Variance and Bias Decomposition and Feature Selection

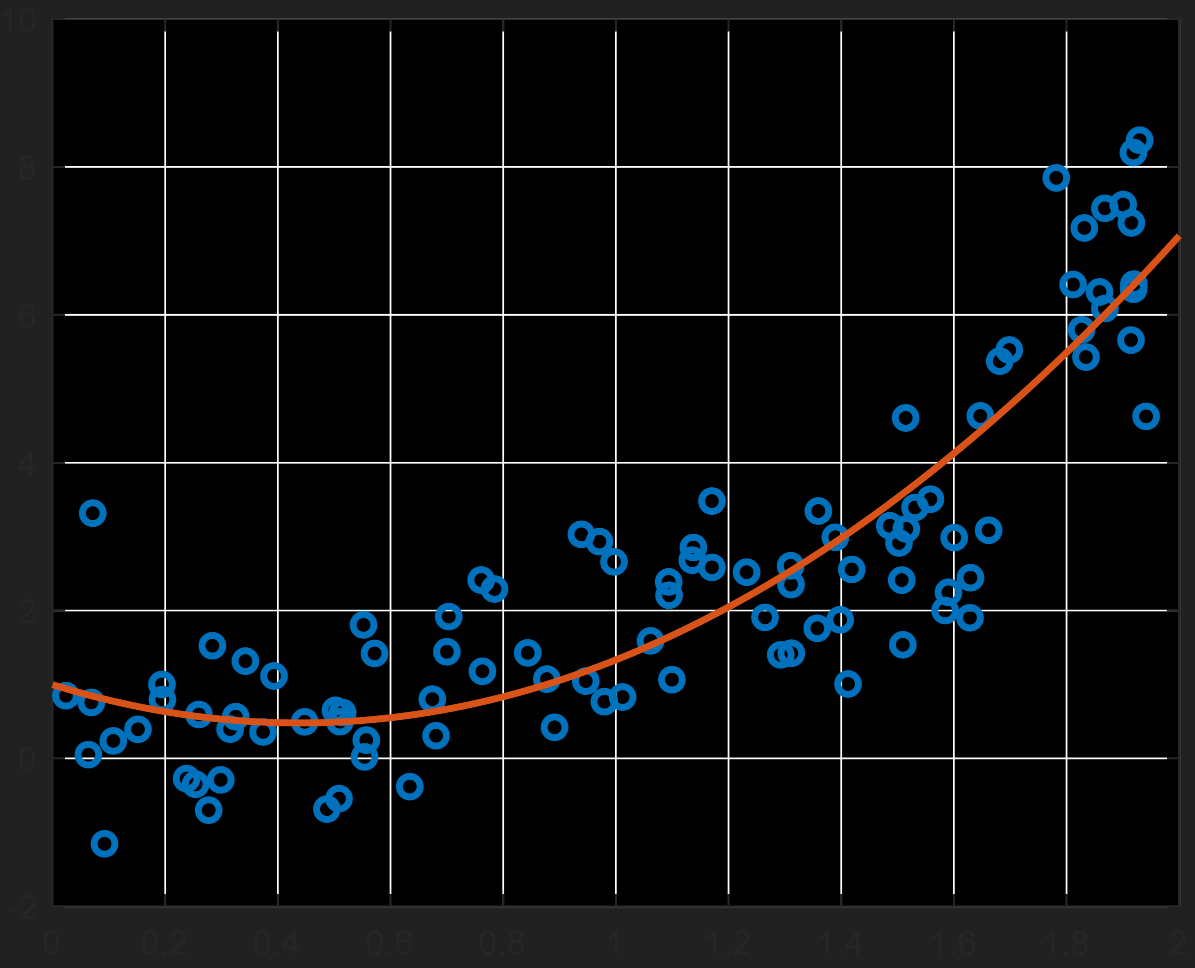
COMS21202, Part III

- Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Objectives

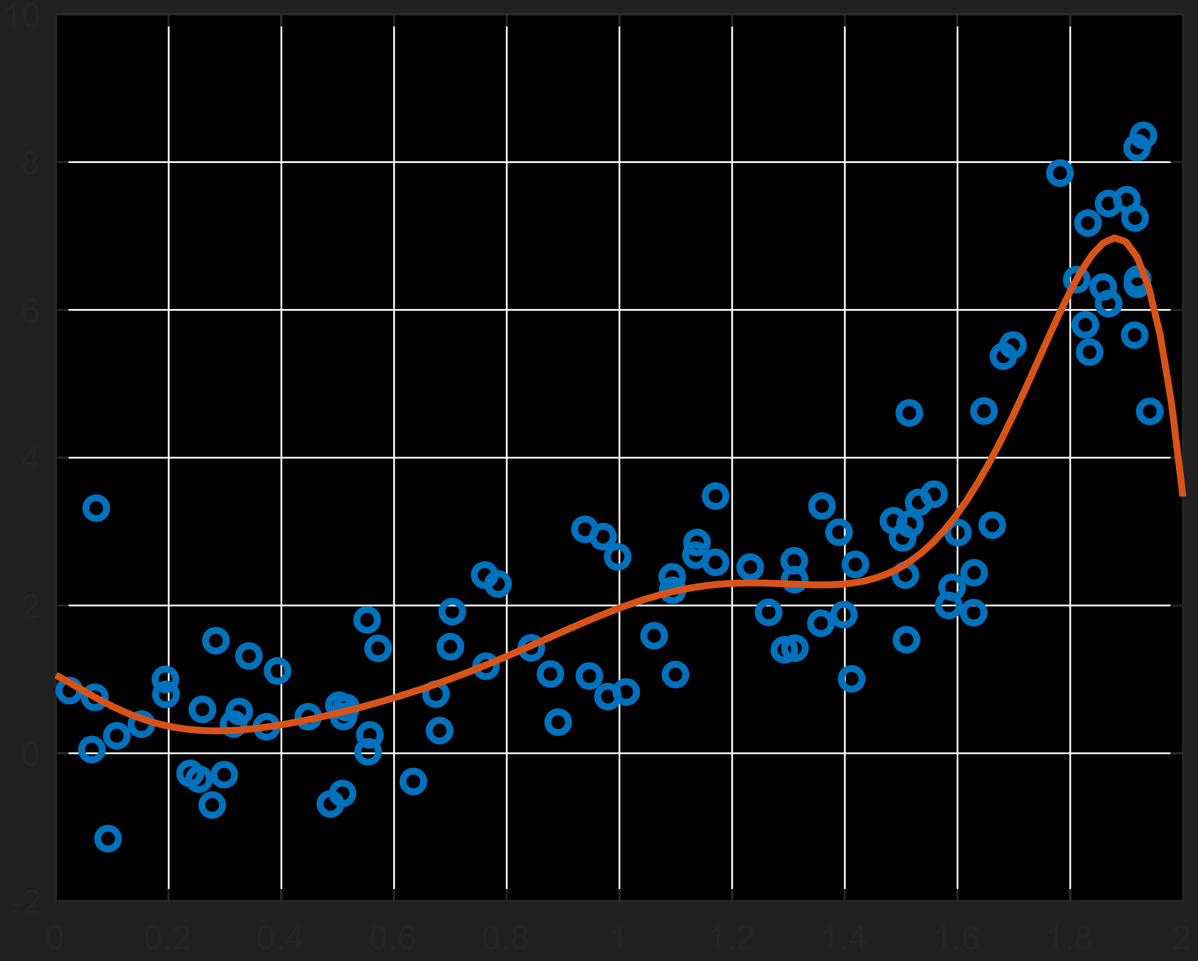
- Understanding how the complexity of feature transforms affects the **training** and **testing** error.
- Decomposing **expected error** into **bias** and **variance**.
- Finding the right feature complexity using **out sample error**.

Recall: $y = \exp(1.5x - 1) + \epsilon,\ \epsilon \sim N(0,1)$



- Polynomial transform with $b = 2$.
- Square error:108.97

Recall: $y = \exp(1.5x - 1) + \epsilon, \epsilon \sim N(0,1)$



- Polynomial transform with $b = 8$.
- Square error:78.87

Observation

- The more complex $\boldsymbol{f}$ is, the more flexible our model $\hat{y}$ is.
- If $\hat{y}$ is too flexible, we start to fit noises rather than the underlying function!

💡

- Regenerate y_i with different ϵ_i and measure squared error again!

