^2 \right] = \dots$$

► Problem? [Whiteboard]

Sparsity

Key Condition: Sparsity

Rescue comes from believing only few β_j 's nonzero.

Sparsity means

$$s := s_n := \sum_{j=1}^p \mathbf{1}\{\beta_j \neq 0\} \text{ is 'small' (relative to } n)$$

Lasso (below) useful when

$$s/n \rightarrow 0.$$

Outperforms LS when

$$p/s \rightarrow \infty.$$

(More generally: Rescue comes from low-dimensional structure.)

Exact vs. Approximate Sparsity

Two types of sparsity: **Exact** and **approximate**.

Exact sparsity:

$$s = \sum_{j=1}^p \mathbf{1}\{\beta_j \neq 0\} \text{ is small}$$

► E.g. $\beta = (1, 3, 0, \dots, 0, 2, 0, \dots, 0)'$

Approximate sparsity: Most \approx zero + few far from.

► E.g. β_j 's (ordered) geometrically decaying, e.g.

$$\beta = (1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27} \dots, \frac{1}{3^{p-1}})'$$

Relevance?

What Determines Economic Growth? (ctnd)

Cross-country growth regression

$$\text{long-run GDP growth} = X'\beta + \varepsilon, \quad E[\varepsilon|X] = 0.$$

X : 62 country-level variables.

Sala-i-Martin: Regressions of form:

3 vars

- ▶ Always include 1960 GDP, life expectancy, school enrollment.
- ▶ Fix 1 additional variable in turn, and run $\binom{58}{3}$ regressions.
- ▶ Report variables that are “significant the most” ($>.<$)

Implicit (sparsity) assumption? $\frac{\text{s less than or equal to 7}}{3+3+1}$

Sparsity-Inducing Estimation

Estimation with Sparsity Constraint

If we believe in sparsity (simplicity)—encourage it.

Suppose s (but not $\{j; \beta_j \neq 0\}$) known.

Let $\|b\|_0$ denote number of nonzero components in b .
norm zero = count the number of non-zero elements in vector 'b'

Could add constraint and (try to) solve

$$\min_{\substack{b \in \mathbf{R}^p, \\ \|b\|_0 \leq s}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2.$$

again, minimizing the sq. errors

We don't know s . But could consider

R = tolerance for the number of non-zeros

$$\tilde{\beta}(R) \in \operatorname{argmin}_{\substack{b \in \mathbf{R}^p, \\ \|b\|_0 \leq R}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2$$

for various $R \geq 0$.

Fit/Complexity Trade-Off

$$\tilde{\beta}(R) \in \underset{\substack{b \in \mathbf{R}^p, \\ \|b\|_0 \leq R}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2$$

$R \geq 0$: Complexity tolerance of our choosing (more later).

the bigger the R , the more complexity I can accept

Governs (mis)fit/complexity trade-off.

- ▶ $R = 0$ forces all zeros (null model).
- ▶ $R \geq p$ recovers (unconstrained) LS.

Best Subset Selection Estimator

$$\tilde{\beta}(R) \in \underset{\substack{b \in \mathbf{R}^p, \\ \|b\|_0 \leq R}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2$$

Best subset selection (of size R) estimator

Nonconvex optimization problem! (Think $R = 1$ in 2D)

Deal-breaker when $p > 40$ or so.

- ▶ $p = 100$ requires solving 2^{100} problems(!)
- ▶ Computationally infeasible.

Convex Relaxation of Constraint

Swapping $\|b\|_0$ for $\|b\|_1$, we get minimization problem

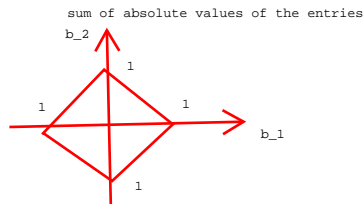
$$\min_{\substack{b \in \mathbb{R}^p, \\ \|b\|_1 \leq R}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2, \quad \|b\|_1 = \sum_{j=1}^p |b_j|.$$

Convex minimization problem!

