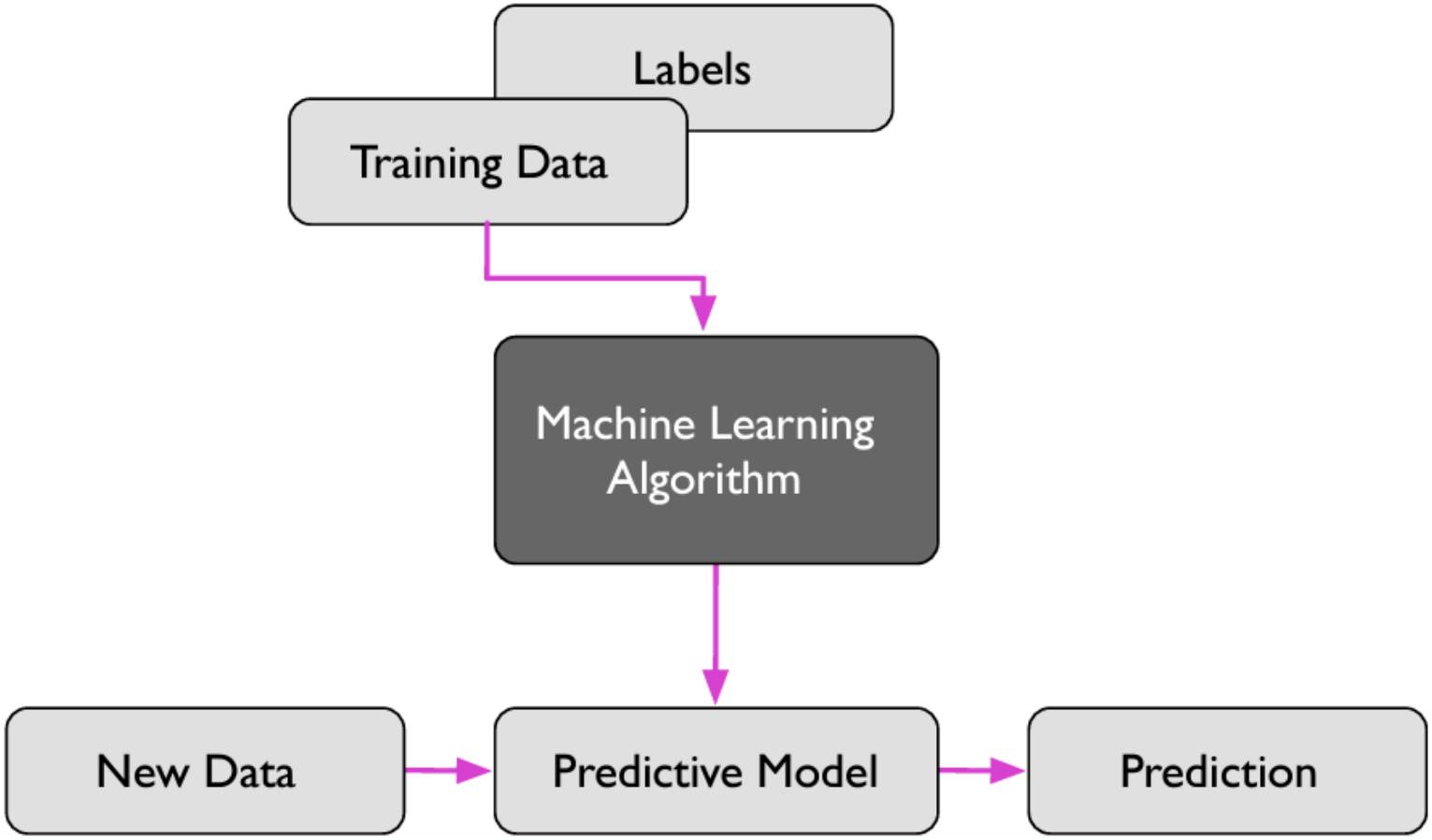
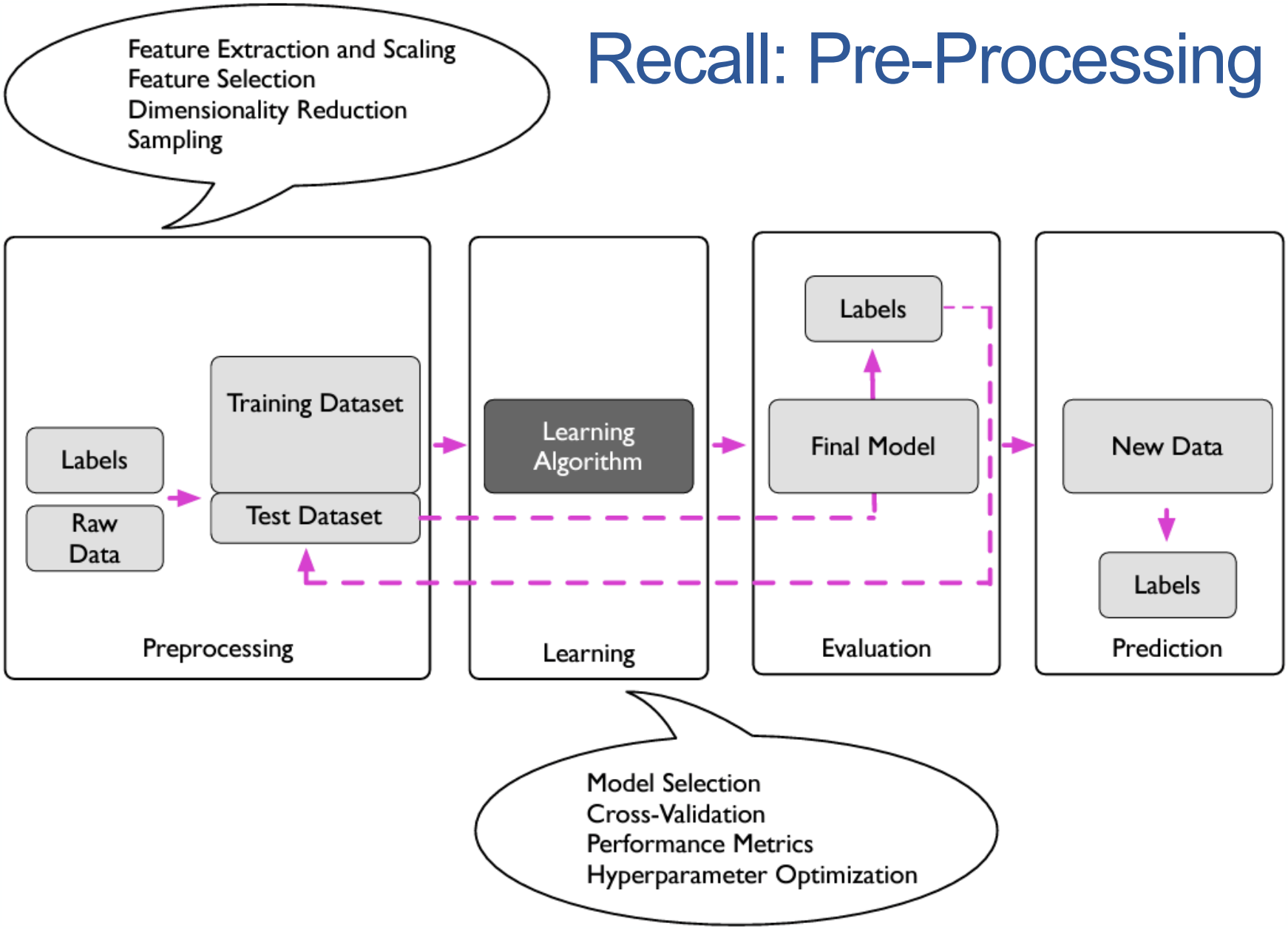
LECTURE 4: MODEL ORDER SELECTION AND CROSS VALIDATION

Ehsan Aryafar
earyafar@pdx.edu
http://web.cecs.pdx.edu/~aryafare/ML.html

Recall: Supervised Learning



Recall: Pre-Processing



Recall: Multivariable Linear Model for Glucose

Attributes

$$y \approx \hat{y} = f(x_1, \ldots, x_{10})$$

Age, Sex, BMI,BP,S1, …, S6

$\boldsymbol{x} = [x_1, \ldots, x_{10}]$

Target

y =Glucose level

- Goal: Find a function to predict glucose level from the 10 attributes
- Linear Model: Assume glucose is a linear function of the predictors:

$$[\text{glucose}] \approx [\text{prediction}] = \beta_0 + \beta_1[\text{Age}] + \cdots + \beta_4[BP] + \beta_5[\text{S1}] + \cdots + \beta_{10}[\text{S6}]$$

