ML extensions to OLS

Oxford Spring School in Advanced Research Methods, 2022

Dr Thomas Robinson, Durham University

Day 2/5

Introduction

Yesterday we explored how a familiar estimator (logistic regression) incorporates some fundamental aspects of ML

- ML is not some entirely new, alien type of doing statistics
- ML is typically focused on prediction problems
- Lots of really useful ML models are extensions of regression framework

So how should we understand OLS within a prediction context?

How do other popular forms of ML come out of it?
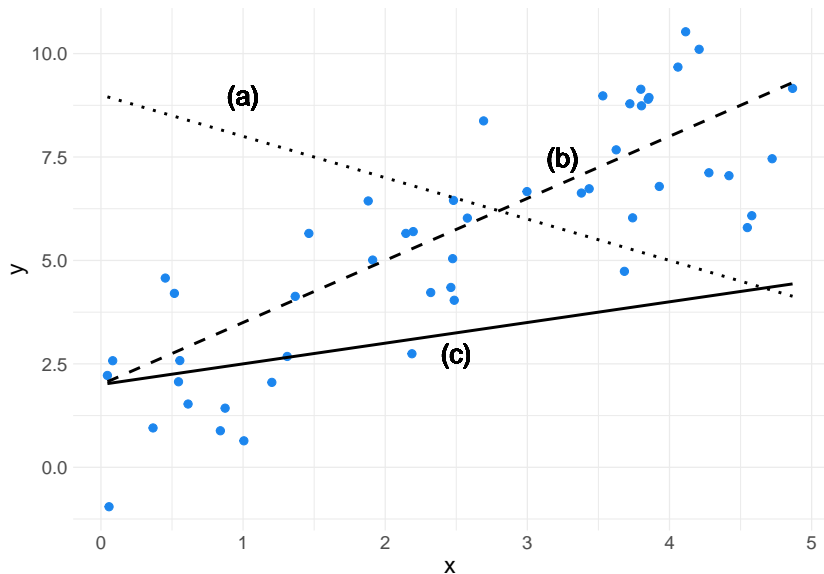
Today's session

1. Recap OLS from prediction perspective

   ▶ How does OLS work?
   ▶ Optimisation criteria
   ▶ Bias-variance trade-off

2. LASSO estimator

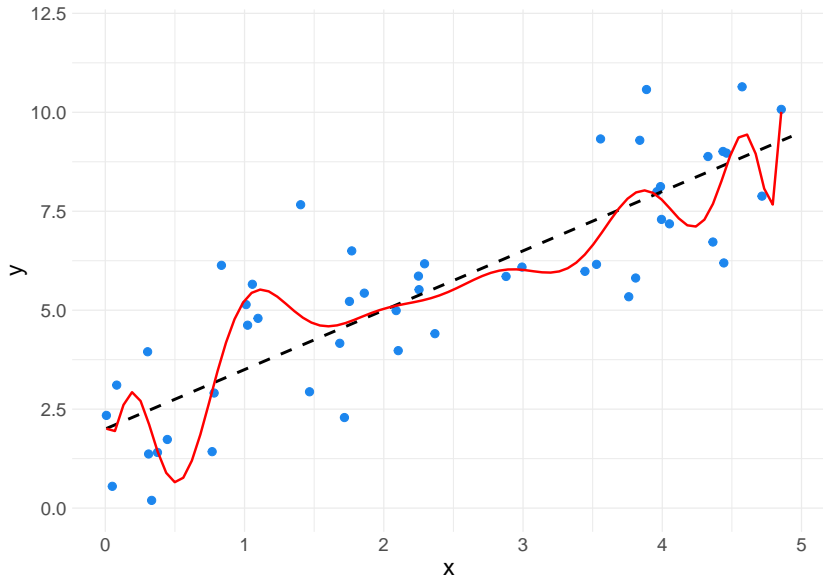3. Hyperparameter tuning

4. Practical application

Key topics:

