

Variance and Bias Decomposition and Feature Complexity

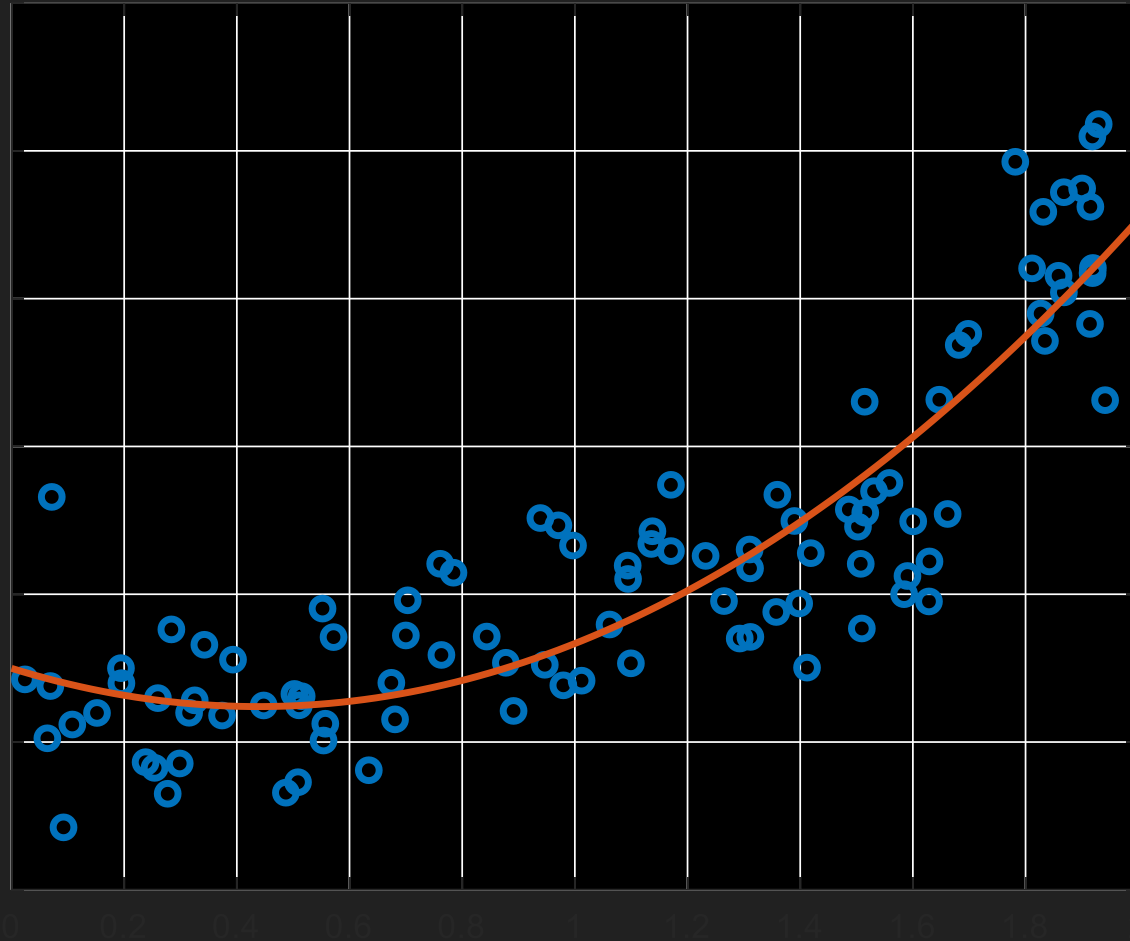
COMS21202, Part III

- 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**.

