EE2211 Introduction to Machine Learning

Lecture 7

Yueming Jin
ymjin@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

Acknowledgement: EE2211 development team
Thomas, Helen, Xinchao, Kar-Ann, Chen Khong, Robby and Haizhou

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Probability and Statistics
- Fundamental Machine Learning Algorithms I (Yueming)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Yueming)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

Fundamental ML Algorithms: Linear Regression

References for Lectures 7-9:

Main

- [Book1] Andriy Burkov, "**The Hundred-Page Machine Learning Book**", 2019. (**read first, buy later**: http://themlbook.com/wiki/doku.php)
  - Chapters 3, 4, 5, & 7.
- [Book2] Andreas C. Muller and Sarah Guido, "**Introduction to Machine Learning with Python**: A Guide for Data Scientists", O'Reilly Media, Inc., 2017

Supplementary

- [Book3] Jeff Leek, "**The Elements of Data Analytic Style**: A guide for people who want to analyze data", Lean Publishing, 2015.
- [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (available online) http://vmls-book.stanford.edu/
- [Ref 5] **Professor Vincent Tan's notes (chapters 7-9): (useful) https://vyftan.github.io/papers/ee2211book.pdf**

Fundamental ML Algorithms: Overfitting, Bias-Variance Tradeoff

Module III Contents

- Overfitting, underfitting & model complexity
- Feature selection & Regularization
- Bias-variance trade-off
- Loss function
- Optimization
- Gradient descent
- Decision trees
- Random forest

Regression Review

- Goal: Given feature(s) x, we want to predict target y
  - x can be 1-D or more than 1-D
  - y is 1-D
- Two types of input data
  - Training set $\{x_i, y_i\}$, from $i = 1, \cdots, N$
  - Test set $\{x_j, y_j\}$, from $j = 1, \cdots, M$
- Learning/Training
  - Training set used to estimate regression coefficients $\hat{w}$
- Prediction/Testing/Evaluation
  - Prediction performed on test set to evaluate performance

Regression Review: Linear Case

- x is 1D & y is 1-D
- Linear relationship between x & y
- Illustration (4 training samples):

$$\mathbf{X}_{train} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{pmatrix} \qquad \mathbf{y}_{train} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

- Training/Learning (primal) on training set

$$\hat{\mathbf{w}} = (\mathbf{X}_{train}^T \mathbf{X}_{train})^{-1} \mathbf{X}_{train}^T \mathbf{y}_{train}$$

- Prediction/Testing/Evaluation on test set

$$\hat{\mathbf{y}}_{test} = \mathbf{X}_{test} \hat{\mathbf{w}}$$

Regression Review: Polynomial

- x is 1-D (or more than 1-D) & y is 1-D
- Polynomial relationship between x & y
- Quadratic illustration (4 training samples, x is 1-D):

$$\mathbf{X}_{train} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{pmatrix} \qquad \mathbf{y}_{train} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

- Training/Learning (primal) on training set

$$\hat{\mathbf{w}} = (\mathbf{X}_{train}^T \mathbf{X}_{train})^{-1} \mathbf{X}_{train}^T \mathbf{y}_{train}$$

- Prediction/Testing/Evaluation on test set

$$\hat{\mathbf{y}}_{test} = \mathbf{X}_{test} \hat{\mathbf{w}}$$

Regression Review: Polynomial

- x is 1-D (or more than 1-D) & y is 1-D
- Polynomial relationship between x & y
- Quadratic illustration (4 training samples, x is 1-D):

$$\underline{\mathbf{P}_{train}} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{pmatrix} \qquad \mathbf{y}_{train} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

- Training/Learning (primal) on training set

$$\hat{\mathbf{w}} = \underline{(\mathbf{P}_{train}^T \mathbf{P}_{train})^{-1} \mathbf{P}_{train}^T} \mathbf{y}_{train}$$

- Prediction/Testing/Evaluation on test set

$$\hat{\mathbf{y}}_{test} = \underline{\mathbf{P}_{test}} \hat{\mathbf{w}}$$

Note on Training & Test Sets

- Linear is special case of polynomial => use "**P**" instead of "**X**" from now on
- Training/Learning (primal) on training set

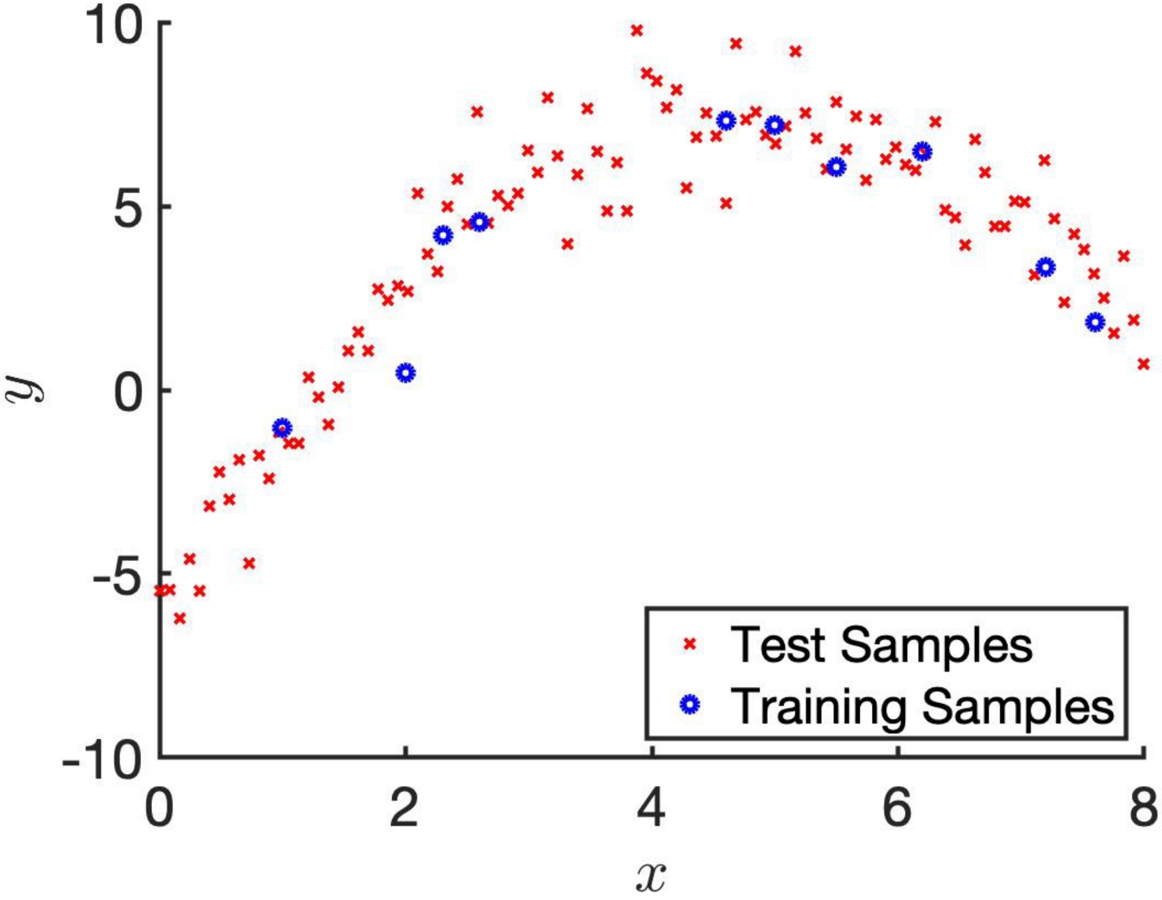
$$\hat{\mathbf{w}} = (\mathbf{P}_{train}^T \mathbf{P}_{train})^{-1} \mathbf{P}_{train}^T \mathbf{y}_{train}$$

- Prediction/Testing/Evaluation on test set

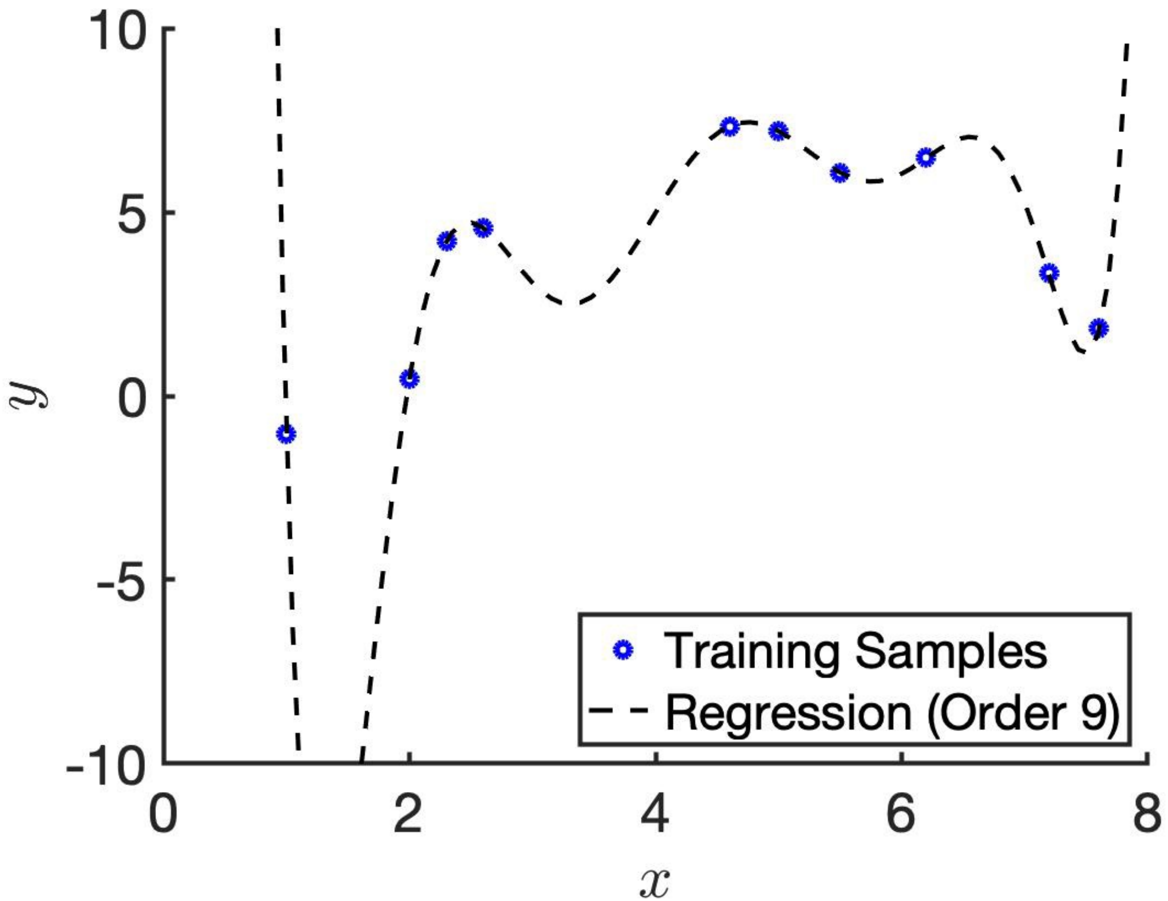
$$\hat{\mathbf{y}}_{test} = \mathbf{P}_{test} \hat{\mathbf{w}}$$

- There should be **zero** overlap between training & test sets
- Important goal of regression: prediction on **new unseen data**, i.e., test set
- Why is test set important for evaluation?

