Regularization and the Bias-Variance Tradeoff

Park Effects: Estimate the park effect α of each MLB ballpark, which represents the expected runs scored in one half-inning at that park above that of an average park, if an average offense faces an average defense

data all half-innings 2017-2019

variables i is the index of the i^{th} half-inning

y_i runs scored in half-inning i

$park(i)$

weather/day of the year

offensive team strength

defensive team strength

$ot(i)$ = offensive team at half-inning i

$dt(i)$ = defensive team

model $$y_i = \beta_0 + \alpha_{park(i)} + \varepsilon_i, \quad \mathbb{E}\varepsilon_i = 0$$

model $$y_i = \beta_0 + \alpha_{park(i)} + \beta_{ot(i)} + \gamma_{dt(i)} + \varepsilon_i$$

$$\left\{\begin{array}{l} \beta_{ot(i)} \\ \text{30 off. teams Phi, LAD, ...} \\ ot(i) = \text{the off. team in half-inning } i \\ \beta_{LAD}, \beta_{Phi}, \dots \end{array}\right.$$

Equivalently, $y_i = x_i^T \beta + \varepsilon_i$

X is a matrix whose i^{th} Row is defined by

Row i

$$x_i^T = [1, \underbrace{\text{0's everywhere except 1 at park(i)}}, \underbrace{\text{0's everywhere except 1 at ot(i)}}, \underbrace{\text{0's everywhere except 1 at dt(i)}}]$$

$$x_i^T = [\overset{\text{intercept}}{1}, \overset{\text{park 1}}{\bullet} \dots \overset{\text{park 30}}{\bullet}, \overset{\text{off 1}}{\bullet} \dots \overset{\text{off 30}}{\bullet}, \overset{\text{def 1}}{\bullet} \dots \overset{\text{def 30}}{\bullet}]$$

$$\beta = \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \vdots \\ \alpha_{30} \\ \beta_1 \\ \vdots \\ \beta_{30} \\ \gamma_1 \\ \vdots \\ \gamma_{30} \end{pmatrix}$$

Problem: Multicollinearity

When home team is on offence, $park(i) = ot(i)$
when road team is on offense, $park(i) = dt(i)$
So it is tough to disentangle
$\alpha_{park(i)}$ from $\beta_{ot(i)}, \gamma_{dt(i)}$

Are the runs scored in this half-inning due to the offensive home team being good or the park being easy?

$$\underbrace{y_i}_{\text{Runs scored in half-inning } i} = \beta_0 + \underbrace{\alpha_{park(i)}}_{\text{Park effect}} + \beta_{ot(i)} + \gamma_{dt(i)} + \varepsilon_i$$

To disentangle these effects, we need a huge number of instances of Road teams on offence to estimate $\beta_{ot(i)}$ well, we need a huge number of instances of Home teams on offence to estimate $\gamma_{dt(i)}$ well, with a good $\beta_{ot(i)}$ and $\gamma_{dt(i)}$ we can estimate $\alpha_{park(i)}$ well.

Our dataset of all half-innings from 2017–2019 consists of $\approx 121,000$ half-innings (rows). This may seem like a huge amount of data. But, with this multicollinearity, is it really?

