Unit 4
Model Order Selection

EE-UY 4563/EL-GY 9143: INTRODUCTION TO MACHINE LEARNING

PROF. SUNDEEP RANGAN (WITH MODIFICATION BY YAO WANG)

Learning Objectives

- ❑Compute the model order for a given model class
- ❑Visually identify overfitting and underfitting of a model in a scatterplot
- ❑Determine if there is under-modeling for a given true function and model class
- ❑Compute the bias and variance for linear models (advanced)
- ❑Perform cross-validation for selecting an optimal order selection

Outline

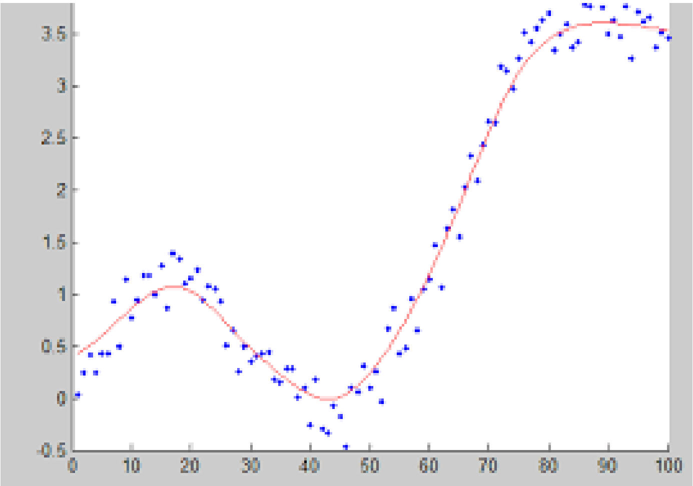
- ❑ Motivating Example: What polynomial degree should a model use?
- ❑ Bias and variance
- ❑ Bias and variance for linear models (Advanced)
- ❑ Cross-validation

Polynomial Fitting

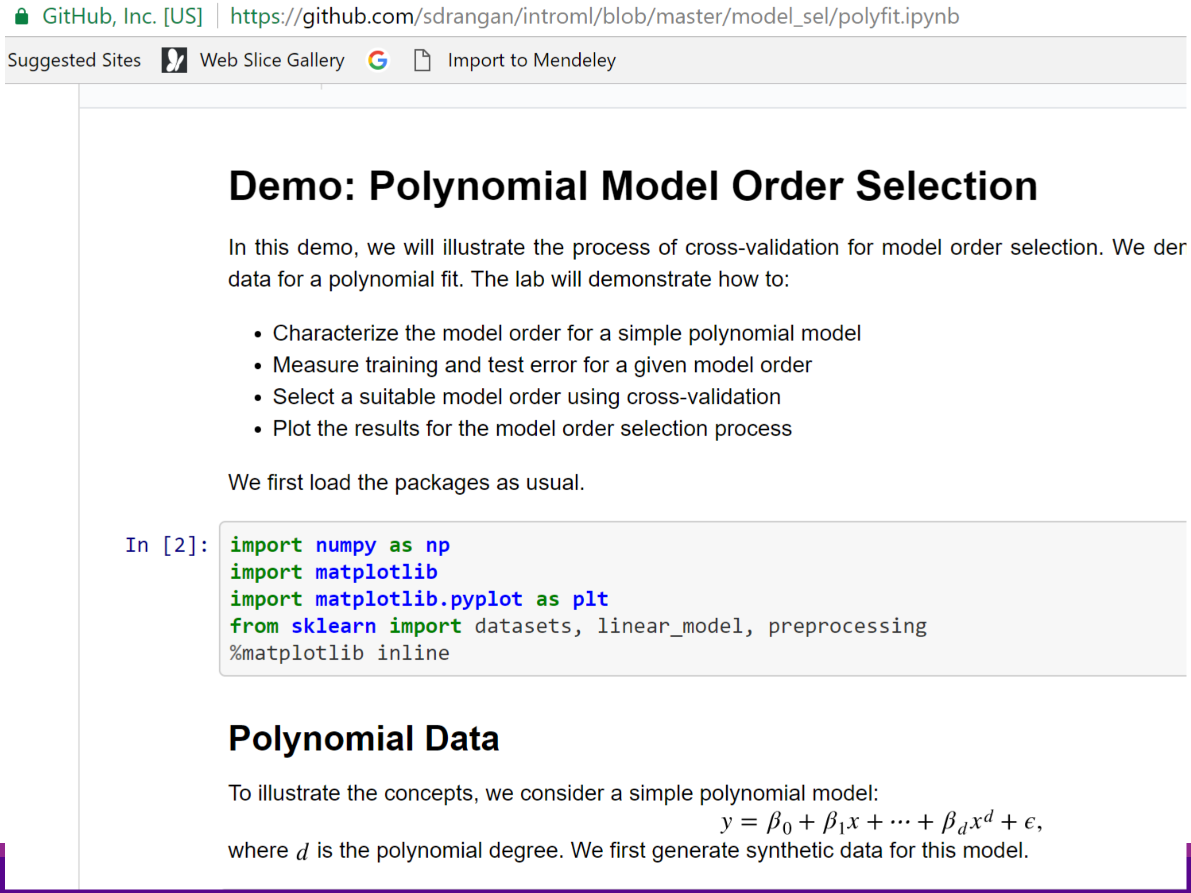
- Last lecture: polynomial regression
- Given data $(x_i, y_i), i = 1, \ldots, N$
- Learn a polynomial relationship:

$$y = \beta_0 + \beta_1 x + \cdots + \beta_d x^d + \epsilon$$

  - d = degree of polynomial. Called model order
  - $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_d)$ = coefficient vector
- Given d, can find $\boldsymbol{\beta}$ via least squares
- How do we select d from data?
- This problem is called model order selection.



Demo on Github

- Demo on github: https://github.com/sdrangan/introml/blob/master/unit04_model_sel/demo1_polyfit.ipynb
- 

GitHub, Inc. [US]
https://github.com/sdrangan/introml/blob/master/model_sel/polyfit.ipynb
Suggested Sites
Web Slice Gallery
Import to Mendeley

Demo: Polynomial Model Order Selection

In this demo, we will illustrate the process of cross-validation for model order selection. We der
data for a polynomial fit. The lab will demonstrate how to:

- Characterize the model order for a simple polynomial model
- Measure training and test error for a given model order
- Select a suitable model order using cross-validation
- Plot the results for the model order selection process

We first load the packages as usual.

```
In [2]: import numpy as np
        import matplotlib
        import matplotlib.pyplot as plt
        from sklearn import datasets, linear_model, preprocessing
        %matplotlib inline
```

Polynomial Data

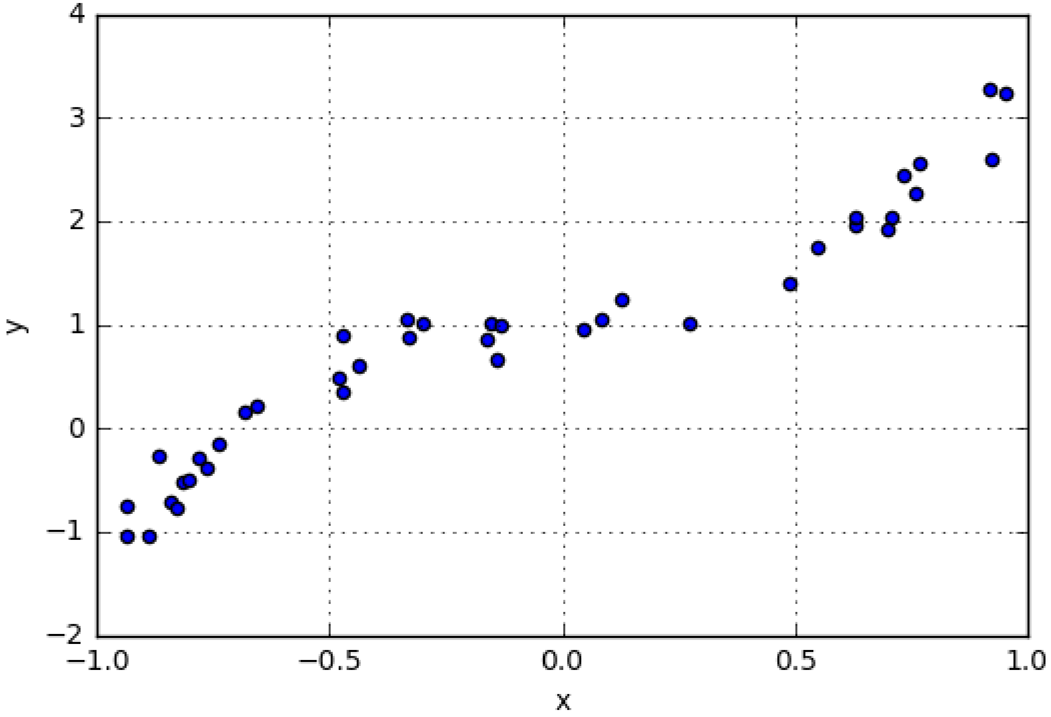
To illustrate the concepts, we consider a simple polynomial model:

$$y = \beta_0 + \beta_1 x + \cdots + \beta_d x^d + \epsilon,$$

where d is the polynomial degree. We first generate synthetic data for this model.

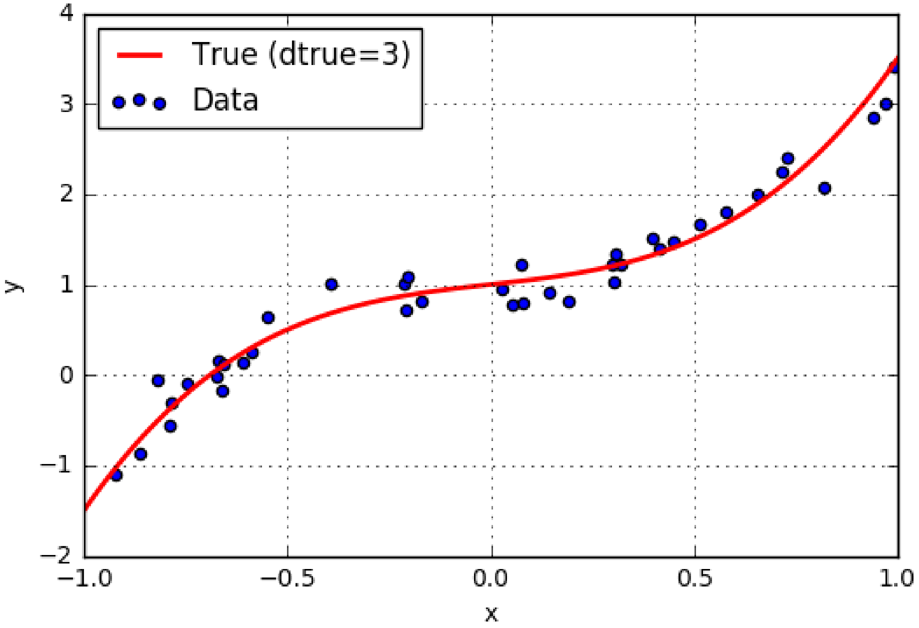
Example Question

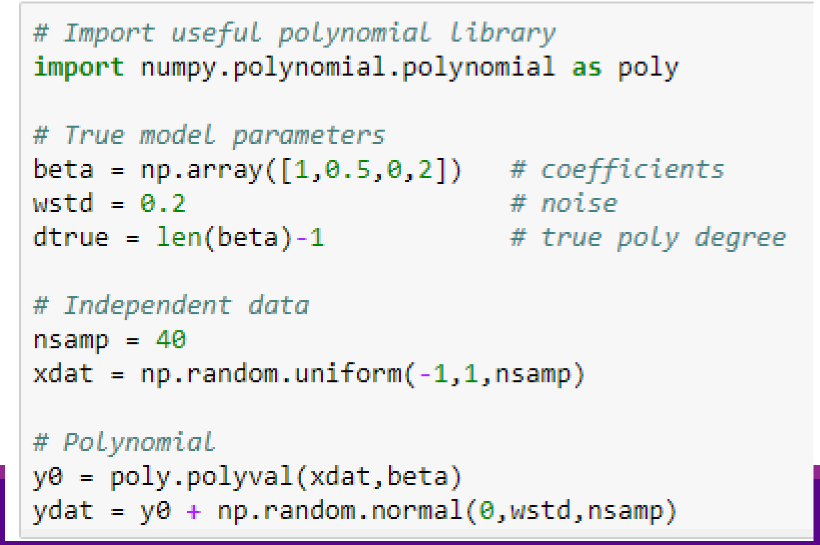
- ❑ You are given some data.
- ❑ Want to fit a model: $y \approx f(x)$
- ❑ Decide to use a polynomial:
  $$f(x) = \beta_0 + \beta_1 x + \cdots + \beta_d x^d$$
- ❑ What model order d should we use?
- ❑ Thoughts?