Easy to solve even with very large p .

Note: ‘Complexity’ of b now distance $\|b\|_1$ to origin.

To make sense, X_j ’s should be (brought) on(to) same scale.



Penalized Least Squares and Lasso

Tibshirani (1996) “Regression shrinkage and selection via the Lasso”

$$\min_{\substack{b \in \mathbf{R}^p, \\ \|b\|_1 \leq R}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2, \quad \|b\|_1 = \sum_{j=1}^p |b_j|.$$

May equivalently solve penalized version

$$\underbrace{\hat{\beta}(\lambda)}_{\text{Lasso}} \in \operatorname{argmin}_{b \in \mathbf{R}^p} \left\{ \underbrace{\frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2}_{(\text{mis})\text{fit}} + \underbrace{\lambda \|b\|_1}_{\text{penalty}} \right\}.$$

Penalty level $\lambda \geq 0$ of our choosing (later)

Acronym: **L**east **A**bsolute **S**hrinkage and **S**election **O**perator.

Shrinkage: Orthonormal Design, I

When $n^{-1} \sum_i X_i X_i' = \mathbf{I}_p$, explicit solution:

$$\hat{\beta}_j(\lambda) = \text{sgn}(\hat{\beta}_j^{\text{LS}}) \left(|\hat{\beta}_j^{\text{LS}}| - \frac{\lambda}{2} \right)_+, \quad j = 1, 2, \dots, p,$$

$$\text{sgn}(z) := \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases} \quad (\text{sign})$$

$$(z)_+ := \max(0, z). \quad (\text{positive part})$$

- ▶ Shrinkage towards origin apparent.
- ▶ Lasso = here **soft thresholding** of LS.
- ▶ Analytic solution unavailable \Rightarrow numerical optimization.

Shrinkage: Orthonormal Design, II

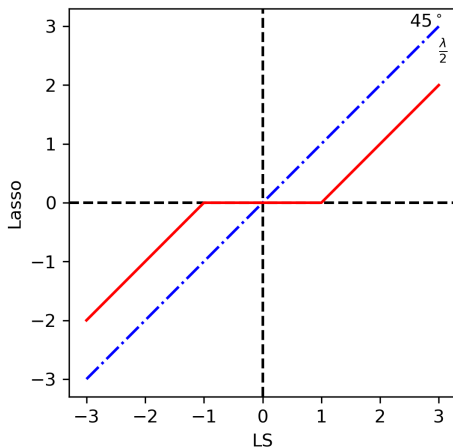


Figure: Lasso vs. Least Squares ($\lambda = 2$)

Selection

$$\text{Lasso: } \hat{\beta}(\lambda) \in \operatorname{argmin}_{b \in \mathbf{R}^p} \left\{ \frac{1}{n} \sum_{i=1}^n (Y_i - X_i' b)^2 + \lambda \sum_{j=1}^p |b_j| \right\}$$

Since $|\cdot|$ has kink at zero, $\hat{\beta}_j(\lambda)$'s tend to be exactly zero.

(Easier to see from constrained formulation.)

If λ large enough, will see exact zeros.

Thus, Lasso does **variable selection**:

- ▶ Variable j is selected if $\hat{\beta}_j(\lambda) \neq 0$.
- ▶ Variable j not selected if $\hat{\beta}_j(\lambda) = 0$.

Penalty Selection

How do we Choose λ ?

Four methods:

1. Sample splitting (validation)
2. Cross-validation
3. Bickel-Ritov-Tsybakov rule
4. Belloni-Chen-Chernozhukov-Hansen rule

See also [Chetverikov & Sørensen \[2021\]](#) for novel approach going far beyond the linear model

Sample Splitting, I

Randomly divide sample in two.

- ▶ **Training**/estimation sample
- ▶ **Validation** sample

$$\underbrace{(X_1, Y_1), \dots, (X_m, Y_m)}_{\text{training sample}}, \underbrace{(X_{m+1}, Y_{m+1}), \dots, (X_n, Y_n)}_{\text{validation sample}}$$

Hastie et al.: Reserve also **testing** sample to evaluate. (Ignored.)