Simulation Study

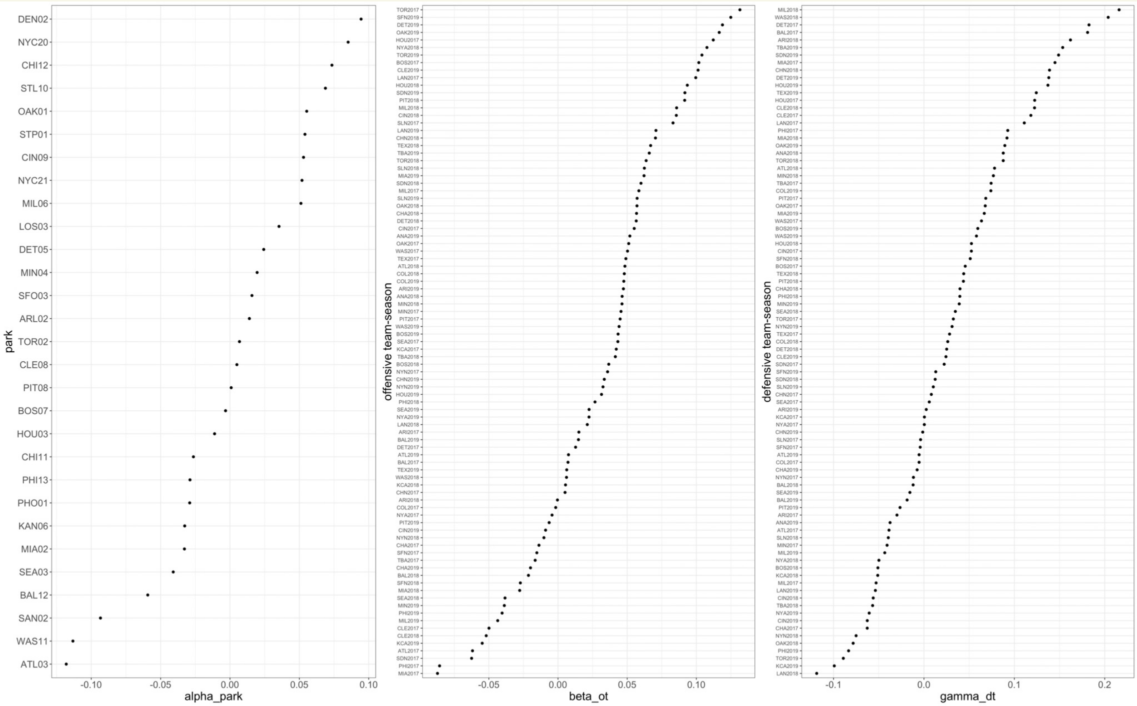
<u>Idea</u> Pretend we knew α, γ, β

Generate fake historical data y (121,000 fake innings)

Then estimate α (and β, γ) from this fake data.

Then compare $\hat{\alpha}$ to "true" α which we know.

* Suppose the "true" coefficients are



Then, given these, lets generate y according to

$$y_i = \underbrace{\beta_0 + \alpha_{park(i)} + \beta_{ot(i)} + \gamma_{dt(i)}}_{x_i^T\beta} + \varepsilon_i$$

$$y_i = \text{Round}\left(\mathcal{N}_+\left(x_i^T\beta, 1\right)\right)$$

$$\mathcal{N}_+(x_i^T\beta, 1) = x_i^T\beta + \varepsilon_i$$

there exits some ε_i which satisfing the equation

&

$\mathbb{E}\varepsilon_i = 0$

$\mathrm{VaR}(\varepsilon_i) = 1$

$$y = X\beta + \varepsilon$$

if we're only interested in point estimate $\hat{\beta}$, we only need to know $\mathbb{E}\varepsilon = 0$ and anything about the actual distribution of ε is irrelevant.

$\hat{\beta}$ is unbiased

$$\mathbb{E}\hat{\beta}_{OLS} = \mathbb{E}\left[(X^TX)^{-1}X^Ty\right] = (X^TX)^{-1}X^T\,\mathbb{E}[y]$$

$$= (X^TX)^{-1}X^T\,\mathbb{E}\left[X\beta + \varepsilon\right]$$

$$= \boxed{(X^TX)^{-1}X^T(X}\beta) + (X^TX)^{-1}X^T\,\cancelto{0}{\mathbb{E}[\varepsilon]}$$

$$= \beta$$

$\varepsilon \sim N(0, \sigma^2) \to$ CI on β

$\hat{\beta} \pm 2\delta$

$$y = X\beta + \varepsilon$$

$$\begin{array}{cccc} & & X\beta & \\ & & \text{1000 parallel universes} & \\ \swarrow & \downarrow & \cdots & \searrow \\ y & y & & y \\ \downarrow & \downarrow & & \downarrow \\ \hat{\beta} & \hat{\beta} & & \hat{\beta} \end{array}$$

where did x_i^T come from? → used the actual X matrix (the actual schedule)

where is ε_i? → implicit in the $\mathcal{N}$

* Now we have "true" known coefficients (β, γ, α), we have the observed schedule matrix X, the fake generated runs scored vector y.

* Using our fake historical dataset (X, y), let's use linear regression to estimate the coefficients, $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$.

