Linear Model in High Dimensions, I: Introduction and Implementation

Jesper Riis–Vestergaard Sørensen

University of Copenhagen, Department of Economics

October 6, 2022

Overview

Introduction

Linear Mean Regression Model

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

Y: Dependent variable (scalar)

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

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

ε: Noise/unobservables

p: Number of covariates

Classical Approach to Modeling and Sampling

Independent sampling n times in accordance with

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

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

From the (infinite) array

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

we have access to the 'first' n observations.

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

Classical Approach to Estimation and Inference

Least squares (LS) estimator:

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

LLN + CLT $\Rightarrow$ asymptotic distribution:

$$\sqrt{n}(\widehat{\beta}^{\mathrm{LS}} - \beta) \xrightarrow{d} \mathrm{N}\left(\mathbf{0}_{p \times 1}, \mathbf{V}\right) \text{ as } n \to \infty,$$

where

$$\begin{aligned} \mathbf{V} &:= \mathbf{A}^{-1}\mathbf{B}\mathbf{A}^{-1}, \\ \mathbf{A} &:= E\left[XX'\right], \\ \mathbf{B} &:= E\left[\varepsilon^2 XX'\right]. \end{aligned}$$

think the CDF F_(X)

(Review Q: What does $\to_d$ mean?)

Asymptotics yield Approximations

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

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

Allows us to gauge uncertainty. For example,

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

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

CIs valid in asymptotic sense,

whatever the statistical significance level I evaluate by jacking up the sample size

$$\mathrm{P}\left(\beta_j \in \mathrm{CI}_j(.95)\right) \to .95 \text{ as } n \to \infty.$$

Problem in High Dimensions

ndim for \beta

Implicit in 'as $n \to \infty$': p is fixed, so $p/n \to 0$.

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

in modern datasets, not always true

With p/n nonnegligible? May be poor.

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

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

X: 62 country-level variables.

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

Only 200ish countries...

Only 90 complete observations

$p/n \approx$ 2/3 '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

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

$n \approx 300,000$ 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):

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

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

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

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

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

Approximations should take 'large p' into account.

Sampling in High Dimensions

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

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

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

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* $(\widehat{Y}_i)$ *outcome* (Y_i).

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

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

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

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

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 $n \to \infty$.

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?

$$E\left[\sum_{j=1}^{p}\left(\widehat{\beta}_j^{\mathrm{LS}} - \beta_j\right)^2\right] = \ldots$$

- Problem? [Whiteboard]

Sparsity

Key Condition: Sparsity

Rescue comes from believing only few β_j's nonzero.

Sparsity means

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

Lasso (below) useful when

$$s/n \to 0.$$

Outperforms LS when

$$p/s \to \infty.$$

(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. $\beta = (1, 3, 0, \cdots, 0, 2, 0, \cdots, 0)'$

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

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

$$\beta = (1, \tfrac{1}{3}, \tfrac{1}{9}, \tfrac{1}{27} \ldots, \tfrac{1}{3^{p-1}})'.$$

Relevance?

What Determines Economic Growth? (ctnd)

Cross-country growth regression

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

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? 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

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

again, minizing the sq. errors

We don't know s. But could consider

R = tolerance for the number of non-zeros

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

for various $R \geqslant 0$.

Fit/Complexity Trade-Off

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

$R \geqslant 0$: 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).
- $R \geqslant p$ recovers (unconstrained) LS.

Best Subset Selection Estimator

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

Best subset selection (of size R) estimator

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

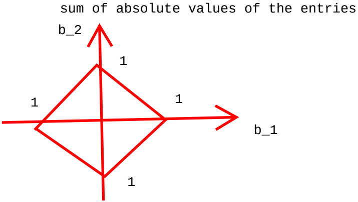
Deal-breaker when $p > 40$ or so.

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

Convex Relaxation of Constraint

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

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



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"

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

May equivalently solve penalized version

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

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

Acronym: Least Absolute Shrinkage and Selection Operator.