Sample Splitting, II

Construct Lasso $\hat{\beta}_m(\lambda)$ using **training** sample, $\{(X_i, Y_i)\}_1^m$

Check (mis)fit using **validation** sample:

$$\mathcal{F}(\lambda) := \sum_{i=m+1}^n \{Y_i - X_i' \hat{\beta}_m(\lambda)\}^2$$

Choose λ to obtain best possible fit:

$$\hat{\lambda}^{\text{ss}} := \underset{\lambda}{\operatorname{argmin}} \mathcal{F}(\lambda)$$

Minimizing out-of-sample prediction error.

CV = (intuitively) data-efficient way of sample splitting

Cross-Validation (CV)

K -fold Cross-Validation:

1. Split into K subsamples (typically, $K = 5$ or 10) of equal size.
2. For each subsample $k = 1, \dots, K$:
 - ▶ Use subsample k for validation and all others for training.
 - ▶ Calculate (mis)fit $\mathcal{F}_k(\lambda)$ as with sample splitting.
3. Minimize *sum* of (mis)fits:

$$\hat{\lambda}^{\text{cv}} := \underset{\lambda}{\operatorname{argmin}} \sum_{k=1}^K \mathcal{F}_k(\lambda).$$

Estimator: Lasso with $\lambda = \hat{\lambda}^{\text{cv}}$ and all data.

Bickel-Ritov-Tsybakov Rule

Relies on **two conditions**:

- ▶ ε independent of X (\Rightarrow cond'l homoskedasticity).
- ▶ Variance σ^2 of ε is known.

Three steps:

1. Choose $\alpha \in (0, 1)$, typically $\alpha = .05$ probability tolerance, similar to 'sig level'
2. Choose $c > 1$, typically $c = 1.1$
3. Declare

$$\widehat{\lambda}^{\text{BRT}} := \frac{2c\sigma}{\sqrt{n}} \Phi^{-1} \left(1 - \frac{\alpha}{2p} \right) \sqrt{\max_{1 \leq j \leq p} \frac{1}{n} \sum_{i=1}^n X_{ij}^2}$$

with Φ = standard normal CDF.

Belloni-Chen-Chernozhukov-Hansen Rule

- ▶ Allows conditional heteroscedasticity.
- ▶ Requires no preliminary (variance) knowledge.

Three steps:

1. Choose α and c as with BRT.
2. Run pilot Lasso $\hat{\beta}^{\text{pilot}} := \hat{\beta}(\hat{\lambda}^{\text{pilot}})$ with

$$\hat{\lambda}^{\text{pilot}} := \frac{2c}{\sqrt{n}} \Phi^{-1} \left(1 - \frac{\alpha}{2p} \right) \max_{1 \leq j \leq p} \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y})^2 X_{ij}^2}.$$

3. Calculate residuals and declare penalty:

$$\hat{\varepsilon}_i := Y_i - X_i' \hat{\beta}^{\text{pilot}},$$

$$\hat{\lambda}^{\text{BCCH}} := \frac{2c}{\sqrt{n}} \Phi^{-1} \left(1 - \frac{\alpha}{2p} \right) \max_{1 \leq j \leq p} \sqrt{\frac{1}{n} \sum_{i=1}^n \hat{\varepsilon}_i^2 X_{ij}^2}.$$

Choice of Penalty Level Depends on Objective

Sample Splitting and Cross-Validation:

Tends (in simulations) to be

- ▶ Very good at out-of-sample prediction.
- ▶ Very bad at variable selection (too many). aggressive in their variable selection

Bickel-Ritov-Tsybakov and Belloni-Chen-Chernozhukov-Hansen¹

Provably good at

- ▶ Out-of-sample prediction.
- ▶ Coefficient estimation.
- ▶ (Variable selection.)