$(X^T X)^{-1} X^T y$

X known fixed schedule matrix

β "true" coeffs

know

pretend we don't know

$\swarrow \quad \downarrow \quad \ldots \quad \searrow$

S parallel universes

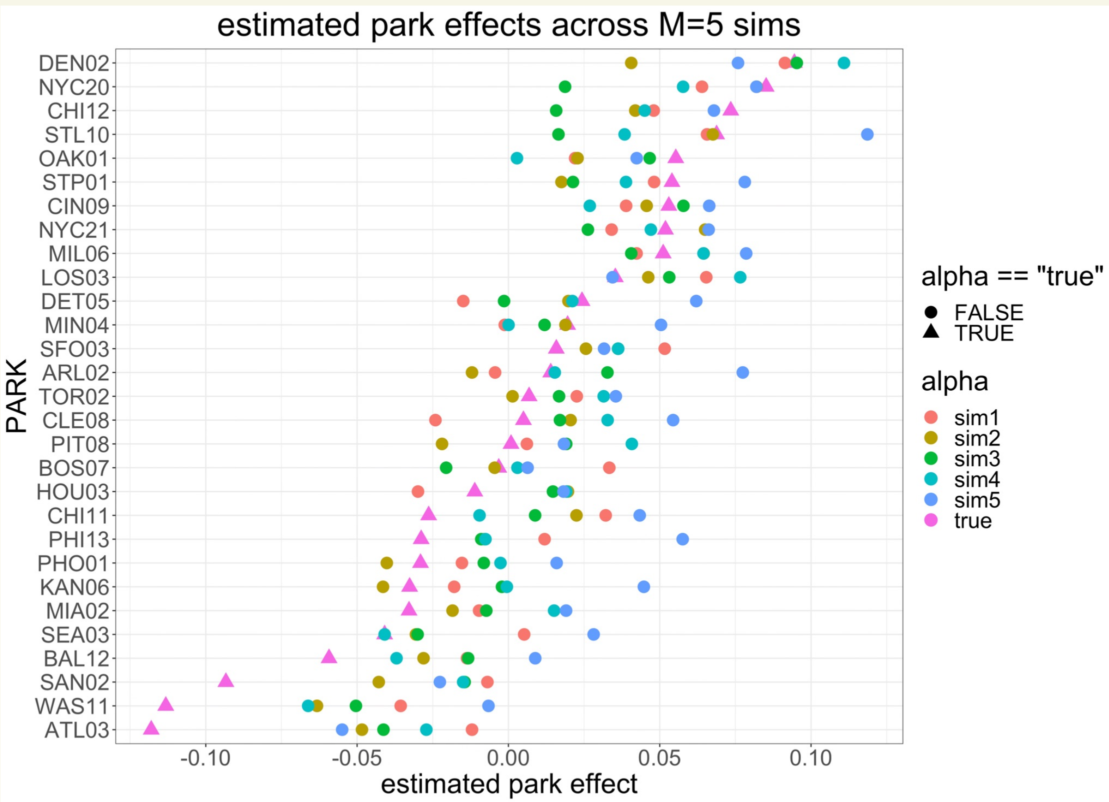
$y^{(1)} \quad y^{(2)} \quad \ldots \quad y^{(S)}$

generated using

$\varepsilon^{(1)} \quad \varepsilon^{(2)} \quad \ldots \quad \varepsilon^{(S)}$

$\downarrow \quad \downarrow \quad \downarrow$

$\hat{\beta} = (\hat{\alpha}, \hat{\beta}, \hat{\gamma}) \quad \hat{\beta} \quad \hat{\beta}$



* Due to Randomness in the training dataset, from $\underbrace{\text{the noise in generating } y}_{\varepsilon}$,

Each simulation yields very different park effect estimates $\hat{\alpha}$ even though the "true" park effects are the same.

* These OLS (ordinary least squares) coefficients $\hat{\alpha}$ are quite sensitive to the noise of the training set, even though we have 121,000 innings of data.

* How can we make the estimated coefficients <u>less sensitive</u> to the Random idiosyncracies of our training dataset?

from data (X, y)

Q What's the least sensitive, dumb, estimator you can think of?