▶ Bias and variance
▶ Regularisation
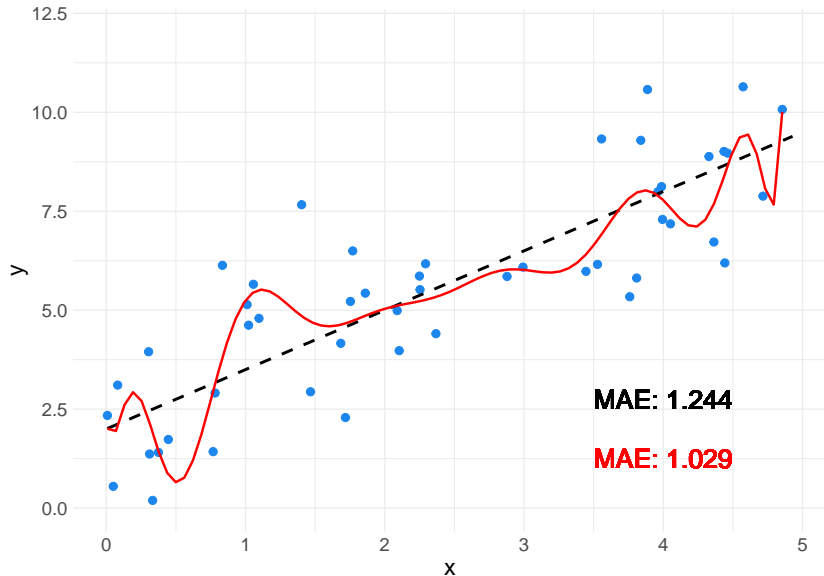▶ $K-$fold cross validation

Ordinary Least Squares Regression

Refresher: which is the best line?

Refresher: which is the best line? 2

Refresher: which is the best line? 2

MAE: 1.244

MAE: 1.029

OLS as a tool for inference

In inference terms, OLS estimates $\hat{\beta}$ that:

- ▶ Captures the linear relationship between $\mathbf{X}$ and $\mathbf{y}$

- ▶ Yields individual estimates of the "effects" of X on y

- ▶ Allows us to understand the uncertainty over $\hat{\beta}$

    - ▶ E.g., how confident are we that there is +/- effect of x_1 on y

OLS: Optimisation

OLS regression minimises the **sum of the squared error** between the regression line ($X\beta$) and the observed outcome (y):

$$\arg\min_{\beta} \sum_{i=1}^{N} (y_i - x_i\beta)^2,$$

where $x_i\beta$ is the linear regression function.

How might we solve this?

▶ Calculus – there is a closed form solution (unlike logistic regression)
▶ Maximum likelihood estimation

Why do we like OLS?

Not only does OLS have a closed form solution, we also know that, under the Gauss Markov (GM) assumptions, OLS is:

- ▶ **B**est
- ▶ ~~Linear~~ (Hansen 2021)
- ▶ **U**nbiased
- ▶ **E**stimator

GM Assumptions

In other words, in terms of estimating the parameters β, you won't find a linear model that is unbiased with a lower variance

OLS is fantastic for inference:

- ▶ Typically concerned with generating **unbiased** estimate of $\hat{\beta}$
- ▶ So we can perform valid significance testing

Bias

Bias is a feature of the estimator:

- $\text{Bias}_\beta = (\mathbb{E}[\hat{\boldsymbol{\beta}}] - \boldsymbol{\beta})$

- On average, the estimated parameters are equal to the true parameters

- Under GM, we know that $(\mathbb{E}[\hat{\boldsymbol{\beta}}] - \boldsymbol{\beta}) = 0$

Variance

As we sample new (GM-satisfying) data, the parameters of our model will shift:

- Hence, there will be variance over our parameter estimates
- $\mathbb{V}_{\hat{\boldsymbol{\beta}}} = \mathbb{E}\big[(\mathbb{E}[\hat{\boldsymbol{\beta}}] - \hat{\boldsymbol{\beta}})^2\big]$
- The average distance between a particular parameter estimate and the mean of parameter estimates over multiple samples

Visualising bias and variance
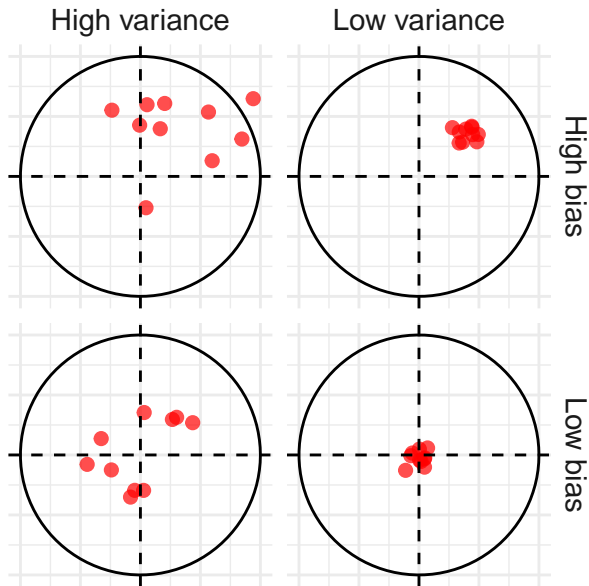
Predicting *new* values