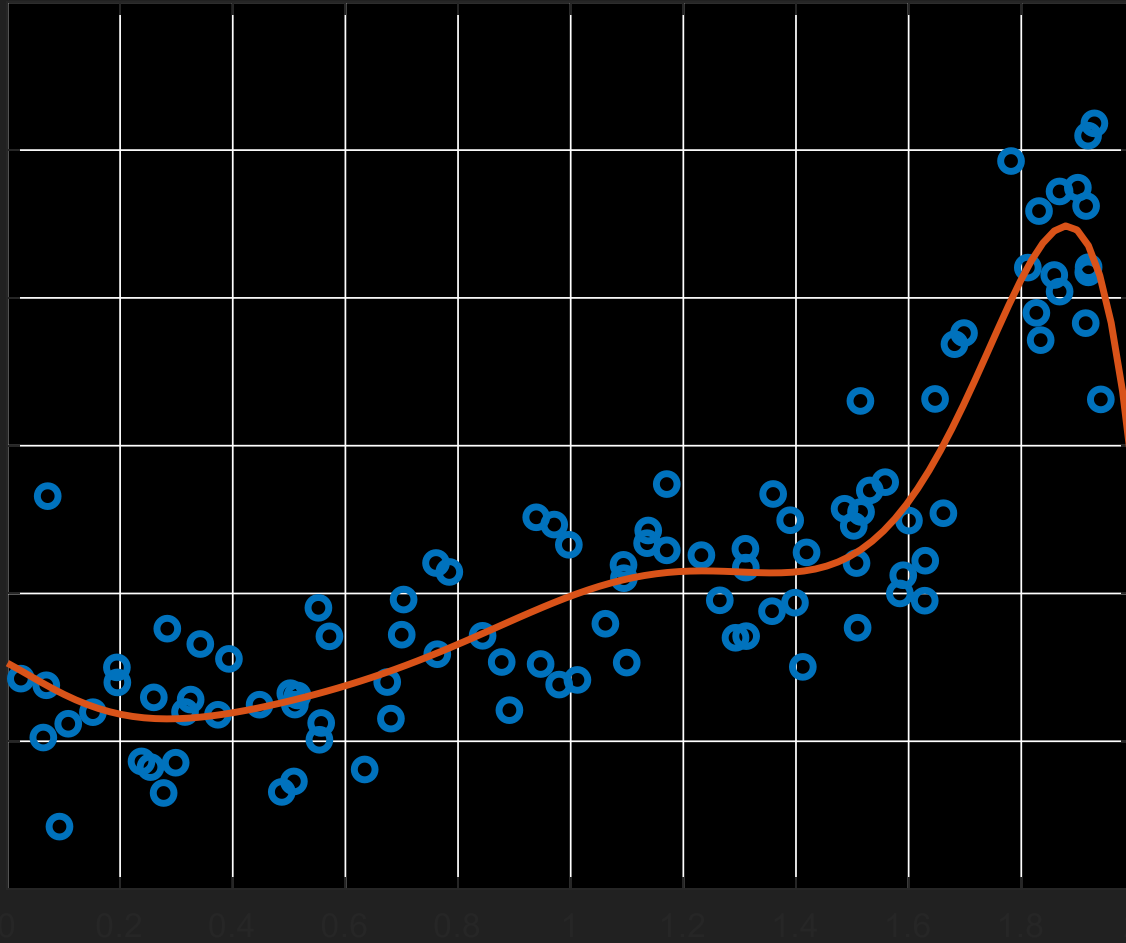
Recall: $y = \exp(1.5x - 1) + \epsilon, \epsilon \sim N(0,1)$



○ 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 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!