- General form:

$$y \approx \hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_4 x_4 + \beta_5 x_5 + \cdots + \beta_{10} x_{10}$$

Target

Intercept

10 Features

Recall: Matrix Form of Linear Regression

- Data: $(\boldsymbol{x}_i, y_i), i = 1, \ldots, n$
- Predicted value for i-th sample: $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$
- Matrix form

$\hat{\boldsymbol{y}}$ a n predicted values

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$$

$\boldsymbol{\beta}$ with $p = k + 1$ coefficient vector

$\boldsymbol{A}$ a $n \times p$ feature matrix

- Matrix equation: $\hat{\boldsymbol{y}} = \boldsymbol{A}\,\boldsymbol{\beta}$

Recall: Least Squares Solution

- Consider cost function of the RSS:

$$\mathrm{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 , \qquad \hat{y}_i = \sum_{j=0}^{p} A_{ij}\beta_j$$

  - Vector $\boldsymbol{\beta}$ that minimizes RSS called the least-squares solution

- Least squares solution: The vector $\boldsymbol{\beta}$ that minimizes the RSS is:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y}$$

  - Can compute the best coefficient vector analytically
  - Just solve a linear set of equations
  - Will show the proof below

Recall: Fitting Transformed Linear Models

- Consider transformed linear model

$$\hat{y} = \beta_1 \phi_1(\boldsymbol{x}) + \cdots + \beta_p \phi_p(\boldsymbol{x})$$

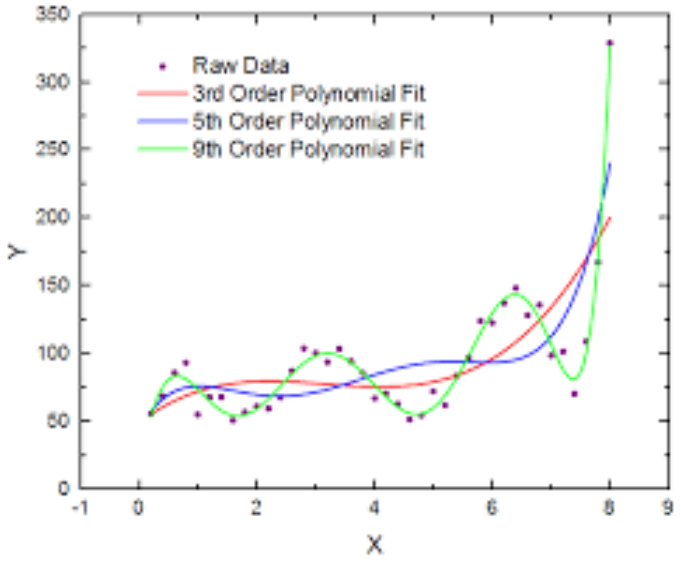
- We can fit this model exactly as before
  - Given data $(\boldsymbol{x}_i, y_i), i = 1, \ldots, N$
  - Want to fit the model from the transformed variables $\phi_j(\boldsymbol{x})$ to target y
  - Define the transformed matrix:

$$A = \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_p(\boldsymbol{x}_1) \\ \vdots & \vdots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \cdots & \phi_p(\boldsymbol{x}_N) \end{bmatrix}$$

  - Predictions: $\hat{y} = A\beta$
  - Least squares fit $\beta = (A^T A)^{-1} A^T y$

Recall: Example: Polynomial Fitting

- Suppose y only depends on a single variable x,
- Want to fit a polynomial model
  - $y \approx \beta_0 + \beta_1 x + \cdots \beta_d x^d$
- Given data $(x_i, y_i), i = 1, \ldots, n$
- Take basis functions $\phi_j(x) = x^j, j = 0, \ldots, d$
- Transformed model: $\hat{y} = \beta_0 \phi_0(x) + \cdots + \beta_d \phi_d(x)$
- Transformed matrix is:



$$\boldsymbol{A} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^d \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & \cdots & x_n^d \end{bmatrix}, \qquad \beta = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_d \end{bmatrix}$$

  - $p = d + 1$ transformed features from **1 original feature**
- Will discuss how to select d in the next lecture

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

- Math Background
- Motivating Example: What polynomial degree should a model use?
- Bias and variance
- Cross-validation

Recall From Probability Theory

- What is a random variable?
  - Give an example of a random variable?
  - Is a random variable discrete or continuous?

Recall From Probability Theory

- What is a random variable?
  - Give an example of a random variable?
  - Is a random variable discrete or continuous?

- A random variable is a variable whose possible values are outcomes of a random phenomenon
  - We use a capital letter, like X, to denote a random variable
    - Think of it as default variable name in a programming language
  - The value of a random variable will be denoted with a lower case letter, in this case x
    - For example, P(X = x)
  - There are two types of random variables: discrete and continuous

Discrete Random Variable

- The space of outcomes (also referred to as state space) is discrete

- Example: X is a discrete random variable that shows the outcome of a fair dice roll
  - State space?

Discrete Random Variable

- The space of outcomes (also referred to as state space) is discrete

- Example: X is a discrete random variable that shows the outcome of a fair dice roll
  - State space = {1, 2, 3, 4, 5, 6}
  - Let x_i (i= 1, …, 6) denote the possible outcomes
  - $P(X = x_i) = 1/6$

- Example: X is a fair coin toss
  - State space = {heads, tails}
  - Let x_1 denote heads and x_2 denote tails
  - $P(X = x_1) = P(X = x_2) = ½$

Advanced math: countably infinite
Not in Exam

- Size of state space could be infinite
  - Example: X is a discrete random variable with state space {1, 2, 3, …}

Continuous Random Variable

- Formal definition (advanced): a continuous RV differs from a discrete RV in that it takes on an un-countably infinite number of possible outcomes

- Example: A random variable that can take any real number between 0 and 1 with equal probability
  - Q: How many real numbers are between 0 and 1?
  - Q: What is the size of the state space?

Probability Models

- A probability model of a random variable consists of:
  - The collection of all possible values of a random variable
  - The probabilities that the values occur
- For a discrete random variable the probability model lists the possible values the random variable takes and the probability which it takes over those values
- Note that these are theoretical distributions as opposed to empirical distributions which come from data

Example – Discrete Probability Model

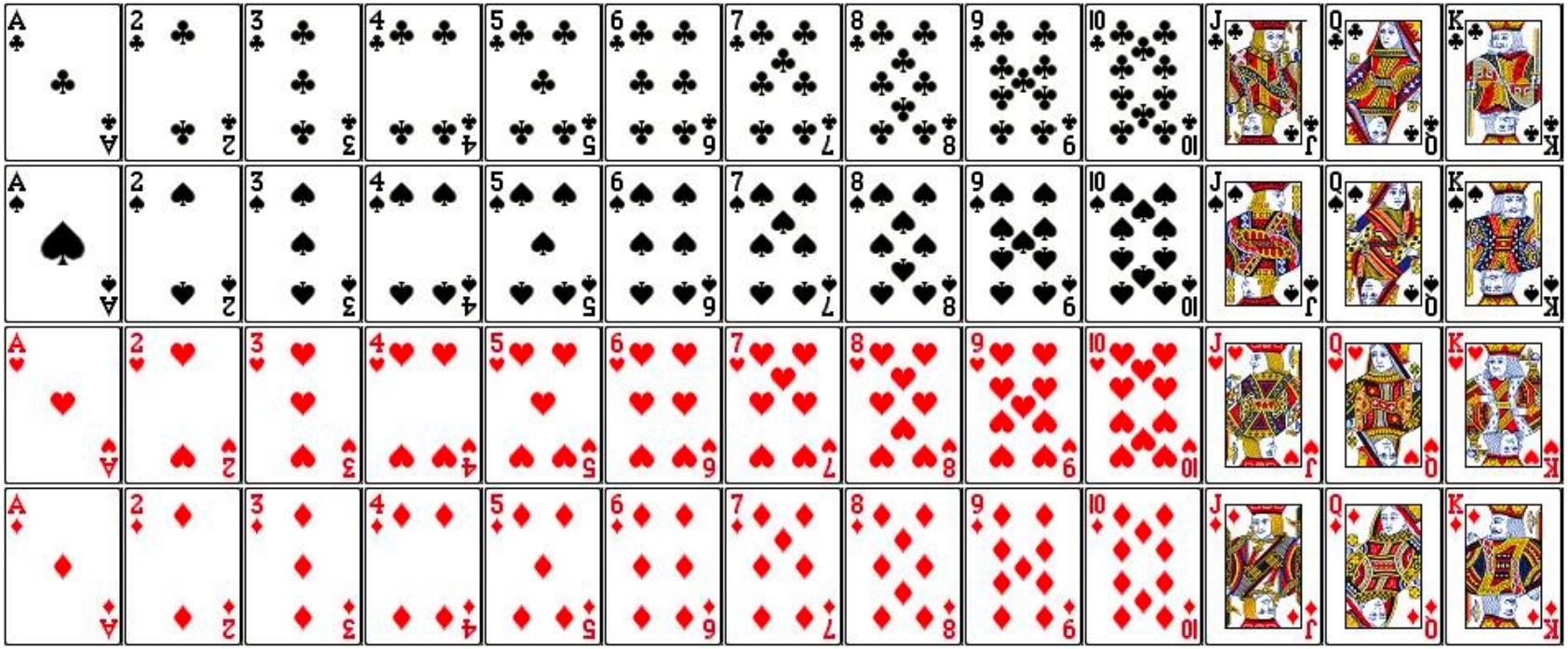
- You usually show it through a table
- Example: Probability distribution of a coin toss