In the remainder of today's session, we are going to consider the following generic supervised learning problem:

- We observe $(\boldsymbol{y}, \boldsymbol{X}) \in \mathcal{D}$
  - A training sample that is taken from a wider possible set of data
  - I.e. we can think, counterfactually, of resampling to get a new sample $(\boldsymbol{y}_{\text{New}}, \boldsymbol{X}_{\text{New}})$
- We also observe a "test" dataset $\boldsymbol{X}'$

The goal is to estimate $\boldsymbol{y}'$ by training a model $\hat{f}$

- The outcomes that correspond to $\boldsymbol{X}'$

OLS as a tool for prediction

When we run OLS, we also get a "trained model":

- $\hat{f}$ – that has parameters equal to $\hat{\beta}$
- Can be applied to a new "test" dataset X'
- To generate new predictions y'

Bias and variance of predictions

We can also think of bias in terms of the predictions:

- $\text{Bias}_{\boldsymbol{y}} = (\mathbb{E}[\hat{\boldsymbol{y}}] - \boldsymbol{y})$

- We ideally want low bias

- High bias suggests the model is not sensitive enough

And we can think about the variance of the prediction:

- $\mathbb{V}_{\hat{\boldsymbol{y}}} = \mathbb{E}\big[(\mathbb{E}[\hat{\boldsymbol{y}}] - \hat{\boldsymbol{y}})^2\big]$

High variance means that the model is very sensitive to $\boldsymbol{X}$ – the training data – but will perform poorly on new samples of data

- With the new data, and high variance, we would expect quite different predictions

Bias-variance trade off

So can't we just choose a low-variance, low-bias modeling strategy? Not quite!

Assume we calculate the mean squared error of some new data $\boldsymbol{X'}$ given a trained model $\hat{f}$:

$$\mathsf{MSE} = \mathbb{E}[(\hat{f}(\boldsymbol{X'}) - \boldsymbol{y})^2].$$
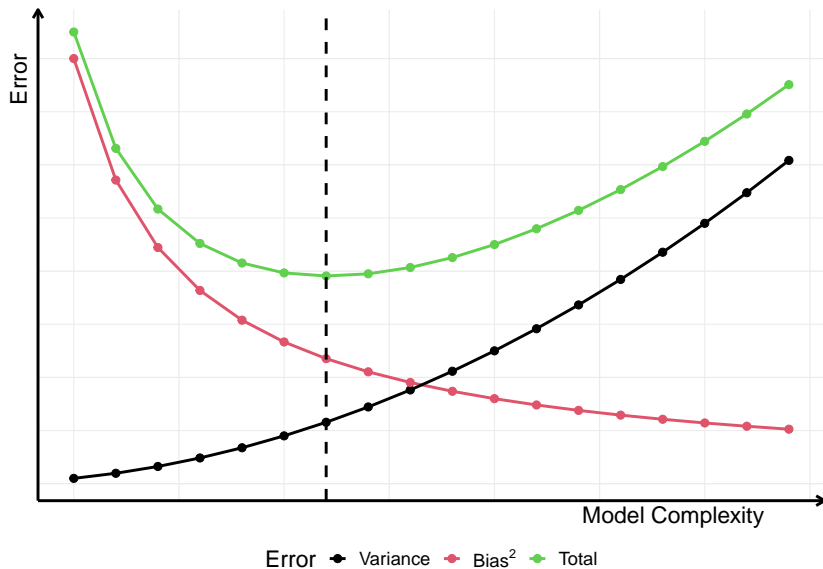
We can decompose this further:

$$MSE = \underbrace{\mathbb{E}[(\hat{f}(\boldsymbol{X'}) - \mathbb{E}[\hat{\boldsymbol{y}}])^2]}_{\text{Variance}} + \underbrace{(\mathbb{E}[\hat{\boldsymbol{y}}] - \boldsymbol{y})^2}_{\text{Bias}^2}$$

