

Lecture 4


Model Order Selection

EE-UY 4563/EL-GY 9123: INTRODUCTION TO MACHINE LEARNING
PROF. SUNDEEP RANGAN (WITH MODIFICATION BY YAO WANG)

Learning Objectives

- ❑ Understand the concept of model class and model order
- ❑ 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
- ❑ Understand the concept of bias, variance and the irreducible error for a model
 - Know how to compute each from synthetically generated data
- ❑ Understand the cross-validation process
 - Use it to assess the test error for a given model
 - Use it to select an optimal model order and for feature selection

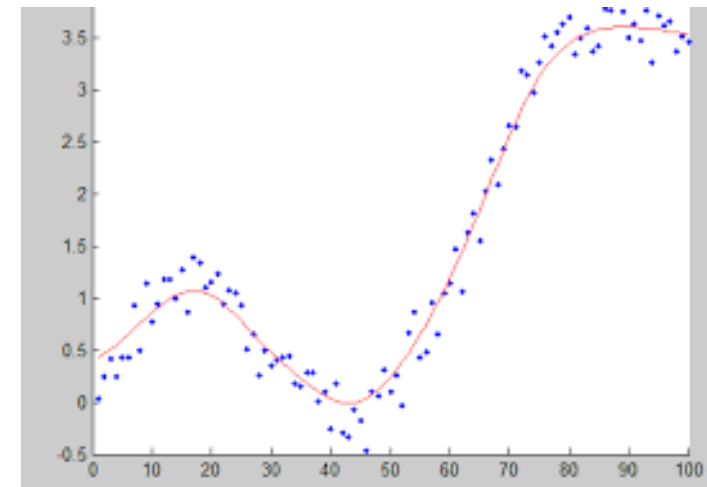
Outline

 Motivating example: What polynomial degree should a model use?

- ☐ Bias and variance
- ☐ Cross-validation
- ☐ Extension to feature selection


Polynomial Fitting



- Last lecture: polynomial regression
- Given data $(x_i, y_i), i = 1, \dots, N$
- Learn a polynomial relationship:
$$y = \beta_0 + \beta_1 x + \dots + \beta_d x^d + \epsilon$$
 - d = degree of polynomial. Called **model order**
 - $\boldsymbol{\beta} = (\beta_0, \dots, \beta_d)$ = 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/unit03_model_sel/demo03_1_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 derive 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

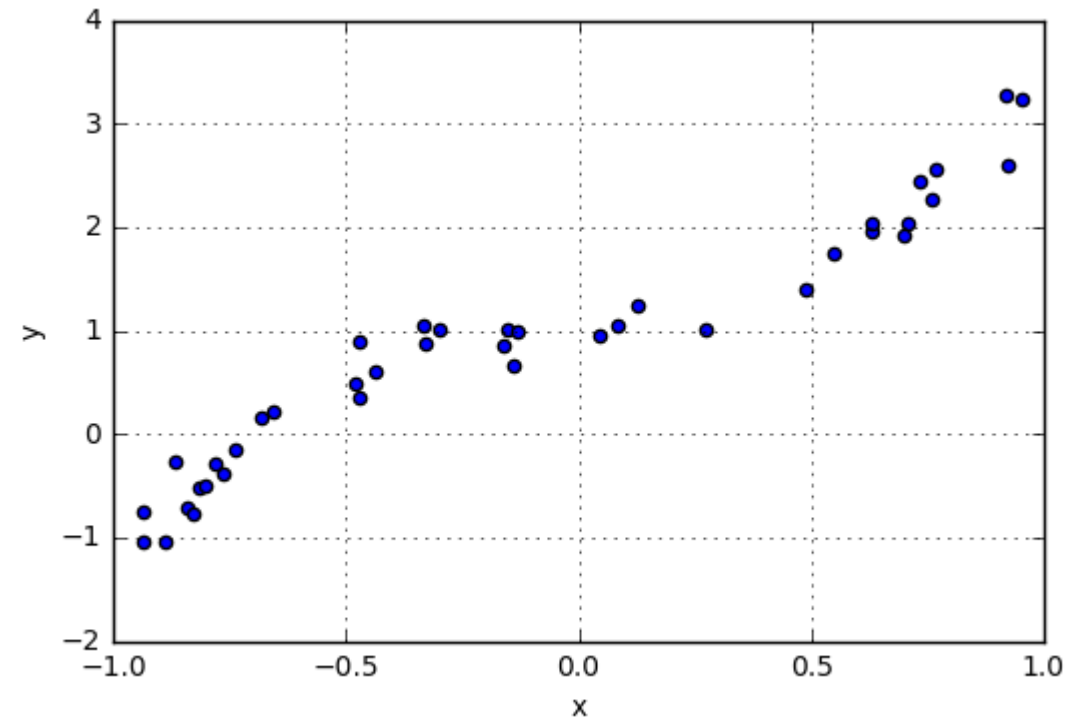
To illustrate the concepts, we consider a simple polynomial model:

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

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

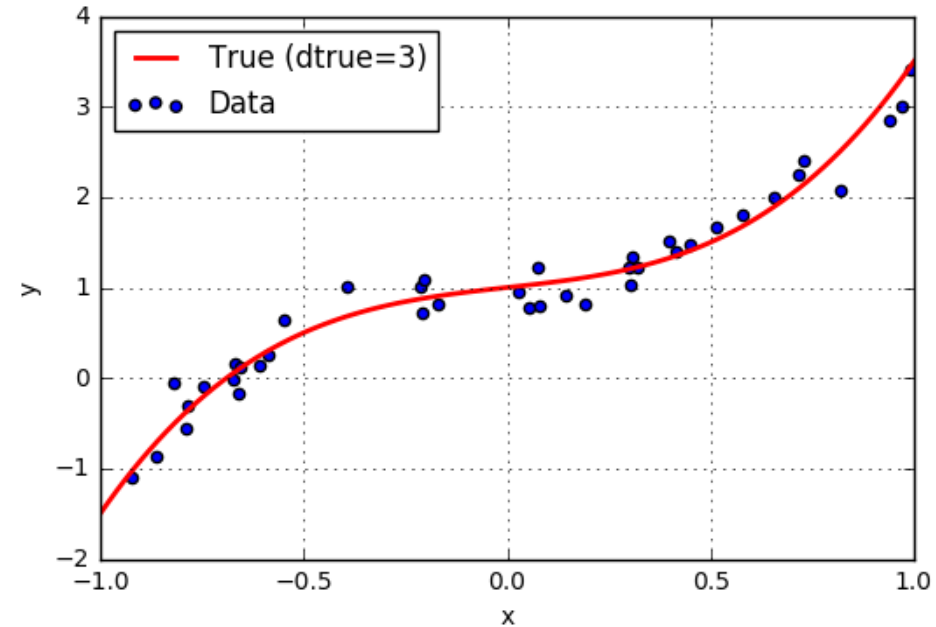
Example Question

- ❑ 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 $[-1,1]$
- $y = f(x) + \epsilon$,
 - $f(x) = \beta_0 + \beta_1 x + \dots + \beta_d x^d = \text{"true relation"}$
 - $d = 3$, $\epsilon \sim N(0, \sigma^2)$
- Synthetic data useful for analysis
 - Know "ground truth"
 - Can measure performance of various estimators



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

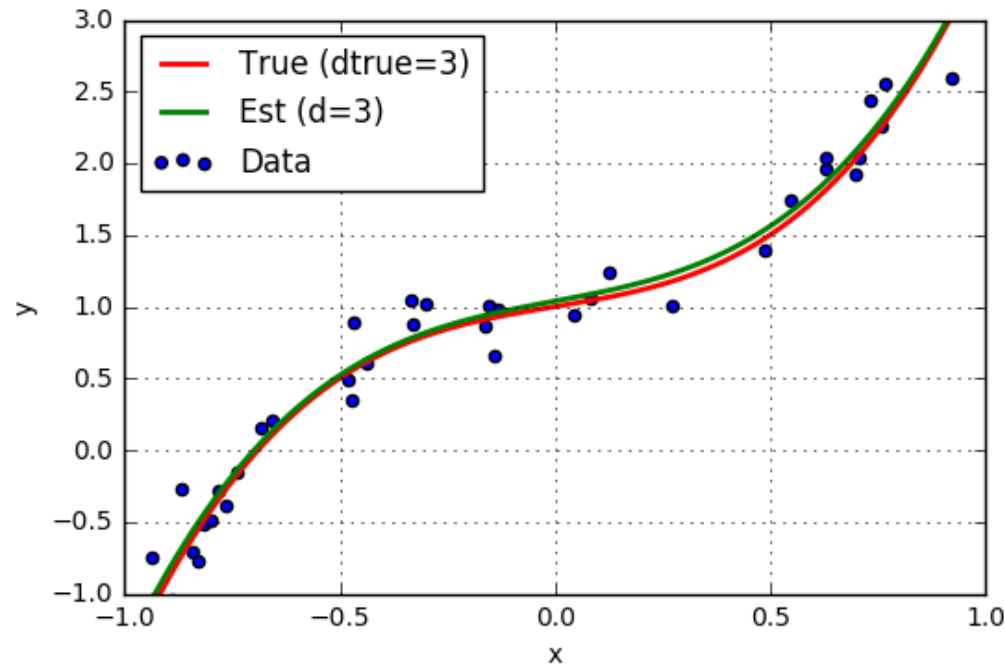
# True model parameters
beta = np.array([1,0.5,0,2]) # coefficients
wstd = 0.2                  # noise
dtrue = len(beta)-1         # true poly degree

