Ridge, LASSO, and Learning to Predict

Gov 51 Section — Week 10

George
Harvard University

April 1, 2026

Why Does OLS Fail at Prediction?

OLS is built for explanation — prediction is a different criterion

Explanation

Goal: what causes Y?
Coefficients: causal meaning
Success: unbiased, low SE

OLS: excellent

Prediction

Goal: minimize test RMSE
Coefficients: just weights
Success: low out-of-sample error

OLS: can fail

Today: we add more features than OLS can handle, watch it fail, then fix it

$\text{Bias}^2 + \text{Variance} + \text{Noise}$ — **regularization attacks variance**

$$\mathbb{E}[(\hat{Y} - Y)^2] = \underbrace{\text{Bias}^2}_{\text{wrong assumptions}} + \underbrace{\text{Variance}}_{\text{sensitivity to sample}} + \underbrace{\sigma^2}_{\text{irreducible noise}}$$

High variance (overfitting)

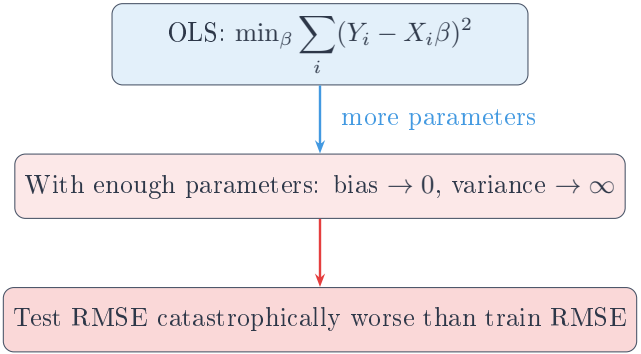
Too many parameters
Noise memorized, not signal
Test RMSE $\gg$ train RMSE

High bias (underfitting)

Too few parameters
Real patterns missed
High error in- *and* out-of-sample

Regularization trades a little bias for a large variance reduction

OLS minimizes in-sample error — that is not the same goal



We will demonstrate this deliberately on real data

Seeing the Tradeoff: A Controlled Experiment

Boston housing is our controlled lab for the bias-variance tradeoff

The dataset

506 Boston neighborhoods
Outcome: `medv` (median home value)
Predictors: 13 (crime, jobs, taxes, schools. . .)

80/20 train/test, `set.seed(51)`

The experiment

Step 1: OLS, 13 main effects

Step 2: OLS, all 3-way interactions (13 predictors $\Rightarrow$ 377 features)

Step 3: Ridge, LASSO, EN on the 377-feature matrix

Metric: test RMSE throughout

We deliberately blow up the feature space

```
# All three-way interactions: 13 vars taken 3 at a time
f_kitchen <- medv ~ (crim + zn + indus + chas + nox + rm
    +
                      age + dis + rad + tax + ptratio +
                      black + lstat)^3

X_train <- model.matrix(f_kitchen, data = train)[, -1]
```

377	**404**	**1.07**
features	training observations	ratio n/p

When $n/p \approx 1$, OLS is nearly rank-deficient
— it will fit training data almost perfectly

Kitchen sink OLS: bias crushed, variance explodes

Model	Train RMSE	Test RMSE
OLS (13 main effects)	3.44	4.11
OLS (377 features)	1.07	123.37

Train RMSE = 1.07

Bias ≈ 0

404 observations: near-perfect fit

Not learning — memorizing

Test RMSE = 123.37

Variance exploded

102 held-out obs: catastrophic failure

The gap *is* the variance

Ridge Regression (L2 Penalty)

Ridge shrinks all coefficients toward zero — none set to zero exactly

OLS:

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2$$

Ridge:

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Key properties

$\lambda > 0$**:** controls penalty strength
$\lambda \uparrow$**:** more shrinkage toward zero
Never: coefficients set to zero exactly

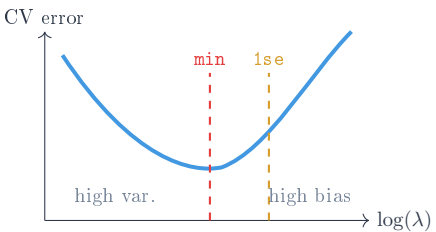
In R: `alpha = 0`

Cross-validation picks the right λ

```
cv_ridge <- cv.glmnet(X_train, y_train,
                      alpha  = 0,     # alpha=0 is Ridge
                      nfolds = 10)
plot(cv_ridge)
```

Reading the CV plot

x-axis: $\log(\lambda)$, *decreasing* left to right
top axis: number of non-zero coefficients
left dashed: `lambda.min` (lowest CV error)
right dashed: `lambda.1se` (simplest within 1 SE)



Ridge: test RMSE drops from 123 to ∼5

```
ridge_pred <- predict(cv_ridge, s = "lambda.min", newx =
    X_test)
ridge_rmse <- rmse(y_test, ridge_pred)
```

Model	Test RMSE	Vars kept
OLS (377 features)	123.37	377
Ridge	≈ 5.0	377

Ridge shrinks all 377 coefficients but drops none
— variance controlled, but no variable selection

LASSO
(L1 Penalty)