So holding the MSE fixed, if we reduce the variance we must increase the bias

▶ I.e. there is a **bias-variance trade-off**

Visualising the trade-off

Error   •— Variance   •— $Bias^2$ •— Total

Out of sample performance of OLS

This trade-off explains why we might not want to use OLS for prediction tasks:

- ▶ By virtue of GM assumptions and BLUE, MSE is explained entirely by variance

  - ▶ Averaging across models, the model parameters are centred on the true values (an unbiased estimator)

- ▶ So we cannot tweak the model to get slightly better out-of-sample predictions at the expense of some added bias

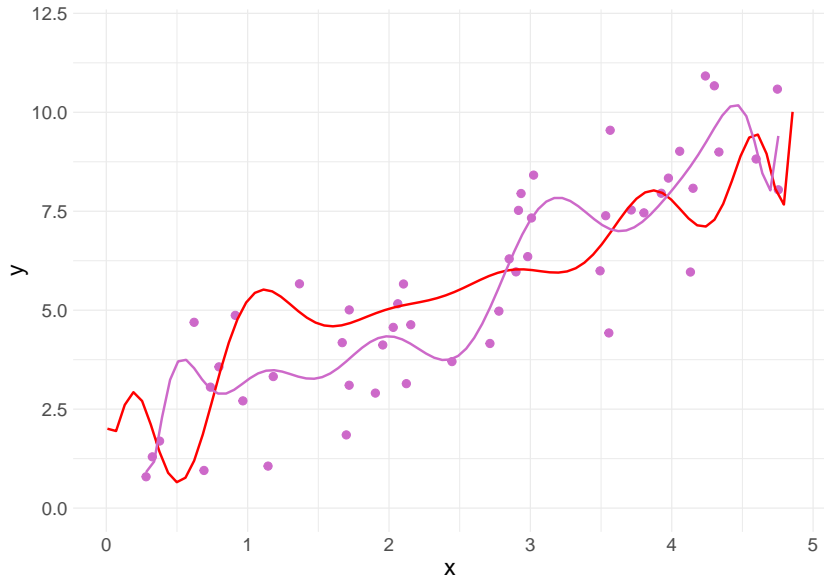 - ▶ In other words, we cannot leverage the bias-variance trade-off

OLS: Complex OLS model trained on $\mathbf{X}$
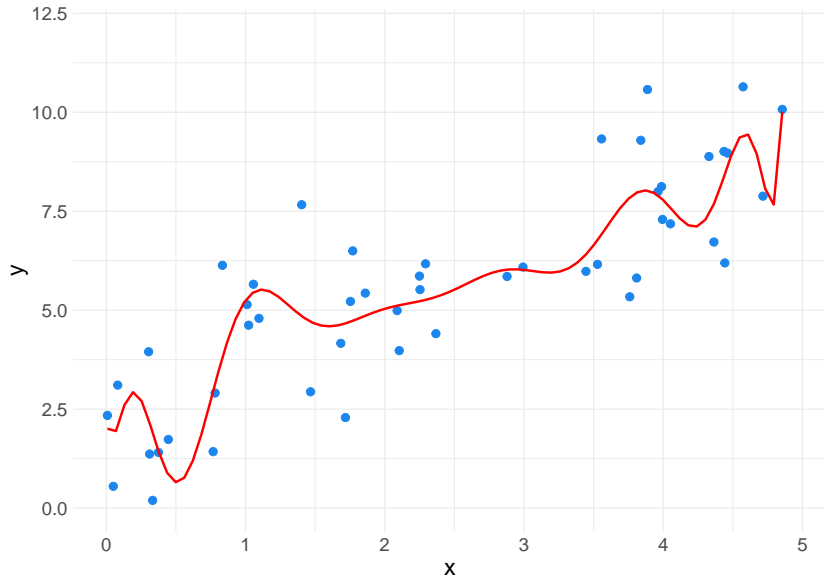
OLS: Compare models' predictions to $\mathbf{X}'$
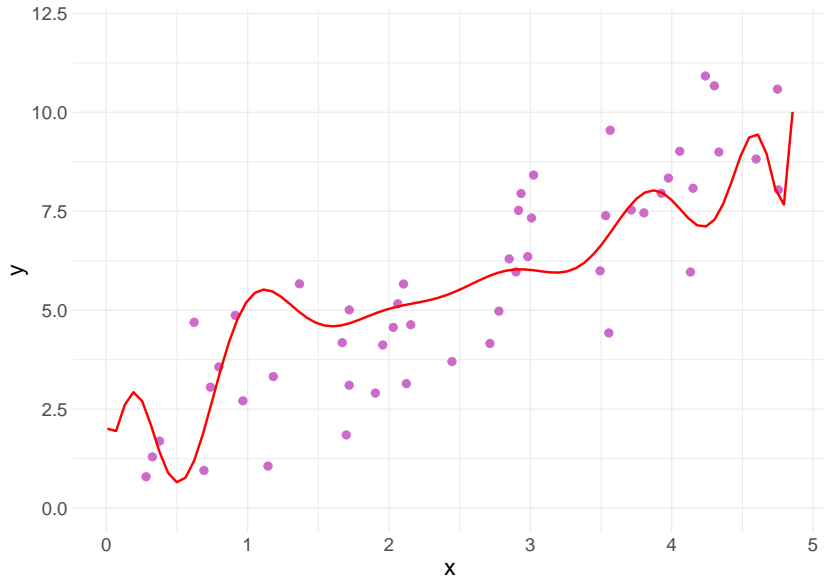
OLS: Variance in model predictions over $\mathbf{X}$