Synthetic Data

- Previous example is synthetic data
- x_i: 40 samples uniform in [-1,1]
- $y = f(x) + \epsilon$,
  - $f(x) = \beta_0 + \beta_1 x + \cdots + \beta_d x^d$ = "true relation"
  - $d = 3,\ \epsilon \sim N(0, \sigma^2)$
- Synthetic data useful for analysis
  - Know "ground truth"
  - Can measure performance of various estimators

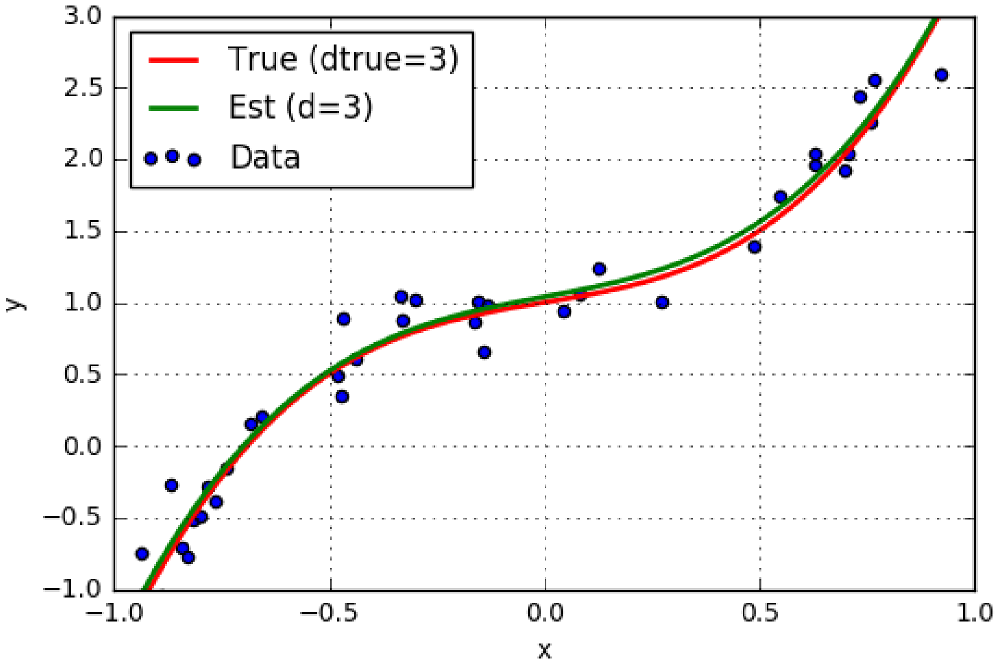


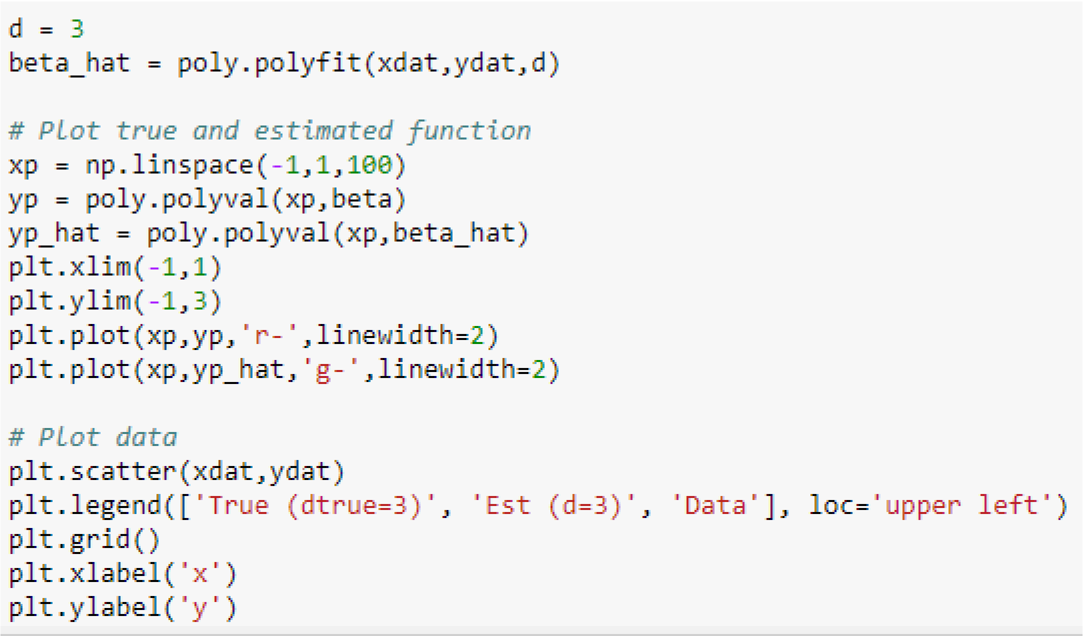


```
# Import useful polynomial library
import numpy.polynomial.polynomial as poly

# True model parameters
beta = np.array([1,0.5,0,2])   # coefficients
wstd = 0.2                     # noise
dtrue = len(beta)-1            # true poly degree

# Independent data
nsamp = 40
xdat = np.random.uniform(-1,1,nsamp)

# Polynomial
y0 = poly.polyval(xdat,beta)
ydat = y0 + np.random.normal(0,wstd,nsamp)
```

Fitting with True Model Order

- ❑ Suppose true polynomial order, d=3, is known
- ❑ Use linear regression
  - numpy.polynomial package
- ❑ Get very good fit

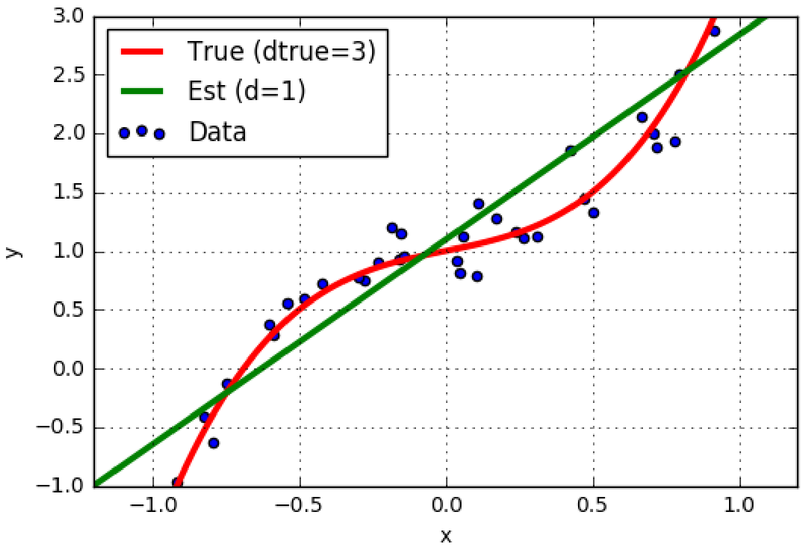




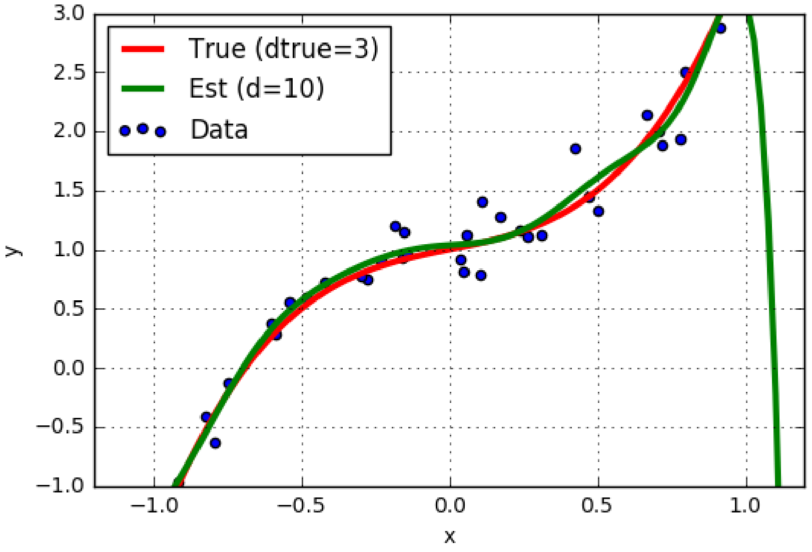
```
d = 3
beta_hat = poly.polyfit(xdat,ydat,d)

# Plot true and estimated function
xp = np.linspace(-1,1,100)
yp = poly.polyval(xp,beta)
yp_hat = poly.polyval(xp,beta_hat)
plt.xlim(-1,1)
plt.ylim(-1,3)
plt.plot(xp,yp,'r-',linewidth=2)
plt.plot(xp,yp_hat,'g-',linewidth=2)

# Plot data
plt.scatter(xdat,ydat)
plt.legend(['True (dtrue=3)', 'Est (d=3)', 'Data'], loc='upper left')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
```

But, True Model Order not Known

- Suppose we guess the wrong model order?

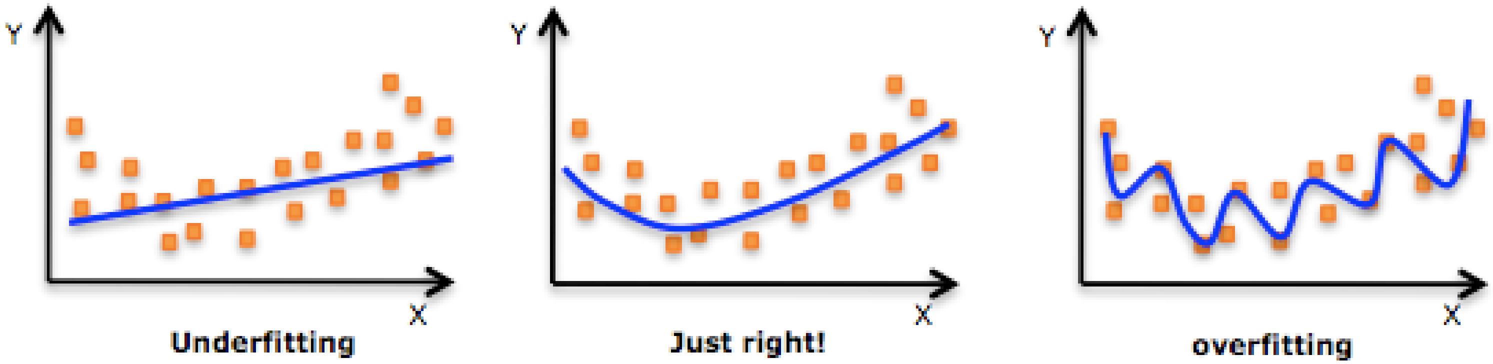


d=1 “Underfitting”



d=10 “Overfitting”

How Can You Tell from Data?



- ❑ Is there a way to tell what is the correct model order to use?
- ❑ Must use the data. Do not have access to the true d?
- ❑ What happens if we guess:
  - ◦ d too big?
  - ◦ d too small?

Using RSS on Training Data?

❑Simple (but bad) idea:

- For each model order, d, find estimate $\widehat{\boldsymbol{\beta}}$
- Compute predicted values on training data

$$\hat{y}_i = \widehat{\boldsymbol{\beta}}^T \boldsymbol{x}_i$$

- Compute RSS

$$RSS(d) = \sum_i (y_i - \hat{y}_i)^2$$

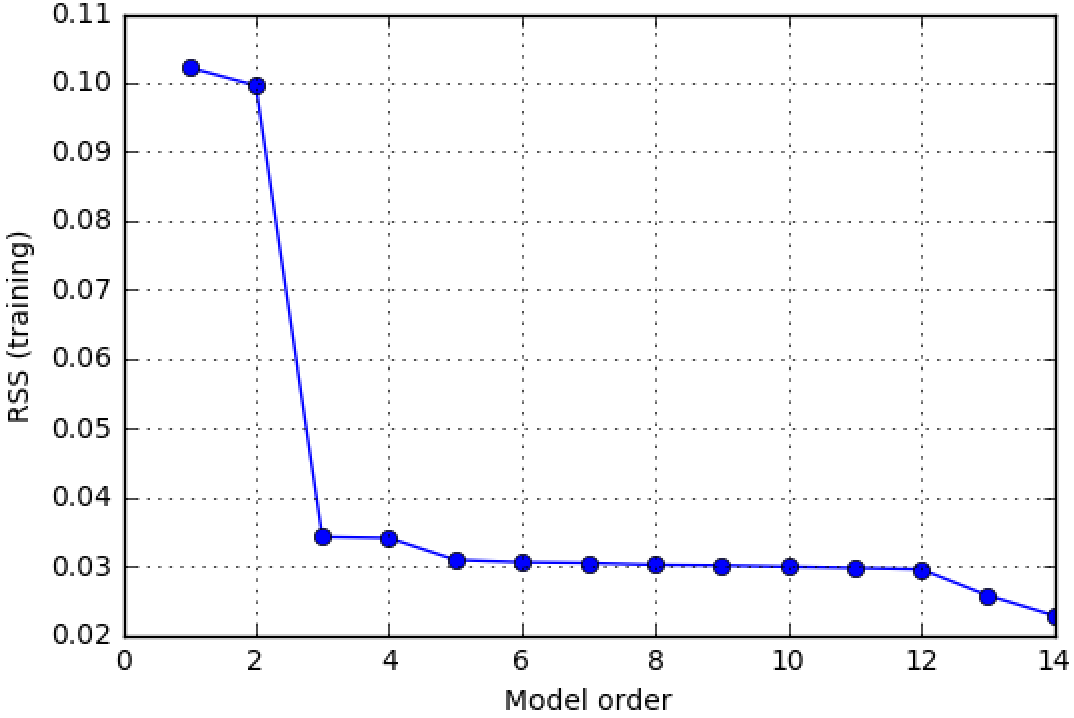
- Find d with lowest RSS

❑This doesn't work

- $RSS(d)$ is always decreasing (Question: Why?)
- Minimizing $RSS(d)$ will pick d as large as possible
- Leads to overfitting

❑What went wrong?