# 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



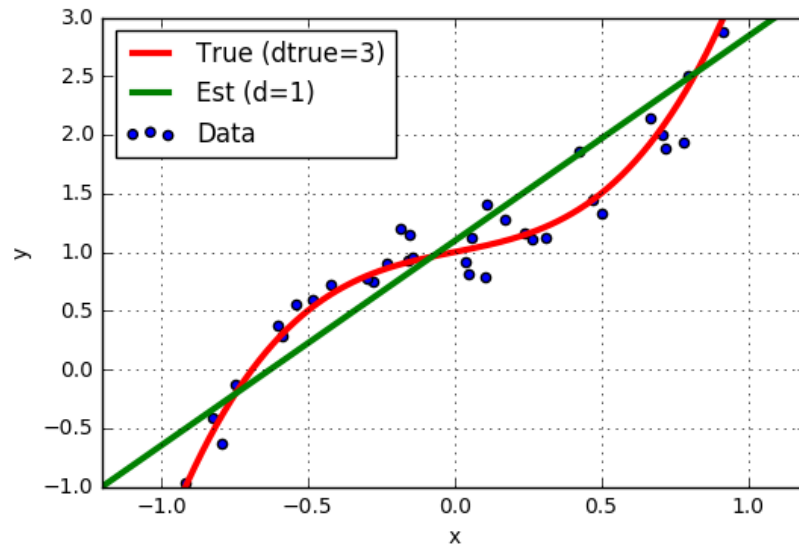
```
d = 3
beta_hat = poly.polyfit(xdat,ydat,d)

# 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)

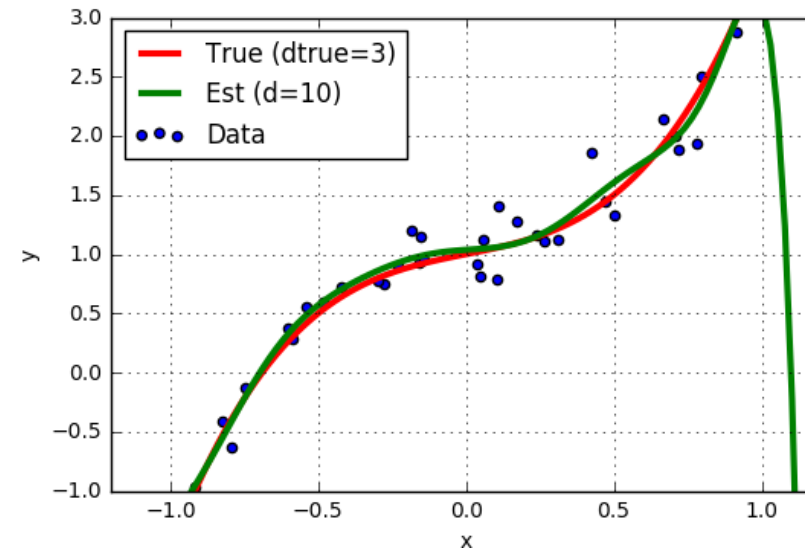
# Plot data
plt.scatter(xdat,ydat)
plt.legend(['True (dtrue=3)', 'Est (d=3)', 'Data'], loc='upper left')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
```


But, True Model Order not Known

□ Suppose we guess the wrong model order?