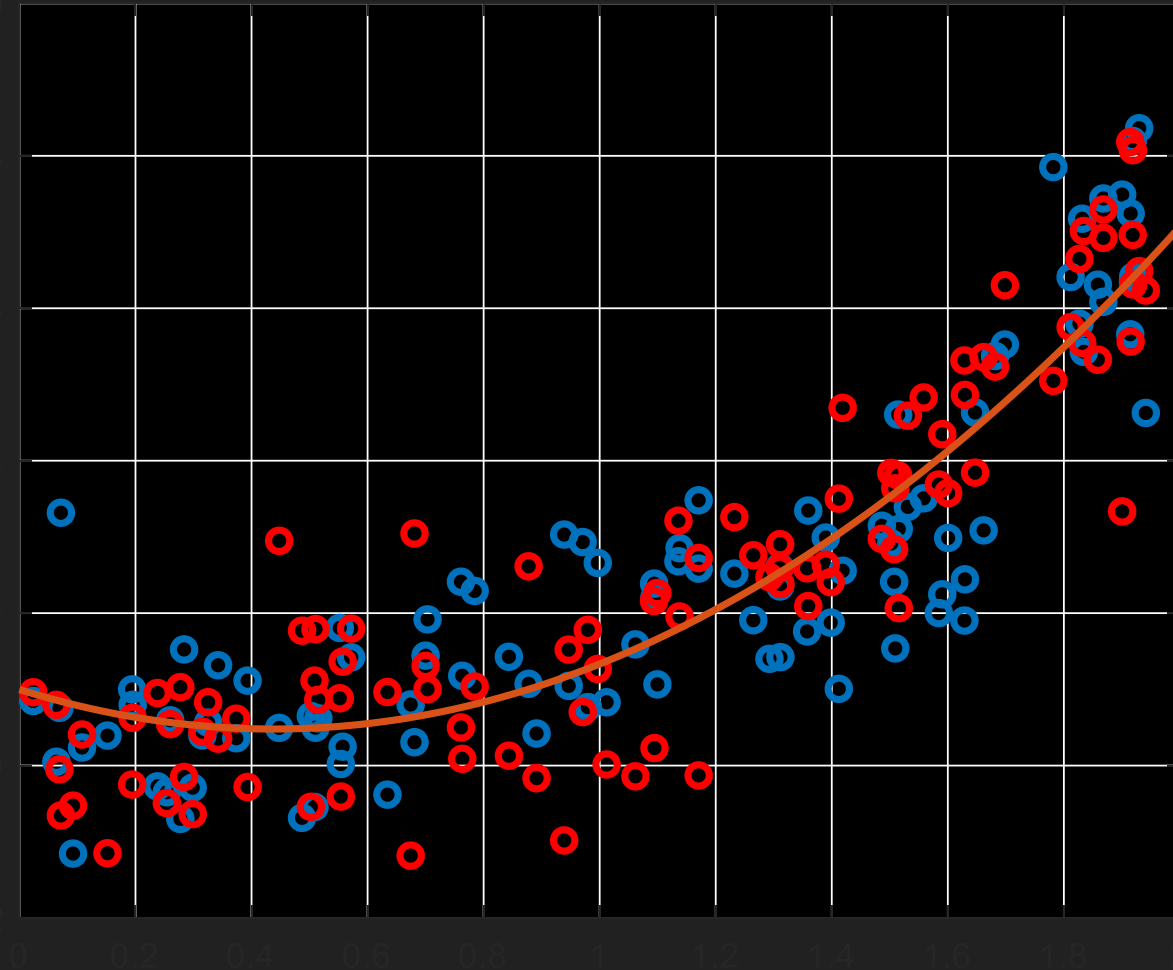
Testing Set & Testing Squared Error

- Denote $D := \{(y_i, x_i)\}_{i=1}^n$.
 - i.e., our training data.
- Now generate a **new** dataset D' :
- $\forall x_i \in X, y'_i = \underbrace{\exp(1.5x_i - 1)}_{g(x)} + \epsilon'$,
 - $\epsilon' \sim N(0,1)$ $g(x)$, “real function”
 - 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: $\sum_{i=1}^n (y'_i - \hat{y}_i)^2$
- 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: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$