Shrinkage: Orthonormal Design, I

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

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

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

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

- Shrinkage towards origin apparent.
- Lasso = here soft thresholding of LS.
- Analytic solution unavailable ⇒ numerical optimization.

Shrinkage: Orthonormal Design, II

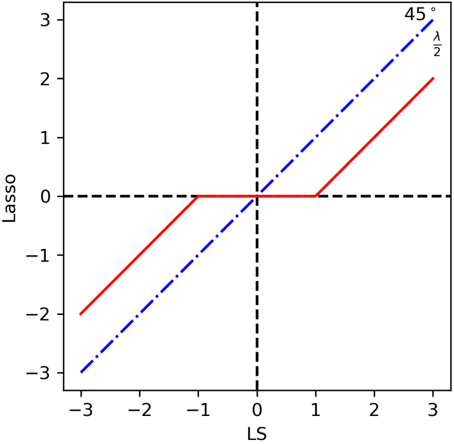


Figure: Lasso vs. Least Squares ($\lambda = 2$)

Selection

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

Since $|\cdot|$ has kink at zero, $\widehat{\beta}_j(\lambda)$'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 $\widehat{\beta}_j(\lambda) \neq 0$.
- Variable j not selected if $\widehat{\beta}_j(\lambda) = 0$.

Penalty Selection

How do we Choose λ?

which summarizes the noise in the problem. We would like to choose the smaller penalty level so that

$$\lambda \geqslant cn\|S\|_\infty \text{ with probability at least } 1-\alpha, \qquad (16)$$

where $1-\alpha$ needs to be close to one, and c is a constant such that $c > 1$. Following [7] and [8], respectively, we consider two choices of λ that achieve the above:

$$X\text{-independent penalty:} \quad \lambda := 2c\sigma\sqrt{n}\Phi^{-1}(1-\alpha/2p), \qquad (17)$$

$$X\text{-dependent penalty:} \quad \lambda := 2c\sigma\Lambda(1-\alpha|X), \qquad (18)$$

where $\alpha \in (0,1)$ and $c > 1$ is constant, and

From: Belloni and Chernozukov (2011) - High Dimensional Sparse Econometric Models, An Introduction

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

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

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

Sample Splitting, II

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

Check (mis)fit using validation sample:

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

Choose λ to obtain best possible fit:

$$\widehat{\lambda}^{\mathsf{SS}} := \operatorname*{argmin}_{\lambda} \mathcal{F}(\lambda)$$

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 $k = 1, \ldots, K$:
   - 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:

$$\widehat{\lambda}^{\mathsf{CV}} := \underset{\lambda}{\operatorname{argmin}} \sum_{k=1}^{K} \mathcal{F}_k(\lambda).$$

Estimator: Lasso with $\lambda = \widehat{\lambda}^{\mathsf{CV}}$ 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

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

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 $\widehat{\beta}^{\text{pilot}} := \widehat{\beta}(\widehat{\lambda}^{\text{pilot}})$ with

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

3. Calculate residuals and declare penalty:

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

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

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). `aggresive in their variable selection`

Bickel-Ritov-Tsybakov and Belloni-Chen-Chernozhukov-Hansen[1]

Provably good at

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

[1] And 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)

```
import numpy as np
from sklearn import datasets
from sklearn.linear_model import Lasso
boston = datasets.load_boston()
X = boston.data
y = boston.target
fit = Lasso(alpha = 500).fit(X,y) # alpha=penalty
y_pred = fit.predict(X)
coef = fit.coef_
sel = (coef != 0)
XNames = boston.feature_names
print(XNames)
print(np.round(coef,2))
print(XNames[sel])
print(np.round(coef[sel],2))
```

Results (Garbage)

Estimates: [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

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

Also, constant/intercept included by default,

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

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

Just as in ridge regression, we can re-parametrize the constant β_0 by standardizing the predictors; the solution for $\hat{\beta}_0$ is $\bar{y}$, and thereafter we fit a model without an intercept (Exercise 3.5). In the signal processing literature, the lasso is also known as *basis pursuit* (Chen et al., 1998).