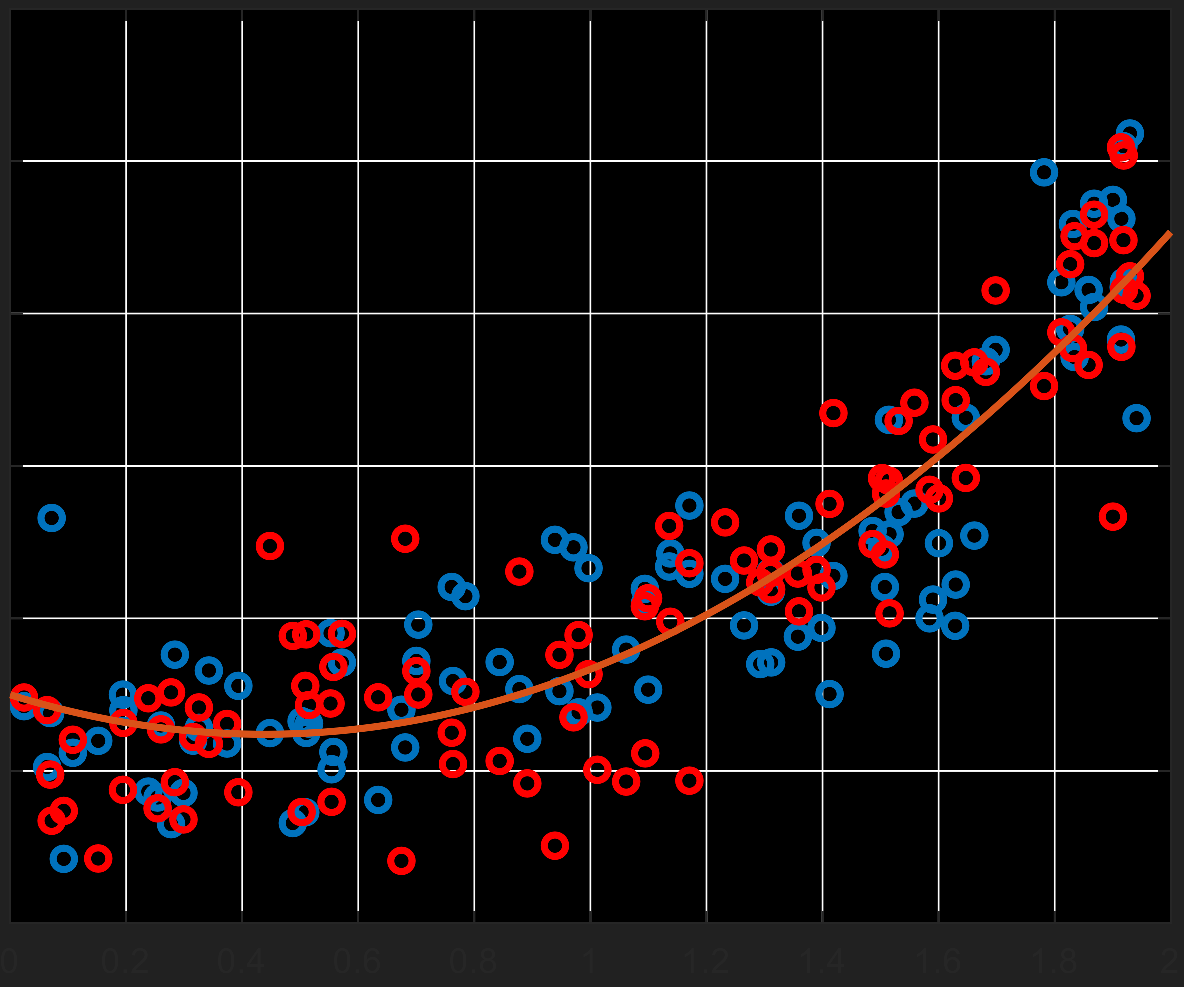
Testing Set & Testing Squared Error

- Denote $D := \{(y_i, x_i)\}_{i=1}^{n}$.
  - i.e., our training data.
- Now generate a **new** dataset D':
- $\forall x_i \in X,\ y'_i = \underbrace{\exp(1.5x_i - 1)}_{g(x),\ \text{“real function”}} + \epsilon'$,
  - $\epsilon' \sim N(0,1)$
  - ϵ' is independent from ϵ.
- $D' := \{(y_i', x_i)\}_{i=1}^{n}$, i.e., testing set.