Overfitting Example

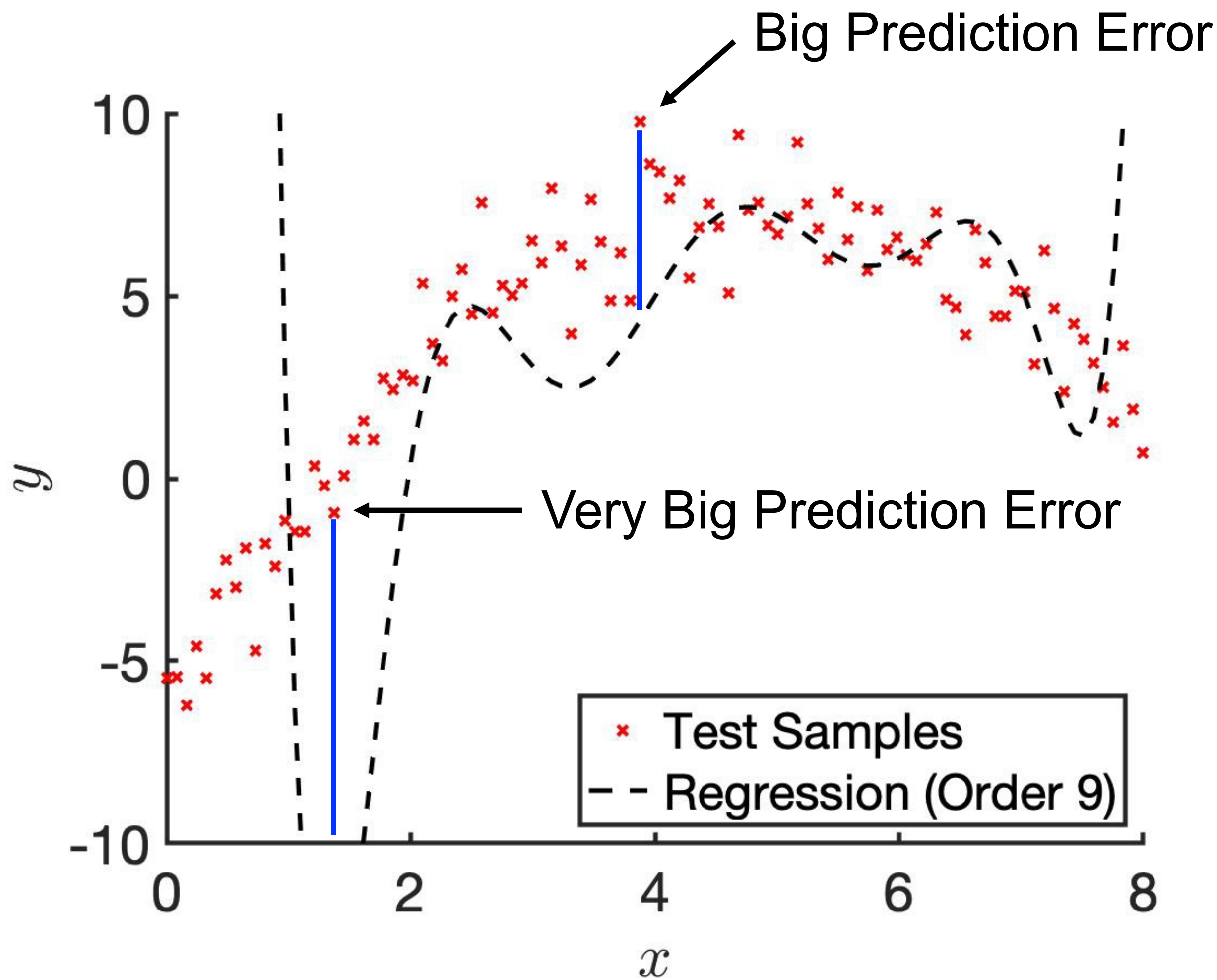


Overfitting Example



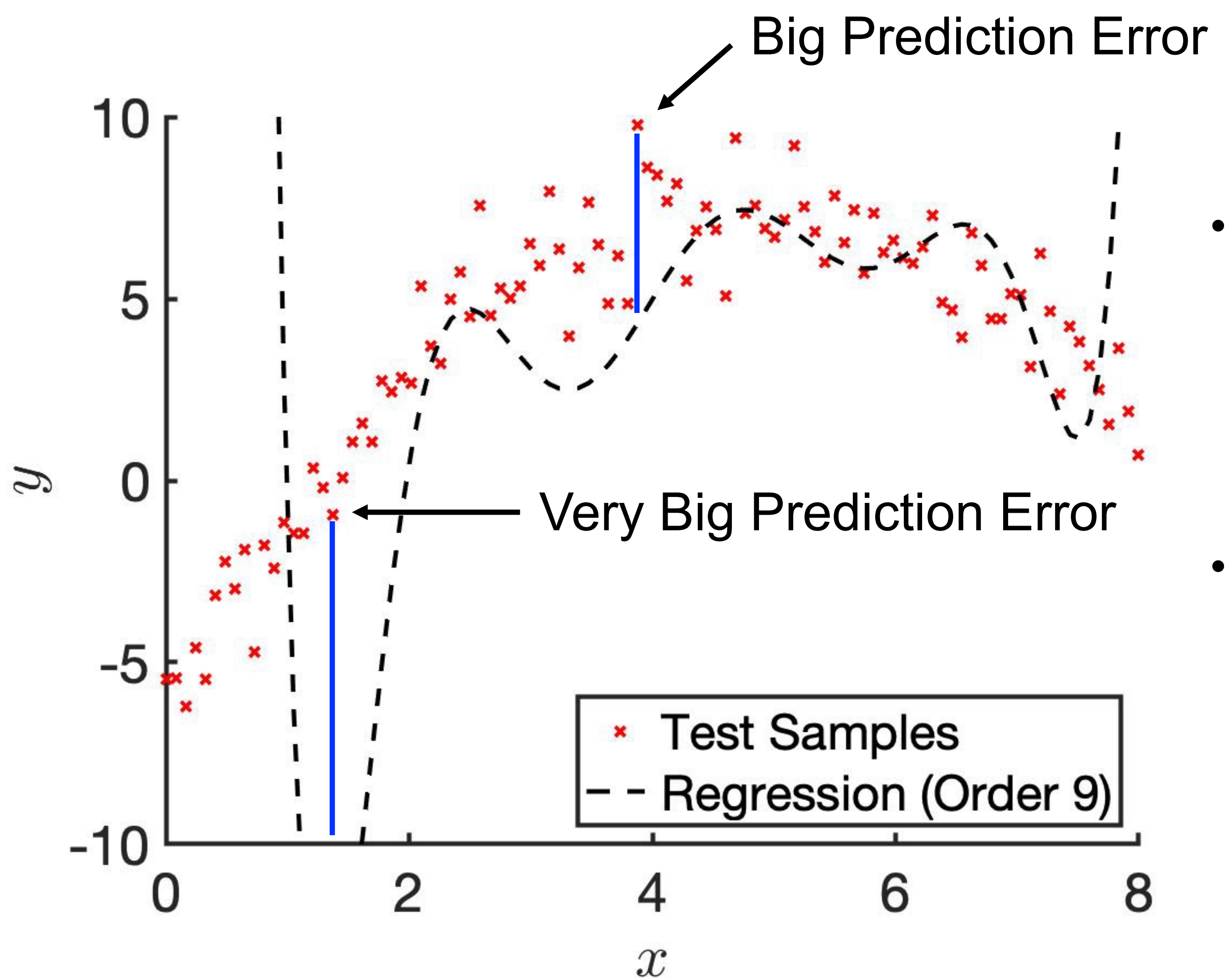
	Training Set Fit	
Order 9	Good	

Overfitting Example



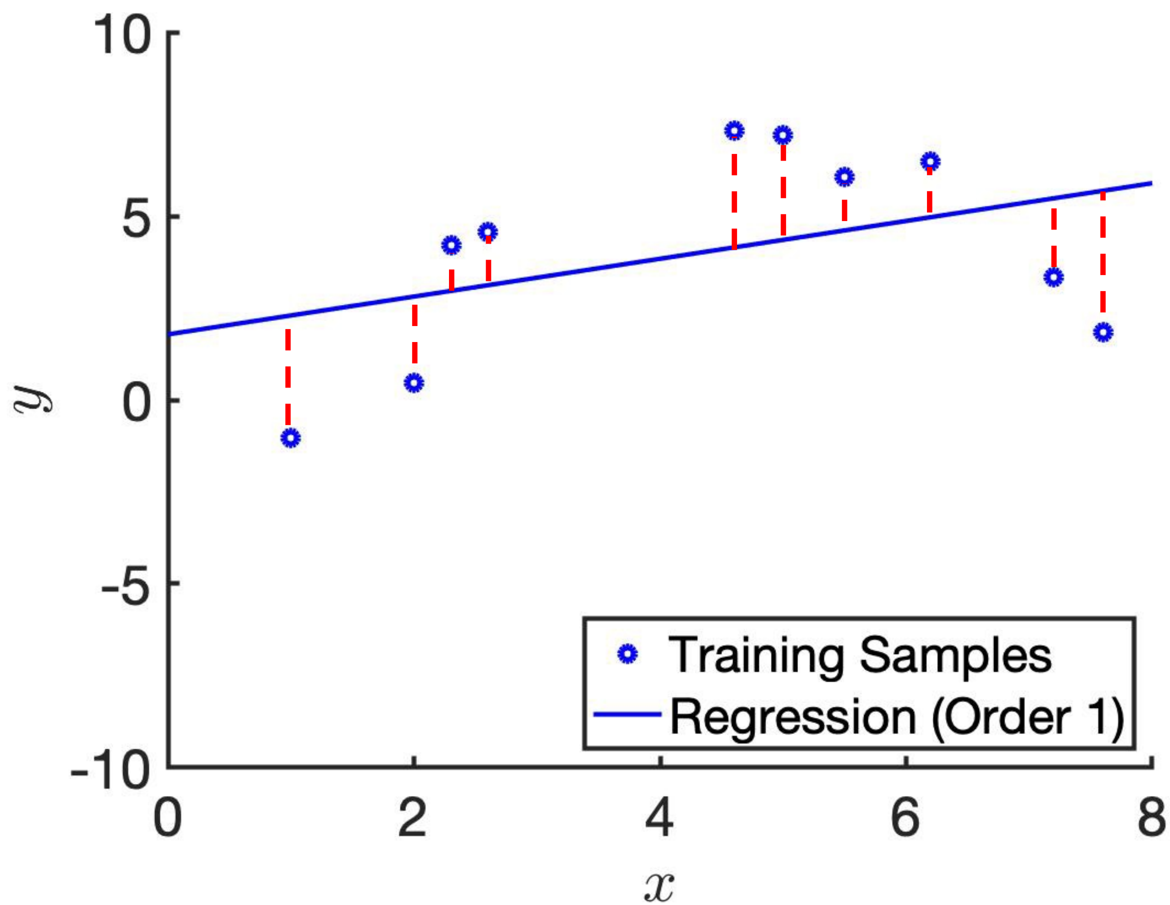
	Training Set Fit	Test Set Fit
Order 9	Good	Bad