Event	Heads	Tails
Probability	0.5	0.5

- Rules for discrete probability distributions
  - The events listed must be disjoint
  - Each probability must be between 0 and 1
  - The sum of probabilities must equal 1

Example – Discrete Probability Model

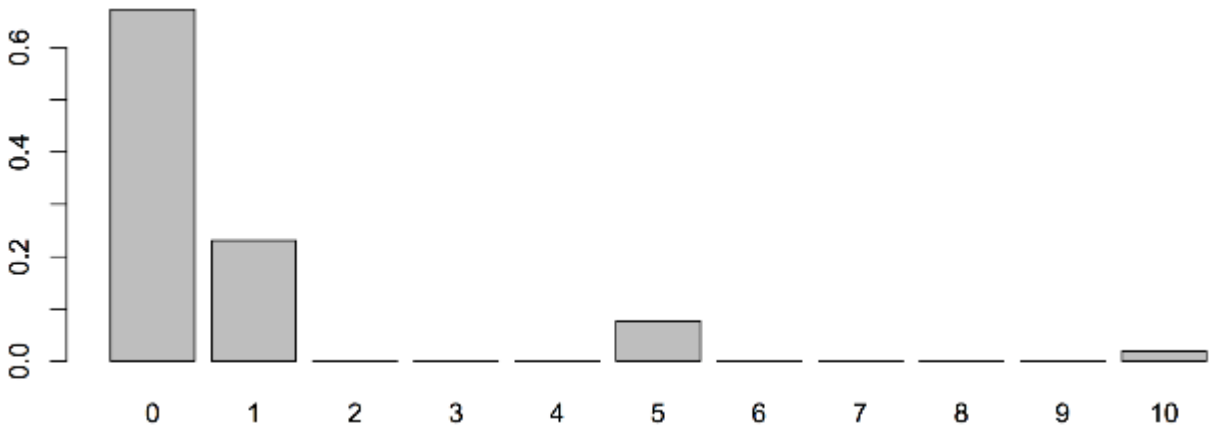
- In a game of cards you win $1 if you draw a heart, $5 if you draw an ace (including the ace of hearts), $10 if you draw the king of spades and nothing for any other card you draw. Write the probability model for your winnings.



Example – Discrete Probability Model

- In a game of cards you win $1 if you draw a heart, $5 if you draw an ace (including the ace of hearts), $10 if you draw the king of spades and nothing for any other card you draw. Write the probability model for your winnings.

Event X	0	1	5	10
Probability P(X)	35/52	12/52	4/52	1/52



Example – Continuous Probability Model

- Note that the goal of a probability model is to show the probability of an event happening

- Example: Let X denote a continuous random variable that can be any real number between 0 and 1
  - What is the probability that X = 0.5?

How to show probability model for continuous variables?

Example – Continuous Probability Model / Continuous Random Variable: Uniform Distribution

X~ U(a, d) means uniform distribution between a and d

- i.e., any value in the interval between a and d is equally probable
- **Question:** what is P(X = x) for some x in [a, d]?
- Let a < b<c<d
  - P (X in interval [b c]) = (c-b)/(d-a)

What is the average value of a uniform random variable?
How do we show the probability model (CDF and PDF)?

Cumulative Distribution Function (Also Known as CDF Plot)

- In probability theory and statistics, the **cumulative distribution function (CDF)** of a real-valued random variable X evaluated at x, is the probability that X will take a value less than or equal to x
  - $F_X(x) = P(X <= x)$
  - Very commonly used
- Example I: suppose X is uniformly distributed on the interval [0, 1]. Then the CDF of X is given by

Plot the CDFs!

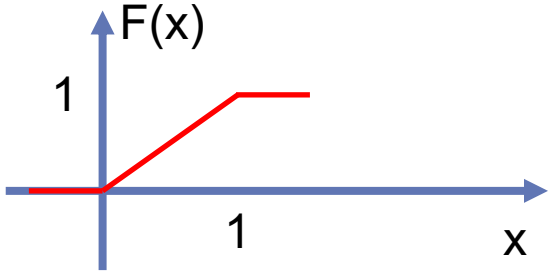
- Example II: Let X take the discrete values 0 and 1 with equal probability. Then the CDF of X is

Plot the CDFs!

Cumulative Distribution Function (Also Known as CDF Plot)

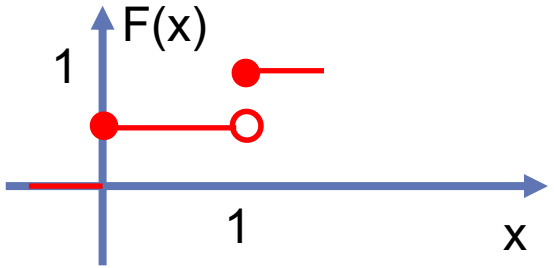
- In probability theory and statistics, the **cumulative distribution function (CDF)** of a real-valued random variable X evaluated at x, is the probability that X will take a value less than or equal to x
  - $F_X(x) = P(X <= x)$
  - Very commonly used
- Example I: suppose X is uniformly distributed on the interval [0, 1]. Then the CDF of X is given by

$$F(x) = \begin{cases} 0 & : x < 0 \\ x & : 0 \leq x < 1 \\ 1 & : x \geq 1. \end{cases}$$



- Example II: Let X take the discrete values 0 and 1 with equal probability. Then the CDF of X is

$$F(x) = \begin{cases} 0 & : x < 0 \\ 1/2 & : 0 \leq x < 1 \\ 1 & : x \geq 1. \end{cases}$$



CDF Continued

- If X is a purely discrete random variable, then it attains values x_1, x_2, … with probability p_i=P(xi) and the CDF of X will be discontinuous at the points x_i and constant in between:

$$F(x) = \mathrm{P}(X \leq x) = \sum_{x_i \leq x} \mathrm{P}(X = x_i) = \sum_{x_i \leq x} p(x_i).$$

- The CDF of a continuous random variable X can be expressed as the integral of its probability density function f_X as follows:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt.$$

Probability Density Function (PDF)

- Probability density function (f_x) specifies the probability of a random variable X falling **within a range of values**
  - The probability is the integral of the random variable over the range
  - For a continuous random variable probability of X being an exact value is 0

- If $F_X(x)$ is the CDF of X, then:

$$F_X(x) = \int_{-\infty}^{x} f_X(u)\,du,$$

$$f_X(x) = \frac{d}{dx} F_X(x).$$

PDF is an advanced concept! Not in Exam!