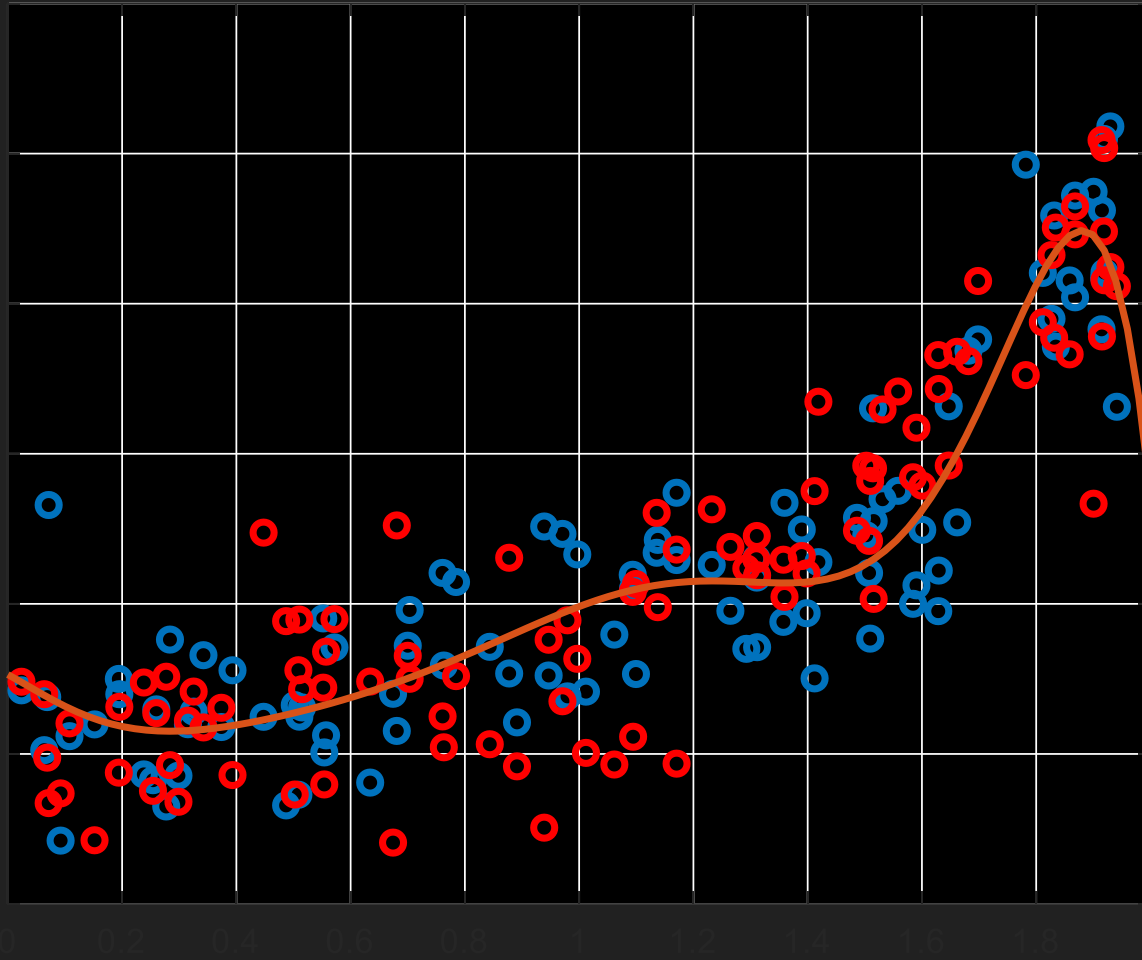


Red dots is testing set.

Polynomial transform with $b = 2$.