Overfitting Example



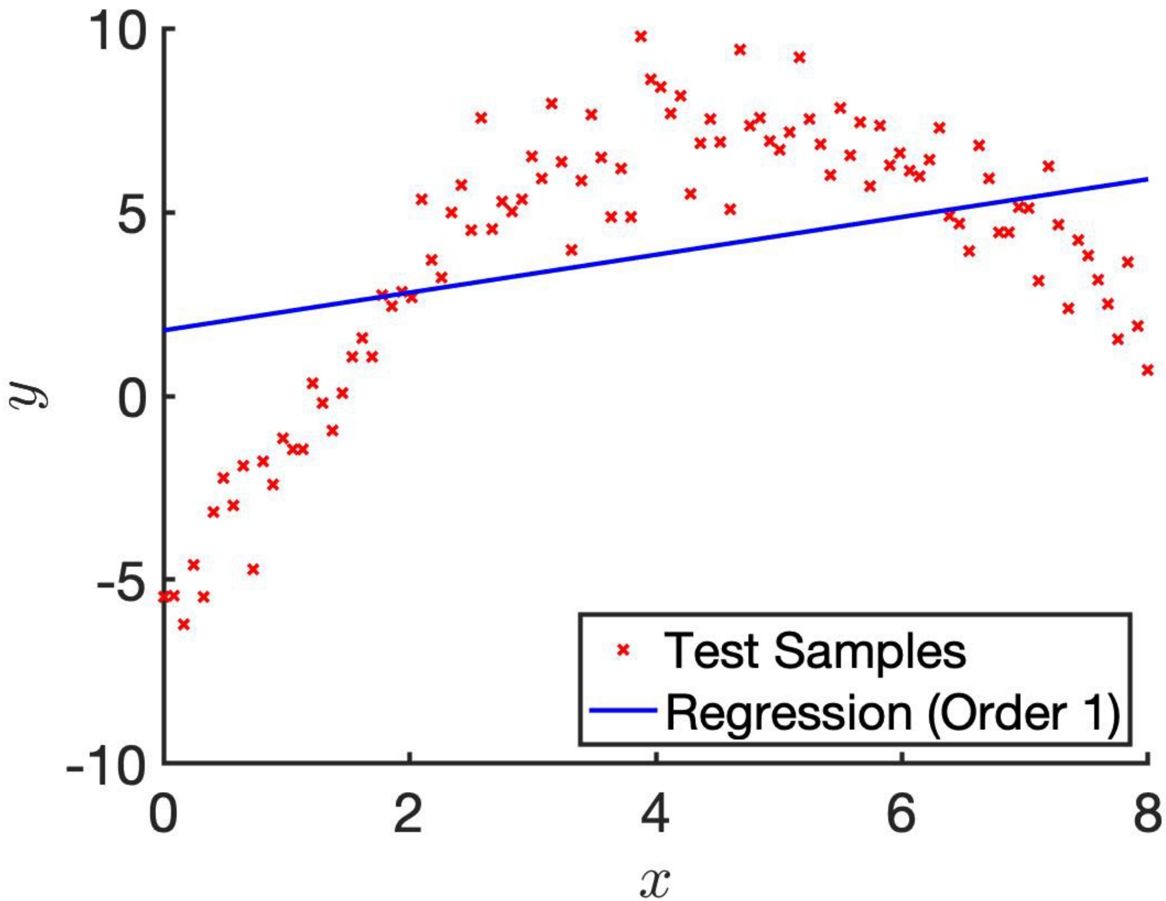
- If we take one of the blue lines and compute the square of its length, this is called **"squared error" for that particular data point**
- If we average squared errors across all the red crosses, it's called **mean squared error (MSE)** in the test set

Underfitting Example



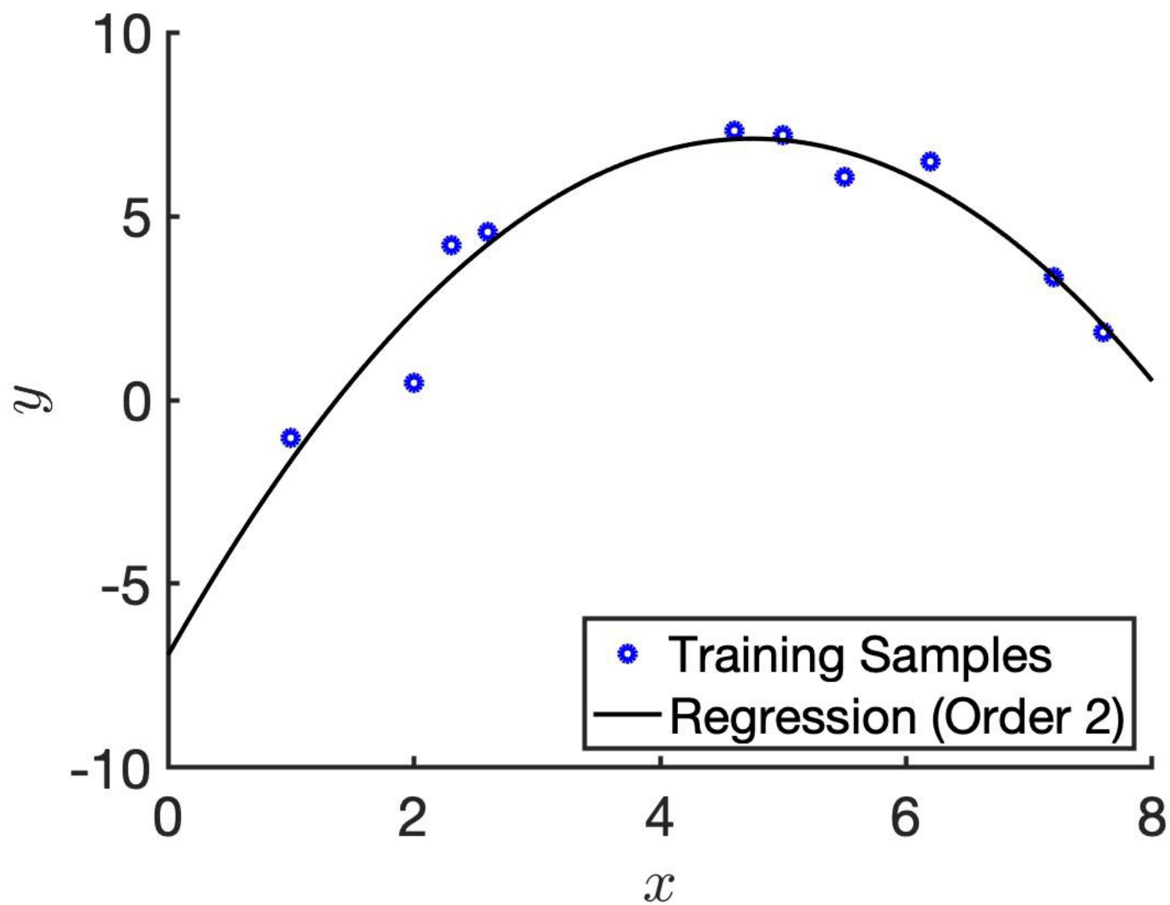
	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 1	Bad	

Underfitting Example



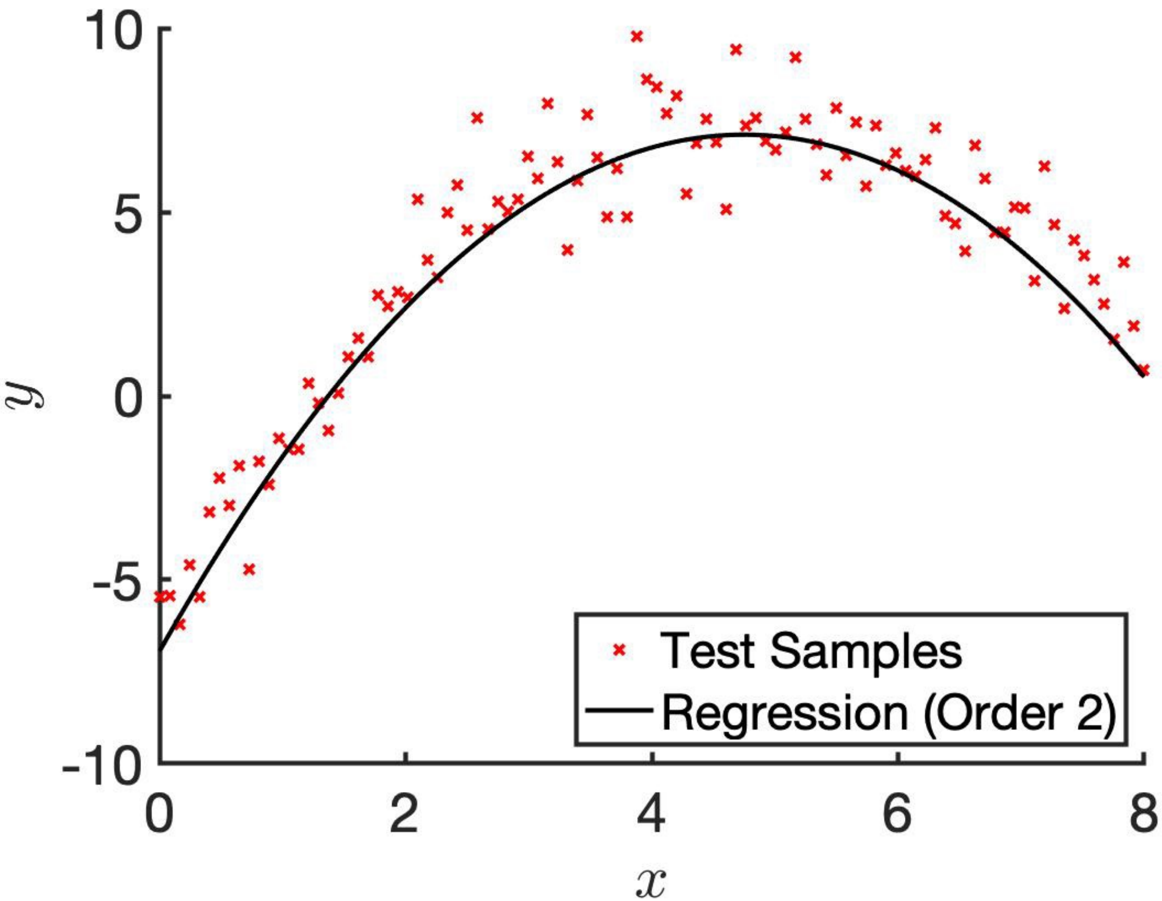
	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 1	Bad	Bad

"Just Nice"



	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 1	Bad	Bad
Order 2	Good	

"Just Nice"



	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 1	Bad	Bad
Order 2	Good	Good

Overfitting & Underfitting

Overfitting → Order 9
Underfitting → Order 1

	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 1	Bad	Bad
Order 2	Good	Good