ZERO

Idea Blend the strengths of OLS and zero:

- Constant values like zero are not sensitive to the random idiosyncracies of training data, but are wrong (on average) for many parks

- OLS estimates are very sensitive to the randomness of the training set, especially in the presence of multicollinearity, but are Unbiased (on average, they are right)

— tradeoff b/t sensitivity and unbiasedness

Idea Shrink the OLS estimates towards zero.
Make them smaller! And hence less sensitive.

OLS: $\hat{\beta}^{(OLS)} = \underset{\beta}{\operatorname{argmin}}\ RSS(\beta) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (x_i^T\beta - y_i)^2$

Now:

$$\underset{\beta}{\operatorname{argmin}} \overbrace{\sum_{i=1}^{n}(y_i - x_i^T\beta)^2}^{\text{want } x_i^T\beta \text{ close to } y} + \overbrace{\lambda \underbrace{\sum_j \beta_j^2}_{\text{Penalty term}}}^{\text{make } \beta \text{ small}} = L(\beta)$$

Shrinkage

A larger β_j has larger β_j^2
so if we are minimizing $L(\beta)$
we want β_j to be smaller

* This technique of adding a <u>penalty term</u> to our loss function is called <u>**Regularization**</u>.

This is L_2 Regularization (β_j^2)

$$|\beta_j| \rightarrow L_1$$

* The hyperparameter $\lambda > 0$ describes by how much we are penalized for having larger β_j.

λ is simply a number which is tuned to optimize predictive performance.

* <u>Ridge Regression</u> = OLS + L_2 Regularization

$$\hat{\beta}^{(Ridge)} = \underset{\beta}{\mathrm{argmin}} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$= \underset{\beta}{\operatorname{argmin}} \; (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$$

$$\beta^T \beta = (\beta_1 \cdots \beta_p) \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} = \beta_1^2 + \dots + \beta_p^2$$

Calculus: gradient $= 0$ and solve

$$\nabla_\beta L(\beta) = \nabla_\beta \left[(y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta \right]$$

$$= \nabla_\beta \left[y^T y - 2\beta^T X^T y + \beta^T X^T X \beta \right] + \lambda \nabla_\beta \left[\beta^T \beta \right]$$

$$= \left[-2X^T y + 2X^T X \beta \right] + \lambda \left[2\beta \right]$$

$$\frac{\partial}{\partial \beta_j} \left[\beta^T \beta \right] = \frac{\partial}{\partial \beta_j} (\beta_1^2 + \dots + \beta_p^2) = 2\beta_j$$

$$\nabla_\beta \left[\beta^T \beta \right] = \left(\frac{\partial}{\partial \beta_1} [\beta^T \beta], \dots, \frac{\partial}{\partial \beta_p} [\beta^T \beta] \right)$$

$$= 0$$

$$\Longrightarrow \quad (2\underbrace{X^TX}_{p\times p} + 2\underbrace{\lambda}_{1\times 1}\underbrace{I}_{p\times p})\beta = 2X^Ty$$

$$2\lambda\beta \longrightarrow 2\lambda I\beta$$

$$I = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}$$

$$\Longrightarrow \quad (X^TX + \lambda I)\cdot\beta = X^Ty$$

$$\underbrace{(X^TX+\lambda I)^{-1}(X^TX+\lambda I)}_{I}\beta = (X^TX+\lambda I)^{-1}X^Ty$$

$$\Longrightarrow \quad \boxed{\hat{\beta}^{(Ridge)} = (X^TX+\lambda I)^{-1}X^Ty}$$

$$\hat{\beta}^{(OLS)} = (X^TX)^{-1}X^Ty$$

$$\lambda I = \begin{pmatrix} \lambda & & & 0 \\ & \lambda & & \\ & & \ddots & \\ 0 & & & \lambda \end{pmatrix}$$