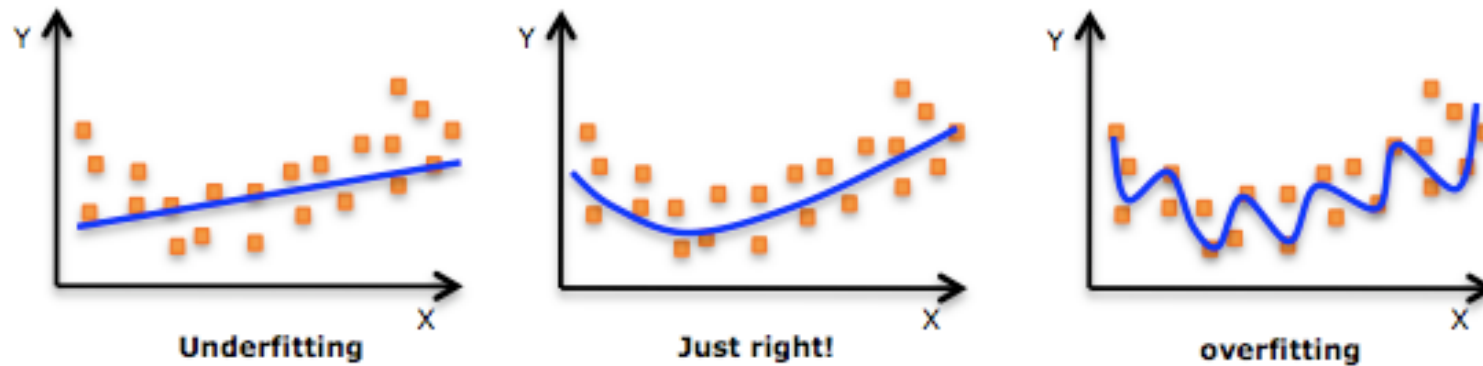


$d=1$ "Underfitting"



$d=10$ "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 $\hat{\beta}$
- Compute predicted values on training data

$$\hat{y}_i = \hat{\beta}^T x_i$$

- Compute RSS

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

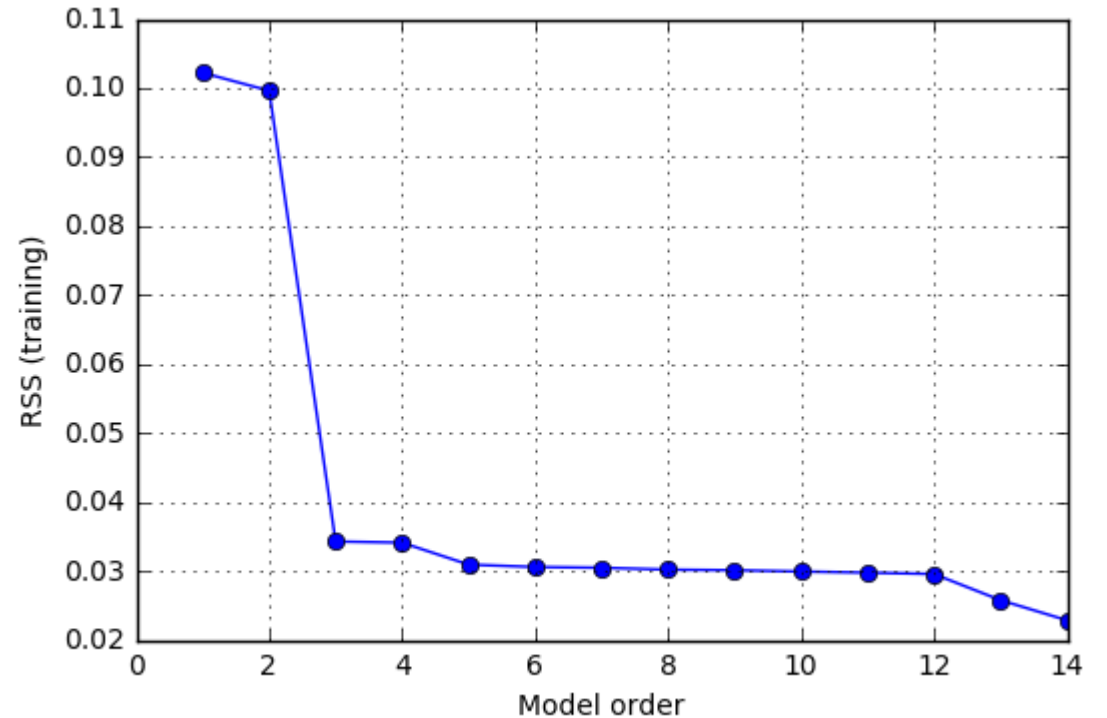
- Find d with lowest RSS

□ 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

☐ Cross-validation

Model Class

❑ Consider general estimation problem

- Given data (x_i, y_i) want to learn a functional relation: $y \approx \hat{y} = f(x)$