❑How do we do better?



Outline

□ Motivating Example: What polynomial degree should a model use?

□ Bias and variance

□ Bias and variance for linear models (Advanced)

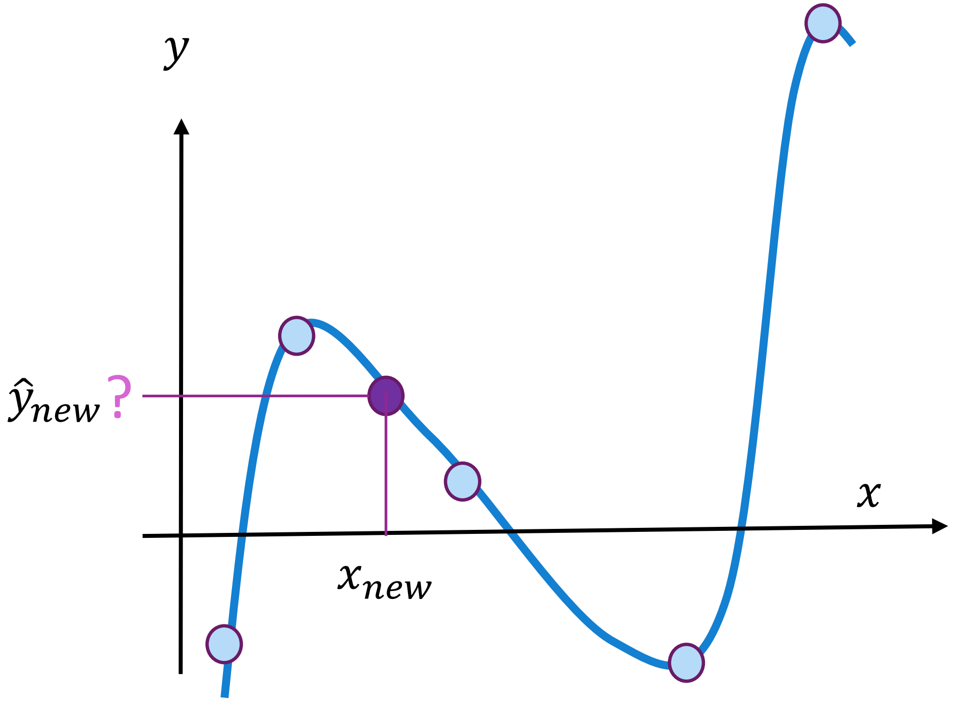
□ Cross-validation

Generalization

- ❑ Machine learning:
  - Get data points $(x_i, y_i), i = 1, \dots, n$
  - Learn some function $\hat{y} = f(x)$

- ❑ Implicitly, we are
  - Inferring the value of y at new values of x
  - Called generalization

- ❑ Basic question for all ML:
  - *How well do models we train generalize to new samples?*

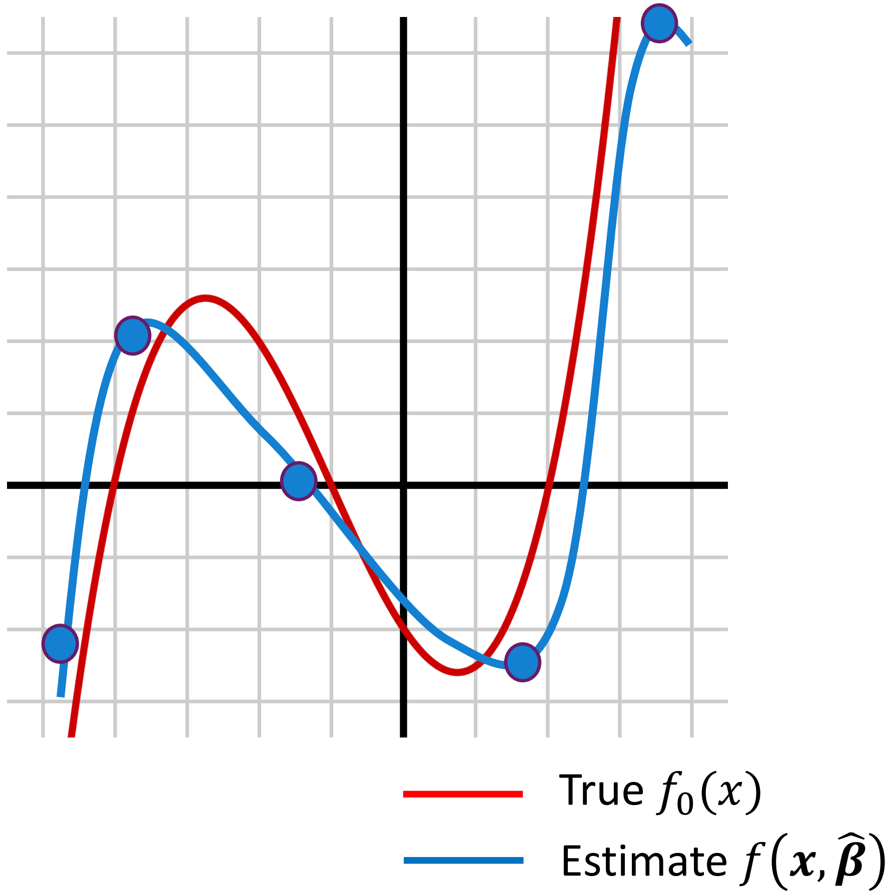


A Model To Understand Generalization

- □ Assume a true relation: $y = f_0(x) + \epsilon,\ \epsilon \sim N(0, \sigma_\epsilon^2)$
- □ Get data points $(x_i, y_i), i = 1, \ldots, n$

  $$y_i = f_0(x_i) + \epsilon_i$$

- □ Assume a model $\hat{y} = f(\boldsymbol{x}, \boldsymbol{\beta})$
  - Parameters $\boldsymbol{\beta}$
- □ Fit a parameter $\widehat{\boldsymbol{\beta}}$ from training data
  - Results in estimated function $f(\boldsymbol{x}, \widehat{\boldsymbol{\beta}})$
- □ Question: How “good” is the estimated function?

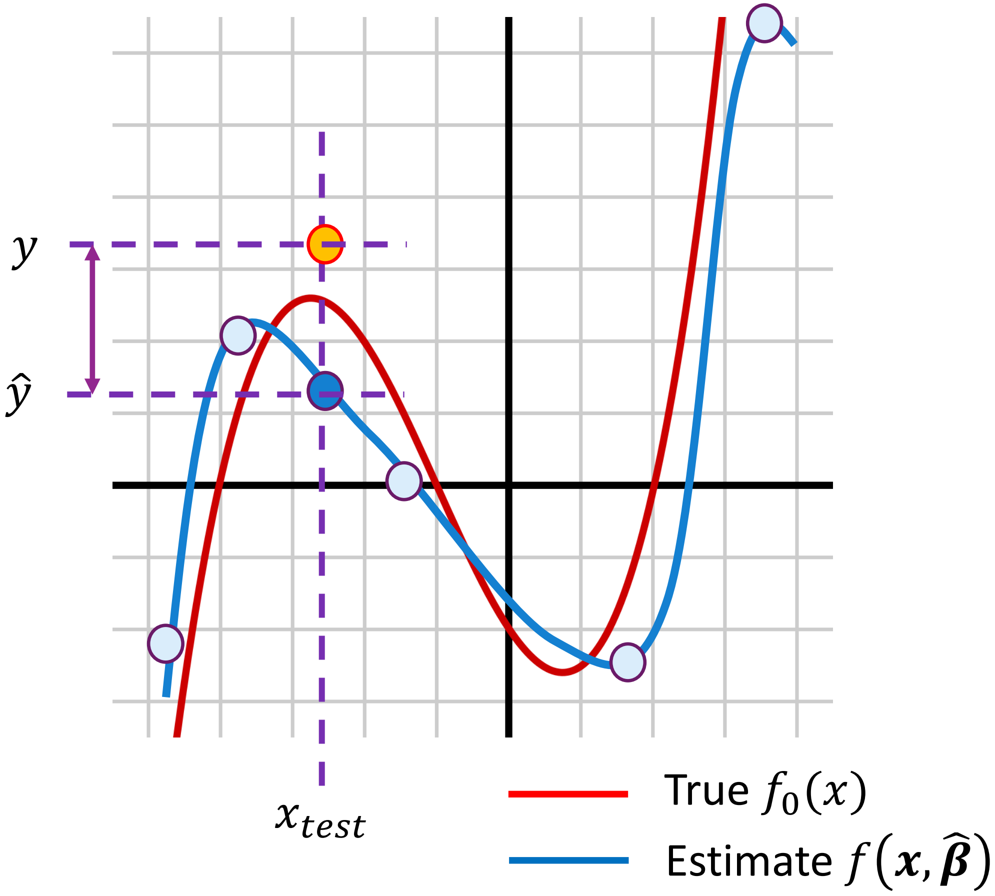


Output Mean Squared Error

- ❑ To evaluate generalization, suppose we are given:
  - A test point $\boldsymbol{x}_{test}$
  - New point, generally different from training samples.
- ❑ Actual value: $y = f_0(\boldsymbol{x}_{test}) + \boldsymbol{\epsilon}$
- ❑ Predicted value: $\hat{y} = f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)$
  - Note that $\widehat{\boldsymbol{\beta}}$ is random due to noise in training data
- ❑ Define output mean squared error :

$$MSE_y(\boldsymbol{x}_{test}) \coloneqq E[y - \hat{y}]^2$$

  - Expectation is over noise $\boldsymbol{\epsilon}$ on the training and test data.



Function MSE and Irreducible Error

❑ Output MSE decomposition: Output MSE, $MSE_y(\boldsymbol{x}_{test}) := E[y - \hat{y}]^2$, satisfies:

$$\boxed{MSE_y(\boldsymbol{x}_{test}) = MSE_f(\boldsymbol{x}_{test}) + \sigma_\epsilon^2}$$

❑ Function MSE: $MSE_f(\boldsymbol{x}_{test}) = E\left[f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]^2$

- Represents difference between estimated and true function

❑ Irreducible error: $\sigma_\epsilon^2 = E(\epsilon^2)$ in output $y = f_0(x) + \epsilon$

- Occurs since y is influenced by other factors than $\boldsymbol{x}$
- Fundamental limit on ability to predict y
- Lower bound on $MSE_y\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right) \geq \sigma_\epsilon^2$

Proof of the MSE Decomposition

- Output MSE decomposition: Output MSE $MSE_y(\boldsymbol{x}_{test}) := E[y - \hat{y}]^2$ is:

$$MSE_y(\boldsymbol{x}_{test}) = MSE_f(\boldsymbol{x}_{test}) + \sigma_\epsilon^2$$

- Proof:
  - $MSE_y(\boldsymbol{x}_{test}) := E[y - \hat{y}]^2 = E\left[f_0(\boldsymbol{x}_{test}) + \epsilon - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]^2 = M_1 + M_2 + 2M_3$
  - $M_1 = E\left[f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]^2 = MSE_f(\boldsymbol{x}_{test})$
  - $M_2 = E\left[\epsilon^{\mathbf{2}}\right] = \sigma^2$
  - Noise on test sample is independent of $\widehat{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{test}$ and $\mathrm{E}(\epsilon) = 0$
  - Therefore $M_3 = E\left[\epsilon(f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right))\right] = E(\epsilon)E\left[f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right] = 0$

Model Class and Under-Modeling

- ❑ Model class: The set of possible functions, $\hat{y} = f(\boldsymbol{x}, \boldsymbol{\beta})$
  - ◦ Set is parametrized by $\boldsymbol{\beta}$

- ❑ Definition: A true function $f_0(\boldsymbol{x})$ is in the model class $\hat{y} = f(\boldsymbol{x}, \boldsymbol{\beta})$ if:

$$f_0(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{\beta}_0) \ \text{ for all } \boldsymbol{x}$$

for some parameter $\boldsymbol{\beta}_0$.