Ridge

pretend $x^T x$ is a number

$(X^T X)^{-1}$ is like $\frac{1}{x^T x}$

$(X^T X + \lambda I)^{-1}$ is like $\frac{1}{x^T x + \lambda}$

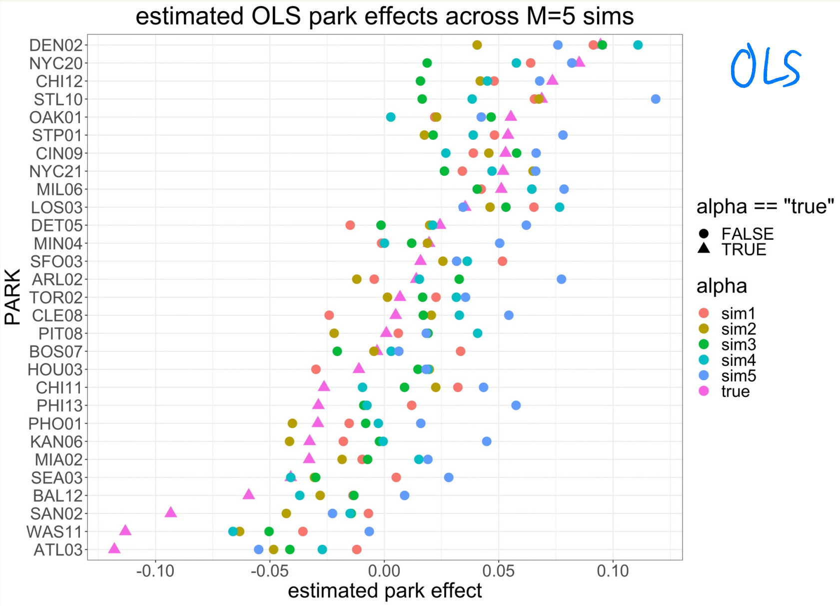
if $x^T x \approx 0$, $\frac{1}{x^T x}$ is large and unstable

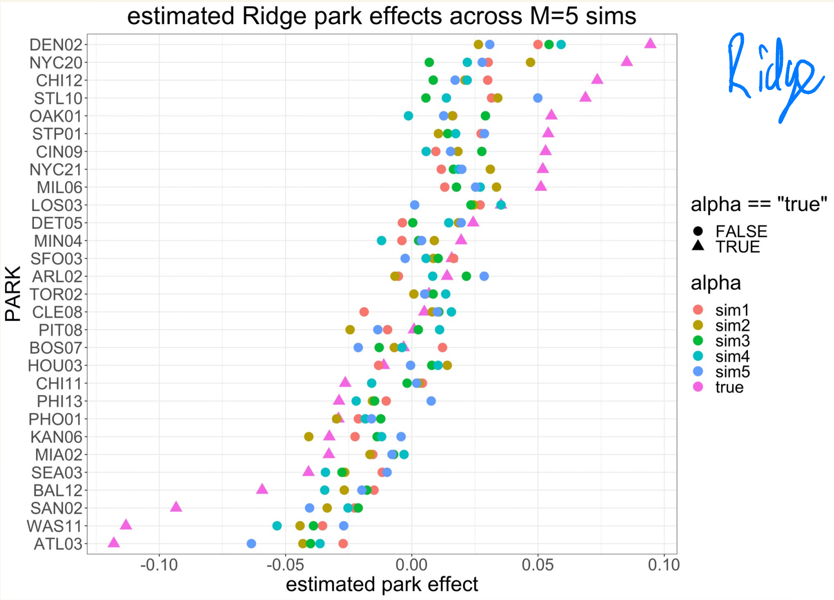
$\frac{1}{x^T x + \lambda}$ is more stable

In the presence of multicollinearity, $(X^T X)^{-1}$ is like dividing by zero in some way.