Testing Set & Testing Square Error

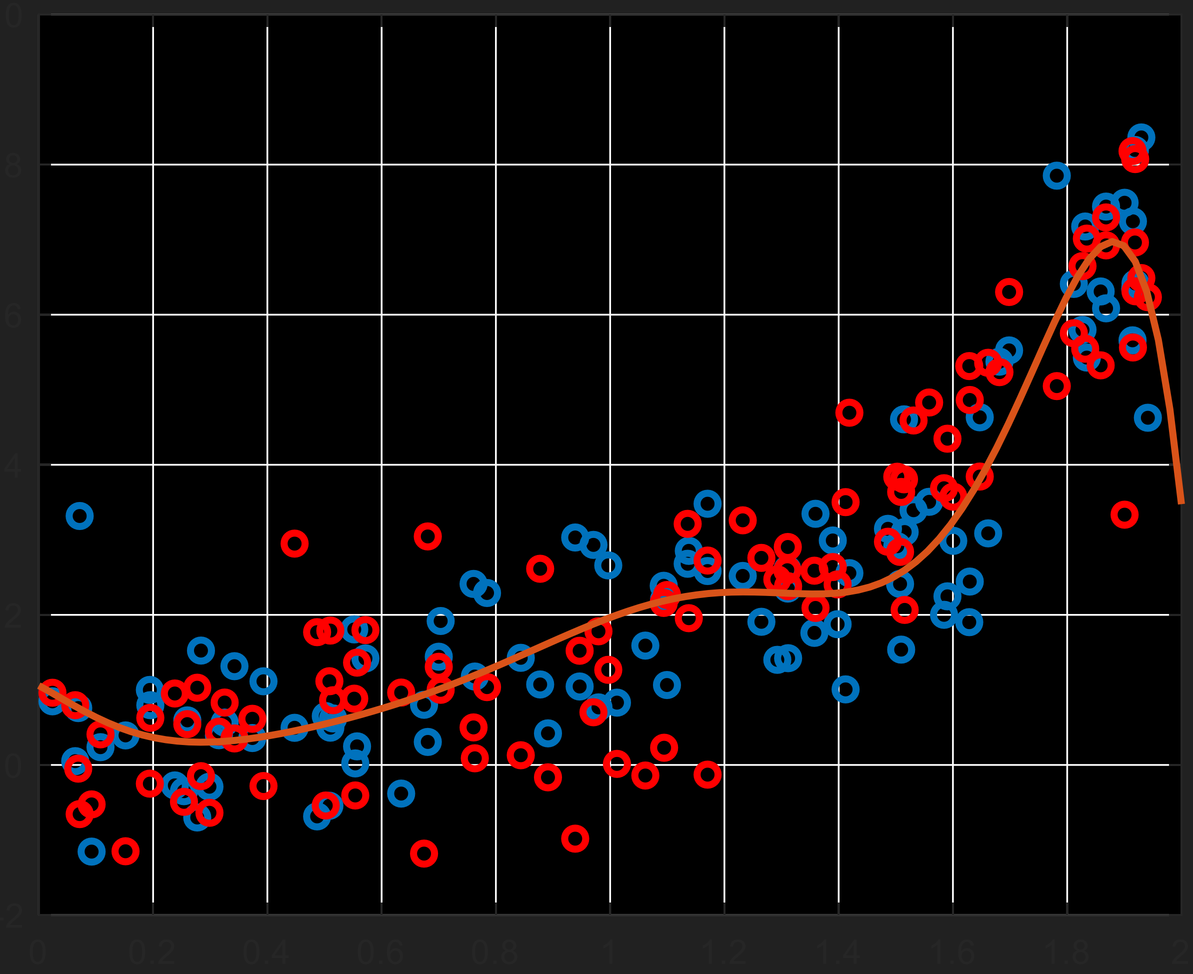
- Testing square error: $\sum_{i=1}^{n}\left(y_i' - \widehat{y_i}\right)^2$
- We **cannot** generate D' in this way in practice.
  - We **do not** know the generating mechanism of y in reality.
  - Here, D' is only generated for study purposes.

Example: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$



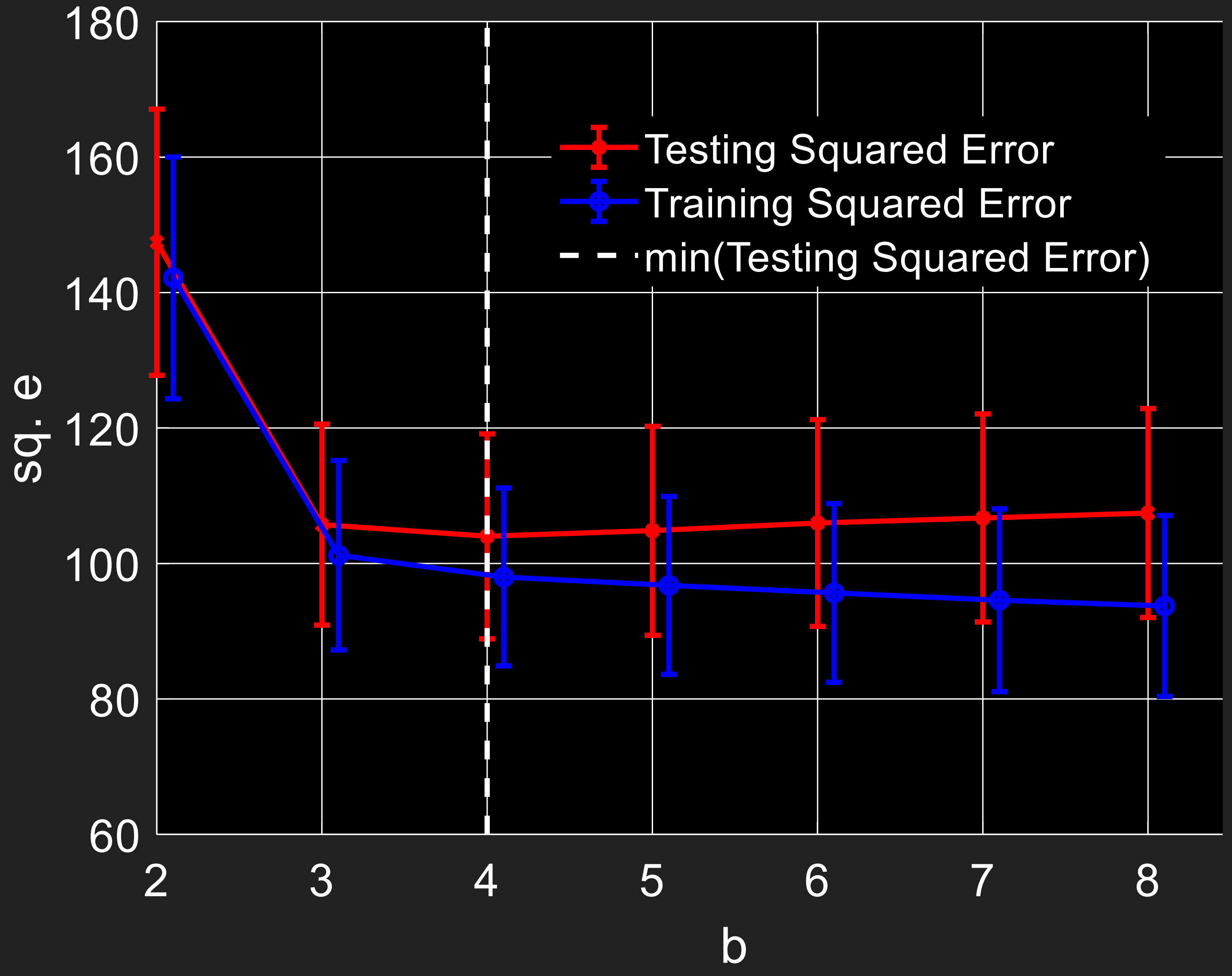
- Red dots is testing set.
- Polynomial transform with $b = 2$.
- Testing error: 99.025