  - ◦ $\boldsymbol{\beta}_0$ called the true parameter

- ❑ Under-modeling: When $f_0(\boldsymbol{x})$ is not in the model class

Sample Questions 1 and 2

❑ For each pair, state if the true function is in the model class or not
  - That is, is there under-modeling or not?
  - If true function is in the model class, state the true parameter

❑ Ex 1:
  - True function: $f_0(x) = 2 + 3x$ Model class: $f(x, \beta) = \beta_0 + \beta_1 x + \beta_2 x^2$
  - No under-modeling. True parameter: $\beta = (2,3,0)$

❑ Ex 2:
  - True function: $f_0(x) = 2 + 3x + 4x^2$ Model class: $f(x, \beta) = \beta_0 + \beta_1 x$
  - There is under-modeling. Model class does not contain x^2 term

Sample Questions 3 and 4

- For each pair, state if the true function is in the model class or not
  - That is, is there under-modeling or not?
  - If true function is in the model class, state the true parameter

- Ex 3:
  - True function: $f_0(x) = \sin(2\pi(5)x + 7)$ Model class: $f(x, \beta) = \beta_0 \sin(2\pi(5)x) + \beta_1 \cos(2\pi(5)x)$
  - No under-modeling. $f_0(x) = \sin(2\pi(5)x + 7) = \sin(2\pi(5)x)\cos(7) + cos(2\pi(5)x)\sin(7)$
  - True parameter $\beta = (\cos 7, \sin 7)$

- Ex 4:
  - True function: $f_0(x) = \sin(2\pi(8)x + 7)$ Model class: $f(x, \beta) = \beta_0 \sin(2\pi(5)x) + \beta_1 \cos(2\pi(5)x)$
  - There is under-modeling. Model class does not contain $\sin(2\pi 8)$ or $\cos(2\pi 8)$ terms

Under-Modeling and Irreducible Error

❑ Suppose that:

- There is no under-modeling: $f_0(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{\beta}_0)$ for some "true" parameter $\boldsymbol{\beta}_0$; and
- Estimator selects the true parameter $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}_0$ (somehow?)

❑ Then, function MSE is zero:

$$MSE_f(\boldsymbol{x}_{test}) = E\left[f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]^2 = E[f_0(\boldsymbol{x}_{test}) - f(\boldsymbol{x}_{test}, \boldsymbol{\beta}_0)]^2 = 0$$

❑ Output MSE = irreducible error

$$MSE_y(\boldsymbol{x}_{test}) \coloneqq MSE_f(\boldsymbol{x}_{test}) + \sigma_\epsilon^2 = \sigma_\epsilon^2$$

What We Have Learned So Far

- ❑ If (A Big If!)
  - There is no undermodeling (i.e. true function is in model class), and
  - We can estimate the true parameter
- ❑ Then:
  - Output error = irreducible error
  - We can achieve the same error as if we knew the true function $f_0(\boldsymbol{x})$
- ❑ This suggests: *Select the model class large!*
  - Guarantees to approximately contains true function
  - Ex: Take model class = set of polynomials with very high degree
- ❑ But, using large models has other problems…

In-Class Exercise

Question

For each model and true function pair below: Determine if there is undermodeling (i.e. the true function is in the model class). If there is no undermodeling, find the true parameters

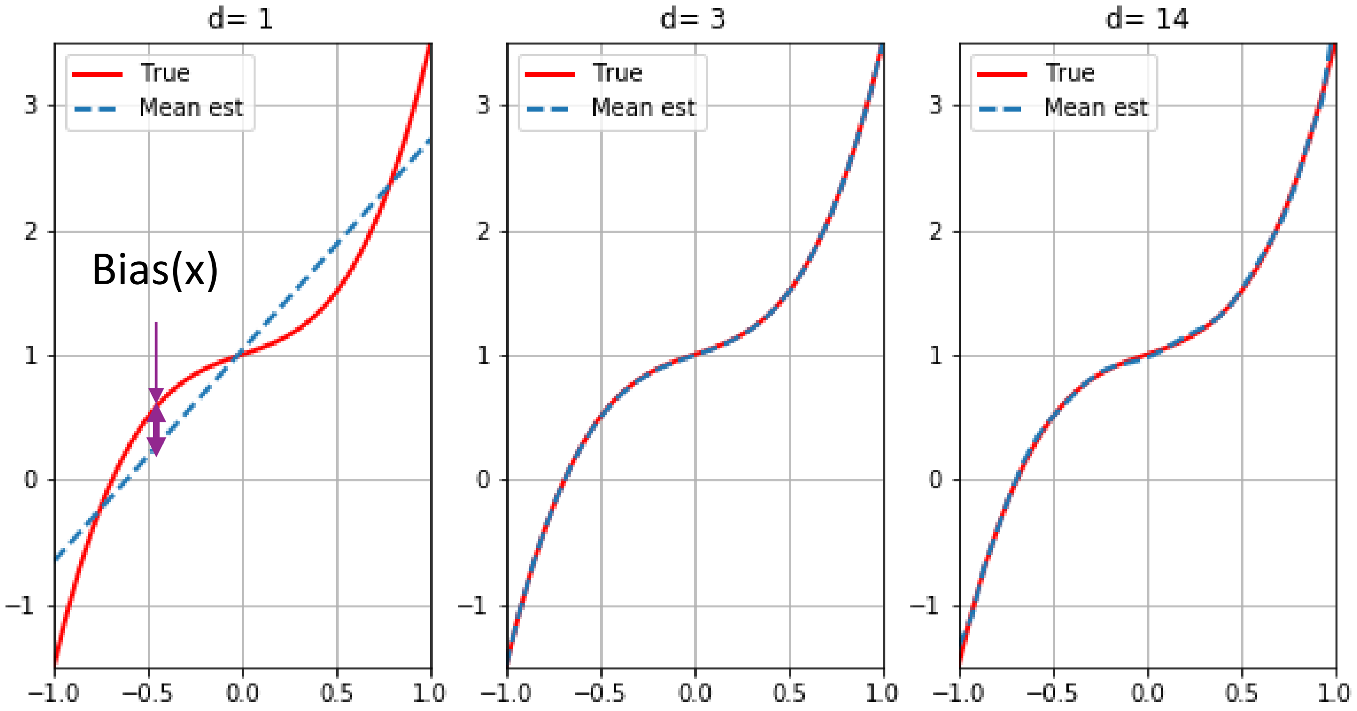
(a) True function: $f_0(x) = (1 + 2x)(3 + 4x)$, Model: $f(x, \beta) = \beta_0 + \beta_1 x + \beta_2 x^2$
(b) True function: $f_0(t) = 2(1 - e^{t-3})$ Model: $f(t, \beta) = a + be^{ct}, \beta = (a, b, c)$
(c) True function: $f_0(t) = 2(1 - e^{t-3})$ Model: $f(t, \beta) = a + be^{-t}, \beta = (a, b)$

Bias and Variance

- ❑To understand potential problem of using a large model class introduce two key quantities:
- ❑Bias: $Bias(\boldsymbol{x}_{test}) := f_0(\boldsymbol{x}_{test}) - E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]$
  - How much the average value of the estimate differs from the true function
- ❑Variance: $Var(\boldsymbol{x}_{test}) \coloneqq E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right) - E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]\right]^2$
  - How much the estimate varies around its average
- ❑Bias and variance are (conceptually) measured as follows:
  - Get many independent training data sets, each with same size N and input values x_i
  - Each dataset has different output values y_i because of independent noise in the training data
  - Obtain $\widehat{\boldsymbol{\beta}}$ for each training data set
  - Bias and variances are computed over the different sets
- ❑Of course, in reality, we have only one training dataset
  - Used to study theoretical averages over different experiments

Bias Illustrated

- ❑Red: True function
- ❑Repeat 100 trials
  - Each trial has independent data
  - Obtain estimate for each trial
- ❑Dashed line:
  Mean estimate among all trials
- ❑Bias=True – Mean estimate
- ❑Conclusions:
  - Low model orders ⇒ bias high
  - High model orders ⇒ bias low



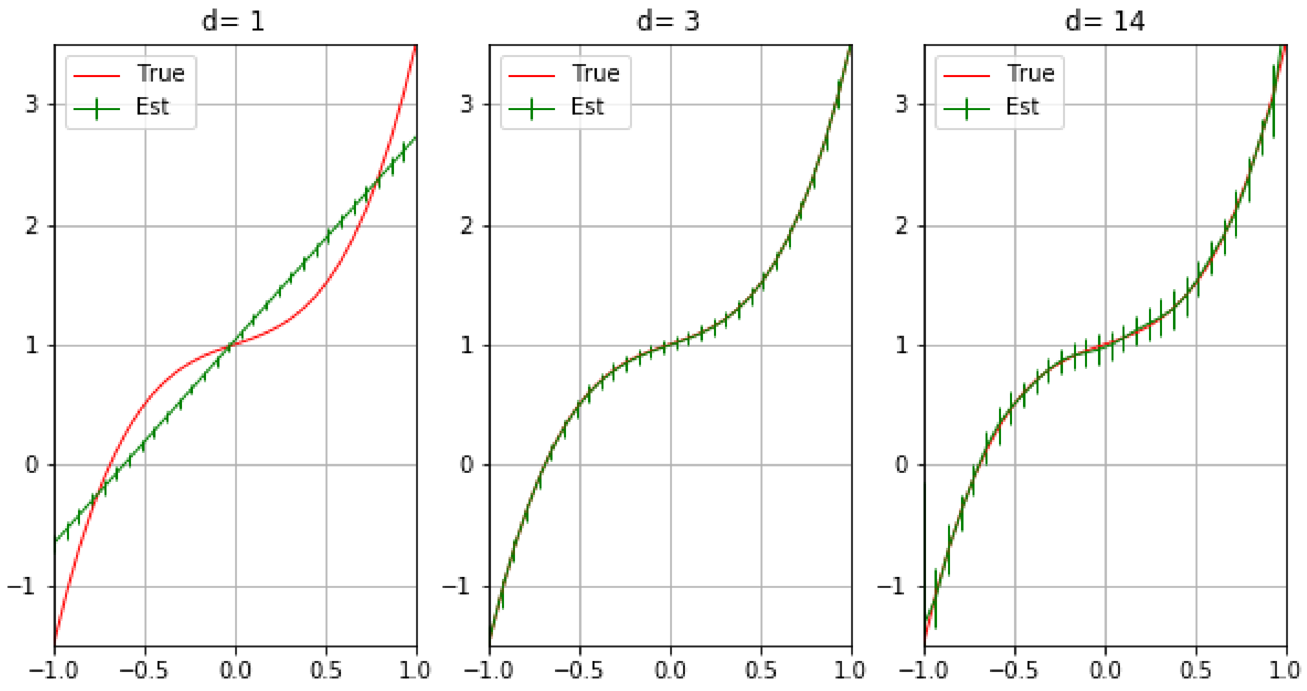
High bias $d < d_{true}$

No bias $d = d_{true}$

No bias $d > d_{true}$

Variance Illustrated

- ❑Red: True function
- ❑Repeat 100 trials
  - Each trial has independent data
  - Obtain estimate for each trial
- ❑Variance=STD around mean
- ❑Conclusions:
  - Low model orders ⇒ low variance
  - High model orders ⇒ high variance



High bias

Low variance

No bias

Med variance

No bias

High variance

Bias-Variance Formula

❑Recall definitions:

- Function MSE: $\mathrm{MSE}_f(\boldsymbol{x}_{test}) \coloneqq E\left[f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]$:
- Bias: $Bias(\boldsymbol{x}_{test}) := f_0(\boldsymbol{x}_{test}) - E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]$
- Variance: $Var(\boldsymbol{x}_{test}) \coloneqq E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right) - E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]\right]^2$

❑Bias-Variance formula : $\boxed{MSE_f(\boldsymbol{x}_{test}) = Bias(\boldsymbol{x}_{test})^2 + Var(\boldsymbol{x}_{test})}$

- Will be proved below

❑Bias-Variance tradeoff

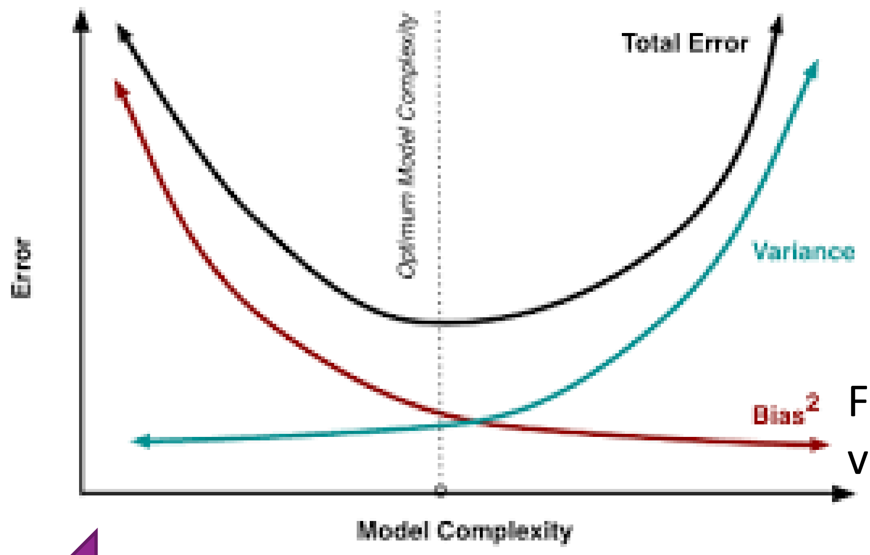
❑Bias due to under-modeling

- Reduced with high model order

❑Variance is due to noise in training data and number of parameters to estimate

- Increases with higher model order

Bias-Variance Tradeoff



Floor is the noise variance

Simpler models
Less parameters
Under-fitting

Richer models
More parameters
Over-fitting

❑ Bias:

- Due to under-modeling
- Reduced with high model order

❑ Variance:

- Increases with noise in training data
- Increase with high model order

❑ Optimal model order depends on:

- Amount of samples available
- Underlying complexity of the relation

Bias-Variance Formula Proof

- Define $\bar{f}(\boldsymbol{x}_{test}) = E[f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}})]$ = average value of estimated function
- $\text{MSE}_f(\boldsymbol{x}_{test}) = E[f_0(\boldsymbol{x}_{test}) - f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}})]^2 = E\left[f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}) + \bar{f}(\boldsymbol{x}_{test}) - f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}})\right]^2$
- Three components: $\text{MSE}_f(\boldsymbol{x}_{test}) = M_1 + M_2 - 2M_3$
  - $M_1 = E\left[f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test})\right]^2 = \left[f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test})\right]^2 = Bias(x_{test})$
  - $M_2 = E\left[f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}) - \bar{f}(\boldsymbol{x}_{test})\right]^2 = Var(x_{test})$
  - $M_3 = E\left[(f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}))(f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}) - \bar{f}(\boldsymbol{x}_{test}))\right]$
    $= (f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}))E\left[f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}) - \bar{f}(\boldsymbol{x}_{test})\right]$
    $= (f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}))(\bar{f}(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test})) = 0$

Summary of Results for Linear Models

- Suppose model is linear with $N =$ num samples, p = num parameters
- Can show the following results (need some math, see next section)
- Result 1: When $N < p$, linear estimate is not unique
  - Need at least as many samples as parameters
- Now assume that $N \geq p$ and parameter estimate is unique
- Result 2: When there is no under-modeling, estimate is unbiased

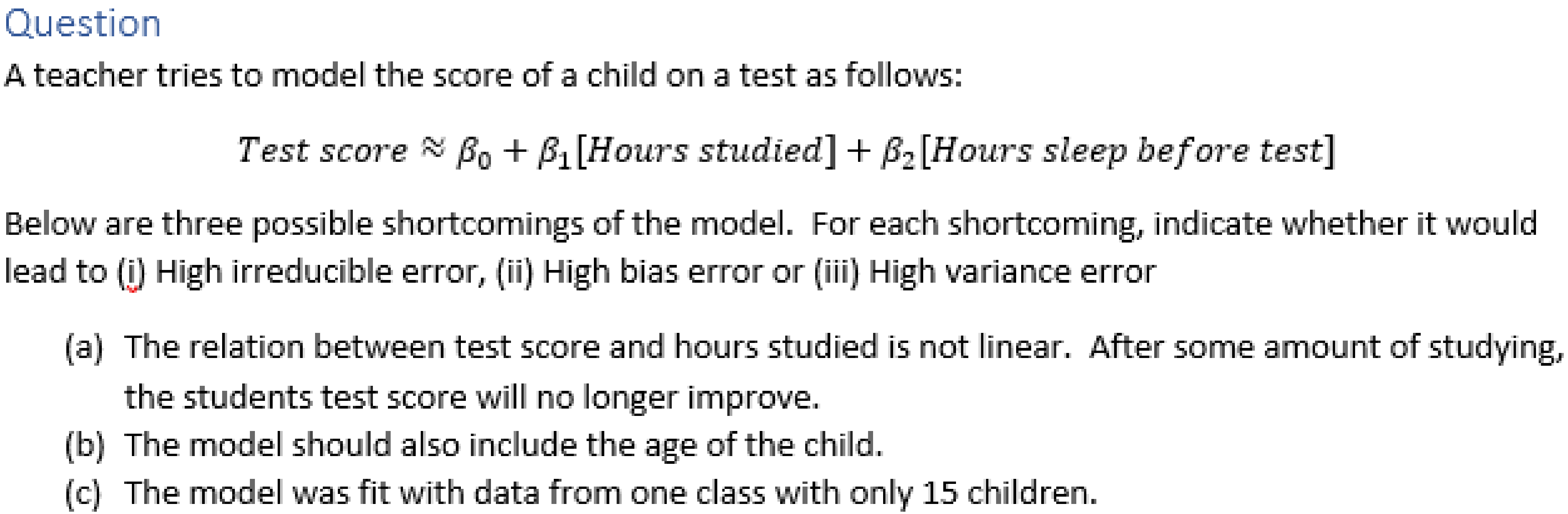
$$E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right] = f_0\ (\boldsymbol{x}_{test},).$$

- Result 3: If test point drawn randomly from the training data:

$$Var = \frac{p}{N}\sigma_\epsilon^2$$

  - Variance increases linearly with number of parameters and inversely with number of samples

In-Class Exercise



Question

A teacher tries to model the score of a child on a test as follows:

$$Test\ score \approx \beta_0 + \beta_1[Hours\ studied] + \beta_2[Hours\ sleep\ before\ test]$$

Below are three possible shortcomings of the model. For each shortcoming, indicate whether it would lead to (i) High irreducible error, (ii) High bias error or (iii) High variance error

(a) The relation between test score and hours studied is not linear. After some amount of studying, the students test score will no longer improve.

(b) The model should also include the age of the child.

(c) The model was fit with data from one class with only 15 children.

Outline

- ❑ Motivating Example: What polynomial degree should a model use?
- ❑ Bias and variance
- ❑ Bias and variance for linear models (Advanced)
- ❑ Cross-validation

This Section is Advanced

- ❑This section requires more advanced probability and linear algebra
- ❑Means and variances of random vectors
- ❑Undergraduates: Skip to final slide for final conclusions
- ❑Graduate students: We will cover this
  - You should review your multi-variable probability and linear algebra

Linear Models

- ❑Consider linear model in general transformed feature space:

$$\hat{y} = f(x, \beta) = \phi(x)^T \beta = \beta_1 \phi_1(x) + \cdots + \beta_p \phi_p(x)$$

  - See previous lecture

- ❑Assume true data relation is: $y = f_0(x) + \epsilon, E(\epsilon) = 0, \quad E(\epsilon^2) = \sigma^2$
- ❑When there is no under-modeling: $f_0(x) = f(x, \beta^0) = \phi(x)^T \beta^0$
  - $\beta^0 = (\beta_0^0, \cdots, \beta_k^0)$ True parameter
- ❑Get data $(x_i, y_i), i = 1, \ldots, N$
- ❑Least squares fit $\hat{\beta} = (A^T A)^{-1} A^T y$

$$A = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_p(x_1) \\ \vdots & \vdots & \vdots \\ \phi_1(x_N) & \cdots & \phi_p(x_N) \end{bmatrix}$$

Minimum Number of Samples

- ❑LS estimate requires $A^T A$ is invertible.
- ❑Linear algebra fact: Since $A \in R^{N \times p}$, we need $Rank(A) \geq p$
  - Otherwise solution is not unique
- ❑Since $Rank(A) \leq \min(N, p)$ we need $N \geq p$.
- ❑Recall:
  - N = number of data samples
  - p = number of parameters
- ❑Conclusion: Number of samples ≥ number of parameters
- ❑This places a basic limit on the model complexity that you can use

Random Vectors Review

❑ To analyze bias and variance in linear models, we need to review random vectors

❑ Random vectors: $\boldsymbol{x} = (x_1, \dots, x_d)^T$: Each component x_j is a random variable

❑ Mean: The vector of means of the components

$$\boldsymbol{\mu} = E\boldsymbol{x} = (Ex_1, \dots, Ex_d)^T = (\mu_1, \dots, \mu_d)^T$$

❑ Covariance components: $\mathrm{Cov}(x_i, x_j) = E\left[(x_i - \mu_i)(x_j - \mu_j)\right]$

❑ Variance matrix $(d \times d)$:

$$\mathrm{Var}(\boldsymbol{x}) \coloneqq E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T] = \begin{bmatrix} \mathrm{Cov}(x_1, x_1) & \cdots & \mathrm{Cov}(x_1, x_d) \\ \vdots & \vdots & \vdots \\ \mathrm{Cov}(x_d, x_1) & \cdots & \mathrm{Cov}(x_d, x_d) \end{bmatrix}$$

Linear Transforms of Random Vectors

- ❑A linear transform is a map: $y = Ax + b$
- ❑$A \in R^{M \times N}$ maps input $x \in R^N$ to $Ax \in R^M$
- ❑Mean and variance matrix under linear map given by
  - Mean: $E(y) = AE(x) + b$
  - Variance: $Var(y) = AVar(x)A^T$

Bias With No Under-Modeling

- ❑Suppose that there is no undermodeling: $f_0(x) = \phi(x)^T \beta^0$
- ❑Then each training sample output is: $y_i = \phi(x_i)^T \beta^0 + \epsilon_i$
- ❑Hence: true data vector $y = A\beta^0 + \epsilon$
- ❑Parameter estimate is:
$$\hat{\beta} = (A^T A)^{-1} A^T y = (A^T A)^{-1} A^T (A\beta^0 + \epsilon) = \beta^0 + (A^T A)^{-1} A^T \epsilon$$
- ❑Since $E\epsilon = 0$, $E\hat{\beta} = \beta^0$. Average of parameter estimate matches true parameter
- ❑$Ef\left(x_{test}, \hat{\beta}\right) = \phi(x_{test})^T E\hat{\beta} = \phi(x_{test})^T \beta^0 = f_0(x_{test})$
- ❑Therefore $Bias(x_{test}) \coloneqq f_0(x_{test}) - Ef\left(x_{test}, \hat{\beta}\right) = 0$
- ❑Conclusion: When the model is linear and there is no under-modeling, there is no bias

Variance of the Parameters in Linear Models

- ❑Since ϵ_i are independent for different samples with $E\epsilon_i = 0, E\epsilon_i^2 = \sigma^2$

$$\mathrm{Cov}(\epsilon_i, \epsilon_j) = \begin{cases} 0 & i \neq j \\ \sigma^2 & i = j \end{cases}$$

- ❑Therefore variance matrix is: $\mathrm{Var}(\epsilon) = \sigma^2 I$
- ❑From last slide: $\hat{\beta} = \beta^0 + (A^T A)^{-1} A^T \epsilon.$
- ❑Applying variance formula of a linear transformation of ϵ

$$E\left((\hat{\beta} - \beta^0)(\hat{\beta} - \beta^0)^T \right) = (A^T A)^{-1} A^T Var(\epsilon) A (A^T A)^{-1}$$

$$= \sigma^2 (A^T A)^{-1} A^T A (A^T A)^{-1} = \sigma^2 (A^T A)^{-1}$$

Variance in Linear Estimate

- To compute variance use trick: Suppose $\boldsymbol{a}$ and $\boldsymbol{z}$ are vectors, $\boldsymbol{a}$ is non-random, $\boldsymbol{z}$ is random:

$$E|\boldsymbol{a}^T\boldsymbol{z}|^2 = \mathrm{E}(\boldsymbol{a}^T\boldsymbol{z}\boldsymbol{z}^T\boldsymbol{a}) = \boldsymbol{a}^T E(\boldsymbol{z}\boldsymbol{z}^T)\boldsymbol{a}$$

- From earlier: $Ef\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right) = \phi(\boldsymbol{x}_{test})^T E\widehat{\boldsymbol{\beta}} = \phi(\boldsymbol{x}_{test})^T \boldsymbol{\beta}^0$

- Therefore variance of linear model:

$$\begin{aligned}\mathrm{Var}(\boldsymbol{x}_{test}) &= E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right) - Ef\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]^2 = E\left[\phi(\boldsymbol{x}_{test})^T(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\right]^2 \\ &= \phi(\boldsymbol{x}_{test})^T E\left[\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\right)\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\right)^T\right]\phi(\boldsymbol{x}_{test}) \\ &= \sigma^2\phi(\boldsymbol{x}_{test})^T(\boldsymbol{A}^T\boldsymbol{A})^{-1}\phi(\boldsymbol{x}_{test})\end{aligned}$$

- Above calculation is for the case of no under-modeling

- But, similar calculation shows variance expression is the same when there is under-modeling