Let's look at $\hat{\beta}^{(Ridge)}$ and $\hat{\beta}^{(OLS)}$

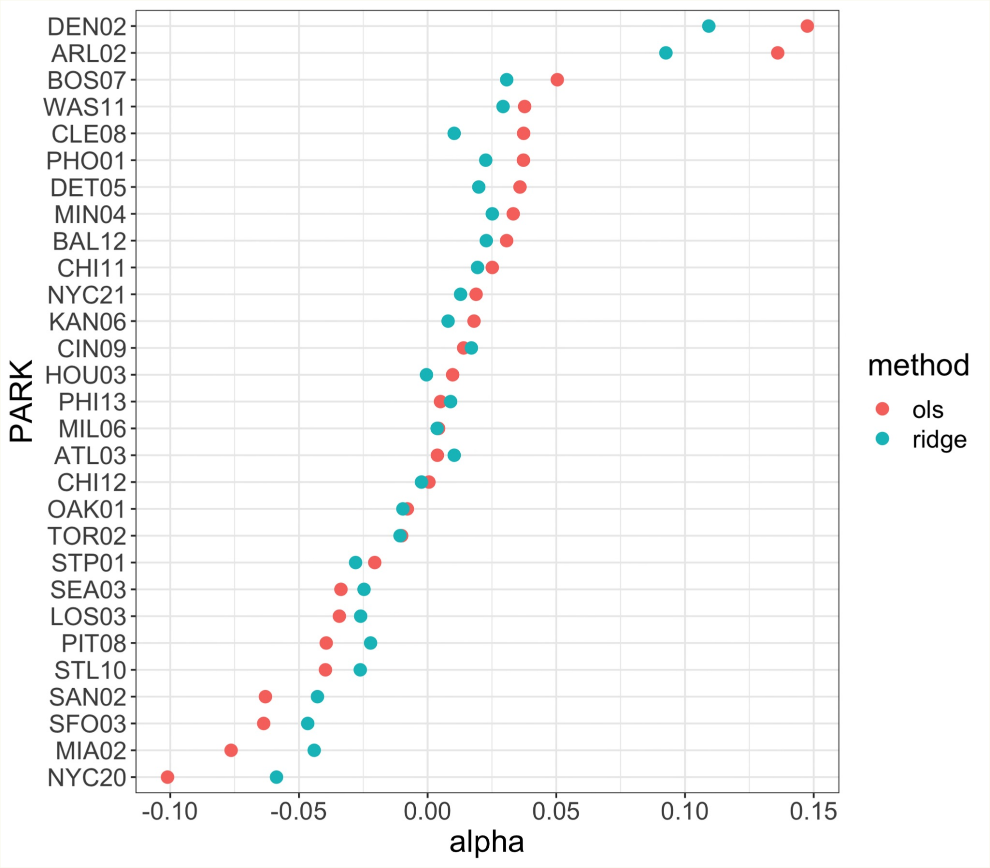
$\lambda = 0.3$





Ridge park effects are more stable across simulations, even though they are unbiased!

```
> ### error
> err(beta.pk.df.sim)
[1] 0.03528335
> err(beta.pk.df.sim_ridge)
[1] 0.03804942
> ### error on non-outliers
> err(beta.pk.df.sim %>% filter( abs(beta.pk.true) < 0.05) )
[1] 0.02533202
> err(beta.pk.df.sim_ridge %>% filter( abs(beta.pk.true) < 0.05) )
[1] 0.01690153
> ### error on outliers
> err(beta.pk.df.sim %>% filter( abs(beta.pk.true) >= 0.05) )
[1] 0.04406852
> err(beta.pk.df.sim_ridge %>% filter( abs(beta.pk.true) >= 0.05) )
[1] 0.05359246
```



How do we quantify the sensitivity of an estimator to the random idiosyncracies of a training dataset?

Model

$$\boxed{y_i = f(x_i) + \varepsilon_i}$$

"true" underlying function f

noise ε_i $\qquad \mathbb{E}\varepsilon_i = 0$

$$\mathbb{E}(y_i | x_i) = f(x_i)$$

Goal of ML: estimate f (obtain $\hat{f}$) as best as possible from a training dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$$

$$\hat{f} = \hat{f}(x; \mathcal{D})$$

<u>Want</u> our estimator $\hat{f}$ to be as "close" to the true f as possible;

On average we want $\hat{f}$ to be as close to f as possible,

$$MSE(f, \hat{f}) := \mathbb{E}\left[\left(f(x) - \hat{f}(x; \mathcal{D})\right)^2\right]$$

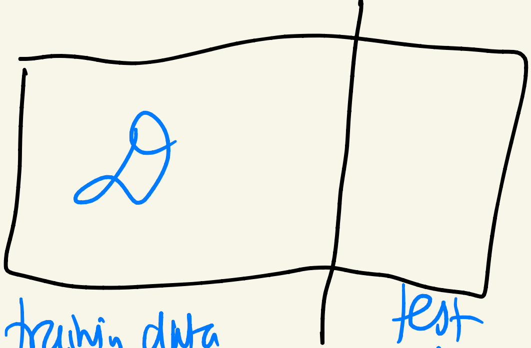
averaging over the randomness in the training set $\mathcal{D}$