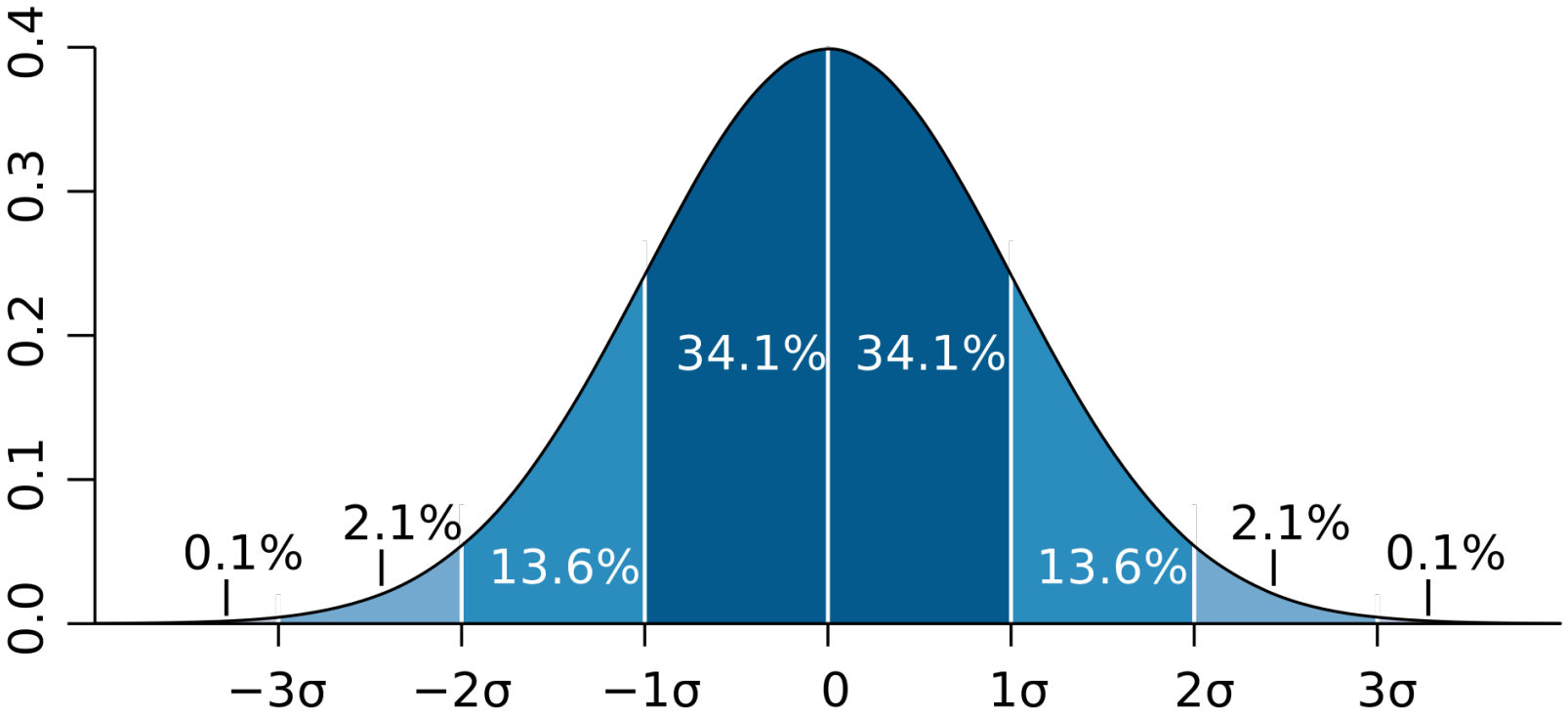
- Intuitively, one can think of $f_X(x)dx$ as the probability of X falling within the infinitesimal interval [x, $x+dx$]

Example: Gaussian (or Normal) Distribution

- Notation $\mathcal{N}(\mu, \sigma^2)$
- Parameters $\mu \in \mathbb{R}$ = mean

  $\sigma^2 \in \mathbb{R}_{>0}$ = variance
- PDF $\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

PDF of a Zero Mean Gaussian RV



For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

Expected Value

- Expected value or mean or average
- The expected value of a random variable, is the long-run average value of repetitions of the experiment it represents.
  - Example: The expected value in rolling a six-sided dice is 3.5
  - Also referred to as mean, average value, **first moment**

Calculating the Expected Value

- Let X be a discrete random variable with a finite number of outcomes x_1, …, x_k occurring with probabilities p_1, …, p_k. The expectation of X is defined as:

$$\mathrm{E}[X] = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k$$

- Example: rolling a fair six-sided dice

$$\mathrm{E}[X] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = 3.5$$

- If X is a continuous random variable, then the expected value is

Not in Exam!

$$\mathrm{E}[X] = \int_{-\infty}^{+\infty} x f(x)\, dx$$

$$\mathrm{E}[X] = \int_{-\infty}^{+\infty} \big(\mathbf{1} - \boldsymbol{FX}(\boldsymbol{x})\big) \boldsymbol{dx}$$

Outline

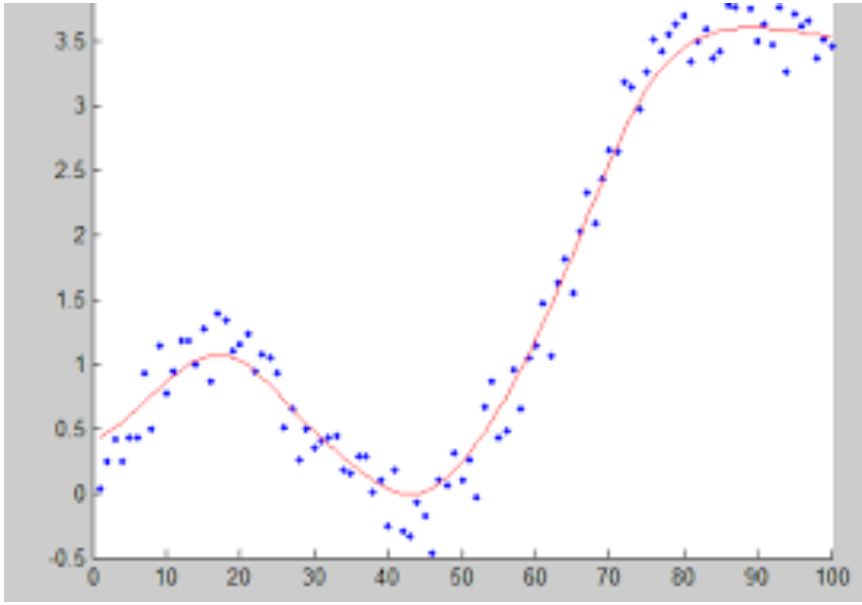
- Math Background
- Motivating Example: What polynomial degree should a model use?
- Bias and variance
- Cross-validation

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

Polynomial Model Order Selection

In this demo, we will illustrate the process of cross-validation for model order selection. We demonstrate the concepts via polynomial fitting using synthetic dat. The lab will demonstrate how to:

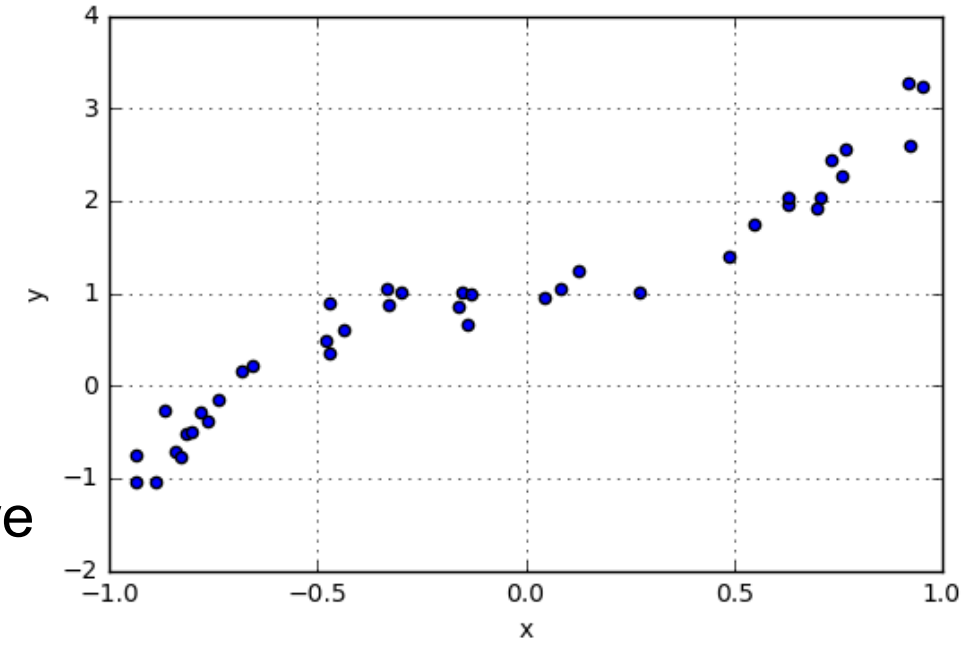
- Characterize the model order for a simple polynomial model
- Measure training and test error for a given model order
- Select a suitable model order using cross-validation
- Plot the results for the model order selection process

We first load the packages as usual.