Case with Equal Test & Training Distributions

- Suppose that test point is distributed identically to training data
  - Training data inputs $\boldsymbol{x}_i, i = 1, \dots, N$
  - $\boldsymbol{x}_{test} = \boldsymbol{x}_i$ with probability $\frac{1}{N}$
- Since rows of A are $\phi(\boldsymbol{x}_i)^T$: $A^T A = \sum_{i=1}^N \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T$
- Now use trick: For random vectors $\boldsymbol{u}$,$\boldsymbol{v}$: $E(\boldsymbol{u}^T \boldsymbol{v}) = Tr\, E(\boldsymbol{v}\boldsymbol{u}^T)$
  - $Tr(A) = \sum_i A_{ii} =$ sum of diagonals
- Therefore, variance averaged over $\boldsymbol{x}_{test}$ is:
  $E\, Var(\boldsymbol{x}_{test}) = \sigma^2 E[\phi(\boldsymbol{x}_{test})^T (\boldsymbol{A}^T \boldsymbol{A})^{-1} \phi(\boldsymbol{x}_{test})] = \sigma^2 Tr\{E[\phi(\boldsymbol{x}_{test})\phi(\boldsymbol{x}_{test})^T](\boldsymbol{A}^T \boldsymbol{A})^{-1}\}$

$$= \frac{\sigma^2}{N} Tr\left\{\sum_i \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T\, (\boldsymbol{A}^T \boldsymbol{A})^{-1}\right\} = \frac{\sigma^2}{N} Tr\{(\boldsymbol{A}^T \boldsymbol{A})(\boldsymbol{A}^T \boldsymbol{A})^{-1}\} = \frac{\sigma^2}{N} Tr\{I_p\} = \frac{\sigma^2 p}{N}$$

Case with Equal Test & Training Distributions

- ❑Assumption on previous slide: Test point $\boldsymbol{x}_{test}$ is randomly selected from training data
- ❑Then, average variance is given by

$$E\ Var(\boldsymbol{x}_{test}) = \frac{\sigma^2 p}{N}$$

- ❑Increases with number of parameters p
  - Shows that increasing model complexity increases variance error
- ❑Decreases with number of samples N
- ❑What if test data point is distributed differently from training data?
  - Then variance may be much larger $\frac{\sigma^2 p}{N}$
  - If test data is not like training data, we are extending model to regions not seen in training data
  - Often leads to high error

Summary of Results for Linear Models

- ❑Suppose model is linear with $N =$ num samples, p = num parameters
- ❑Result 1: When $N < p$, linear estimate is not unique
  - Need at least as many samples as parameters
- ❑Now assume that $N \geq p$ and parameter estimate is unique
- ❑Result 2: When there is no under-modeling, estimate is unbiased

$$E\left[f\left(\boldsymbol{x}_{test},\widehat{\boldsymbol{\beta}}\right)\right] = f_0\ (\boldsymbol{x}_{test},).$$

- ❑Result 3: If test point drawn from same distribution as training data:

$$Var = \frac{p}{n}\sigma_\epsilon^2$$

  - Variance increases linearly with number of parameters and inversely with number of samples

Outline

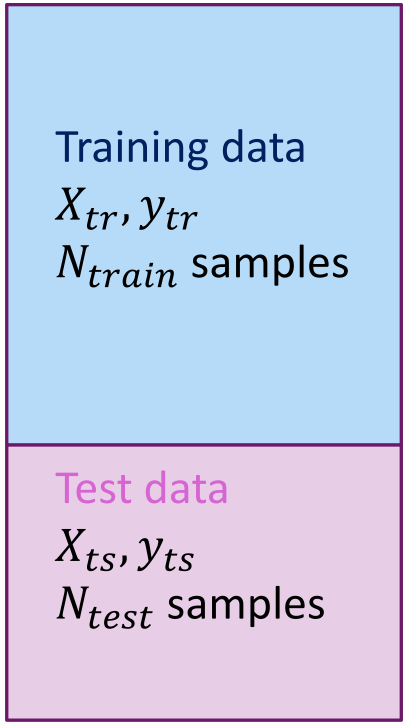
- ❑Motivating Example: What polynomial degree should a model use?
- ❑Bias and variance
- ❑Bias and variance for linear models (Advanced)
- ❑Cross-validation

Cross Validation

- □ Key idea: Evaluate on samples different from training
- □ Get data X, y
- □ Split into training X_{tr}, y_{tr} and test X_{ts}, y_{ts}
- □ For $p = 1$ to p_{max} // Loop over model order
  - Fit on training data with model order p
  - Predict values on test data
  - Score fit on test data (e.g. measure RSS)
- □ Select model order with smallest score:

$$\hat{p} = \arg\min_{p} S[p]$$

  - Maximize if higher score is better



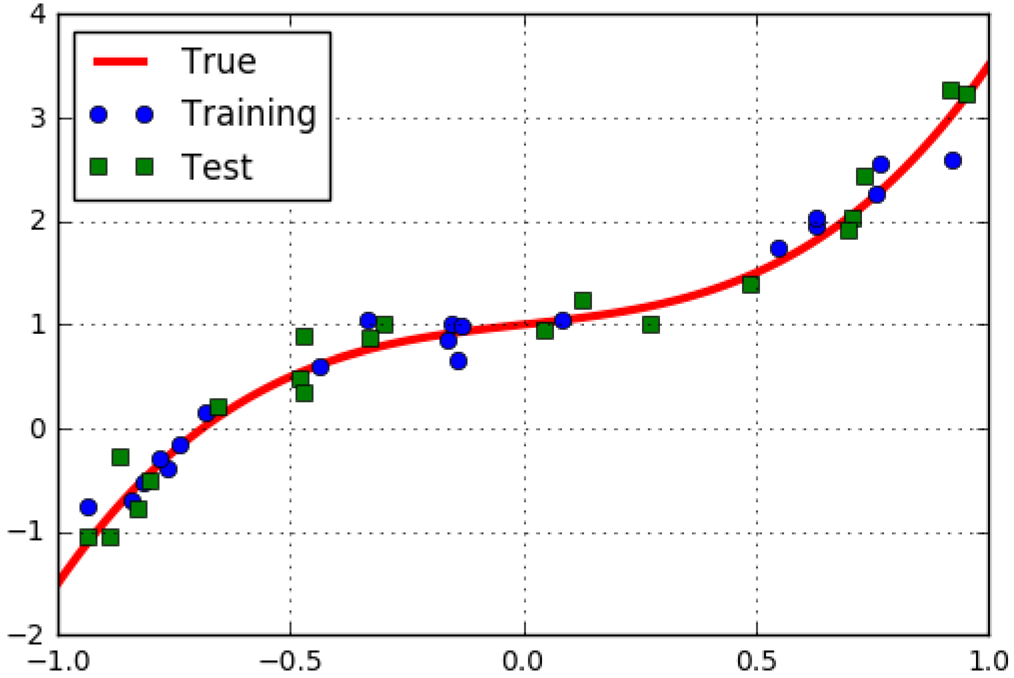
$$\hat{\beta} = \text{fit}(X_{tr}, y_{tr}, p)$$

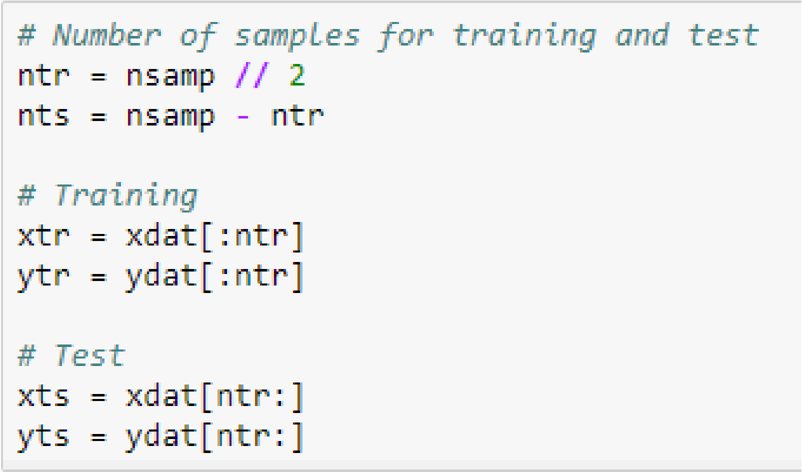
$$\hat{y}_{ts} = \text{predict}\left(X_{ts}, \hat{\beta}\right)$$

$$\text{S}[p] = \text{score}(y_{ts}, \hat{y}_{ts})$$

Polynomial Example: Training Test Split

- Example: Split data into 20 samples for training, 20 for test

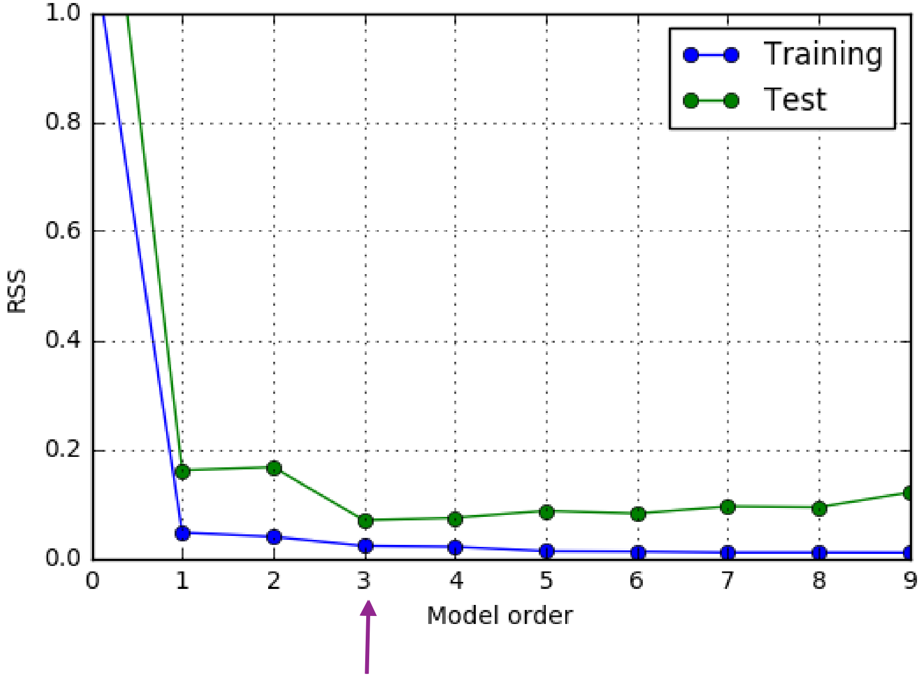




```
# Number of samples for training and test
ntr = nsamp // 2
nts = nsamp - ntr

# Training
xtr = xdat[:ntr]
ytr = ydat[:ntr]

# Test
xts = xdat[ntr:]
yts = ydat[ntr:]
```