We don't actually observe f, so

$$MSE = \mathbb{E}\left[\left(Y(x) - \hat{f}(x; \mathcal{D})\right)^2\right]$$

out-of-sample testing data

full dataset



training data

↓

used to create $\hat{f}$

test data (out-of-sample)

↓

used to evaluate $\hat{f}$

$$\mathrm{MSE}(x; \mathcal{D}) = \mathbb{E}\left[\left(Y - \hat{f}(x; D)\right)^2\right]$$

$$= \mathbb{E}\,(Y - \hat{f})^2$$

$$\hat{f} = \hat{f}(x; D)$$

$$x \mapsto \hat{f}(x)$$

$$= \mathbb{E}\left(Y^2 - 2Y\hat{f} + \hat{f}^2\right)$$

$$= \mathbb{E}\,Y^2 - 2\,\mathbb{E}(Y\hat{f}) + \mathbb{E}(\hat{f}^2)$$

$$Y = f + \varepsilon$$

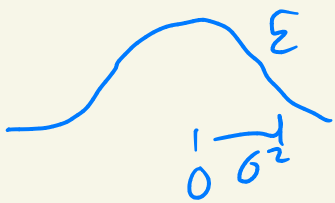
$$= \mathbb{E}(f+\varepsilon)^2 - 2\,\mathbb{E}\big[(f+\varepsilon)\hat{f}\big] + \mathbb{E}\hat{f}^2$$

$$= \mathbb{E}\left[f^2 + 2f\varepsilon + \varepsilon^2\right]$$
$$\quad - 2\,\mathbb{E}\left[f\hat{f} + \varepsilon\hat{f}\right] + \mathbb{E}\hat{f}^2$$

$x \mapsto f(x)$ is (unknown) fixed/constant

$$= f^2 + \cancel{2f\,\mathbb{E}\varepsilon} + \mathbb{E}\varepsilon^2$$
$$\quad - 2f\,\mathbb{E}\hat{f} - \cancel{2\,\mathbb{E}(\hat{f})\,\mathbb{E}(\varepsilon)} + \mathbb{E}\hat{f}^2$$

$\mathbb{E}\varepsilon = 0$



$\mathbb{E}\varepsilon = 0$

$\mathbb{E}\varepsilon^2 = \sigma^2$

$$= f^2 - 2f\,\mathbb{E}\hat{f} + \mathbb{E}\hat{f}^2 + \mathbb{E}\varepsilon^2$$

$$= \underline{f^2} - \underline{2f\,\mathbb{E}\hat{f}} + \underline{(\mathbb{E}\hat{f})^2} + \mathbb{E}\hat{f}^2 - (\mathbb{E}\hat{f})^2 + \mathbb{E}\varepsilon^2$$

$$= \underline{\left(f - \mathbb{E}\hat{f}\right)^2} + \left[\mathbb{E}\hat{f}^2 - \left(\mathbb{E}\hat{f}\right)^2\right] + \mathbb{E}\varepsilon^2$$

$Y = f + \varepsilon$

$$MSE(x; D) = \mathbb{E}\left[\left(Y - \hat{f}(x; D)\right)^2\right]$$

$$= \left(f - \mathbb{E}\hat{f}\right)^2 + \left[\mathbb{E}\hat{f}^2 - \left(\mathbb{E}\hat{f}\right)^2\right] + \mathbb{E}\varepsilon^2$$

out-of-sample

$$\boxed{MSE(\hat{f}) = \text{Bias}(\hat{f})^2 + \text{VAR}(\hat{f}) + \text{IRReducible ERROR}}$$

can't actually compute this

The Bias-Variance Tradeoff

- OLS: unbiased $\mathbb{E}\hat{\beta}^{(OLS)} = \beta$
  higher variance

- Ridge: biased
  lower variance (more stable)

- variance ← sensitivity to the training set

<u>Overfitting</u> = too much variance
= memorizes noise in the training set, rather than getting the underlying trend

$$\text{Bias}(\hat{f}) = \mathbb{E}(f - \hat{f})^2$$

* f in real life is the "truest" underlying function

$\hat{f}$ = OLS → $X\hat{\beta} + \varepsilon$ FALSE