Testing error: 99.025

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

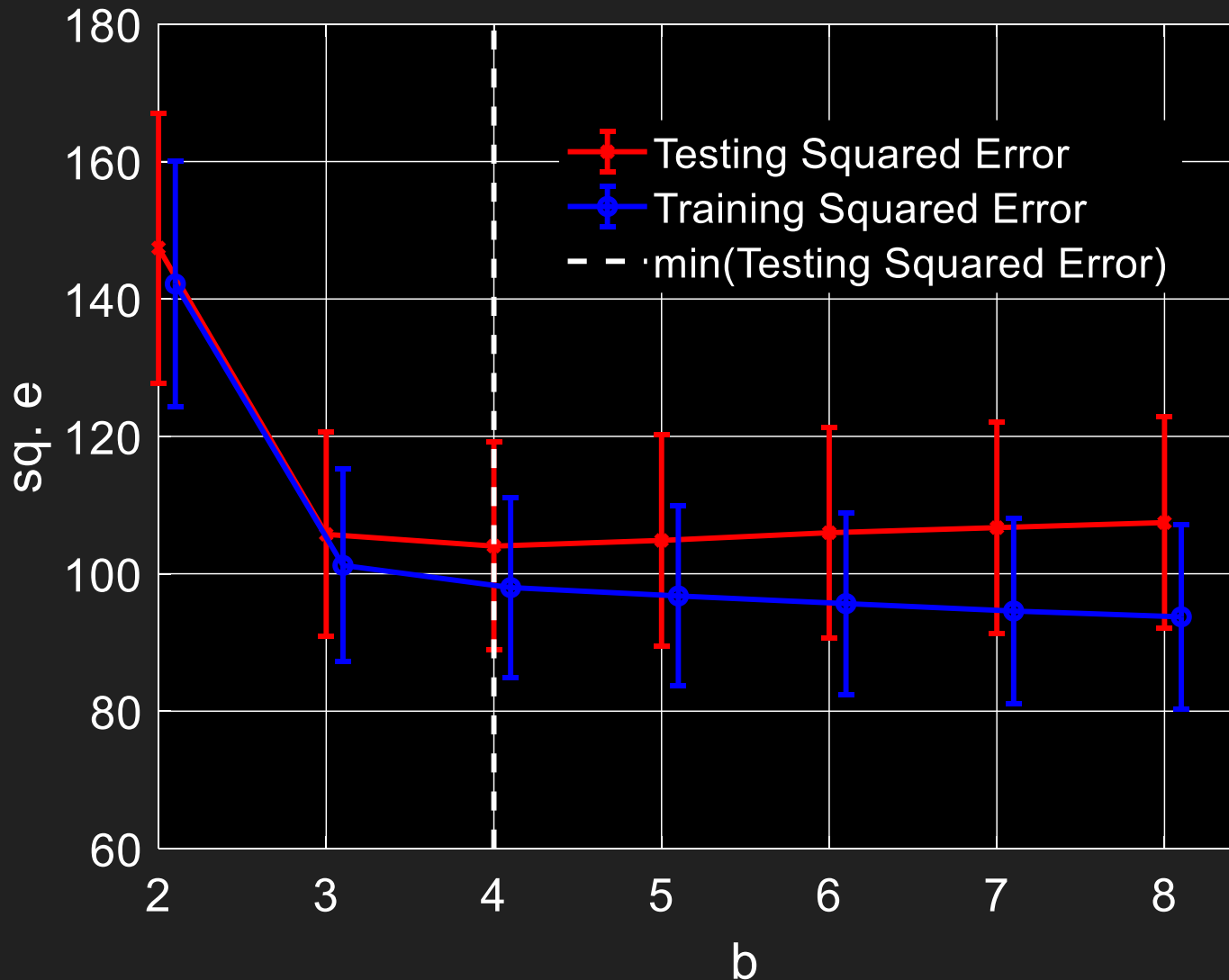


Red dots is testing set.

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?

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 looking 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: $\mathbb{E}_{\epsilon}[(y - \hat{y})^2 | \mathbf{x}_i]$
 - suppose y is generated by $y = g(x) + \epsilon$ (like in the previous case), we can rewrite:
 - $\mathbb{E}_{\epsilon}[(y - \hat{y})^2 | \mathbf{x}_i] = \mathbb{E}_{\epsilon}[(g(\mathbf{x}_i) + \epsilon - \hat{y})^2 | \mathbf{x}_i]$
 - PC: write down the formula using integral.

Expected Square Error Decomposition

- Bias and Variance Decomposition:

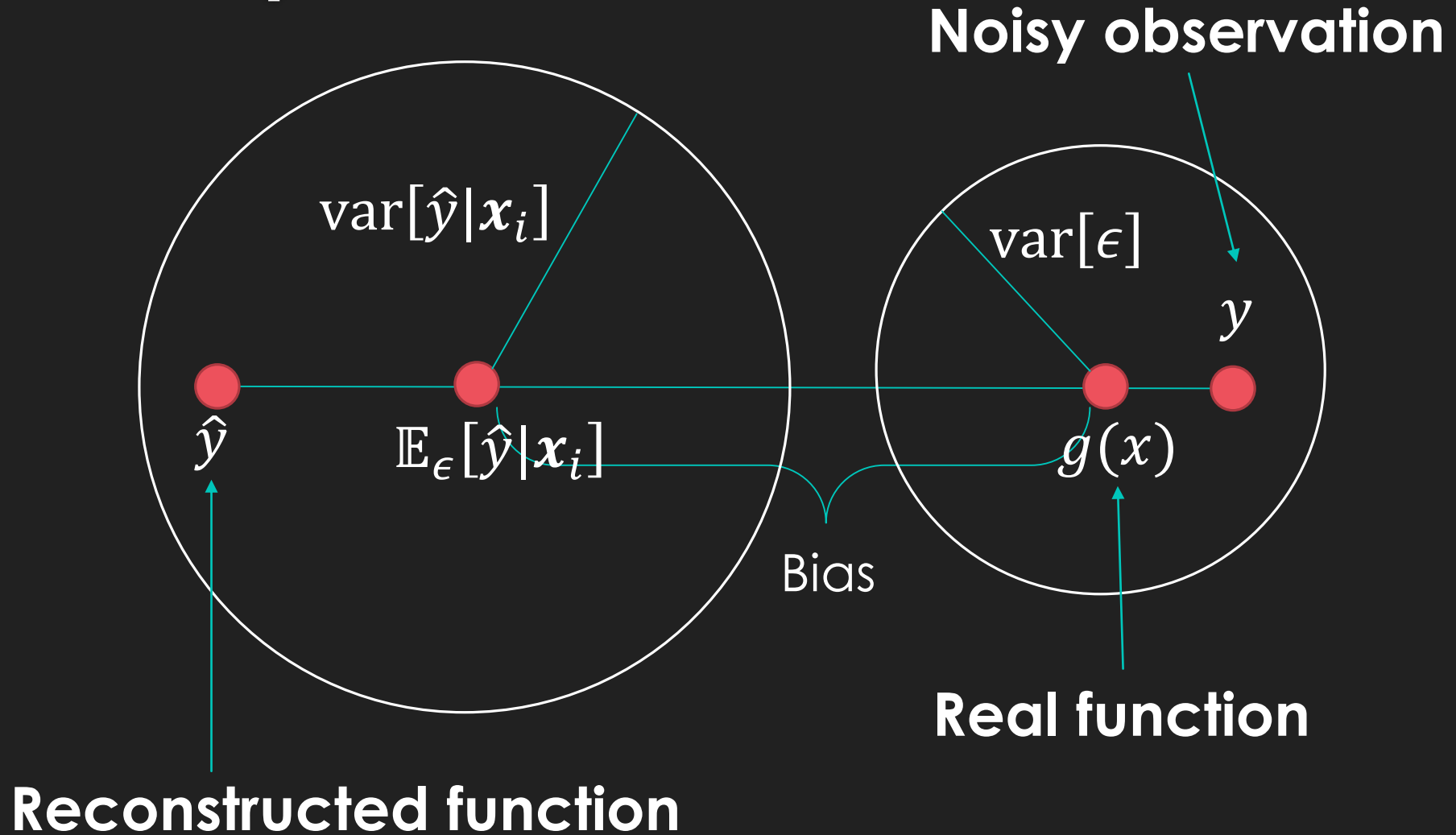
$$\begin{array}{c} \mathbb{E}_{\epsilon}[(y - \hat{y})^2 | \mathbf{x}_i] \\ = \underbrace{\text{var}[\epsilon]}_{\text{Irreducible error}} + \underbrace{\left[g(x) - \mathbb{E}_{\epsilon}[\hat{y} | \mathbf{x}_i]\right]^2}_{\text{bias}} + \underbrace{\text{var}[\hat{y} | \mathbf{x}_i]}_{\text{variance}} \end{array}$$

- “Variance and Bias decomposition”
- Live demonstration

Expected Square Error Decomposition

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



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

Variance and Bias Tradeoff

- $\text{var}[\epsilon] + [g(\mathbf{x}_i) - \mathbb{E}_\epsilon[\hat{y}|\mathbf{x}_i]]^2 + \text{var}[\hat{y}|\mathbf{x}_i]$
 - As we increase b , \hat{y} becomes more **complex** and can adapt to more complex underlying function, thus 2nd term **keeps dropping**.
 - As we 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**.