¹And Chetverikov & Sørensen [2021].

Python Implementation

Scikit-Learn Package

Main Python machine learning package. Includes

- ▶ OLS
- ▶ Lasso
- ▶ Other high-dim. methods to be discussed (e.g. Ridge)

Also include few datasets

- ▶ [Boston house prices](#), Iris plants, Diabetes, Handwritten digits...

Data: Boston House Prices

Contains information on sample of houses in Boston

Sample size is $n = 506$

Target variable: Price.

Basic features (X_{ij} 's) of town where house is located:

- ▶ CRIM - per capita crime rate by town
- ▶ ZN - proportion of residential land zoned for lots over 25000 sq.ft.
- ▶ INDUS - proportion of non-retail business acres per town
- ▶ CHAS - Charles River dummy
- ▶ NOX - nitric oxides concentration
- ▶ RM - average number of rooms per dwelling
- ▶ AGE - proportion of owner-occupied units built prior to 1940
- ▶ DIS - weighted distances to five Boston employment centres
- ▶ RAD - index of accessibility to radial highways
- ▶ TAX - full-value property-tax rate per \$10,000
- ▶ PTRATIO - pupil-teacher ratio by town
- ▶ B - $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks
- ▶ LSTAT - % lower status of the population

Lasso Implementation (Naïve)

```
1 import numpy as np
2 from sklearn import datasets
3 from sklearn.linear_model import Lasso
4 boston = datasets.load_boston()
5 X = boston.data
6 y = boston.target
7 fit = Lasso(alpha = 500).fit(X,y) # alpha=penalty
8 y_pred = fit.predict(X)
9 coef = fit.coef_
10 sel = (coef != 0)
11 XNames = boston.feature_names
12 print(XNames)
13 print(np.round(coef,2))
14 print(XNames[sel])
15 print(np.round(coef[sel],2))
```

Results (Garbage)

Estimates: [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; -0.01; 0; 0; 0]

Selected variables, **emphasized**:

- ▶ CRIM - per capita crime rate by town
- ▶ ZN - proportion of residential land zoned for lots over 25000 sq.ft.
- ▶ INDUS - proportion of non-retail business acres per town
- ▶ CHAS - Charles River dummy
- ▶ NOX - nitric oxides concentration
- ▶ RM - average number of rooms per dwelling
- ▶ AGE - proportion of owner-occupied units built prior to 1940
- ▶ DIS - weighted distances to five Boston employment centres
- ▶ RAD - index of accessibility to radial highways
- ▶ **TAX - full-value property-tax rate per \$10,000**
- ▶ PTRATIO - pupil-teacher ratio by town
- ▶ B - $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks
- ▶ LSTAT - % lower status of the population

Caution: ‘Lasso’ Differs with Software

Python defines Lasso as

$$\hat{\beta}(\lambda) \in \operatorname{argmin}_{b \in \mathbf{R}^p} \left\{ \frac{1}{2n} \sum_{i=1}^n (Y_i - X_i' b)^2 + \lambda \sum_{j=1}^p |b_j| \right\}.$$

Also, constant/intercept included by default,

$$(\hat{\beta}_0(\lambda), \hat{\beta}(\lambda)) \in \operatorname{argmin}_{(b_0, b) \in \mathbf{R}^{1+p}} \left\{ \frac{1}{2n} \sum_{i=1}^n (Y_i - b_0 - X_i' b)^2 + \lambda \sum_{j=1}^p |b_j| \right\}.$$

(Lasso option `fit_intercept = False` removes it.)

Caution: Scaling Matters

$$Y = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon, \quad E[\varepsilon|X] = 0.$$

Suppose we rescale $\tilde{X}_1 = \gamma X_1, \gamma \neq 0$. Then

$$Y = \tilde{\beta}_1 \tilde{X}_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon, \quad \tilde{\beta}_1 := \beta_1 / \gamma.$$

OLS insensitive to rescaling:

- ▶ OLS of Y on X_1, X_2, \dots, X_p gives $\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p$
- ▶ OLS of Y on $\tilde{X}_1, X_2, \dots, X_p$ gives $\hat{\beta}_1 / \gamma, \hat{\beta}_2, \dots, \hat{\beta}_p$

Lasso sensitive to rescaling: Large γ makes $\tilde{\beta}_1$ small.