Example: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$



- Red dots is testing set.
- Polynomial transform with $b = 8$.
- Testing error: 128.01

Testing/Training error vs. *b*, 100 times with error bar



How to make sense of this?

- Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Testing/Training error vs. b

- The training error drops as the complexity of our feature increases.
  - which is a result of “overfitting” as we previously discussed in this unit.
- Why the testing error drops then increases again?
- To answer this, we look at the **expected square error**.

Expected Square Error

- Instead of look at error on a single dataset, we look at **expected error**.
  - Instead of evaluating a student based on one exam score, we look at his/her expected score over the entire course.
- The expected error: $\mathbb{E}_\epsilon\left[(y-\hat{y})^2|\boldsymbol{x}_i\right]$
  - suppose y is generated by $y = g(x) + \epsilon$ (like in the previous case), we can rewrite:
    - $\mathbb{E}_\epsilon\left[(y-\hat{y})^2|\boldsymbol{x}_i\right] = \mathbb{E}_\epsilon[(g(\boldsymbol{x}_i) + \epsilon - \hat{y})^2|\boldsymbol{x}_i]$
  - PC: write down the formula using integral.

Expected Square Error Decomposition

- Bias and Variance Decomposition:

$$\mathbb{E}_\epsilon\left[(y-\hat{y})^2|\boldsymbol{x}_i\right]$$

$$=\mathrm{var}[\epsilon]+\left[g(x)-\mathbb{E}_\epsilon[\hat{y}|\boldsymbol{x}_i]\right]^2+\mathrm{var}[\hat{y}|\boldsymbol{x}_i]$$