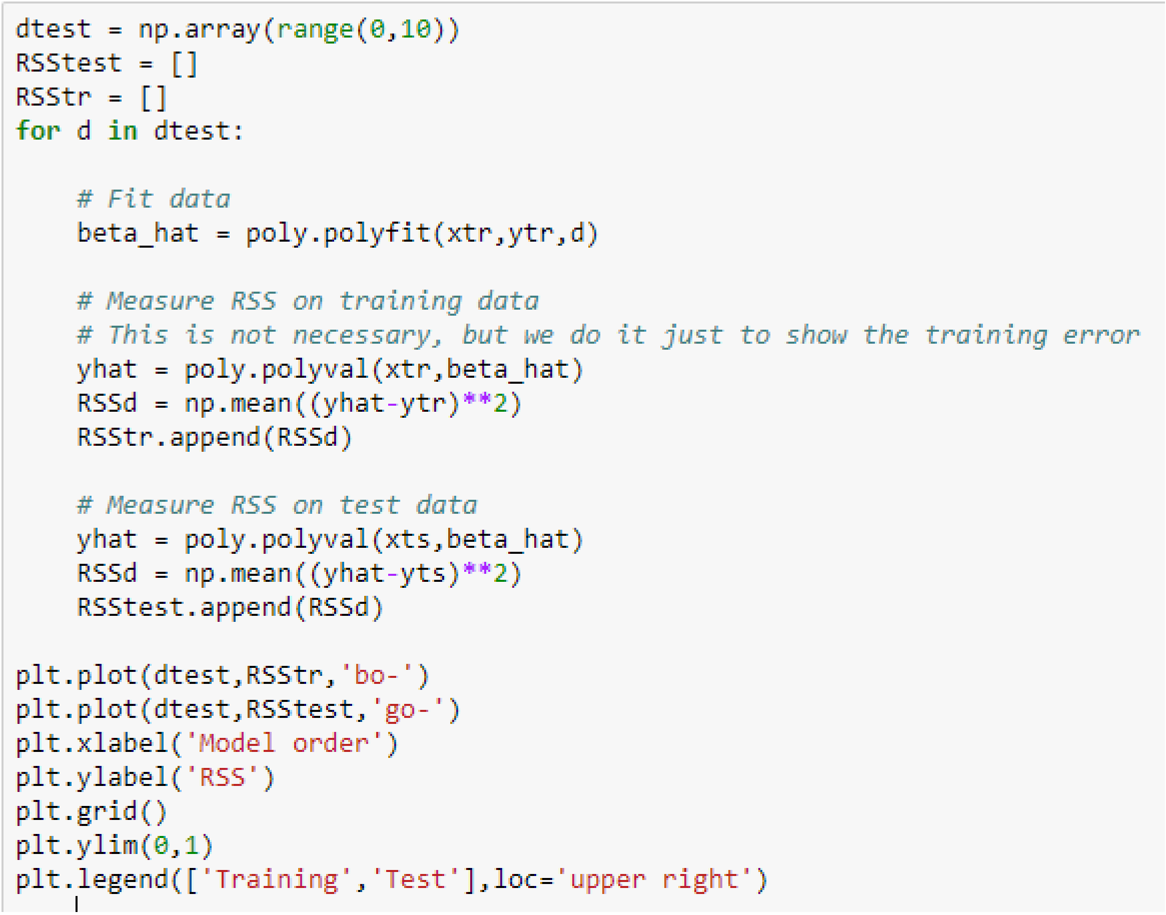
Finding the Model Order

- ❑ Estimated optimal model order = 3



RSS test minimized at $d = 3$

RSS training always decreases



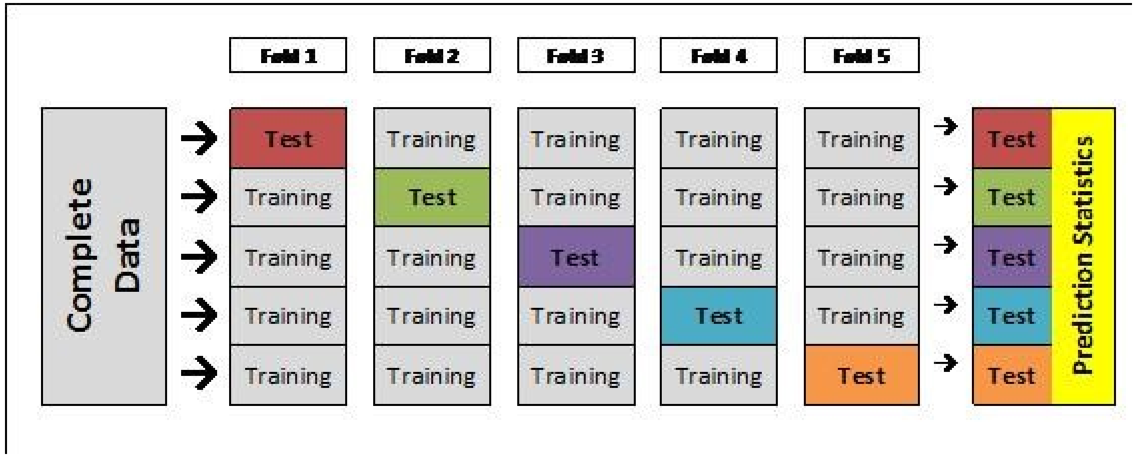
```
dtest = np.array(range(0,10))
RSStest = []
RSStr = []
for d in dtest:

    # Fit data
    beta_hat = poly.polyfit(xtr,ytr,d)

    # Measure RSS on training data
    # This is not necessary, but we do it just to show the training error
    yhat = poly.polyval(xtr,beta_hat)
    RSSd = np.mean((yhat-ytr)**2)
    RSStr.append(RSSd)

    # Measure RSS on test data
    yhat = poly.polyval(xts,beta_hat)
    RSSd = np.mean((yhat-yts)**2)
    RSStest.append(RSSd)

plt.plot(dtest,RSStr,'bo-')
plt.plot(dtest,RSStest,'go-')
plt.xlabel('Model order')
plt.ylabel('RSS')
plt.grid()
plt.ylim(0,1)
plt.legend(['Training','Test'],loc='upper right')
```

Problems with Simple Train/Test Split

- ❑ Test error could vary significantly depending on samples selected
- ❑ Only use limited number of samples for training
- ❑ Problems particularly bad for data with limited number of samples

K-Fold Cross Validation

- ❑ K-fold cross validation
  - Divide data into K parts
  - Use $K-1$ parts for training. Use remaining for test.
  - Average over the K test choices
  - More accurate, but requires K fits of parameters
  - Typical choice: K=5 or 10
  - Average MSE over K folds estimates the total MSE
  - (=Bias^2+Variance+irreducible error)



Complete Data

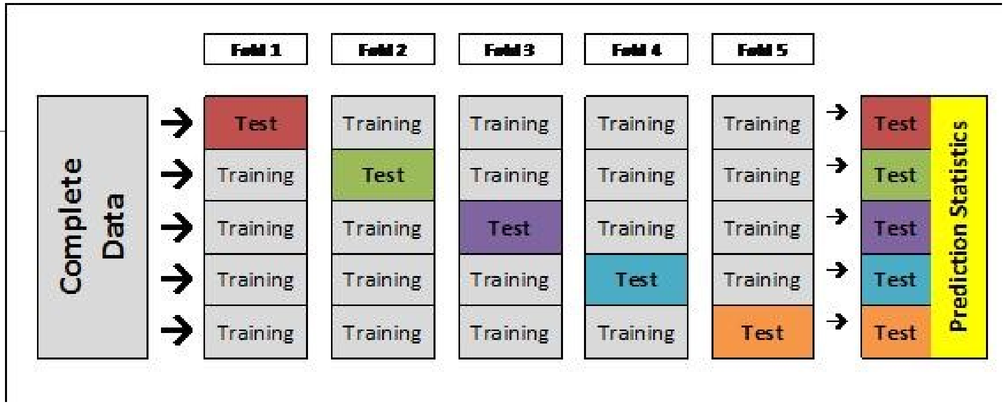
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	
→	Test	Training	Training	Training	Training	→ Test
→	Training	Test	Training	Training	Training	→ Test
→	Training	Training	Test	Training	Training	→ Test
→	Training	Training	Training	Test	Training	→ Test
→	Training	Training	Training	Training	Test	→ Test

Prediction Statistics

- ❑ Leave one out cross validation (LOOCV)
  - Take $K = N$ so one sample is left out.
  - Most accurate, but requires N model fittings
  - Necessary when *N* is small.

From http://blog.goldenhelix.com/goldenadmin/cross-validation-for-genomic-prediction-in-svs/

K-Fold Pseudo-Code

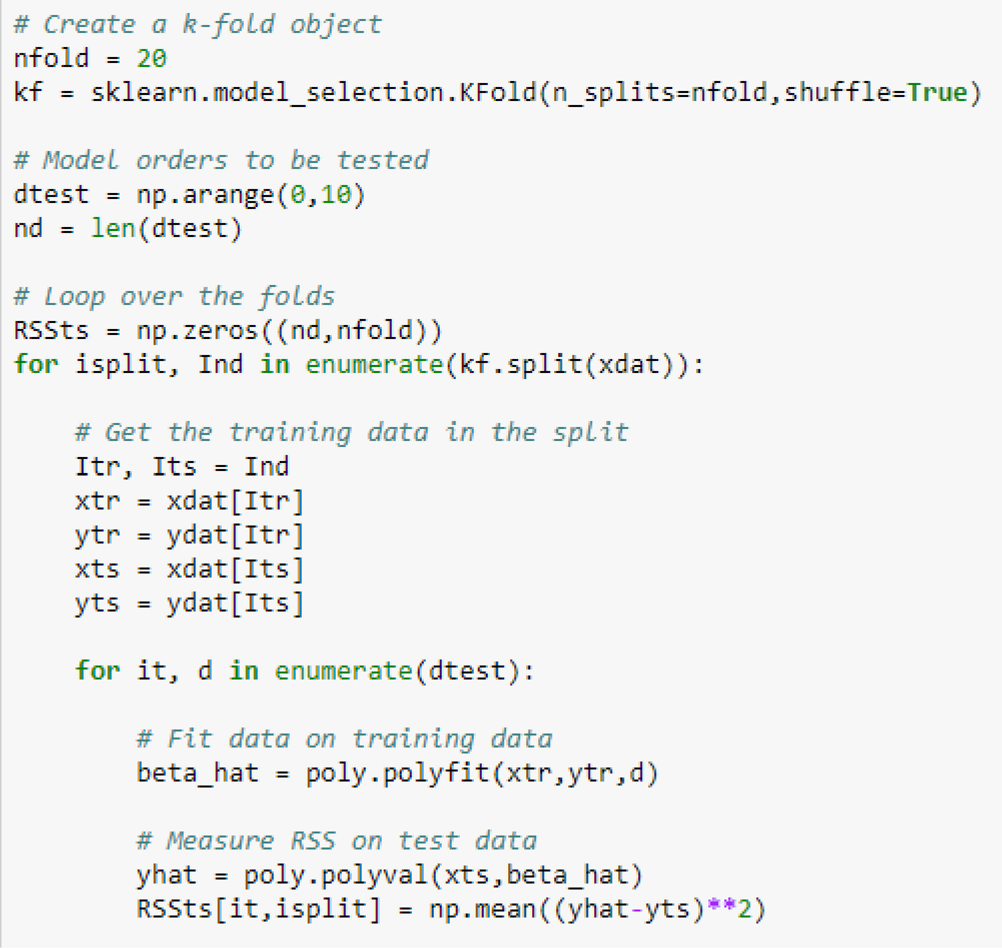


	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5		
Complete Data	Test	Training	Training	Training	Training	Test	Prediction Statistics
	Training	Test	Training	Training	Training	Test	
	Training	Training	Test	Training	Training	Test	
	Training	Training	Training	Test	Training	Test	
	Training	Training	Training	Training	Test	Test	

- Get data X, y
- For $i = 1$ to K // Loop over folds
  - Split into training X_{tr}, y_{tr} and test X_{ts}, y_{ts} for fold i
  - For $p = 1$ to p_{max} // Loop over model order
    - Fit on training data with model order p : $\hat{\beta} = \mathrm{fit}(X_{tr}, y_{tr}, p)$
    - Predict values on test data: $\hat{y}_{ts} = \mathrm{predict}(X_{ts}, \hat{\beta})$
    - Score the fit on test data: $S[p, i] = \mathrm{score}(y_{ts}, \hat{y}_{ts})$
- Find average score for each model order: $\bar{S}[\mathrm{p}] = \frac{1}{K}\sum_{i=1}^{K} S[p, i]$
- Select model order with lowest average score: $\hat{p} = \arg\min_{p} \bar{S}[p]$

Polynomial Example

- ❑ Use sklearn Kfold object
- ❑ Loop
  - Outer loop: Over K folds
  - Inner loop: Over D model orders
  - Measure test error in each fold and order
  - Averaging test errors from K folds for each model order
  - Find the model order with the minimal average test errors
  - Can be time-consuming



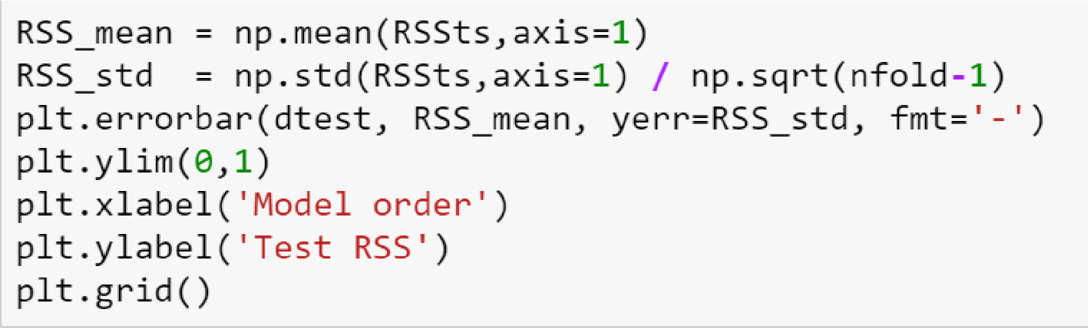
```
# Create a k-fold object
nfold = 20
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)

# Model orders to be tested
dtest = np.arange(0,10)
nd = len(dtest)

# Loop over the folds
RSSts = np.zeros((nd,nfold))
for isplit, Ind in enumerate(kf.split(xdat)):

    # Get the training data in the split
    Itr, Its = Ind
    xtr = xdat[Itr]
    ytr = ydat[Itr]
    xts = xdat[Its]
    yts = ydat[Its]

    for it, d in enumerate(dtest):

        # Fit data on training data
        beta_hat = poly.polyfit(xtr,ytr,d)

        # Measure RSS on test data
        yhat = poly.polyval(xts,beta_hat)
        RSSts[it,isplit] = np.mean((yhat-yts)**2)
```