[1]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model, preprocessing
%matplotlib inline
```

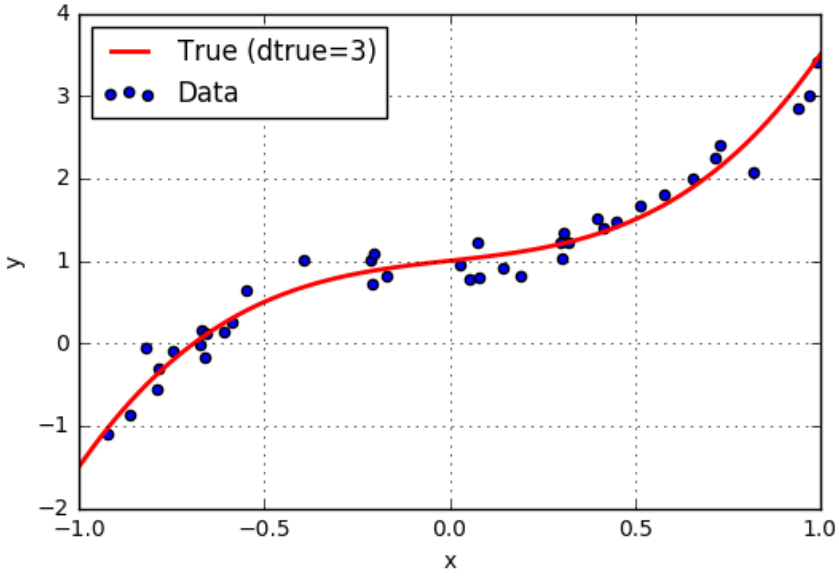
Example Question

- 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

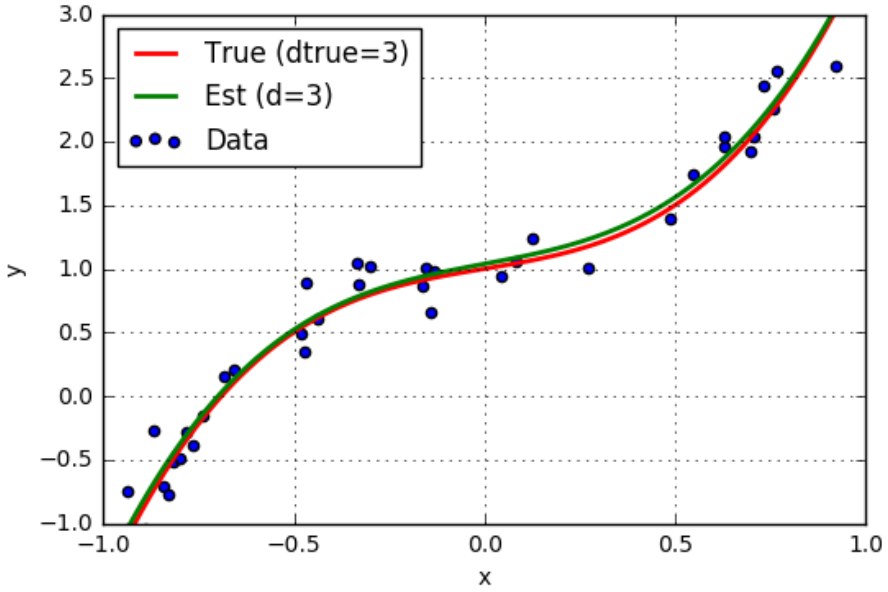
- Previous example is synthetic data
- x_i: 40 samples uniform in [-1,1]
- $y = f(x) + \epsilon$,
  - $f(x) = \beta_0 + \beta_1 x + \cdots + \beta_d x^d$ = "true relation"
  - $d = 3,\ \epsilon \sim N(0, \sigma^2)$
- Synthetic data useful for analysis
  - Know "ground truth"
  - Can measure performance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

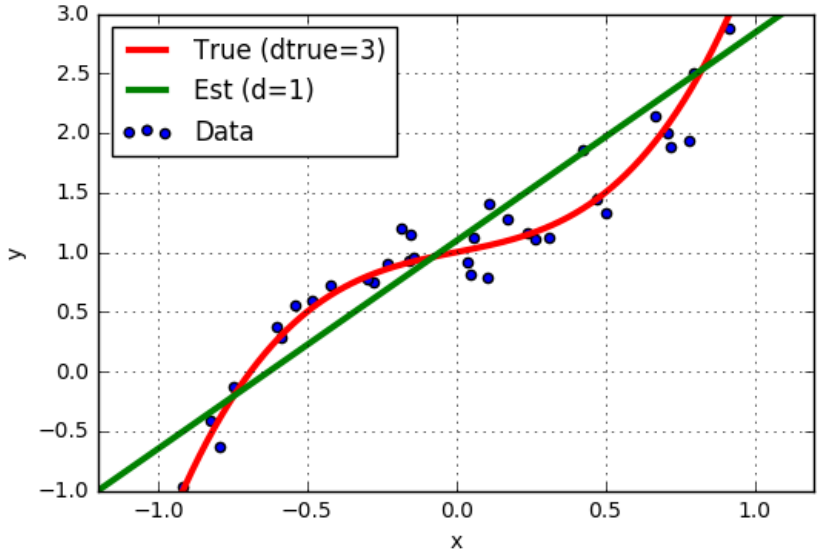
- Suppose true polynomial order, d=3, is known
- Use linear regression
  - numpy.polynomial package
- Get very good fit



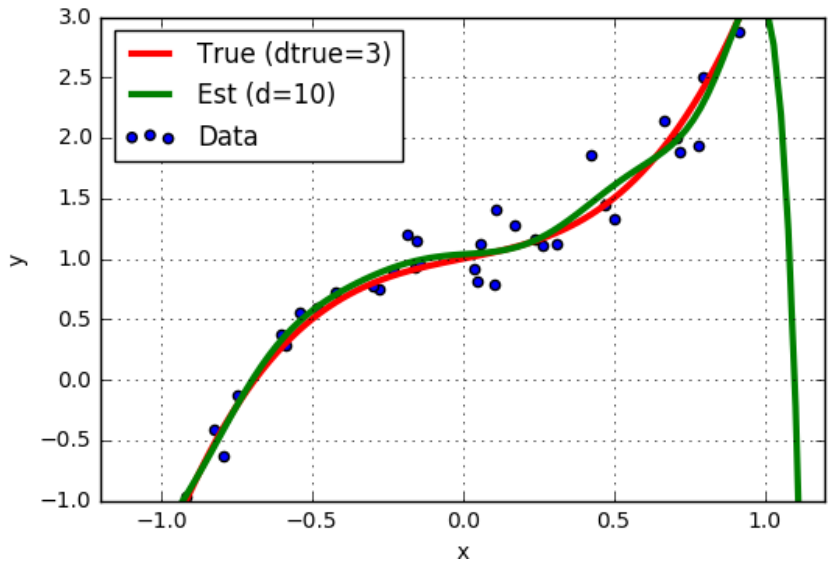

```
d = 3
beta_hat = poly.polyfit(xdat,ydat,d)

# 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?

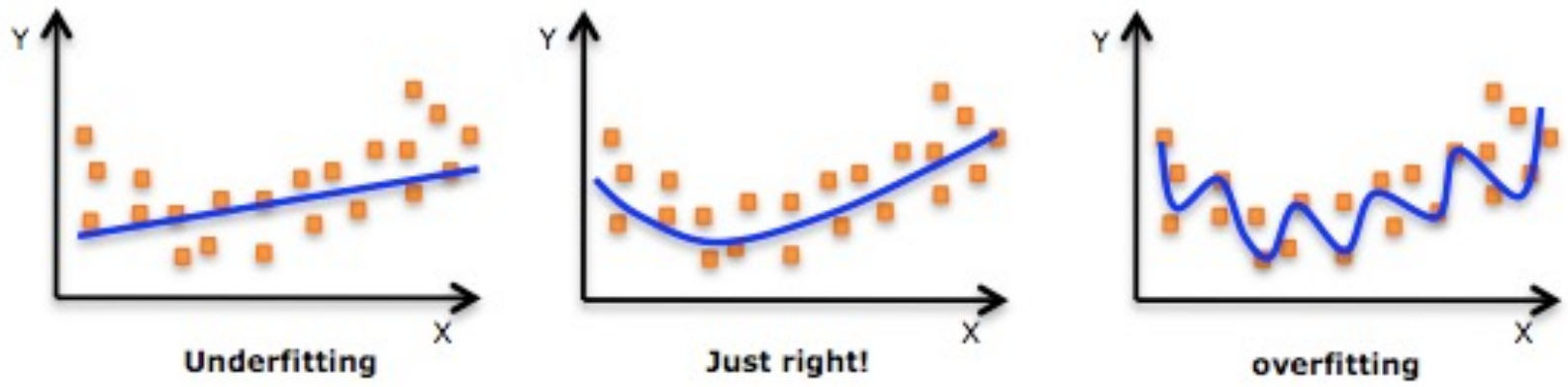


d=1 “Underfitting”



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?