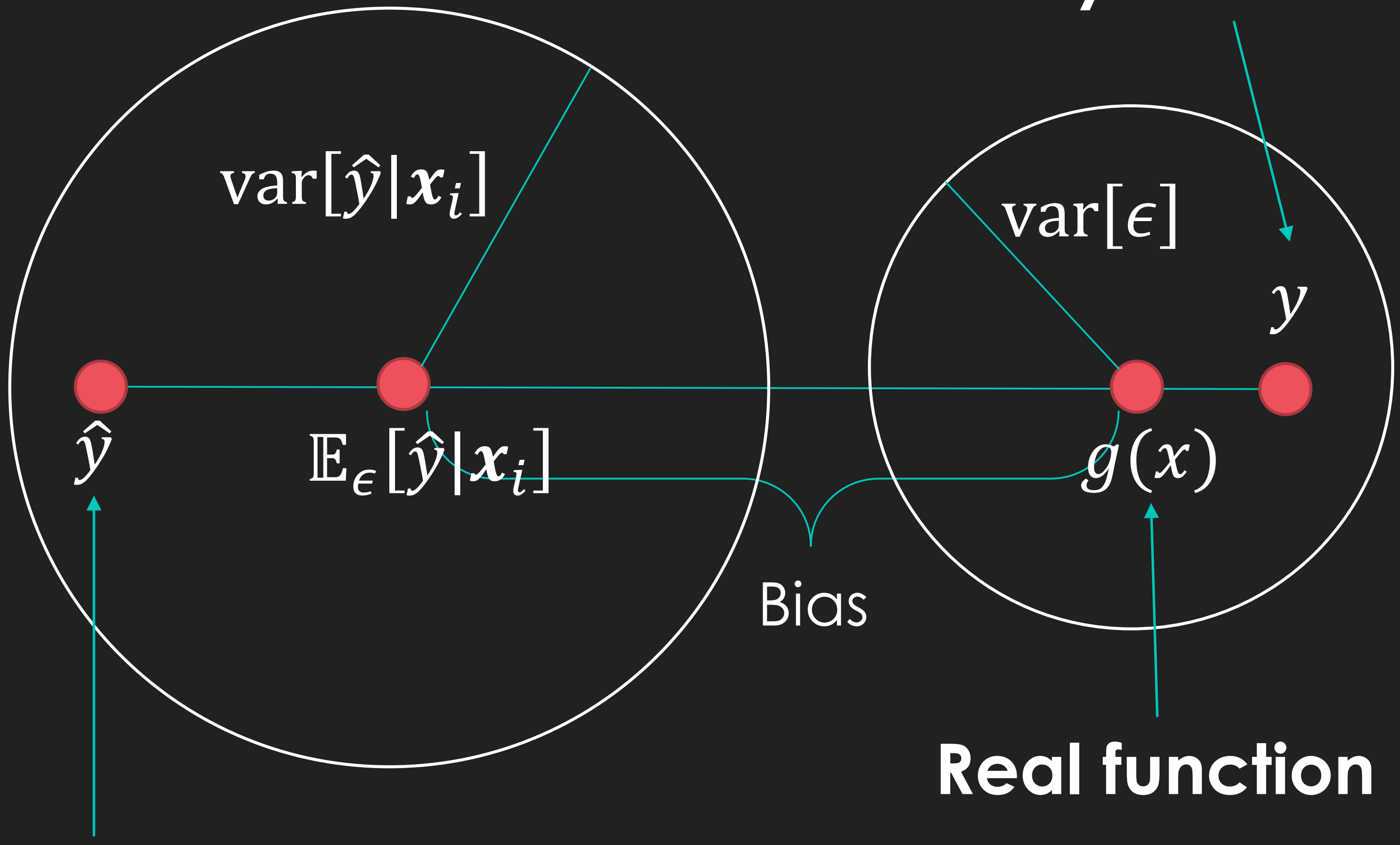
Irreducible error bias variance

- "Variance and Bias decomposition"
- Live demonstration

Expected Square Error Decomposition

- $\mathrm{var}[\epsilon] + \left[g(\boldsymbol{x}_i) - \mathbb{E}_\epsilon[\hat{y}|\boldsymbol{x}_i]\right]^2 + \mathrm{var}[\hat{y}|\boldsymbol{x}_i]$
  - The first term measures the randomness of our data generating process, which is beyond our control.
  - The second term shows the accuracy of our expected prediction.
  - The third term shows how easily our learned function is affected by the randomness of the dataset.

A Visualization of V-B Decomposition



$\text{var}[\hat{y}|\boldsymbol{x}_i]$
$\text{var}[\epsilon]$
y
$\hat{y}$
$\mathbb{E}_\epsilon[\hat{y}|\boldsymbol{x}_i]$
$g(x)$
Bias

Noisy observation

Real function

Reconstructed function

- Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Variance and Bias Tradeoff

- $\mathrm{var}[\epsilon] + \left[g(\boldsymbol{x}_i) - \mathbb{E}_\epsilon[\hat{y}|\boldsymbol{x}_i]\right]^2 + \mathrm{var}[\hat{y}|\boldsymbol{x}_i]$
  - As we increase b, $\hat{y}$ becomes more **complex** and can adapt to more complex underlying function, thus 2nd term keeps dropping.
  - As wee increase b, $\hat{y}$ becomes more **sensitive** to the noise in our dataset, thus 3rd term keeps increasing.
  - A **balance** between 2nd and 3rd term gives the minimum testing error.