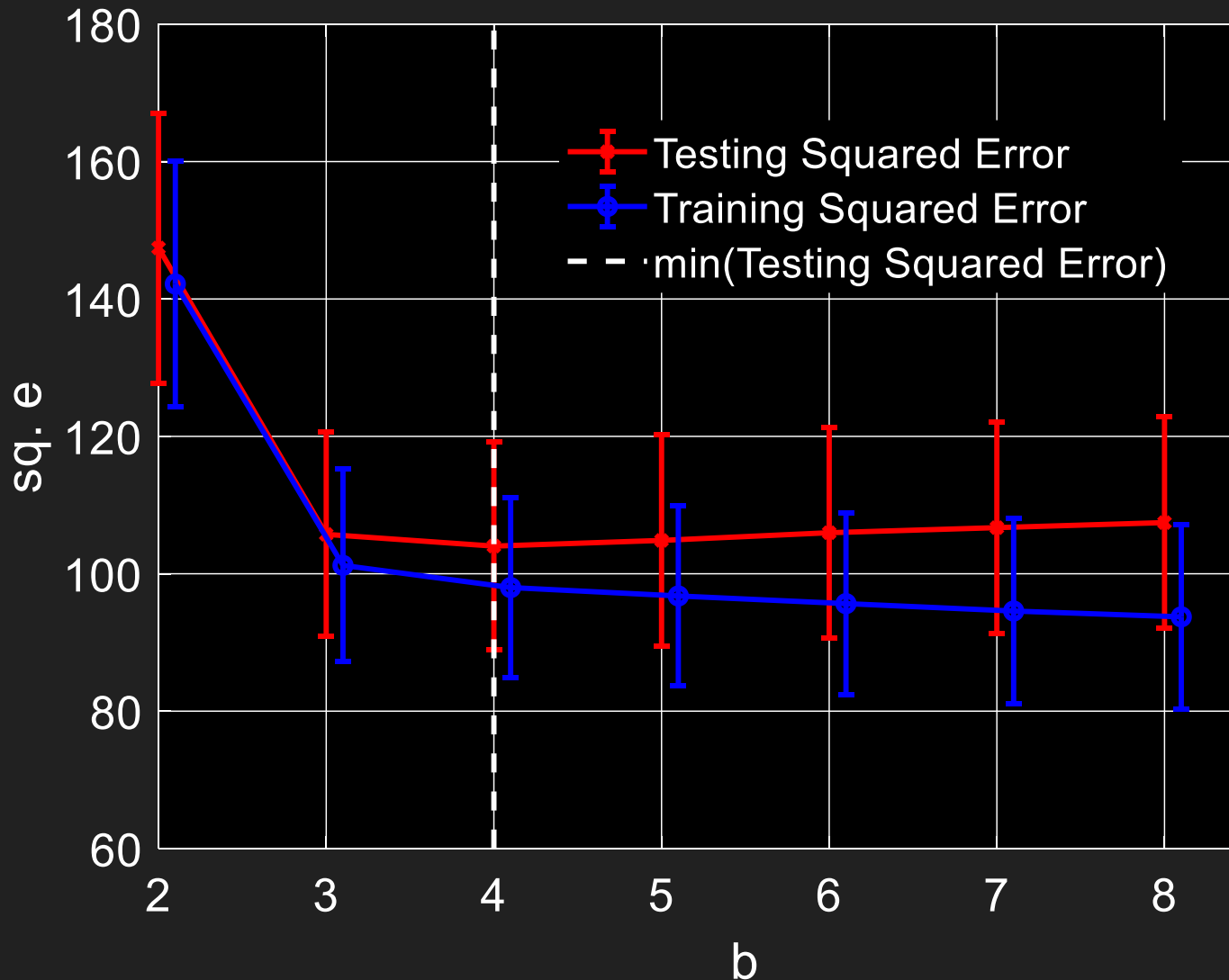
In Sample Error

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

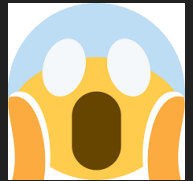
Testing/Training error vs. b , 100 times with error bar



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

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

○ Plug in **LS solution** of \hat{y} in $\text{var}[\hat{y}|\mathbf{x}_i]$:



○ $\hat{y} := \mathbf{x}_i (\mathbf{f}(\mathbf{X})^\top \mathbf{f}(\mathbf{X}))^{-1} \mathbf{f}(\mathbf{X})^\top \mathbf{y},$

○ \mathbf{f} is poly. trans.

○ $\mathbf{y}_i = g(\mathbf{x}_i) + \epsilon, \epsilon \sim N(0, \sigma^2).$

○ $\text{var}[\hat{y}|\mathbf{x}_i] = \langle h(\mathbf{x}_i), h(\mathbf{x}_i) \rangle \cdot \sigma^2$

○ Where $h(\mathbf{x}_i) := \mathbf{f}(\mathbf{x}_i) (\mathbf{f}(\mathbf{X})^\top \mathbf{f}(\mathbf{X}))^{-1} \mathbf{f}(\mathbf{X})^\top$

○ We can show $\frac{1}{n} \sum_{i=1}^n \text{var}[\hat{y}|\mathbf{x}_i] = \frac{m\sigma^2}{n}$

○ Now see why variance increases with b !

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

- 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.