The LASSO estimator

Regularisation and overfitting

In the previous examples, an overly complicated model yields poor out-of-sample predictions. To ensure our model does not have too much variance, we can *penalise* overly-complicated models that will perform poorly on new test data (X')

- ▶ This may introduce bias into the model

- ▶ But, if done correctly, we can reduce the total MSE by offsetting overly-high variance

- ▶ And therefore yield better predictions on X'

This is a generalisable feature of ML:

- ▶ Regularisation constrains model complexity to prevent overfitting

- ▶ Especially important for the models we consider in the remainder of the week that are very powerful

Regularisation of OLS

If we want to continue using a linear predictor we need to modify the loss function L:

- L_{OLS} is known to be unbiased

- So adding a term that is non-zero to L will *add* bias...

- ... and hopefully improve out-of-sample prediction

In other words:

- We sacrifice some variance in order to improve the predictive performance of the model on X'

Generalising OLS with regularization

We can state this problem using the following general linear optimisation problem:

$$\arg \min_f \underbrace{\sum_{i=1}^{N}(y_i - f(\boldsymbol{x_i}))^2}_{\text{Sum of squared error}} + \underbrace{\lambda R(f)}_{\text{Regularisation}}$$

For OLS:

▶ $f \in \mathcal{F}_{\text{linear}}$
▶ $\lambda = \frac{1}{\infty} = 0$

But what if $\lambda \neq 0$?

▶ Then we must decide what $R(\cdot)$ is
▶ And decide on a value of λ – a hyperparameter

$R(\cdot)$ as shrinkage

Consider an OLS model with k parameters:

► The model estimates coefficients for each parameter
► Regardless of how large or small that coefficient is
► In a sense, with non-zero estimates for each parameter these models can be considered "complex"

We can reduce the complexity of the regression model by setting some parameters to zero

► I.e. we **shrink** the coefficient estimates

► Aim to reduce the variance error by more than the increase in bias error

In the linear framework, we need some way to penalize non-zero coefficients

Least absolute shrinkage and selection operator (LASSO)

We can calculate the total magnitude (or **L1 Norm**) of the coefficients in a model as:

$$||\boldsymbol{\beta}||_1 = \sum_j |\beta_j|$$

Next, we can think about restricting the size of this norm:

$$||\boldsymbol{\beta}||_1 \leq t$$

And finally we want to include this in our loss function:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} (y_i - \boldsymbol{x_i}\boldsymbol{\beta})^2 \text{ subject to } ||\boldsymbol{\beta}||_1 \leq t$$