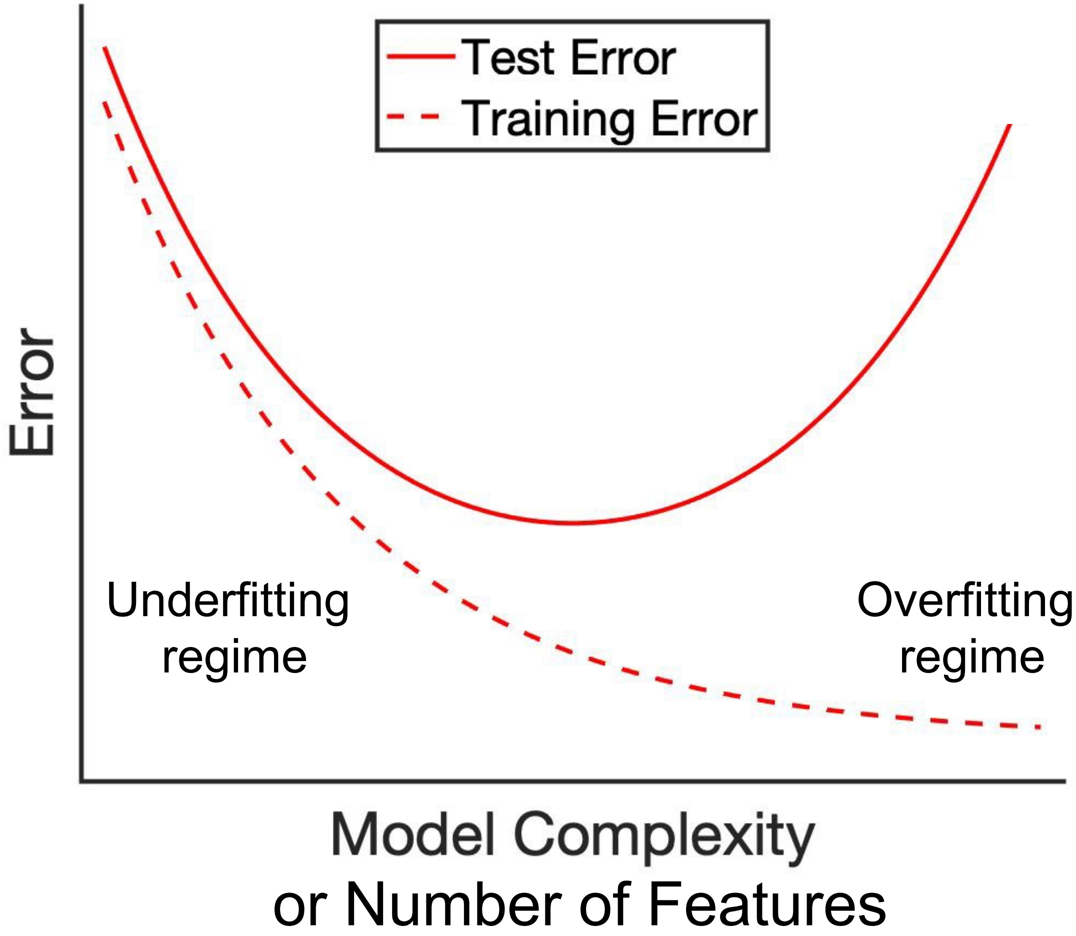
Overfitting & Underfitting

- **Overfitting** occurs when model predicts the training data well, but predicts new data (e.g., from test set) poorly
- **Reason 1**
  – Model is too complex for the data
  – Previous example: Fit order 9 polynomial to 10 data points
- **Reason 2**
  – Too many features but number of training samples too small
  – Even linear model can overfit, e.g., linear model with 9 input features (i.e., w is 10-D) and 10 data points in training set => data might not be enough to estimate 10 unknowns <u>well</u>
- **Solutions**
  – Use **simpler models** (e.g., lower order polynomial)
  – Use **regularization** (see next part of lecture)

Overfitting & Underfitting

- **Underfitting** is the inability of trained model to predict the targets in the training set
- **Reason 1**
  - Model is too simple for the data
  - Previous example: Fit order 1 polynomial to 10 data points that came from an order 2 polynomial
  - **Solution**: Try more complex model
- **Reason 2**
  - Features are not informative enough
  - **Solution**: Try to develop more informative features

Overfitting / Underfitting Schematic

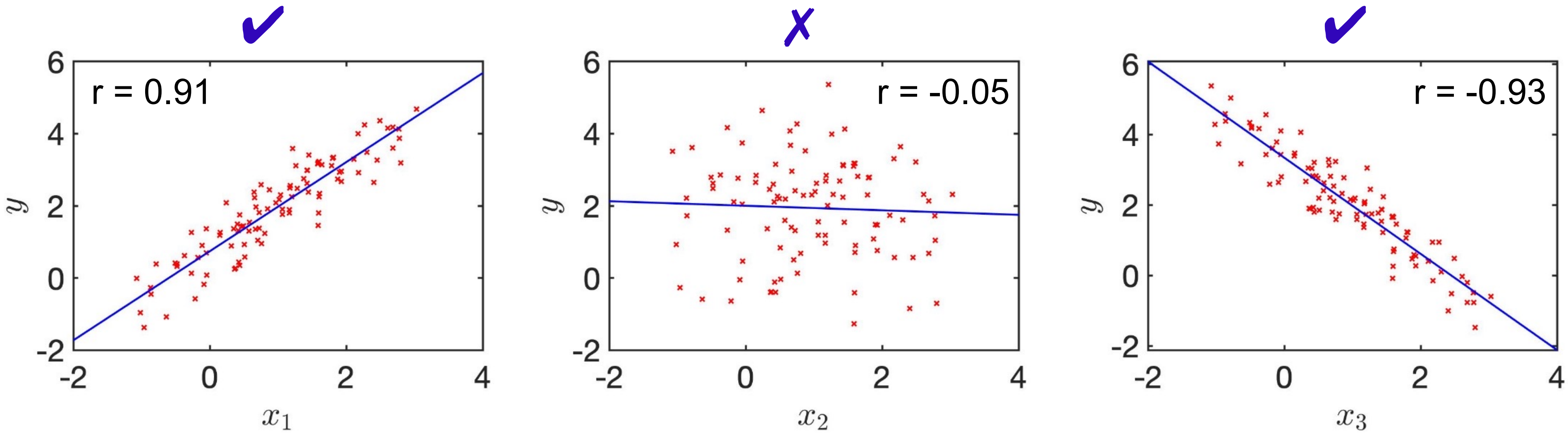


Feature Selection

- Less features might reduce overfitting
  - Want to discard useless features & keep good features, so perform feature selection
- Feature selection procedure
  - Step 1: feature selection in training set
  - Step 2: fit model using selected features in training set
  - Step 3: evaluate trained model using test set
- Very common mistake
  - Feature selection with test set (or full dataset) leads to inflated performance
  - Do not perform feature selection with test data

Selecting Features With Pearson's r

- Given features x, we want to predict target y
- Assume x & y both continuous
- Compute Pearson's correlation coefficient between each feature & target y in the training set
  - Pearson's correlation r measures linear relationship between two variables



Selecting Features With Pearson's r

- Given features x, we want to predict target y
- Assume x & y both continuous
- Compute Pearson's correlation coefficient between each feature & target y in the training set
  – Pearson's correlation r measures linear relationship between two variables
- Two options
  – Option 1: Pick K features with largest absolute correlations
  – Option 2: Pick all features with absolute correlations > C
  – K & C are "magic" numbers set by the ML practitioner
- Other metrics besides Pearson's correlation are possible

Regularization

- Regularization is an umbrella term that includes methods forcing learning algorithm to build less complex models.
- Motivation 1: Solve an ill-posed problem
  - For example, estimate 10th order polynomial with just 5 datapoints
- Motivation 2: Reduce overfitting
- For example, in previous lecture, we added $\lambda \mathbf{w}^T \mathbf{w}$:

$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{P}\mathbf{w} - \mathbf{y})^T (\mathbf{P}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

- Minimizing with respect to $\mathbf{w}$, primal solution is

$$\hat{\mathbf{w}} = (\mathbf{P}^T \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P}^T \mathbf{y}$$

- For $\lambda > 0$, matrix becomes invertible (Motivation 1)

Regularization

- Regularization is an umbrella term that includes methods forcing learning algorithm to build less complex models.
- Motivation 1: Solve an ill-posed problem
  - For example, estimate 10th order polynomial with just 5 datapoints
- Motivation 2: Reduce overfitting
- For example, in previous lecture, we added $\lambda \mathbf{w}^T \mathbf{w}$:

$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{P}\mathbf{w} - \mathbf{y})^T (\mathbf{P}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

- Minimizing with respect to $\mathbf{w}$, primal solution is

$$\hat{\mathbf{w}} = (\mathbf{P}^T \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P}^T \mathbf{y}$$

- $\hat{\mathbf{w}}$ might also perform better in test set, i.e., reduces overfitting (Motivation 2) – will show example later

Regularization

- Consider minimization from previous slide

$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{P}\mathbf{w} - \mathbf{y})^T (\mathbf{P}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

Cost function quantifying data fitting error in training set

Regularization

Regularization

- Consider minimization from previous slide

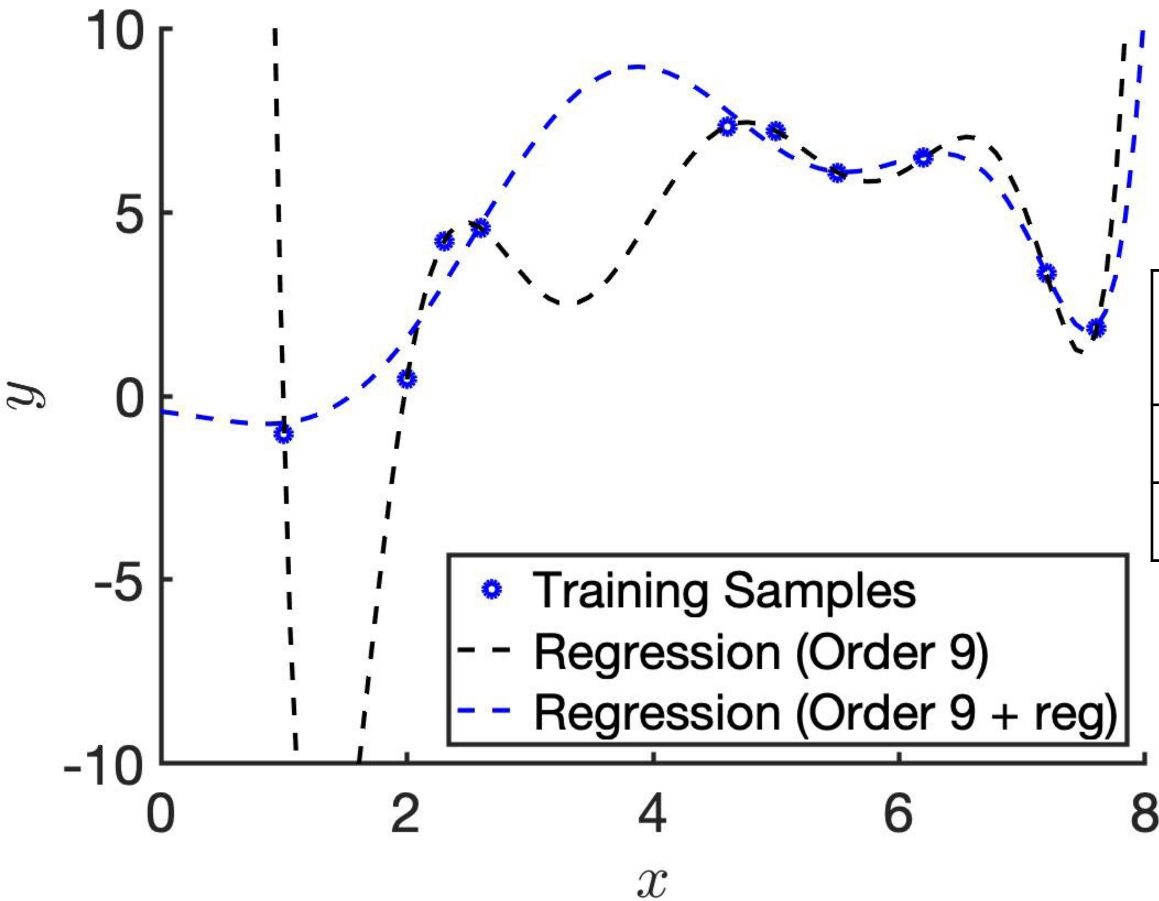
$$\underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{P}\mathbf{w} - \mathbf{y})^T (\mathbf{P}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