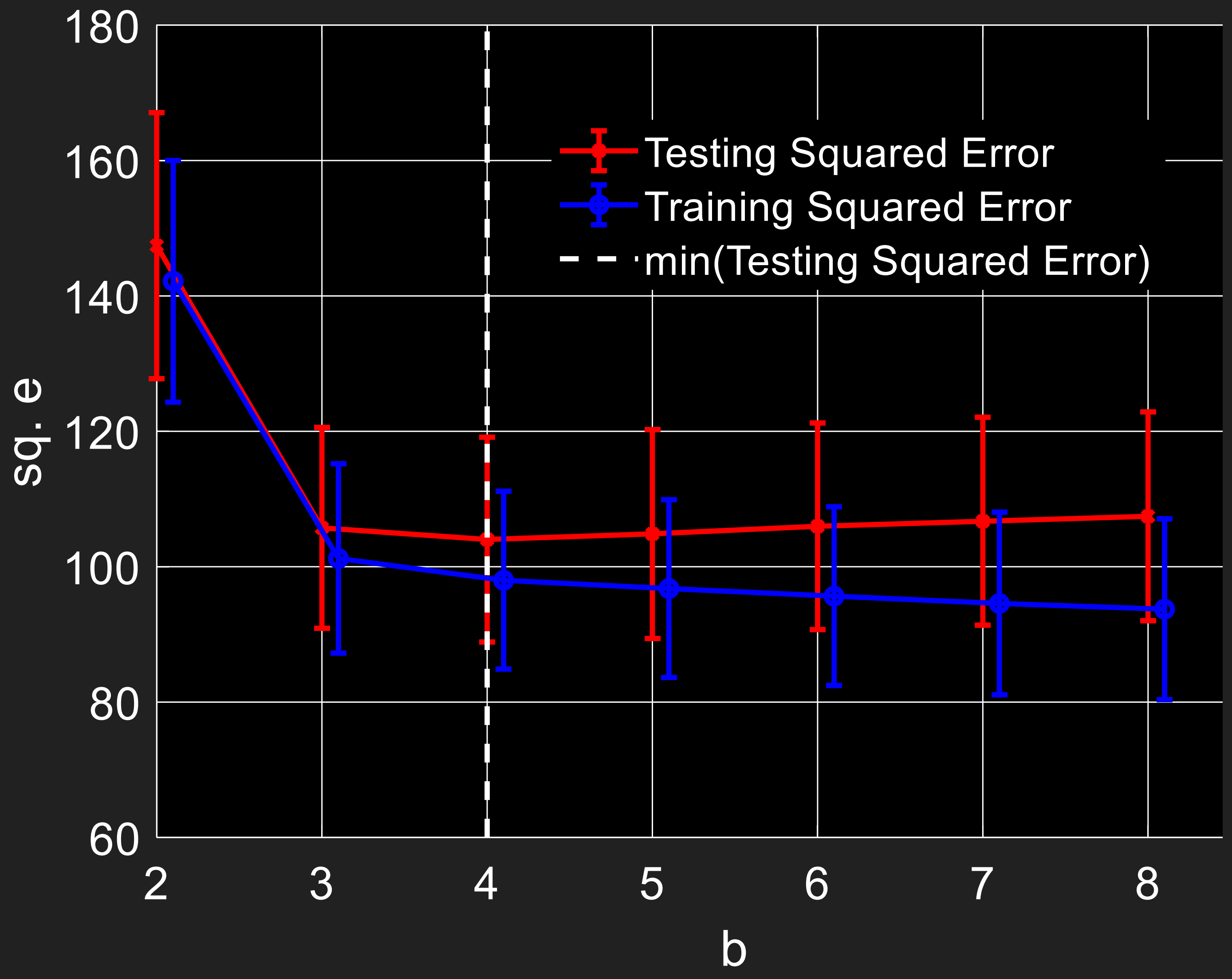
In Sample Error

- We derived $\mathbb{E}_\epsilon\left[(y-\hat{y})^2|\boldsymbol{x}_i\right]$ only with respect to each $\boldsymbol{x}_i$.
- To calculate the collective error, we need to average over all $\boldsymbol{x}_i$.
  - $\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_\epsilon\left[(y-\hat{y})^2|\boldsymbol{x}_i\right]$
  - is called **in sample error**

In Sample Error

- Earlier, the testing error on D' is a (rough) approximation of the in sample error.
- It seems to do a good job for selecting the "right" features.
  - i.e., balancing between bias and variance.

Testing/Training error vs. b, 100 times with error bar



Approx. in sample error selects f with $b = 4$

- Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

A Closer Look at In Sample $\mathrm{var}[\hat{y}]$

- Plug in **LS solution** of $\hat{y}$ in $\mathrm{var}[\hat{y}|\boldsymbol{x}_i]$: 😱
  - $\hat{y} \; := \boldsymbol{x}_i\left(\boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{f}(\boldsymbol{X})\right)^{-1} \boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{y}$,
  - $\boldsymbol{f}$ is poly. trans.
  - $\boldsymbol{y}_i = g(\boldsymbol{x}_i) + \epsilon$, $\epsilon \sim N\left(0, \sigma^2\right)$.
- $\mathrm{var}[\hat{y}|\boldsymbol{x}_i] =< h(\boldsymbol{x}_i), h(\boldsymbol{x}_i) > \cdot\, \sigma^2$
  - Where $h(\boldsymbol{x}_i) := \boldsymbol{f}(\boldsymbol{x}_i)\left(\boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{f}(\boldsymbol{X})\right)^{-1} \boldsymbol{f}(\boldsymbol{X})^\top$
- We can show $\frac{1}{n}\sum_{i=1}^{n} \mathrm{var}[\hat{y}|\boldsymbol{x}_i] = \frac{m\sigma^2}{n}$
  - Now see why variance increases with b!

A Closer Look at in sample $\mathrm{var}[\hat{y}]$

- The derivation of the above formulas will be deferred to the **problem class.**

- However, a box of chocolate will be awarded to the first student who sends me the correct answer **before** the problem class.