This final optimisation constraint is equivalent to:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} (y_i - \boldsymbol{x_i}\boldsymbol{\beta})^2 + \lambda ||\boldsymbol{\beta}||_1.$$
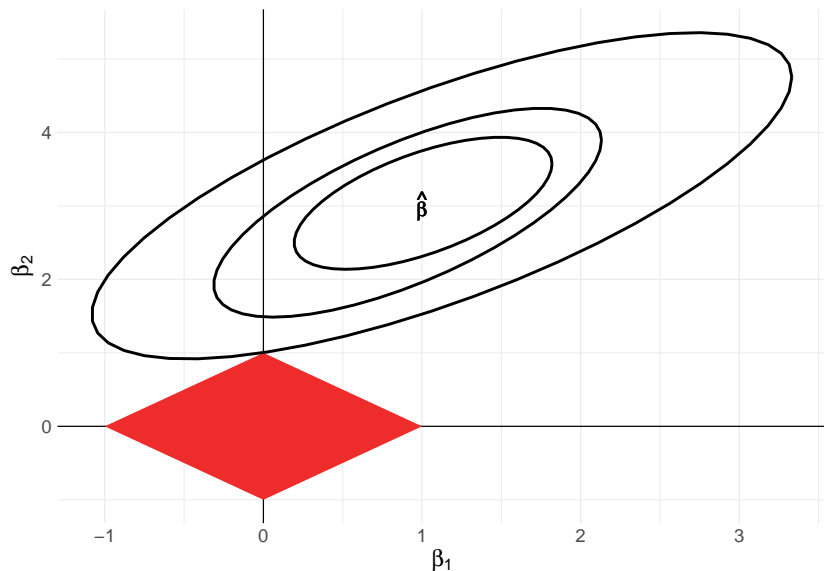
LASSO

Hence, the LASSO estimator conforms to the generalised loss function introduced earlier

▶ $R(f) = ||\hat{\boldsymbol{\beta}}||_1$

LASSO often yields coefficient estimates of exactly zero:

▶ Think about varying the true value of some coefficient β_j:

  ▶ When β_j is large, we might shrink it (relative to OLS) but the importance of this predictor is sufficient to entail a non-zero coefficient

  ▶ But for some small enough value b, the cost of including b in L1-norm is greater than the reduction in squared error

▶ In other words, the L1 norm constraint can lead to "corner solutions"

Example of LASSO corner solution ($||\boldsymbol{\beta}||_1 \leq 1$)

Comparison of $\hat{\beta}_j^{\text{OLS}}$ to $\hat{\beta}_j^{\text{LASSO}}$
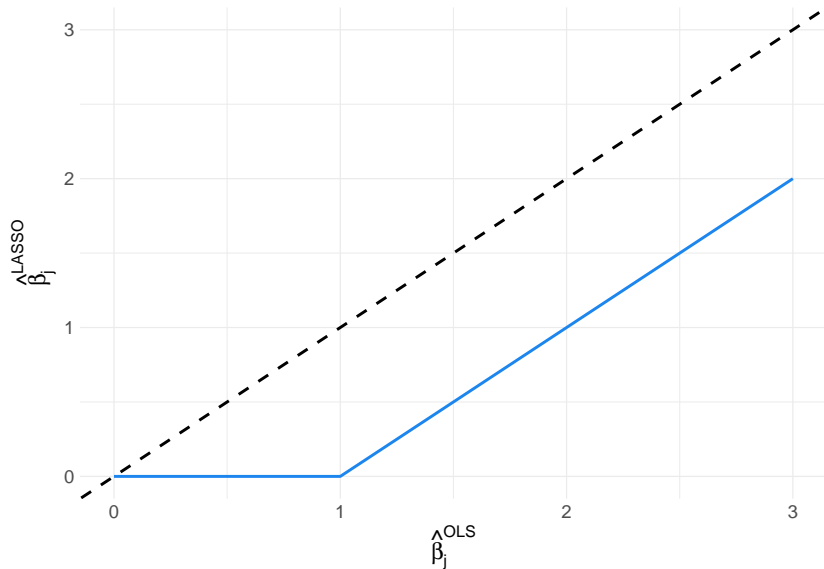
Two helpful properties of LASSO

1. Prediction accuracy
   - ▶ We trade off an amount of bias for a (hopefully) greater reduction in variance, improving out-of-sample prediction
   - ▶ Cf. a non-zero *true* coefficient estimated with a large confidence interval

2. Selection of relevant variables
   - ▶ The possibility of corner-solutions acts as a useful variable selection mechanism
   - ▶ LASSO essentially selects the most important variables for us

Hyperparameter tuning

Choosing λ

The final part of the estimation problem is setting λ. Recall that:

$$\mathcal{L} = \sum_{i=1}^{N} \left(y_i - f(x_i) \right)^2 + \lambda ||\hat{\boldsymbol{\beta}}||_1$$

λ regulates how much bias we add to the model:

- ▶ Too large a value = overly constricted model, large MSE
- ▶ Too small a value = overly complex model, large MSE

We need to find a value that helps us get near the bottom of the total-error curve!