- $\mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \cdots + w_d^2$ L2 - Regularization
- Encourage $w_0, \cdots, w_d$ to be small (also called shrinkage or weight-decay) => constrain model complexity
- More generally, most machine learning algorithms can be formulated as the following optimization problem

$$\underset{\mathbf{w}}{\operatorname{argmin}} \, \textbf{Data-Loss}(\mathbf{w}) + \lambda \textbf{Regularization}(\mathbf{w})$$

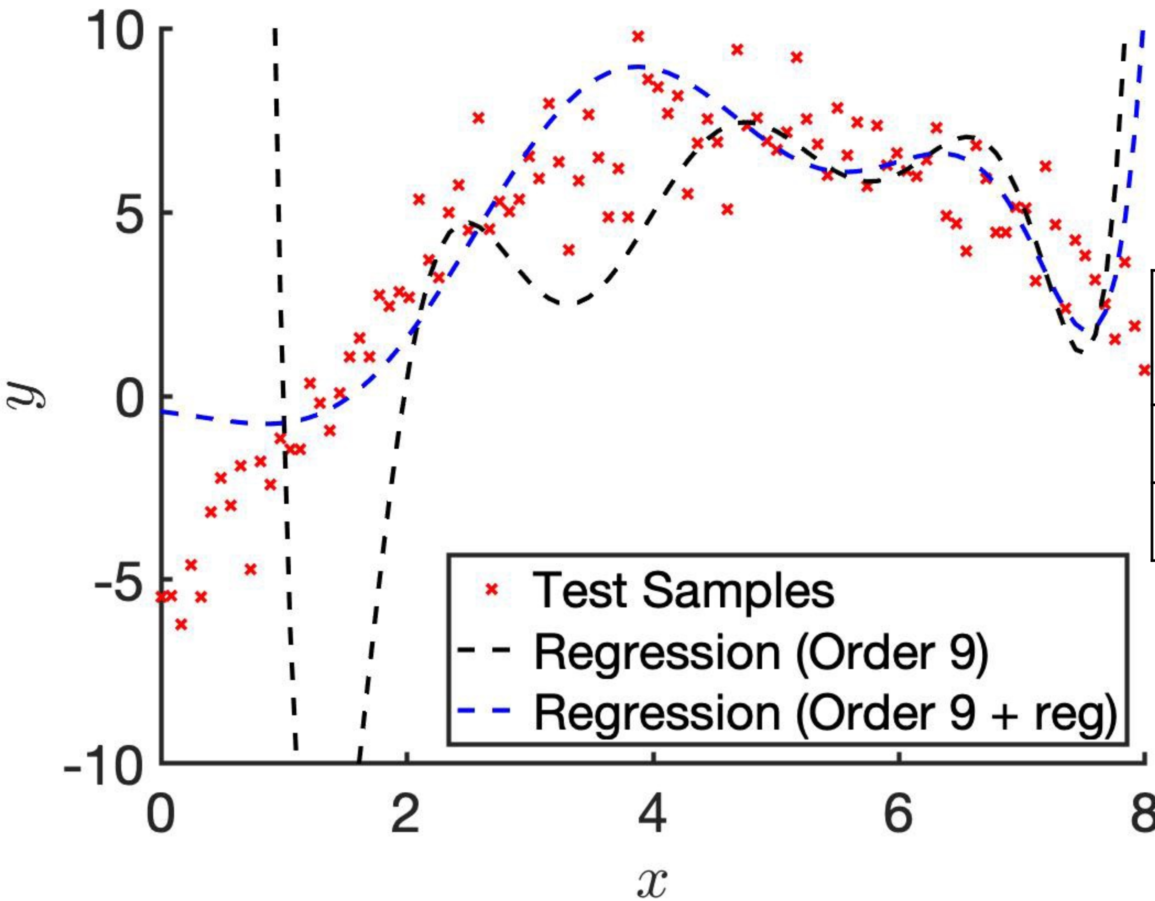
- **Data-Loss(w)** quantifies fitting error to training set given parameters **w:** smaller error => better fit to training data
- **Regularization(w)** penalizes more complex models

Regularization Example



	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 9, $\lambda = 1$	Good	

Regularization Example



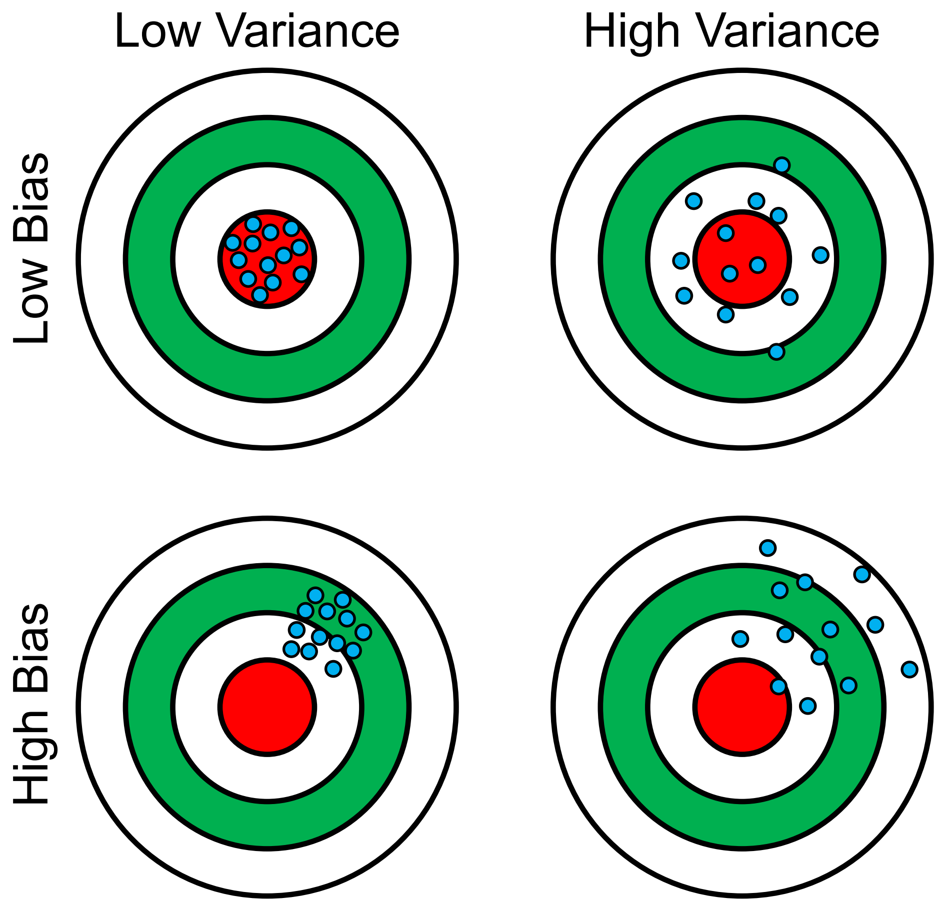
	Training Set Fit	Test Set Fit
Order 9	Good	Bad
Order 9, $\lambda = 1$	Good	Good

Python demo

Bias versus Variance

- Suppose we are trying to predict red target below:

Low Bias: blue predictions on average close to red target
Low Variance: low variability among predictions



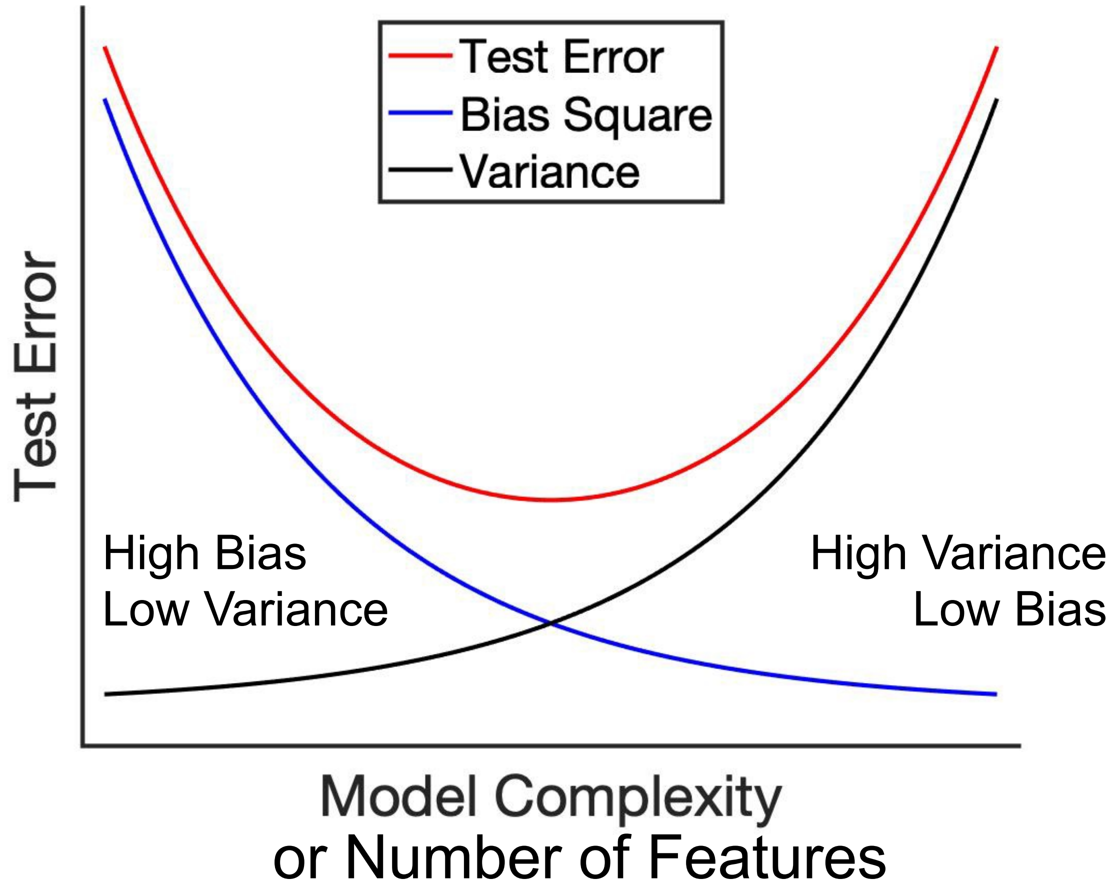
Low Bias: blue predictions on average close to red target
High Variance: large variability among predictions

High Bias: blue predictions on average not close to red target
Low Variance: Low variability among predictions

High Bias: blue predictions on average not close to red target
High Variance: high variability among predictions