LASSO swaps β_j^2 for $|\beta_j|$ — and that changes everything

Ridge (L2 penalty):

$$\text{RSS} + \lambda \sum_j \beta_j^2$$

Shrinks all coefficients
None set exactly to zero

LASSO (L1 penalty):

$$\text{RSS} + \lambda \sum_j |\beta_j|$$

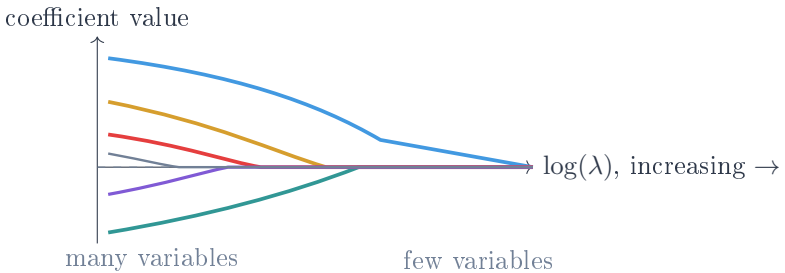
Shrinks *and* selects
Weak coefficients zeroed out exactly

L1 penalty creates corners in the constraint set — at a corner, $\hat{\beta}_j = 0$ exactly

LASSO does variable selection automatically, Ridge does not

The coefficient path shows variables dropping one by one

As λ increases, more variables hit exactly zero



Weakest predictors hit zero first — LASSO ranks variables by predictive importance

LASSO: fewer variables, better prediction than OLS

```
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1,
   nfolds = 10)
lasso_pred <- predict(cv_lasso, s = "lambda.min", newx =
    X_test)
```

Model	Test RMSE	Vars kept
OLS (13 main effects)	4.11	13
OLS (377 features)	123.37	377
Ridge	≈ 5.0	377
LASSO	≈ 3.97	$\sim$20–40

LASSO uses a fraction of the variables and beats simple OLS — selection *and* shrinkage working together

Including `black` optimizes prediction — not fairness

`black`: proportion of Black residents (Harrison & Rubinfeld, 1978)

If LASSO keeps it:

Correlation: race predicts home values
Cause: historical discrimination, not intrinsic property
Algorithm: no knowledge of either

What LASSO cannot do:

Optimize: prediction
Not optimize: fairness
Gap: who decides? who pays the cost?

PS3 Q6.5 asks the same question — with recidivism instead of rent

Boston

Variable: `black` (race proxy)
Outcome: home value
Survivors: whatever predicts price
Who pays? neighborhoods

COMPAS (PS3)

Variable: race, prior arrests...
Outcome: recidivism prediction
Survivors: whatever predicts reoffending
Who pays? defendants

Thursday: Breiman's Two Cultures — when prediction and fairness collide

Elastic Net:
Blending Both Penalties

Elastic Net blends Ridge and LASSO — α is the mixing dial

$$\min_{\beta} \sum_{i=1}^{n}(Y_i - X_i\beta)^2 \;+\; \lambda\Big[\alpha\sum_{j}|\beta_j| \;+\; (1-\alpha)\sum_{j}\beta_j^2\Big]$$

α	**Method**	**When to prefer**
0	Ridge	All predictors contribute, high correlation
0.5	Elastic Net	Correlated predictors, want partial selection
1	LASSO	Sparse signal, automatic variable selection

One line changes α — Ridge, Elastic Net, and LASSO are the same call

```
cv_ridge <- cv.glmnet(X_train, y_train, alpha = 0,
   nfolds = 10)
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1,
   nfolds = 10)
cv_enet  <- cv.glmnet(X_train, y_train, alpha = 0.5,
   nfolds = 10)
```

Only `alpha` changes — everything else is identical

Putting It All Together

All four models: the bias-variance tradeoff in one table

Model	Predictors	Test RMSE	Regime
OLS (main effects)	13	4.11	Low variance, some bias
OLS (kitchen sink)	377	123.37	Variance explodes
Ridge	377	≈ 5.0	Variance controlled
LASSO	$\sim$30	≈ 3.97	Selection + shrinkage
Elastic Net	$\sim$30	≈ 4.04	Blend

More features $\neq$ better

377 features: 30$\times$ worse than 13

High p, low $n/p \Rightarrow$ variance explodes

Regularization works

LASSO and EN: beat OLS on test RMSE

Fewer variables, better generalization

What This Means for PS3

Same workflow, different data — you do the thinking

Today (Boston)

Outcome: `medv`
Features: 13 → 377
OLS kitchen sink: 30× overfit
LASSO: beats OLS on test RMSE
`lambda.min`: > 1, $\log(\lambda) > 0$

PS3 (COMPAS)

Outcome: recidivism
Features: 141 after interactions
Same story: OLS overfits
LASSO: selects ∼10–30 vars
`lambda.min`: < 1, $\log(\lambda) < 0$ — normal

Use `s = "lambda.min"` throughout unless a question says otherwise

Three things that will trip you up on PS3

1. **Negative $\log(\lambda)$: normal**
   - 141 features $\Rightarrow$ optimal penalty < 1
   - CV plot: same U-shape, x-axis shifted left
2. **Column alignment after `model.matrix()`**
   - Train and test must have identical columns
   - Fix: `intersect(colnames(X_train), colnames(X_test))`
3. **Counting non-zero coefficients**
   - `coef()` returns a sparse matrix
   - Always use `[-1]` to drop the intercept first

The bias-variance tradeoff is not a theorem to memorize. It is the reason we don't just throw everything into OLS.