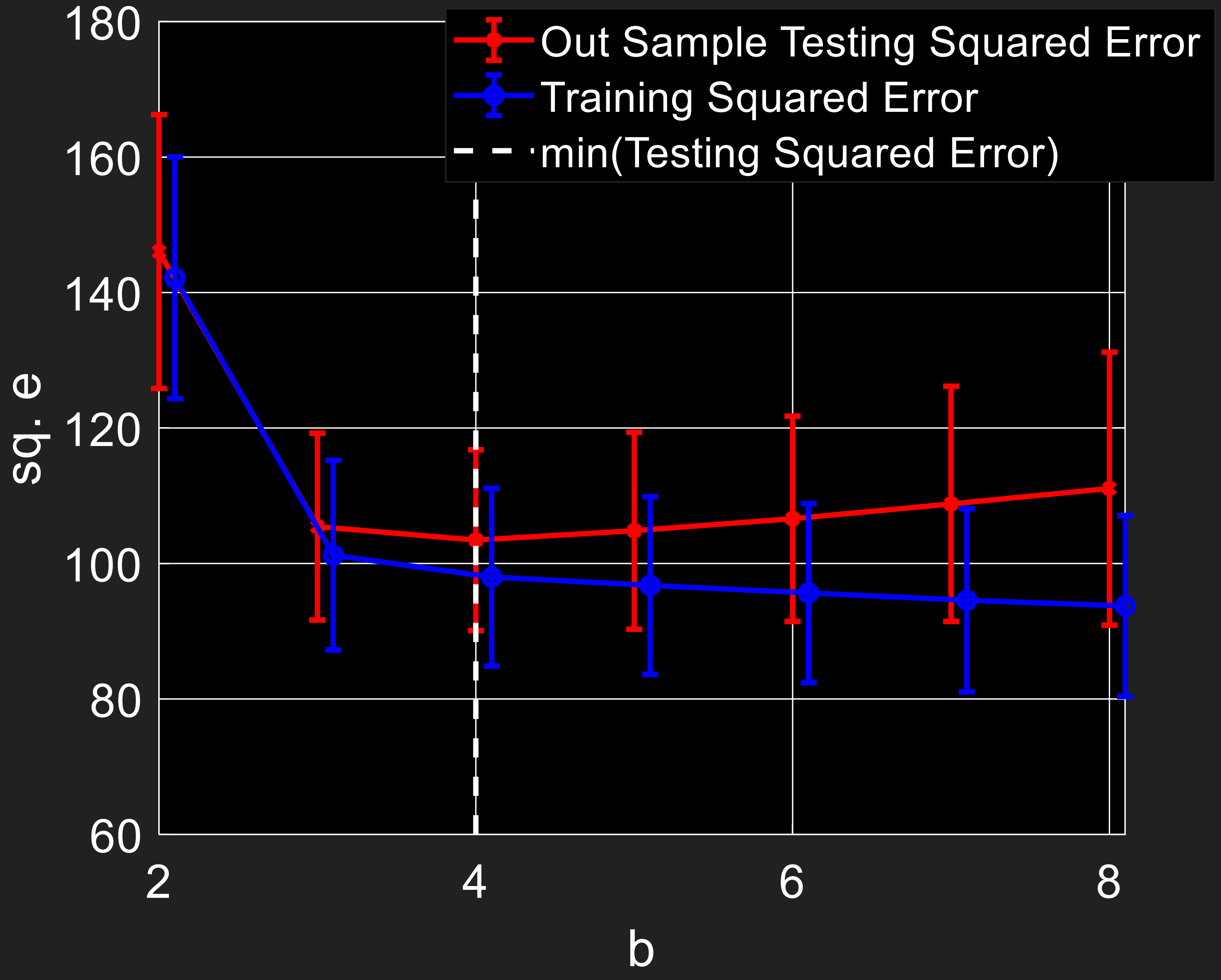
Out Sample Error

- However, we cannot construct D' as we did earlier in reality.
  - We do not know $g(x)$
- Instead, we use **out sample error**:
- $\mathbb{E}_{\boldsymbol{x}}\mathbb{E}_{\epsilon}\left[(y-\hat{y})^2|\boldsymbol{x}\right]$
  - Error over the entire distribution of $\boldsymbol{x}$
  - Requiring assumptions on the distribution of x.

Approximating Out Sample Error

- To approximate Out Sample Error:
  - Calculate $\hat{y}$ on D.
  - Get a fresh batch of observations
    - $D' := \{(y'_i, x'_i)\}_{i=1}^{n'}$
  - Calculate $\frac{1}{n'}\sum_{(y',x')\in D'}\left(y' - \widehat{y'}\right)^2$ (1)
    - $\widehat{y'} \coloneqq \boldsymbol{f}(\boldsymbol{x}')\left(\boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{f}(\boldsymbol{X})\right)^{-1}\boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{y}$
    - The average is an approx. to expectation.
- If D and D' are **independently** taken from the **same** data distribution, (1) is a good approximation of out sample error.

Out Sample Error/Training error vs. b, 100 times with error bar



Out sample error behaves similarly to in sample error!

Approximating Out Sample Error

- The approximation of out sample error using D' is usually referred as "testing error" in machine learning.
    - In contrast to the "training error" obtained using D.
- If you cannot get a fresh batch of data points, just split your dataset into D and D'!
    - Called Hold-out validation.

Conclusion

- Feature complexity affects training and testing errors **in different ways**.
- The behavior of testing error can be explained by decomposition of expected error.
- Two types of expected errors can be used for feature selection:
  - In sample error
  - Out sample error
  - Out sample error can be simply approximated using dataset split!