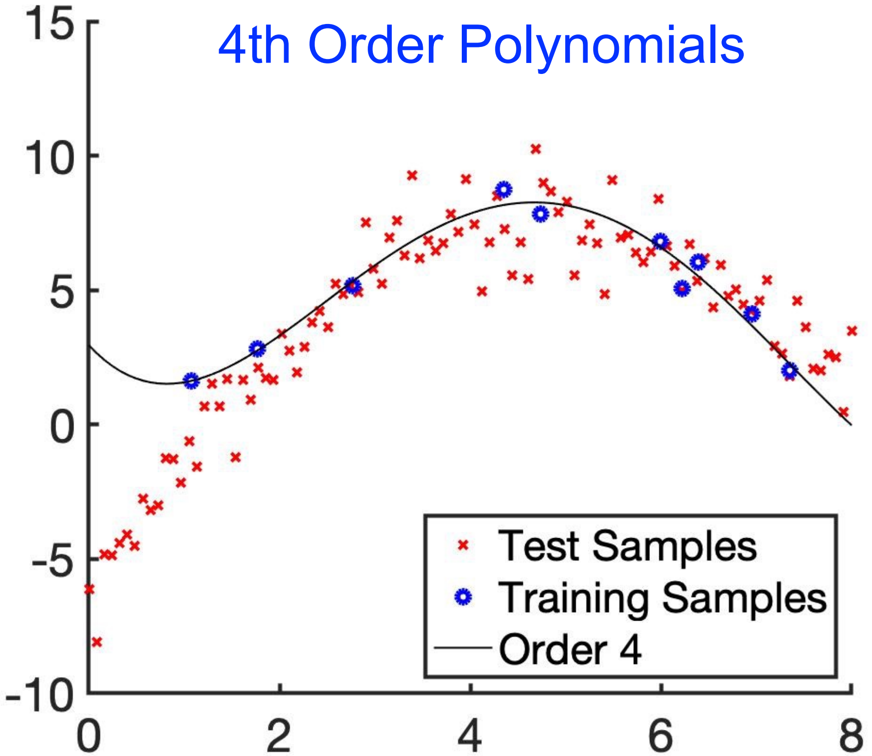
Bias + Variance Trade off

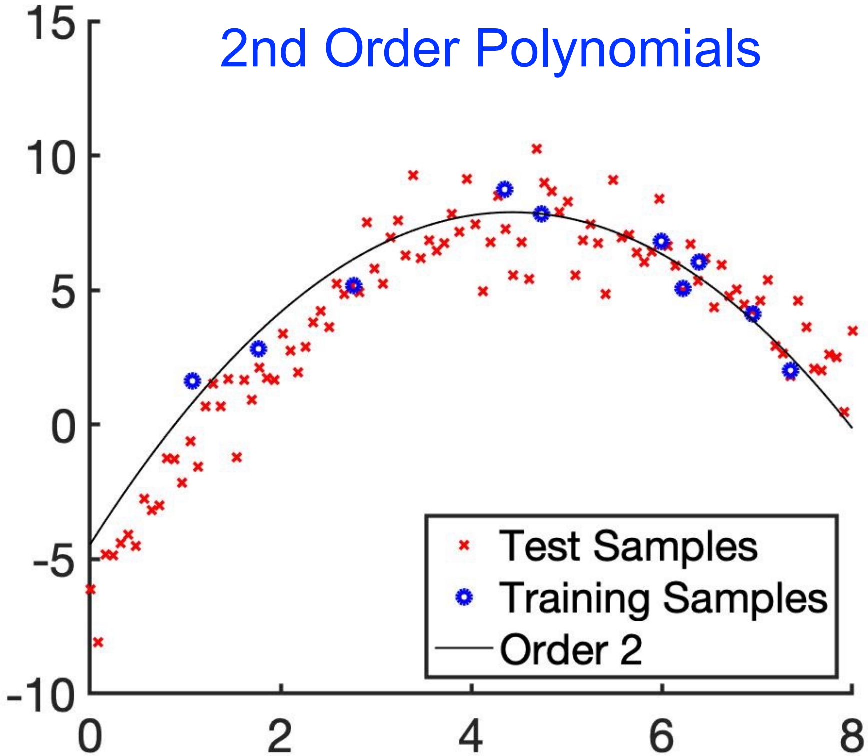
- Test error = Bias Squared + Variance + Irreducible Noise



Bias + Variance Example

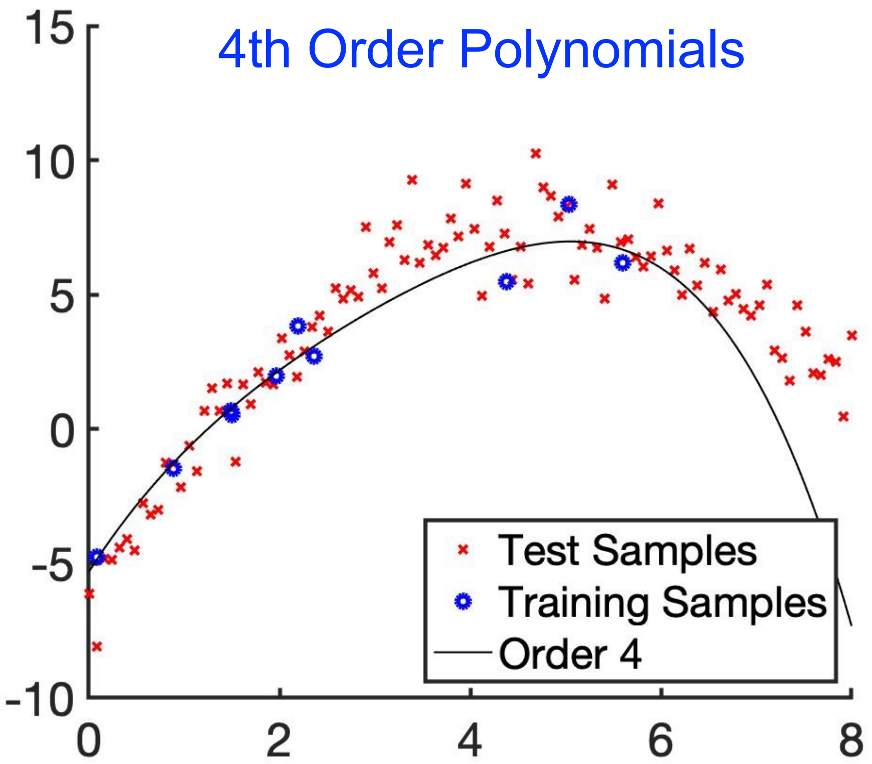
- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial
- Fit with order 4 polynomial

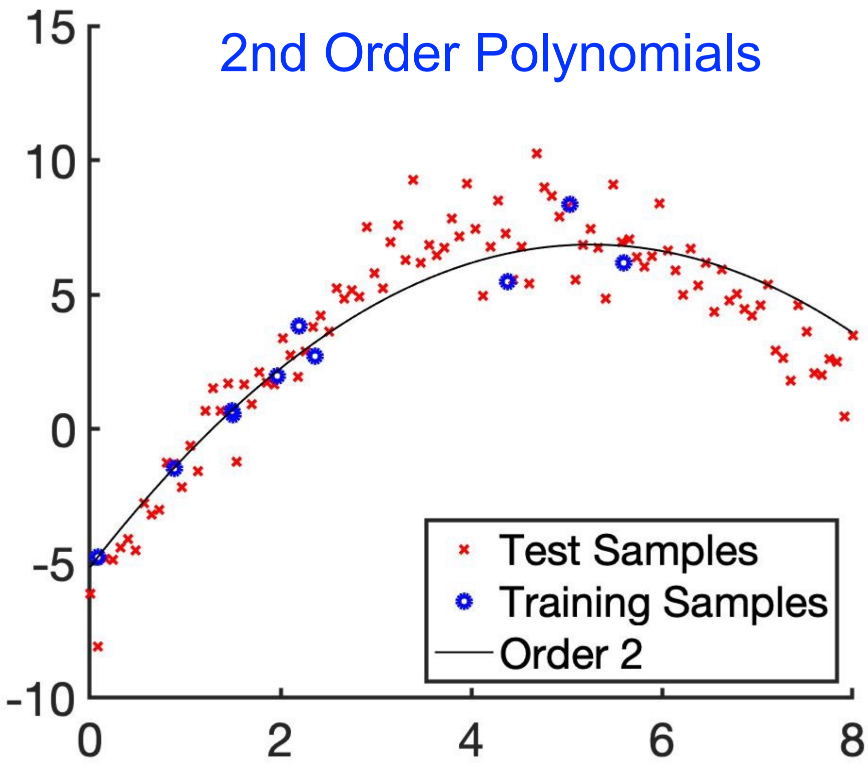




Bias + Variance Example

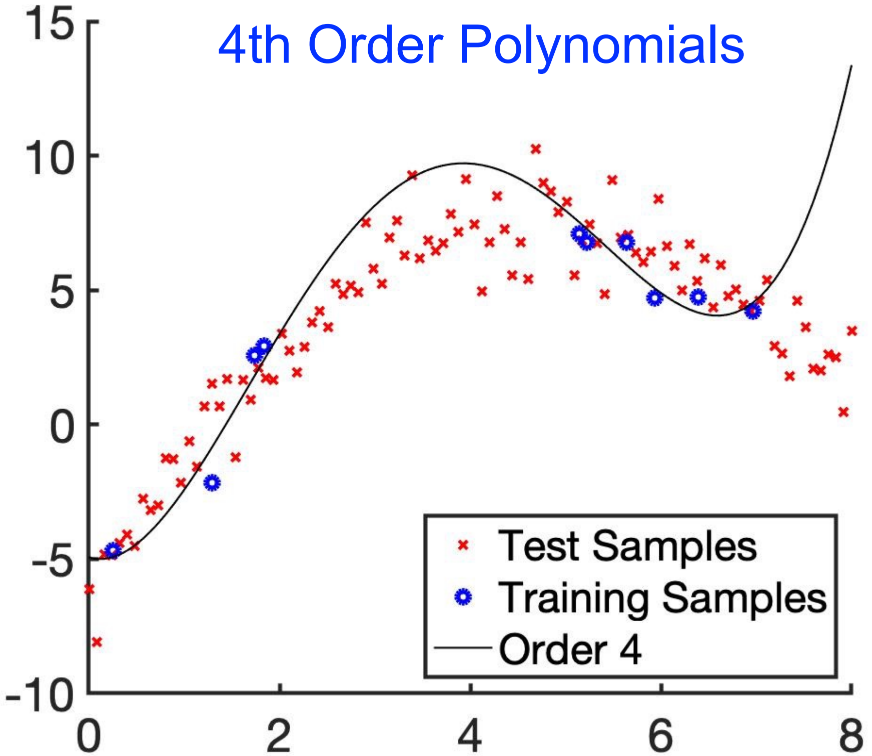
- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial
- Fit with order 4 polynomial