Caution: Scaling Matters

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

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

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

OLS insensitive to rescaling:

- OLS of Y on $X_1, X_2, \ldots, X_p$ gives $\widehat{\beta}_1, \widehat{\beta}_2, \ldots, \widehat{\beta}_p$
- OLS of Y on $\widetilde{X}_1, X_2, \ldots, X_p$ gives $\widehat{\beta}_1/\gamma, \widehat{\beta}_2, \ldots, \widehat{\beta}_p$

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

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

Normalization

Solution: Bring regressors onto same scale.

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

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

Standardize RHS variables:

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

Ensures

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

Then Lasso using Y_i and $\widetilde{X}_{i1}, \ldots, \widetilde{X}_{ip}$.

Implementing Lasso with Normalization

```
boston = datasets.load_boston()
X = boston.data
muhat = np.mean(X, axis = 0)
stdhat = np.std(X, axis = 0)
Xtilde = (X-muhat)/stdhat # standardize
y = boston.target
fit = Lasso(alpha = 1.3).fit(Xtilde,y)
yhat = fit.predict(Xtilde) # Q: Why OK?
coef = fit.coef_
sel = (coef != 0)
XNames = boston.feature_names
print(XNames[sel])
print(np.round(coef[sel],2)) # in std.dev.s
coef_orig = coef / stdhat
print(np.round(coef_orig[sel],2)) # original
```

Results (Std. Dev.s)

Estimates: [0; 0; 0; 0; 0; 2.54; 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

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

Cross-Validation and Lasso in Matlab

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

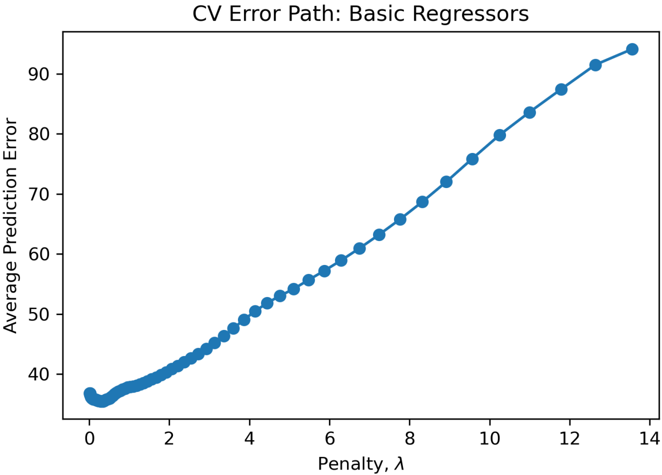
Results

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



$$\Longrightarrow \widehat{\lambda}^{\texttt{CV}} \approx \_\_\_ \text{ Fit} \approx \_\_\_?$$

Adding Quadratics and Interactions

From matrix **X** of (basic) features, add quadratics + interactions

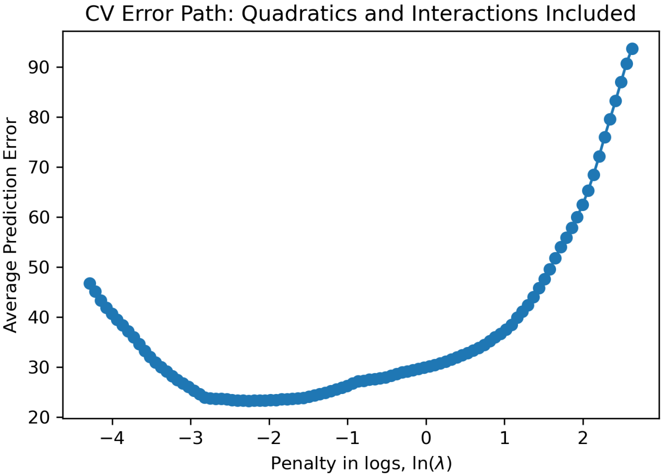
```
from sklearn.preprocessing import PolynomialFeatures

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



$\Longrightarrow\ \widehat{\lambda}^{\texttt{CV}} > 0$. Fit $\approx$____?