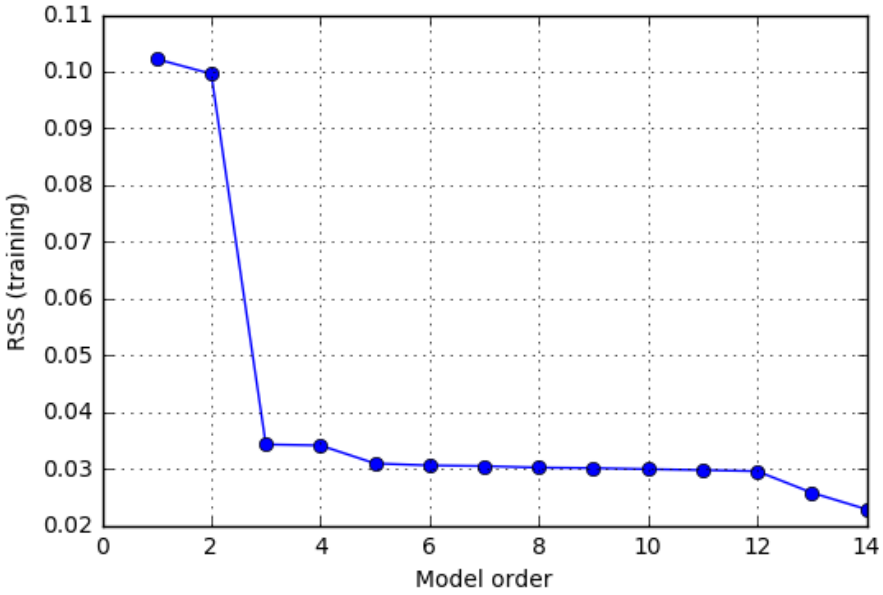
- Simple (but bad) idea:
  - For each model order, d, find estimate $\widehat{\boldsymbol{\beta}}$
  - Compute predicted values on training data

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

  - 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?

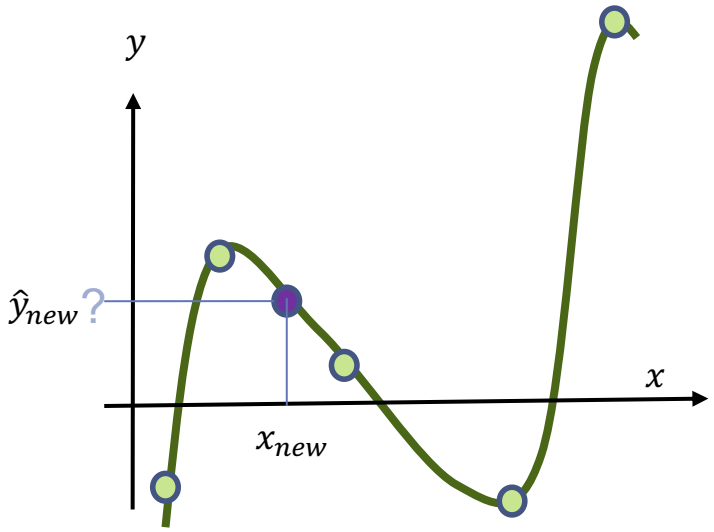


Outline

- Math Background
- Motivating Example: What polynomial degree should a model use?
- Bias and variance
- Cross-validation

Generalization

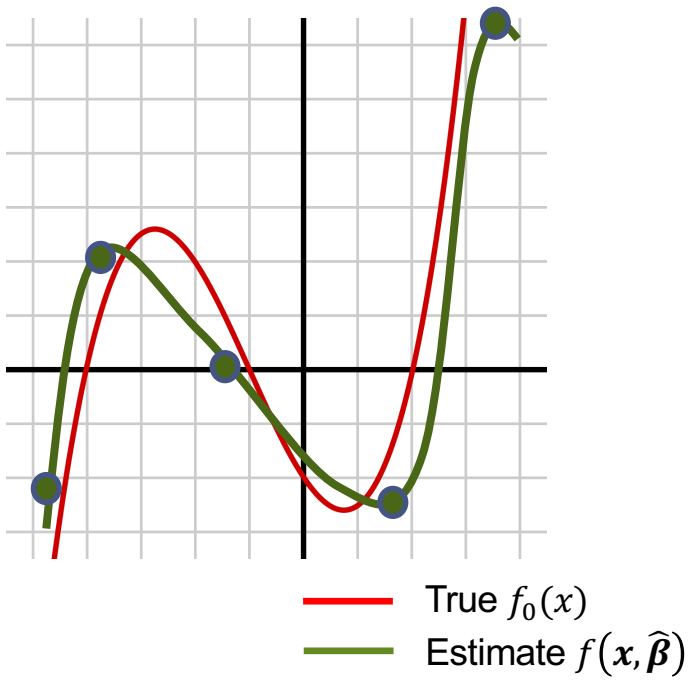
- Machine learning:
  - Get data points $(x_i, y_i), i = 1, \ldots, n$
  - Learn some function $\hat{y} = f(x)$
- Implicitly, we are
  - Inferring the value of y at new values of x
    - x_{new} is unseen data point
  - Called generalization
- Basic question for all ML:
  - *How well do models we train generalize to new samples?*
  - *What can we say about accuracy of new data points that we have not seen*



A Model To Understand Generalization

- Assume a true relation: $y = f_0(x) + \epsilon$, $\epsilon \sim N(0, \sigma_\epsilon^2)$ is noise
  - ϵ is a zero mean normal (Gaussian) random variable!
- 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? or how good the estimate generalizes w.r.t. true function?

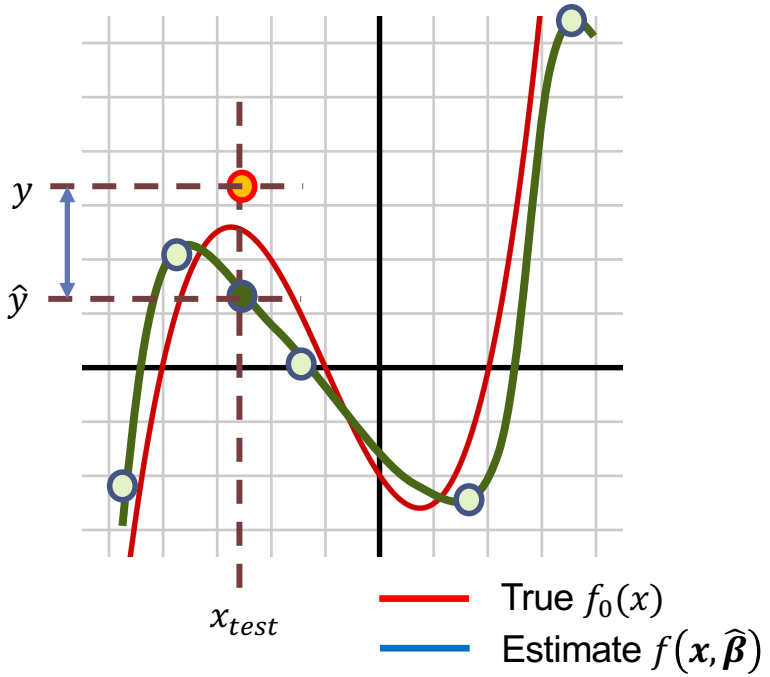


Output Mean Squared Error

- 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 :

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

  - There are two sources of randomness in Expectation: (i) noise $\boldsymbol{\epsilon}$ on the training (as you try to get $\widehat{\boldsymbol{\beta}}$) and test data (as you get y), (ii) evaluation will depend on the test value $\boldsymbol{x}_{test}$.