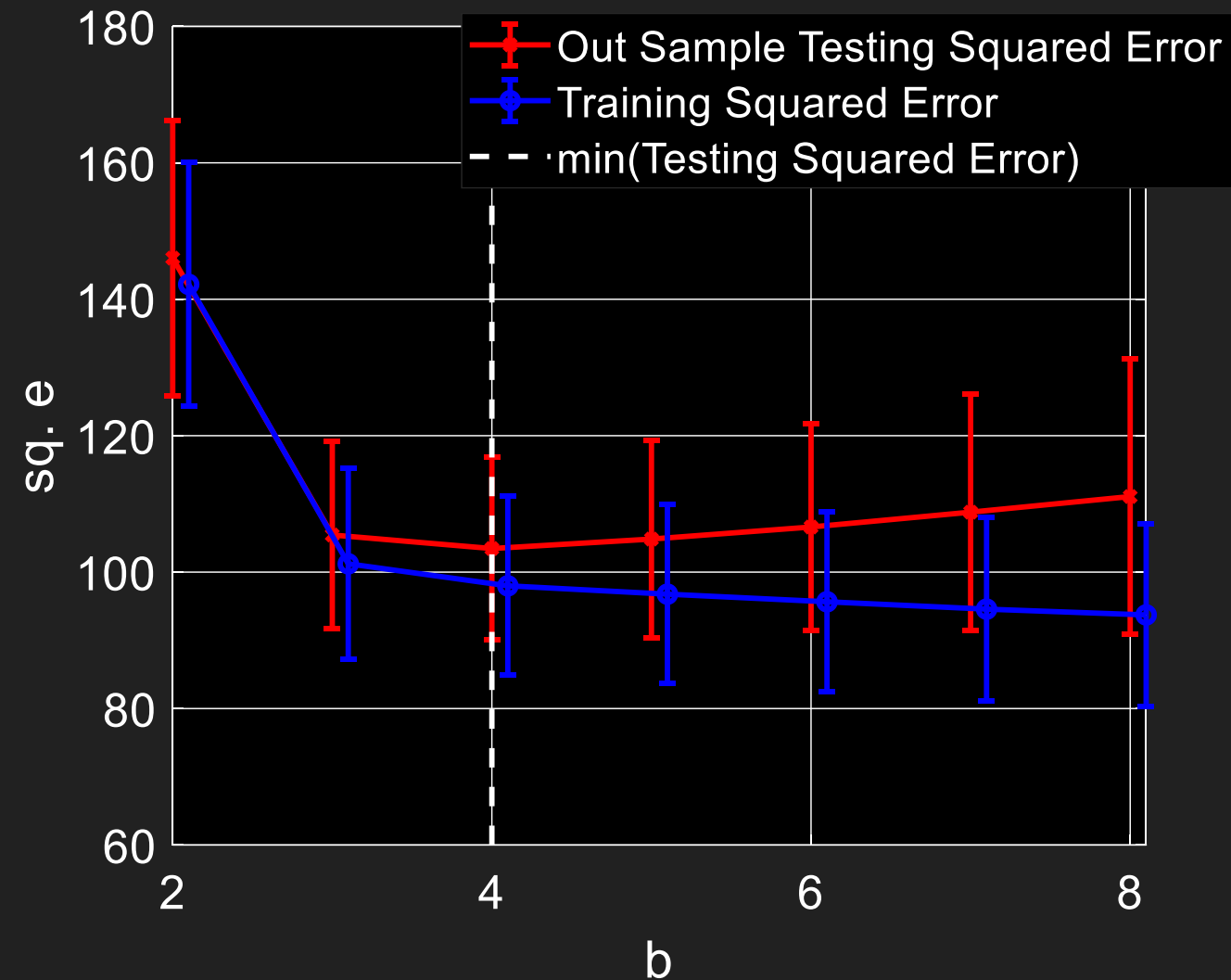
Out Sample Error

- However, we cannot construct D' as we did earlier in reality.
 - We do not know $g(x)$
- Instead, we use **out sample error**:
- $\mathbb{E}_x \mathbb{E}_\epsilon [(y - \hat{y})^2 | x]$
 - Error over the entire distribution of 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 $\frac{1}{n'} \sum_{(y', x') \in D'} (y' - \hat{y}')^2$ (1)
 - $\hat{y}' := f(x')(f(X)^\top f(X))^{-1} f(X)^\top y$
 - 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!