- ▶ This process is called hyperparameter tuning
- ▶ It is a recurrent feature of ML methods

Simple tuning

Simplest way is to simply try a few values:

- ▶ In the case of LASSO, we might try $\lambda = \{0.1, 1, 10\}$
- ▶ Choose λ that yields lowest MSE from $MSE_{\lambda=0.1}, MSE_{\lambda=1}, MSE_{\lambda=10}$
- ▶ Use this value in the final model

But there are some limitations:

- ▶ You are "testing" your model on the same data that it was trained upon
- ▶ So this will inflate the actual accuracy of your model
- ▶ Goes against the train-test ethos of ML prediction

Holdout sample

As an alternative, we could create a holdout sample:

- ▶ Split our training data X into $\{X^{\mathsf{Train}}, X^{\mathsf{Holdout}}\}$
- ▶ Train our model for each value of λ on X^{Train}
- ▶ Then test the predictive accuracy on X^{Holdout}
- ▶ In other words, create a miniature version of the train-test split within our training data

But again, there are limitations:

- ▶ By leaving out some observations, we lose predictive power
- ▶ Even if observations are randomised across the two datasets, the model can never learn from the fixed holdout data

K-fold cross validation

We can generalise holdout sampling to incorporate all of our training data:

1. Randomly assign each *training* observation to one of k **folds**
2. Estimate the model k times, omitting one "fold" from training at a time
3. Calculate the prediction error using the fold not included in the data
4. Average the prediction errors across $k - folds$
5. Repeat 2-4 for each value of λ we want to test

Choose λ where the average cross-validated MSE is lowest

The choice of k will depend on:

▶ The time it takes to train the model
▶ The size of your training data

Application

Blackwell and Olsen (2021)

Suppose we have an outcome y, a treatment d, covariates X, and an "effect moderator" v

- ▶ We want to estimate an *inference* model
- ▶ Understand how the treatment effect is moderated

Naive suggestion:

- ▶ Include an interaction term to model the differential effect of treatment
  - ▶ I.e. $y_i = \beta_0 + \beta_1 d_i + \beta_2 v_i + \beta_3 d_i v_i + \boldsymbol{\beta'} \boldsymbol{X_i}$

What's wrong with this model?

- ▶ We assume that the interactive effect β_3 is constant across covariates
- ▶ This introduces bias into the model if $\boldsymbol{v} \boldsymbol{X}$ is related either to $\boldsymbol{d} \boldsymbol{v}$ or $\boldsymbol{y}$

Prediction and inference problem

Therefore the researcher faces a prediction problem *and* an inference problem:

- ▶ **Inference problem**: How do we control for potential bias introduced between X, v, d, and y?

- ▶ **Prediction problem**: Which interactions within vX are most likely to confound the results?

    - ▶ Let us denote the true non-zero predictors $\mathcal{P}$

    - ▶ Inverting a $\hat{y}$ problem – which variables are useful to predict new data?

From today's session we know that:

- ▶ Bias can be useful to offset variance when making out-of-sample predictions

- ▶ Bias inherently distorts our estimate of β

Combining LASSO and OLS

Blackwell and Olsen propose splitting the problem of interaction estimation into two stages:

1. **Variable selection**
   - ▶ Use LASSO to estimate a series of variable selection models
   - ▶ Attempt to find interaction terms that correlate with either outcome, treatment, or treatment-moderated interaction

2. **Inference**
   - ▶ Use OLS to estimate an inference model
   - ▶ Using only non-zero interaction terms in LASSO models

What makes this strategy so useful (and informative!) is that:

- ▶ We leverage bias to make better predictions in Stage 1
- ▶ We de-bias inference in Stage 2 using OLS + Stage 1 results

Post-double selection method

Stage 1

▶ Estimate LASSO models for:
1. y on $\{v, X, vX\}$
2. d on $\{v, X, vX\}$
3. dv on $\{v, X, vX\}$

▶ Let $Z*$ index all variables with non-zero coefficients in any of models 1-3

Stage 2

▶ Regress y on d, dv and $Z*$

Blackwell and Olson also suggest adding all "base-terms" (i.e X) regardless of LASSO coefficient