Function MSE and Irreducible Error

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

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

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

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

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

Due to independence of RVs

Model Class and Under-Modeling

- Model class: The set of all 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
  - You may not be able to estimate $\boldsymbol{\beta}_0$ but at least it exists

- 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: $f(x, \beta) = \beta_0 + \beta_1 x + \beta_2 x^2$
- Ex 2:
  - True function: $f_0(x) = 2 + 3x + 4x^2$ Model class: $f(x, \beta) = \beta_0 + \beta_1 x$

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: $f(x, \beta) = \beta_0 \sin(2\pi(5)x) + \beta_1 \cos(2\pi(5)x)$
- Ex 4:
  - True function: $f_0(x) = \sin(2\pi(8)x + 7)$ Model class: $f(x, \beta) = \beta_0 \sin(2\pi(5)x) + \beta_1 \cos(2\pi(5)x)$

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 under-modeling (i.e. true function is in model class), and
  - We can estimate the true parameter
- Then:
  - Output MSE = 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…
  - Particularly when you have a limited amount of training data

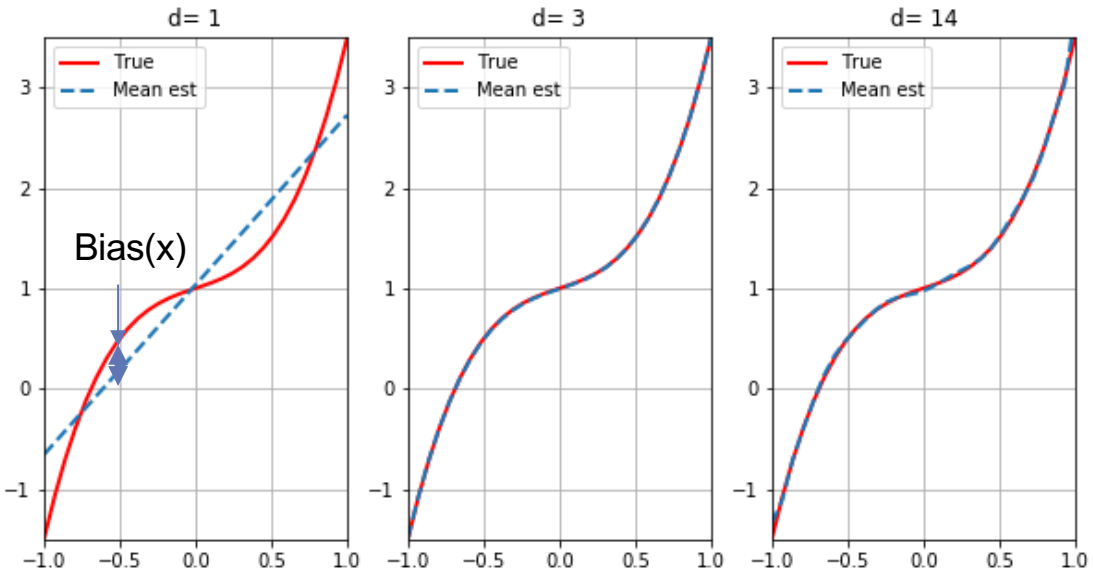
Bias and Variance

- 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: $Var(\boldsymbol{x}_{test}) := E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right) - E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]\right]^2$
  - 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
  - This is a completely theoretical computation, so imagine conceptually you could re-run these experiments
  - Useful to conceptually understand generalization

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
    - True function is cubic, but model is restricted to linear
  - High model orders ⇒ bias low

Three model (polynomial degree) orders!



High bias $d < d_{true}$

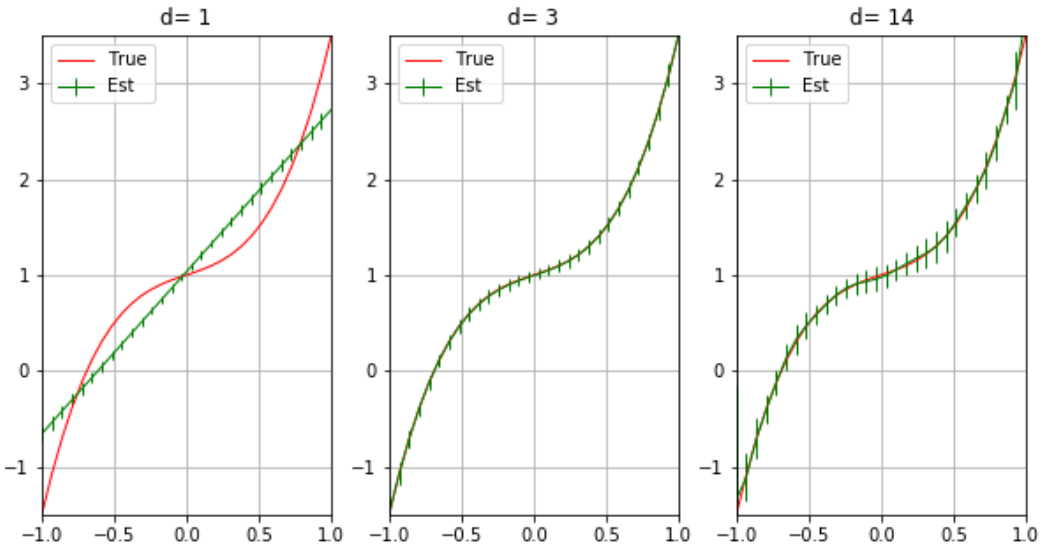
No bias $d = d_{true}$

No bias $d > d_{true}$

When there is very low model order, we have a high bias or high level of under-fitting.

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: $\mathrm{MSE}_f(\boldsymbol{x}_{test}) \coloneqq E\left[f_0(\boldsymbol{x}_{test}) - f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]^2$:
  - Bias: $Bias(\boldsymbol{x}_{test}) \coloneqq f_0(\boldsymbol{x}_{test}) - E\left[f\left(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}\right)\right]$
  - 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 : $MSE_f(\boldsymbol{x}_{test}) = Bias(\boldsymbol{x}_{test})^2 + Var(\boldsymbol{x}_{test})$
  - Will be proved soon
- 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
- $\mathrm{MSE}_f(\boldsymbol{x}_{test}) = E\left[f_0(\boldsymbol{x}_{test}) - f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}})\right]^2 = E\left[f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}) + \bar{f}(\boldsymbol{x}_{test}) - f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}})\right]^2$
- Three components: $\mathrm{MSE}_f(\boldsymbol{x}_{test}) = M_1 + M_2 - 2M_3$
  - $M_1 = E\left[f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test})\right]^2 = \left[f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test})\right]^2 = Bias(x_{test})$
  - $M_2 = E\left[f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}) - \bar{f}(\boldsymbol{x}_{test})\right]^2 = Var(x_{test})$
  - $M_3 = E\left[(f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}))(f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}) - \bar{f}(\boldsymbol{x}_{test}))\right]$
    $= (f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}))E\left[f(\boldsymbol{x}_{test}, \widehat{\boldsymbol{\beta}}) - \bar{f}(\boldsymbol{x}_{test})\right]$
    $= (f_0(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test}))(\bar{f}(\boldsymbol{x}_{test}) - \bar{f}(\boldsymbol{x}_{test})) = 0$

Outline

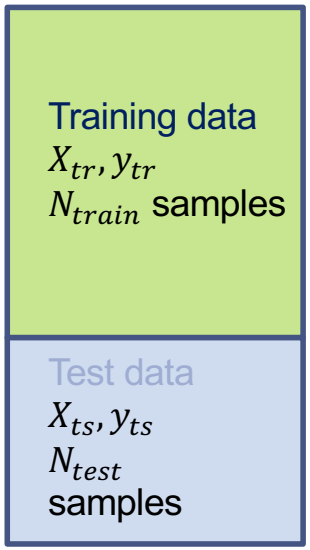
- Math Background
- Motivating Example: What polynomial degree should a model use?
- Bias and variance
- Cross-validation

Cross Validation (CV)

- 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
  - For example, polynomial degree
  - 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