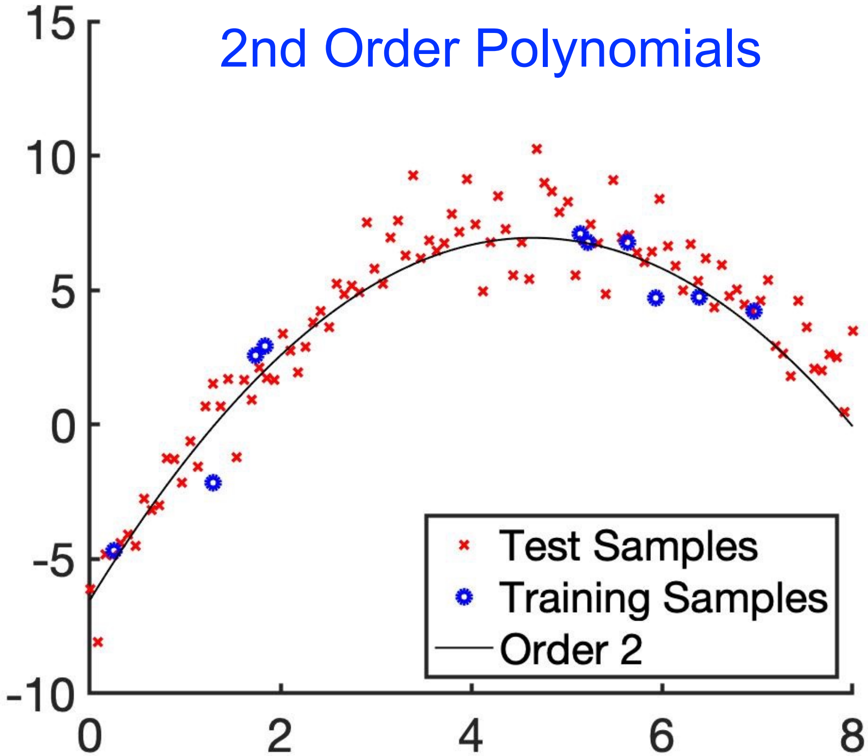




Bias + Variance Example

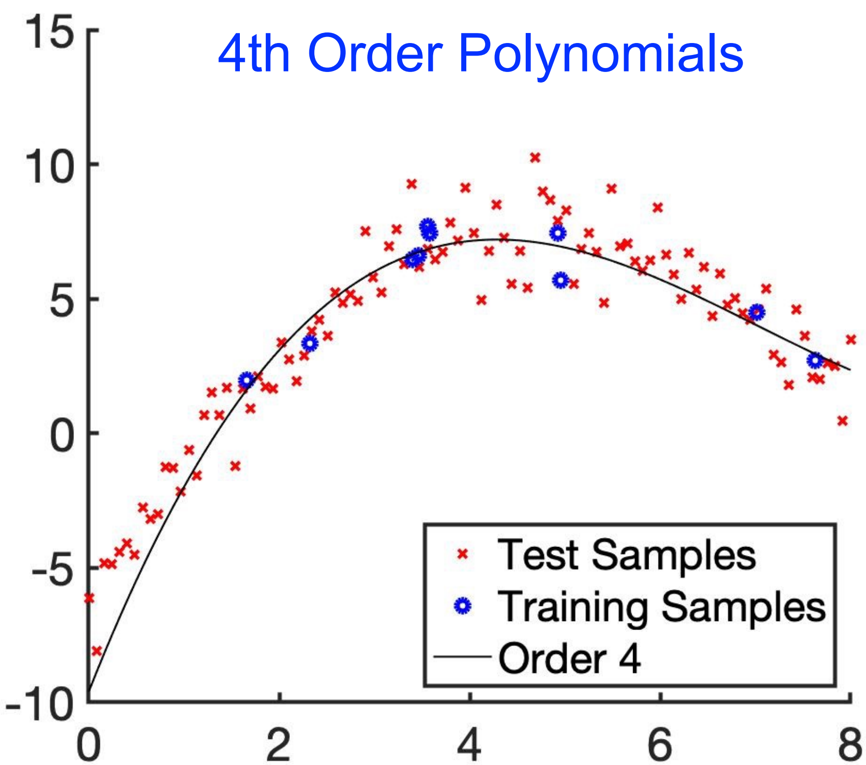
- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial
- Fit with order 4 polynomial

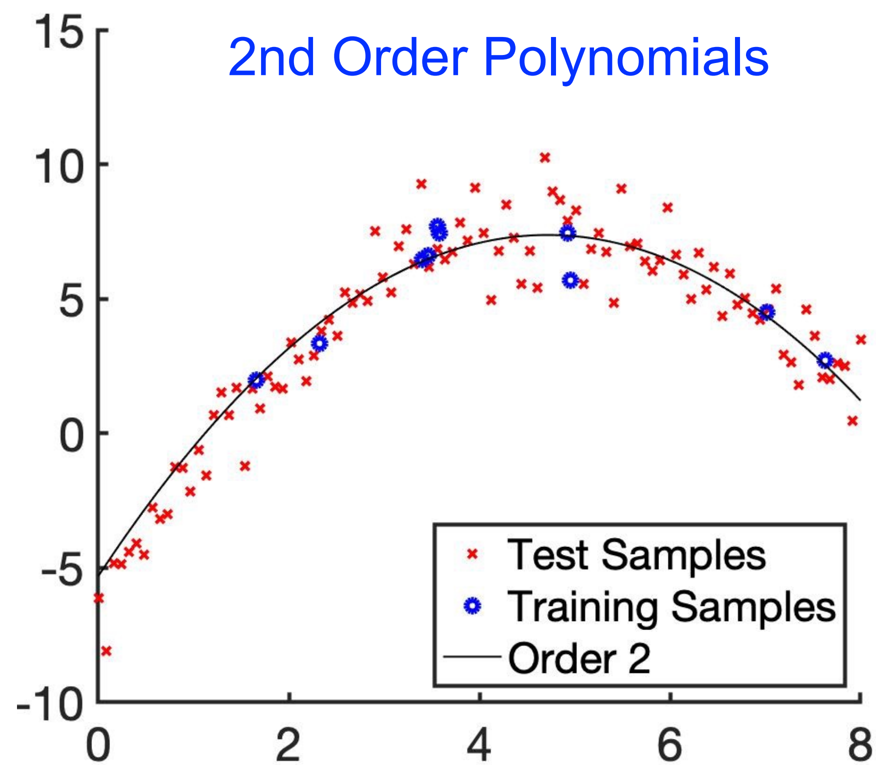




Bias + Variance Example

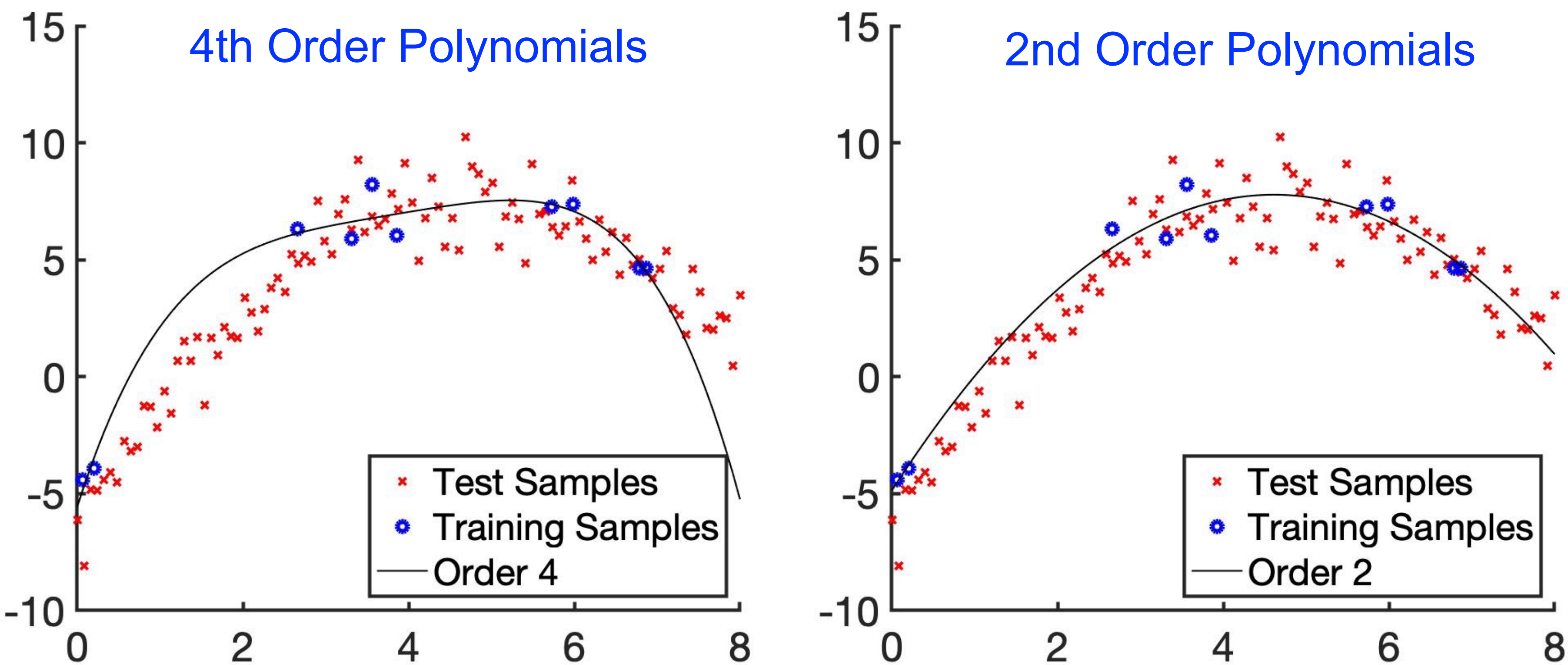
- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial
- Fit with order 4 polynomial





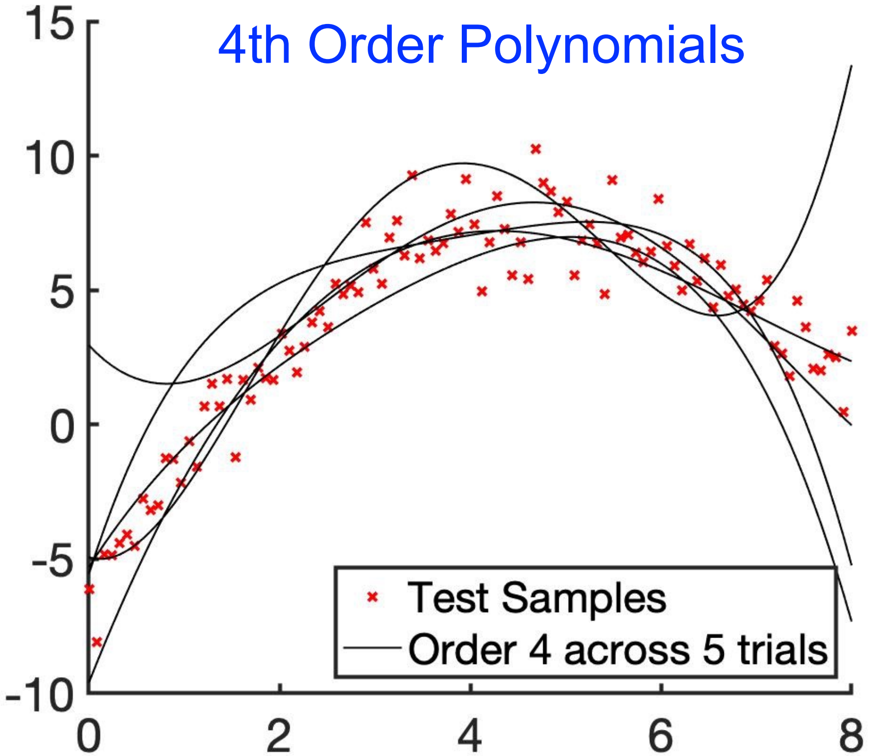
Bias + Variance Example

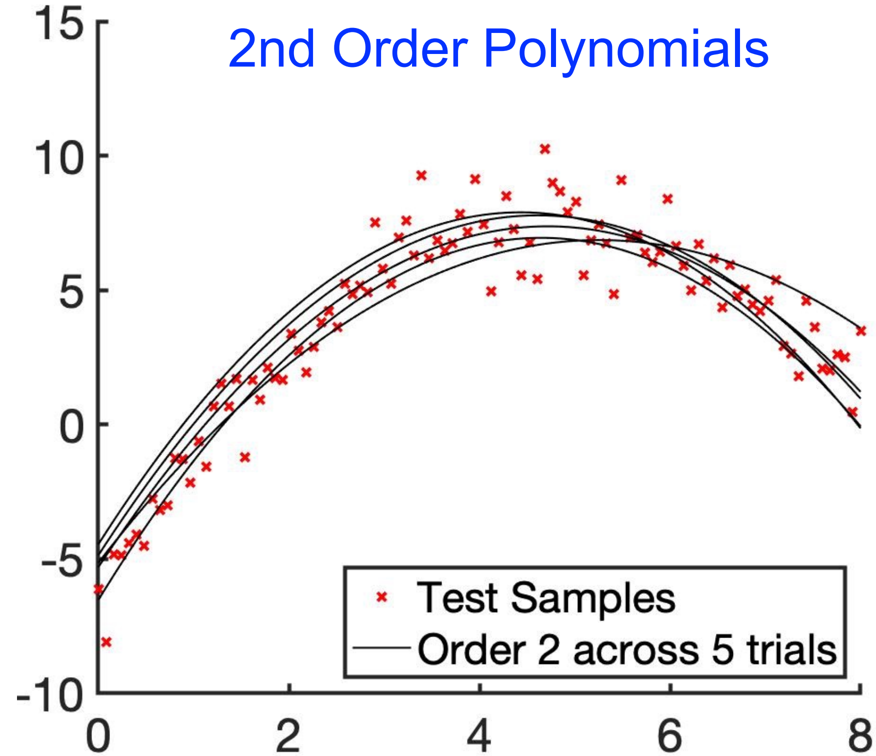
- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial
- Fit with order 4 polynomial