Polynomial Example CV Results

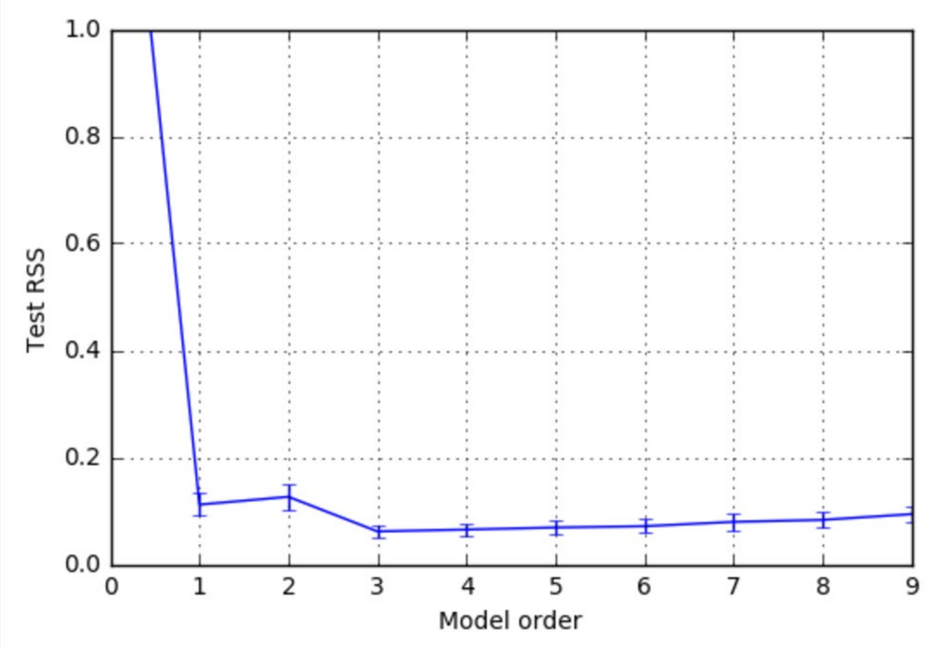
- ❑For each model order d
  - Compute mean test RSS over K folds
  - Compute standard error (SE) of test RSS
  - SE=STD of mean RSS=RSS std/ $\sqrt{K-1}$
  - (expectation over different realizations of data in each fold)

- ❑Simple model selection
  - Select d with lowest mean test RSS

- ❑For this example
  - Estimate model order = 3



```
RSS_mean = np.mean(RSSts,axis=1)
RSS_std  = np.std(RSSts,axis=1) / np.sqrt(nfold-1)
plt.errorbar(dtest, RSS_mean, yerr=RSS_std, fmt='-')
plt.ylim(0,1)
plt.xlabel('Model order')
plt.ylabel('Test RSS')
plt.grid()
```

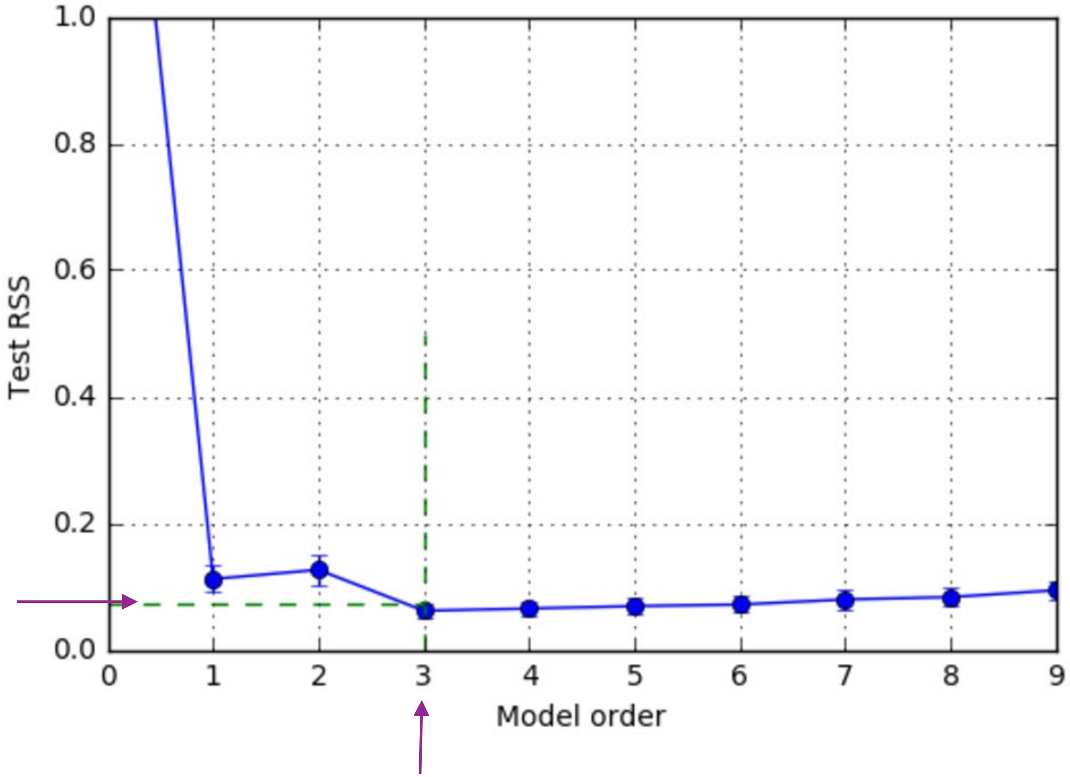


One Standard Error Rule

- ❑ Previous slide: Select d to minimize RSS_mean[d]
  - Average RSS across the folds and select lowest average

- ❑ Problem: Often over-predicts model order

- ❑ One standard deviation rule
  - Use simplest model "within one SE of minimum"



Optimal model order

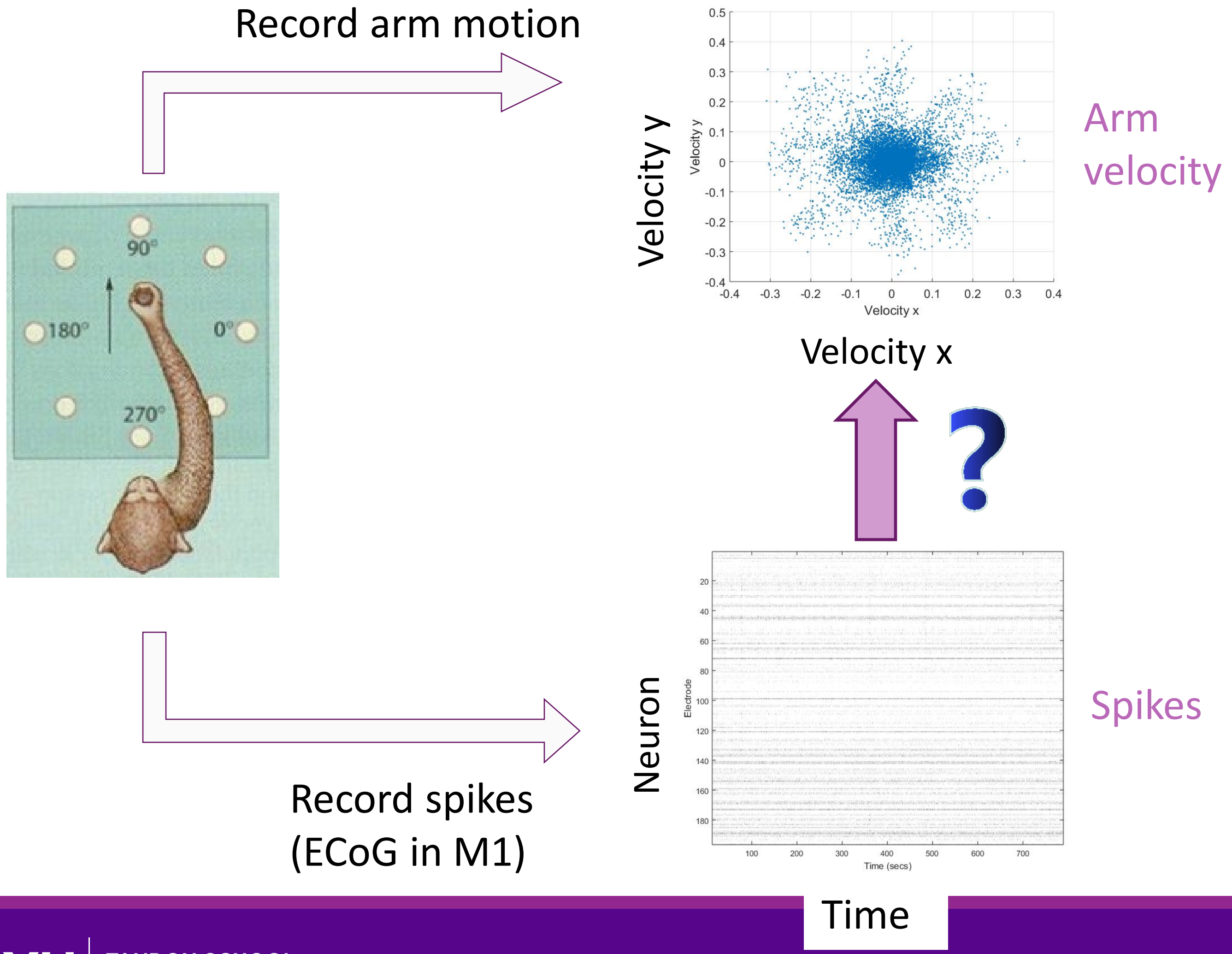
One SE Rule Pseudo-Code

- Get data X, y
- Compute score as before: $\mathrm{S}[p, i]$ = score for model order p on fold i
- Compute average, std deviation and standard error of the scores:
  - $\bar{S}[\mathrm{p}] = \frac{1}{K}\sum_{i=1}^{K} S[p,i]\,,\ \ \sigma^2[p] = \frac{1}{K}\sum_{i=1}^{K}(S[p,i] - \bar{S}[\mathrm{p}])^2\,,\ \ SE[p] = \frac{\sigma[p]}{\sqrt{K-1}}$
- Find model order via normal rule: $\hat{p}_0 = \arg\min_p \bar{S}[p]$ (lowest average score)
- Compute target score: $S_{tgt} = \bar{S}[p_0] + SE[p_0]$
- One SE rule: Find simplest model with score lower than target:

$$\hat{p} = \min\{p \mid \bar{S}[p] \le S_{tgt}\}$$

- Note that one SE rule always produce a model order ≤ normal rule ($\hat{p} \le \hat{p}_0$)

Lab: Neural ECoG Data



Record arm motion

Velocity y

Velocity x

Arm velocity

?

Neuron

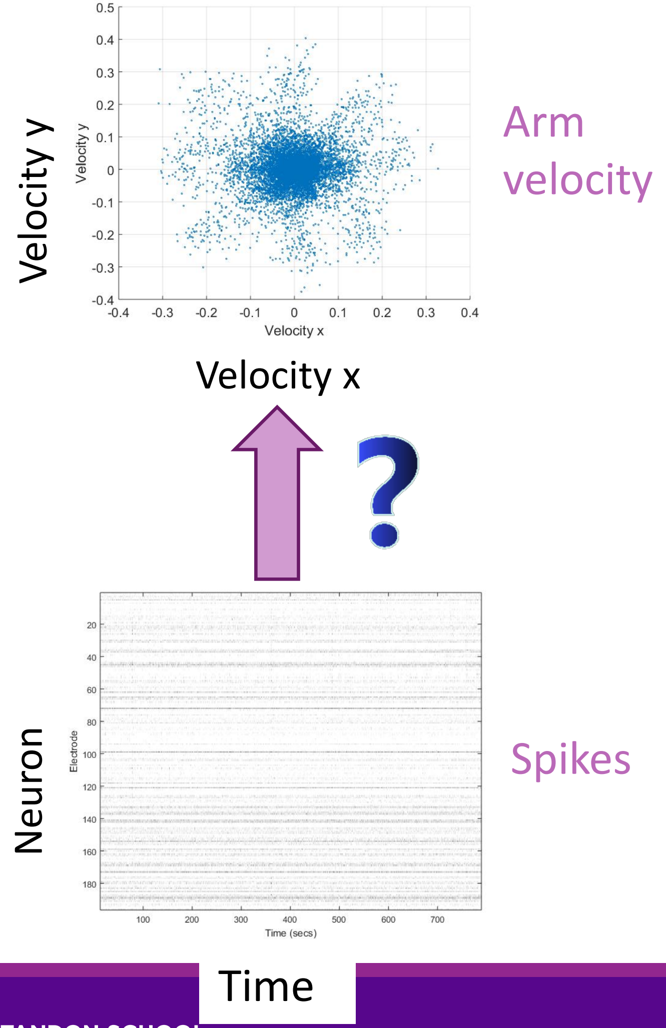
Time

Spikes

Record spikes (ECoG in M1)

- □ Read a monkey’s brain!
- □ Predict hand motion from electrode measurements (number of spikes in each neuron)
- □ Data from https://crcns.org/
  - Open source website on neural data
  - Great for projects

Lab: Neural ECoG Data



- ❑ Linear filter model:

$$y[t,k] = \sum_{\ell=0}^{d} \sum_{j=0}^{p-1} X[t-\ell,j]W[\ell,j,k] + b[k]$$

  - $X[t,j]$ = spikes from neuron j at time t
  - $y[t,k]$ = Output k at time t (two outputs for x and y motion)
  - $W[\ell,j,k]$ = weight from neuron j to output k at time delay ℓ

- ❑ Linear fitting: Find W and b from X and y

- ❑ Model order selection:
  - Find optimal maximum delay d
  - Higher d allows better fit, but uses more parameters