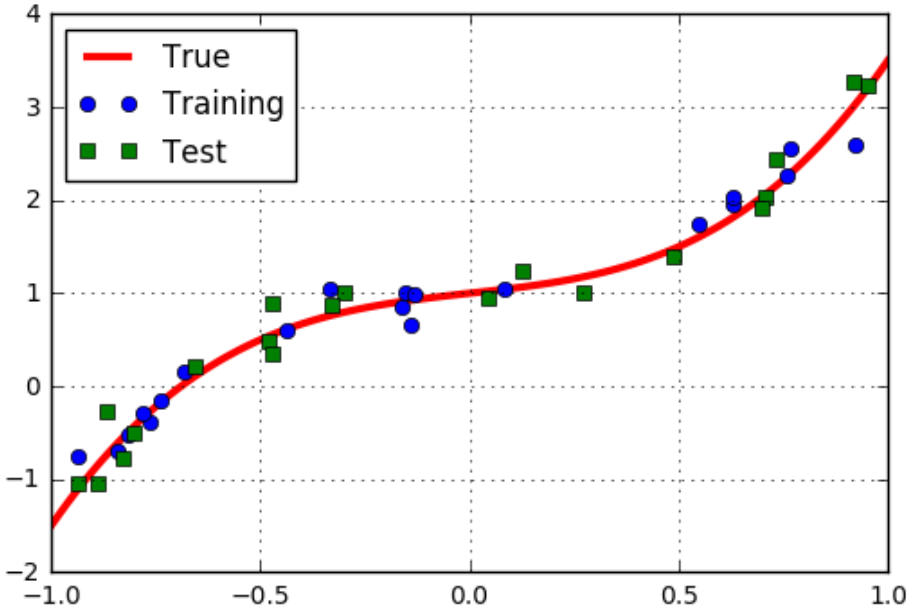
$\hat{\beta} = \text{fit}(X_{tr}, y_{tr}, p)$

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

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

Polynomial Example: Training Test Split

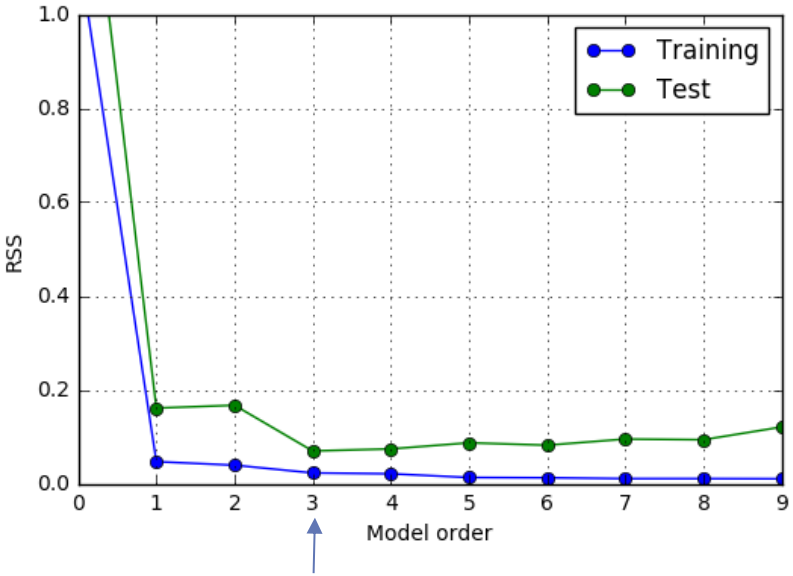
- Example: Split data into 20 samples for training, 20 for test




```
# 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 (i.e., how you do the Train/Test split)
- Only use limited number of samples for training
  - Since you do the split
  - Increases the variance error in bias-variance formula
- Both problems particularly bad for data with limited number of samples
  - You need to use your sample more judiciously!

K-Fold Cross Validation

- K-fold cross validation (break data into k parts instead of 2!)
  - Each part is called a fold
  - Divide data into K parts
  - Use $K-1$ parts for training. Use remaining for test.
  - Average over the K test choices (instead of just 1, which would be less accurate)
  - 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)
- 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

- 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 : $\hat{\beta} = \text{fit}(X_{tr}, y_{tr}, p)$
    - Predict values on test data: $\hat{y}_{ts} = \text{predict}(X_{ts}, \hat{\beta})$
    - Score the fit on test data: $S[p, i] = \text{score}(y_{ts}, \hat{y}_{ts})$
- Find average score for each model order: $\bar{S}[\text{p}] = \frac{1}{K}\sum_{i=1}^{K} S[p, i]$
- Select model order with lowest average score: $\hat{p} = \arg\min_{p} \bar{S}[p]$

Polynomial Example

- Use sklearn Kfold object

```
# 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)
```

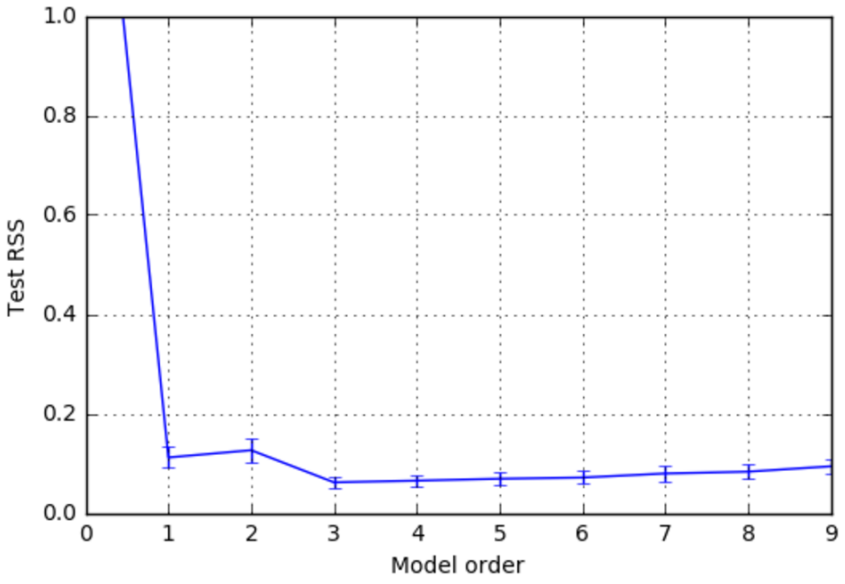
- Loop
  - Outer loop: Over K folds
  - Inner loop: Over D model orders
  - Measure test error in each fold and order
  - Averaging test errors from K folds for each model order
  - Find the model order with the minimal average test errors
  - Can be time-consuming

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)

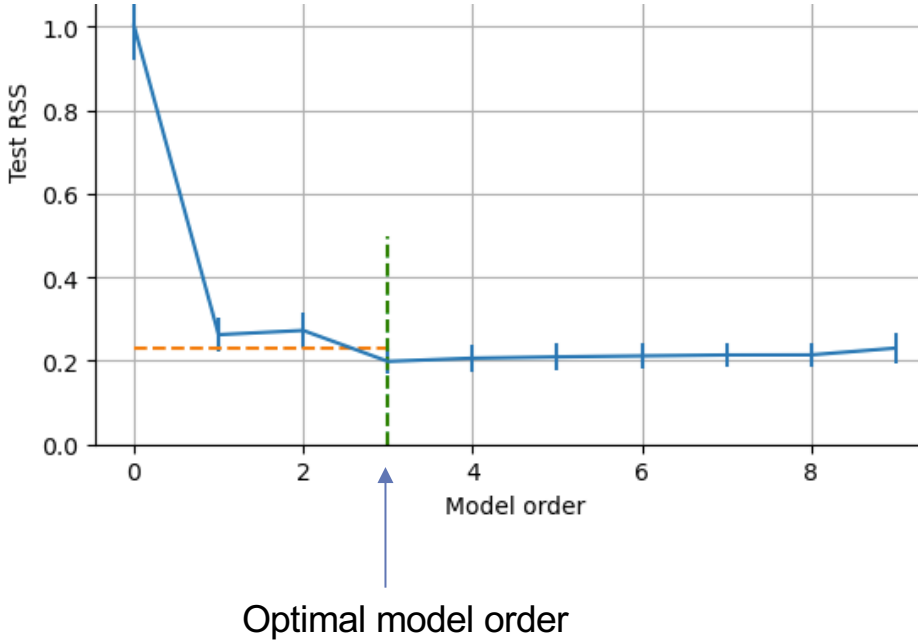
```
RSS_mean = np.mean(RSSts,axis=1)
RSS_std  = np.std(RSSts,axis=1) / np.sqrt(nfold-1)
plt.errorbar(dtest, RSS_mean, yerr=RSS_std, fmt='-')
plt.ylim(0,1)
plt.xlabel('Model order')
plt.ylabel('Test RSS')
plt.grid()
```

- Simple model selection
  - Select d with lowest mean test RSS
- For this example
  - Estimate model order = 3



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”



One SE Rule Pseudo-Code

- Get data X, y
- Compute score as before: $S[p, i]$ = 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:

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

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