- ▶ May drive coefficient to zero. (Orthonormal case...)

Normalization

Solution: Bring regressors onto same scale.

For each $j = 1, \dots, p$, define

$$\hat{\mu}_j := \frac{1}{n} \sum_{i=1}^n X_{ij}, \quad \hat{\sigma}_j^2 := \frac{1}{n} \sum_{i=1}^n (X_{ij} - \hat{\mu}_j)^2$$

Standardize RHS variables:

$$\tilde{X}_{ij} := \frac{X_{ij} - \hat{\mu}_j}{\hat{\sigma}_j}.$$

Ensures

$$\frac{1}{n} \sum_{i=1}^n \tilde{X}_{ij} = 0, \quad \frac{1}{n} \sum_{i=1}^n \tilde{X}_{ij}^2 = 1$$

Then Lasso using Y_i and $\tilde{X}_{i1}, \dots, \tilde{X}_{ip}$.

Implementing Lasso with Normalization

```
1 boston = datasets.load_boston()
2 X = boston.data
3 muhat = np.mean(X, axis = 0)
4 stdhat = np.std(X, axis = 0)
5 Xtilde = (X-muhat)/stdhat # standardize
6 y = boston.target
7 fit = Lasso(alpha = 1.3).fit(Xtilde,y)
8 yhat = fit.predict(Xtilde) # Q: Why OK?
9 coef = fit.coef_
10 sel = (coef != 0)
11 XNames = boston.feature_names
12 print(XNames[sel])
13 print(np.round(coef[sel],2)) # in std.dev.s
14 coef_orig = coef / stdhat
15 print(np.round(coef_orig[sel],2)) # original
```

Results (Std. Dev.s)

Estimates: [0; 0; 0; 0; 0; 0; **2.54**; 0; 0; 0; 0; 0; **-1.16**; 0; **-3.49**]

Important variables, emphasized in red:

- ▶ CRIM - per capita crime rate by town
- ▶ ZN - proportion of residential land zoned for lots over 25000 sq.ft.
- ▶ INDUS - proportion of non-retail business acres per town
- ▶ CHAS - Charles River dummy
- ▶ NOX - nitric oxides concentration
- ▶ **RM - average number of rooms per dwelling**
- ▶ AGE - proportion of owner-occupied units built prior to 1940
- ▶ DIS - weighted distances to five Boston employment centres
- ▶ RAD - index of accessibility to radial highways
- ▶ TAX - full-value property-tax rate per \$10000
- ▶ **PTRATIO - pupil-teacher ratio by town**
- ▶ $B - 1000(B_k - 0.63)^2$ where B_k is the proportion of blacks
- ▶ **LSTAT - % lower status of the population**

Choices of Penalty Above

```
1 import numpy as np
2 from sklearn import datasets
3 from scipy.stats import norm
4 boston = datasets.load_boston()
5 X = boston.data
6 y = boston.target
7 sigma = np.std(y)
8 (n,p) = X.shape
9 Xscale = np.max(np.mean((X ** 2),axis = 0)) ** 0.5
10 c = 1.1; alpha = 0.05
11 lamb=c*sigma*norm.ppf(1-alpha/(2*p))
12     \*Xscale/np.sqrt(n)
13 # BRT as stated --^ & upon normalizing --v
14 lamb1=c*sigma*norm.ppf(1-alpha/(2*p))/np.sqrt(n)
15 print(lamb)
16 print(lamb1)
17 # Note: Dividing by 2 (Python Lasso definition)
```