Bias + Variance Example

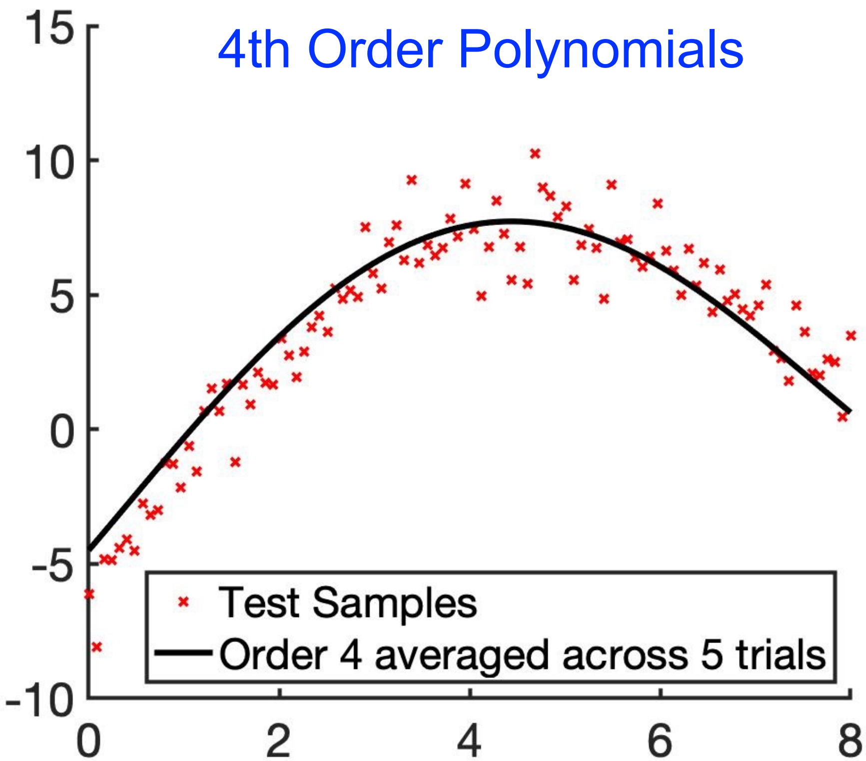
- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial: low variance
- Fit with order 4 polynomial: high variance

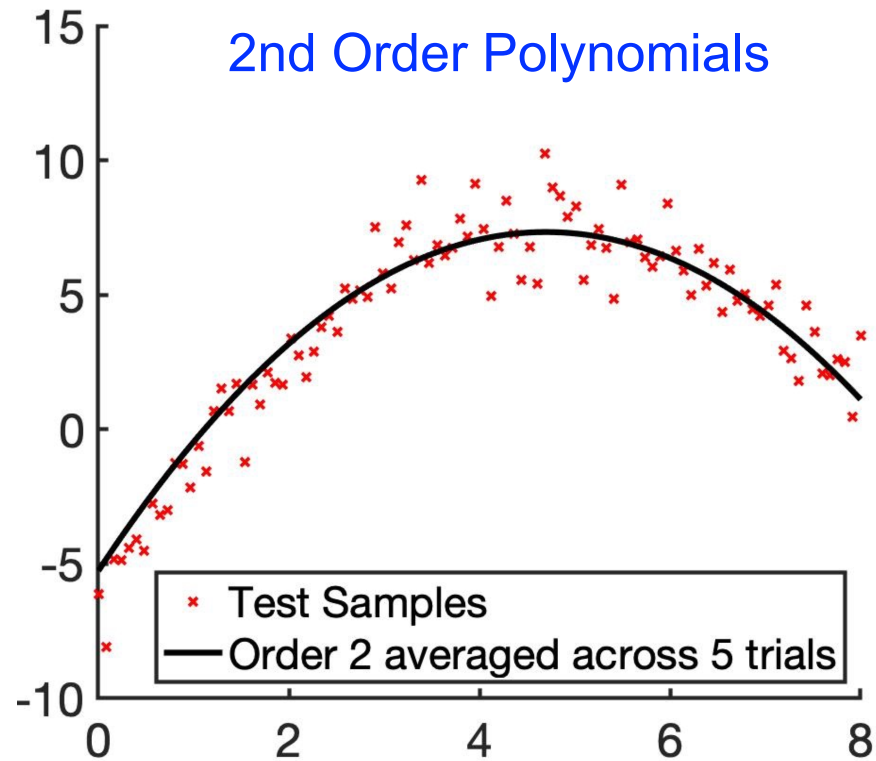




Bias + Variance Example

- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial: low variance, low bias
- Fit with order 4 polynomial: high variance, low bias

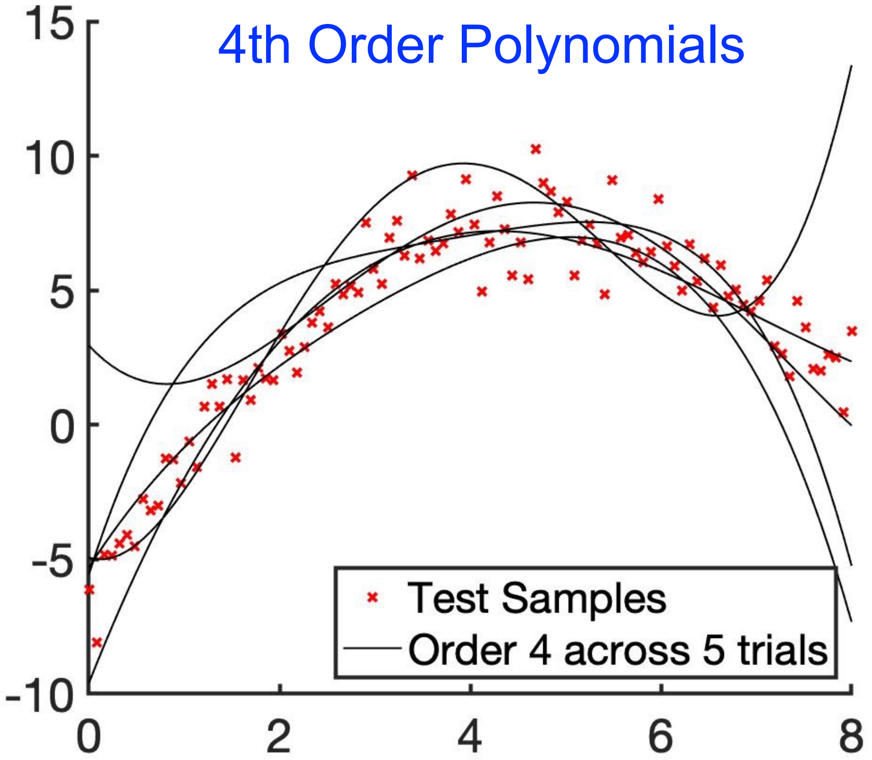


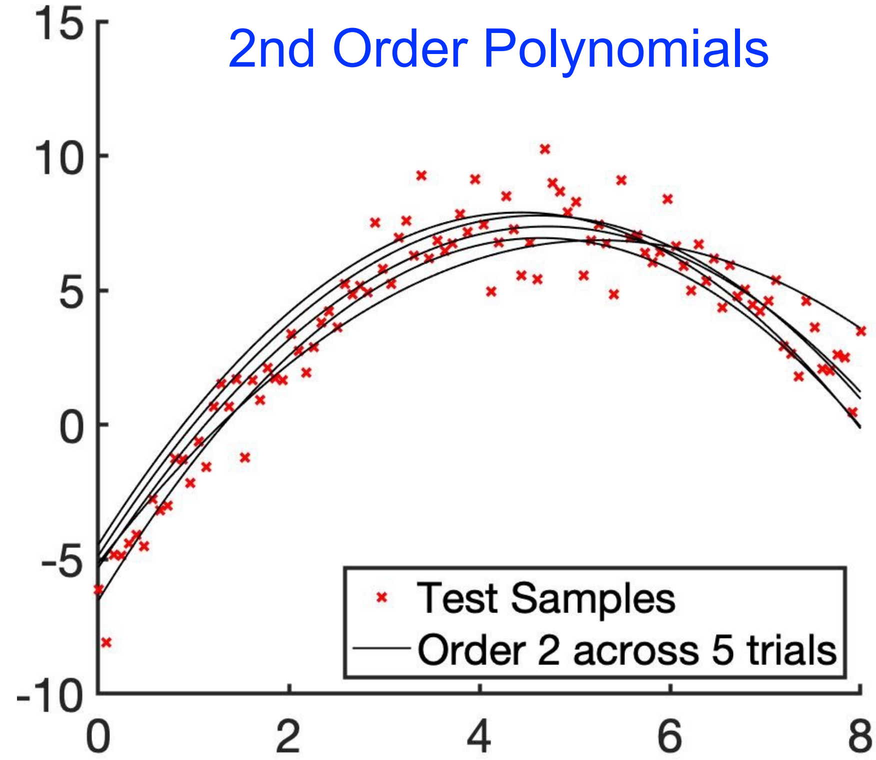


Bias + Variance Example

- Simulate data from order 2 polynomial (+ noise)
- Randomly sample 10 training samples each time
- Fit with order 2 polynomial: low variance, low bias
- Fit with order 4 polynomial: high variance, low bias

Order 2 Achieves Lower Test Error





Bias-Variance Decomposition Theorem

- **Test error** = Bias Squared + Variance + Irreducible Noise
  - Mathematical details in optional uploaded material (won't be tested)
- "**Variance**" refers to variability of prediction models across different training sets
  - In previous example, every time the training set of 10 samples changes, the trained model changes
  - "Variance" quantifies variability across trained models
- "**Bias**" refers to how well an average prediction model will perform
  - In previous example, every time the training set of 10 samples changes, the trained model changes
  - If we average the trained models, how well will this average trained model perform?
- "**Irreducible Noise**" reflects the fact that even if we are perfect modelers, it might not be possible to predict target y with 100% accuracy from feature(s) x

Summary

- Overfitting, underfitting & model complexity
  - Overfitting: low error in training set, high error in test set
  - Underfitting: high error in both training & test sets
  - Overly complex models can overfit; Overly simple models can underfit
- Feature selection
  - Extract useful features from training set
- Regularization (e.g., L2 regularization)
  - Solve “ill-posed” problem (e.g., more unknowns than data points)
  - Reduce overfitting
- Bias-Variance Decomposition Theorem
  - Test error = Bias Squared + Variance + Irreducible Noise
  - Can be interpreted as trading off bias & variance:
    - Overly complex models can have high variance, low bias
    - Overly simple models can have low variance, high bias