Extra Slides

Alternative $R(f)$ to the L1 norm

Following a similar logic to the shrinkage used by LASSO, we can define other measures of magnitude, like the L2 norm $||\boldsymbol{\beta}||_2$:

$$\sqrt{\sum_j |\beta_j|^2}$$

When we plug in the L2 norm into the loss function, we get the **ridge regression** estimator:

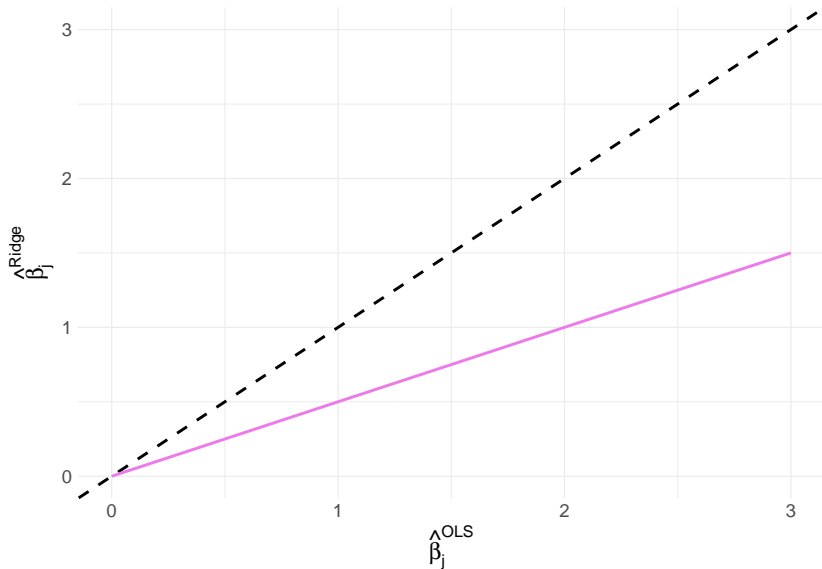$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} (y_i - \boldsymbol{x_i}\boldsymbol{\beta})^2 + \lambda||\boldsymbol{\beta}||_2$$

Unlike the LASSO estimator, ridge regression does not have a sharp cut-off, but rather scales the size of all coefficients in the model

Ridge regression – no corner solutions
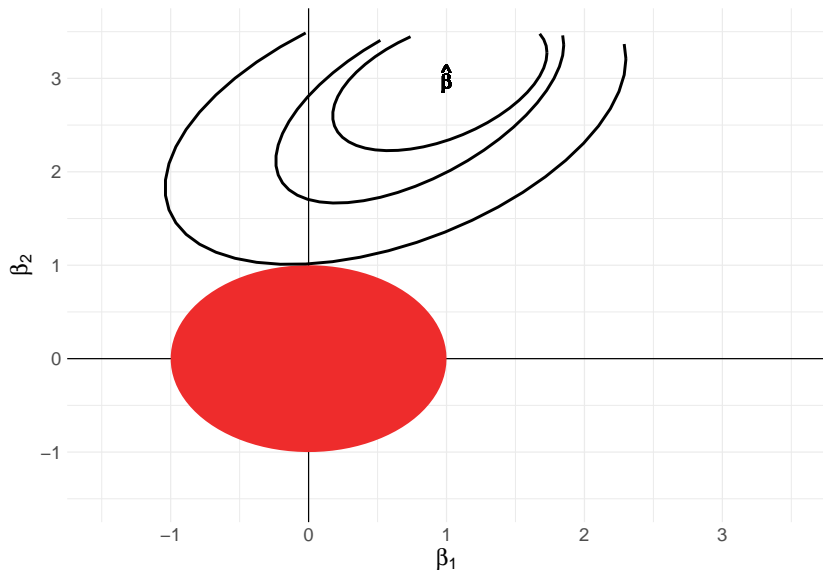
Ridge regression – constant scaling of coefficients

Gauss Markov Assumptions

Five assumptions need to hold:

1. y is a linear function of β
2. $\mathbb{E}[\epsilon_i] = 0$
3. $\mathbb{V}[\epsilon_i] = \sigma_i^2, \forall i$
4. $Cov(\epsilon_i, \epsilon_j) = 0, \forall i \neq j$
5. $Cov(\boldsymbol{x_i}, \epsilon_i) = 0$