Cross-Validation and Lasso in Matlab

```
1 ... # initiation and standardization
2 from sklearn.linear_model import LassoCV
3 fitCV = LassoCV(cv = 5).fit(Xtilde,y) # K = 5 folds
4 coef = fitCV.coef_
5 sel = (coef != 0)
6 print(XNames[sel])
7 print(np.round(coef[sel],2))
8 print(XNames[~sel])
```

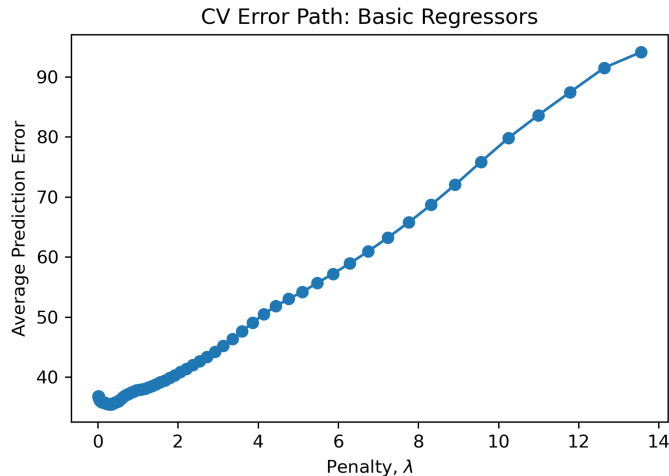
Results

Estimates: [-.47; .5; -.07; .64; -1.31; 2.91; 0; -2.03; .37; -.16; -1.85; .71; -3.72]

Important variables in red:

- ▶ CRIM - per capita crime rate by town
- ▶ ZN - proportion of residential land zoned for lots over 25000 sq.ft.
- ▶ INDUS - proportion of non-retail business acres per town
- ▶ CHAS - Charles River dummy
- ▶ NOX - nitric oxides concentration
- ▶ RM - average number of rooms per dwelling
- ▶ AGE - proportion of owner-occupied units built prior to 1940
- ▶ DIS - weighted distances to five Boston employment centres
- ▶ RAD - index of accessibility to radial highways
- ▶ TAX - full-value property-tax rate per \$10000
- ▶ PTRATIO - pupil-teacher ratio by town
- ▶ B - $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks
- ▶ LSTAT - % lower status of the population

Cross-Validation Error Path



$\Rightarrow \hat{\lambda}^{cv} \approx \text{---}$ Fit $\approx \text{---}$?

Adding Quadratics and Interactions

From matrix \mathbf{X} of (basic) features, add quadratics + interactions

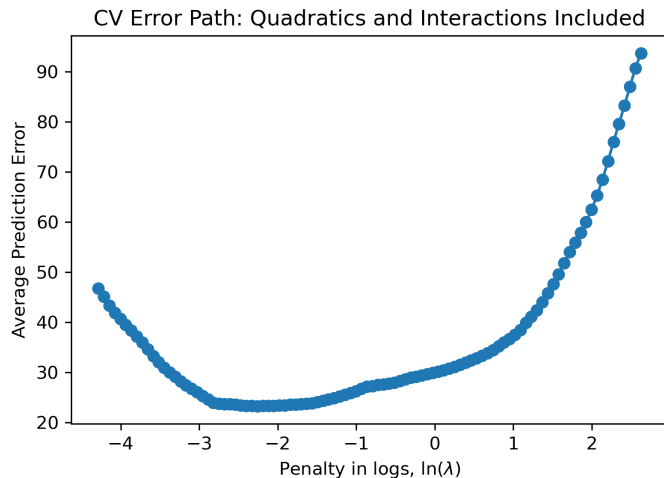
```
from sklearn.preprocessing import PolynomialFeatures
```

```
quad_int = PolynomialFeatures(degree=2,include_bias=False)
```

```
Xnew = quad_int.fit_transform(X)
```

#Regressors now 104 (instead of 13), so $p/n \approx 1/5$.

Cross-Validation Error Path: Technical Regressors



$\Rightarrow \hat{\lambda}^{\text{cv}} > 0$. Fit \approx ___?