❑ Model class: The set of possible estimates:

$$\hat{y} = f(\mathbf{x}, \boldsymbol{\beta})$$

- Set is parametrized by $\boldsymbol{\beta}$

❑ Many possible examples:

- Linear model: $\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$
- Polynomial model: $\hat{y} = \beta_0 + \beta_1 x + \dots + \beta_k x^k$
- Nonlinear: $\hat{y} = \beta_0 + \beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x}$
- ...

Model Class and True Function

□ Analysis set-up:

- Learning algorithm assumes a **model class**: $\hat{y} = f(\mathbf{x}, \boldsymbol{\beta})$
- But, data has **true** relation: $y = f_0(x) + \epsilon, \epsilon \sim N(0, \sigma_\epsilon^2)$

□ Will quantify three key effects:

- Irreducible error
- Under-modeling
- Over-fitting

Output Mean Squared Error

❑ To evaluate prediction error suppose we are given:

- A parameter estimate $\hat{\beta}$ (computed from the learning algorithm for a fixed training set)
- A test point \mathbf{x}_{test}
- Test point is generally different from training samples.

❑ Predicted value: $\hat{y} = f(\mathbf{x}_{test}, \hat{\beta})$

❑ Actual value: $y = f_0(\mathbf{x}_{test}) + \epsilon$

❑ Output mean squared error:

$$MSE_y(\mathbf{x}_{test}, \hat{\beta}) := E[y - \hat{y}]^2$$

- Expectation is over noise ϵ on the test sample.

Irreducible Error

□ Rewrite output MSE:

$$MSE_y(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}}) := E[y - \hat{y}]^2 = E[f_0(\mathbf{x}_{test}) + \epsilon - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2$$

□ Since noise on test sample is independent of $\hat{\boldsymbol{\beta}}$ and \mathbf{x}_{test} :

$$MSE_y(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}}) := [f_0(\mathbf{x}_{test}) - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 + E(\epsilon^2) = [f_0(\mathbf{x}_{test}) - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 + \sigma_\epsilon^2$$

□ Define **irreducible error**: σ_ϵ^2

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

Under-Modeling

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

$$f_0(x) = f(x, \beta_0) \text{ for all } x$$

for some parameter β_0 .

- β_0 called the **true parameter**

□ **Under-modeling:** When $f_0(x)$ is not in the model class

Sample Question

❑ 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

❑ Examples:

- True function: $f_0(x) = 2 + 3x$ Model class: $f(x, \beta) = \beta_0 + \beta_1 x + \beta_2 x^2$
- True function: $f_0(x) = 2 + 3x + 4x^2$ Model class: $f(x, \beta) = \beta_0 + \beta_1 x$
- True function: $f_0(x) = \sin(2\pi(5)x + 7)$ Model class: $f(x, \beta) = \beta_0 \sin(2\pi(5)x) + \beta_1 \cos(2\pi(5)x)$
- True function: $f_0(x) = \sin(2\pi(8)x + 7)$ Model class: $f(x, \beta) = \beta_0 \sin(2\pi(5)x) + \beta_1 \cos(2\pi(5)x)$

❑ Solutions in class

Analysis of Under-Modeling: Noise-Free Case

- Assume true relation has no noise: $y = f_0(x)$
 - Can model noise, but requires more probability theory

- Get training data: $(x_i, y_i), i = 1, \dots, n$

- Fit model parameter from least-squares:

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^n (y_i - f(x_i, \beta))^2 = \arg \min_{\beta} \sum_{i=1}^n (f_0(x_i) - f(x_i, \beta))^2$$

- **Conclusions:** With no noise

- Minimizing training error finds best least squares fit of the true functions in the model class
- If there is a unique true parameter, then $\hat{\beta} = \beta_0$. Estimator identifies correct parameter

Bias: Noise-Free Case

- Let \mathbf{x}_{test} = some test point
 - Can be different from the training data set

□ **Definition:** When there is no noise, the **bias** at a test point \mathbf{x}_{test} is:

$$Bias(\mathbf{x}_{test}) := f_0(\mathbf{x}_{test}) - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})$$

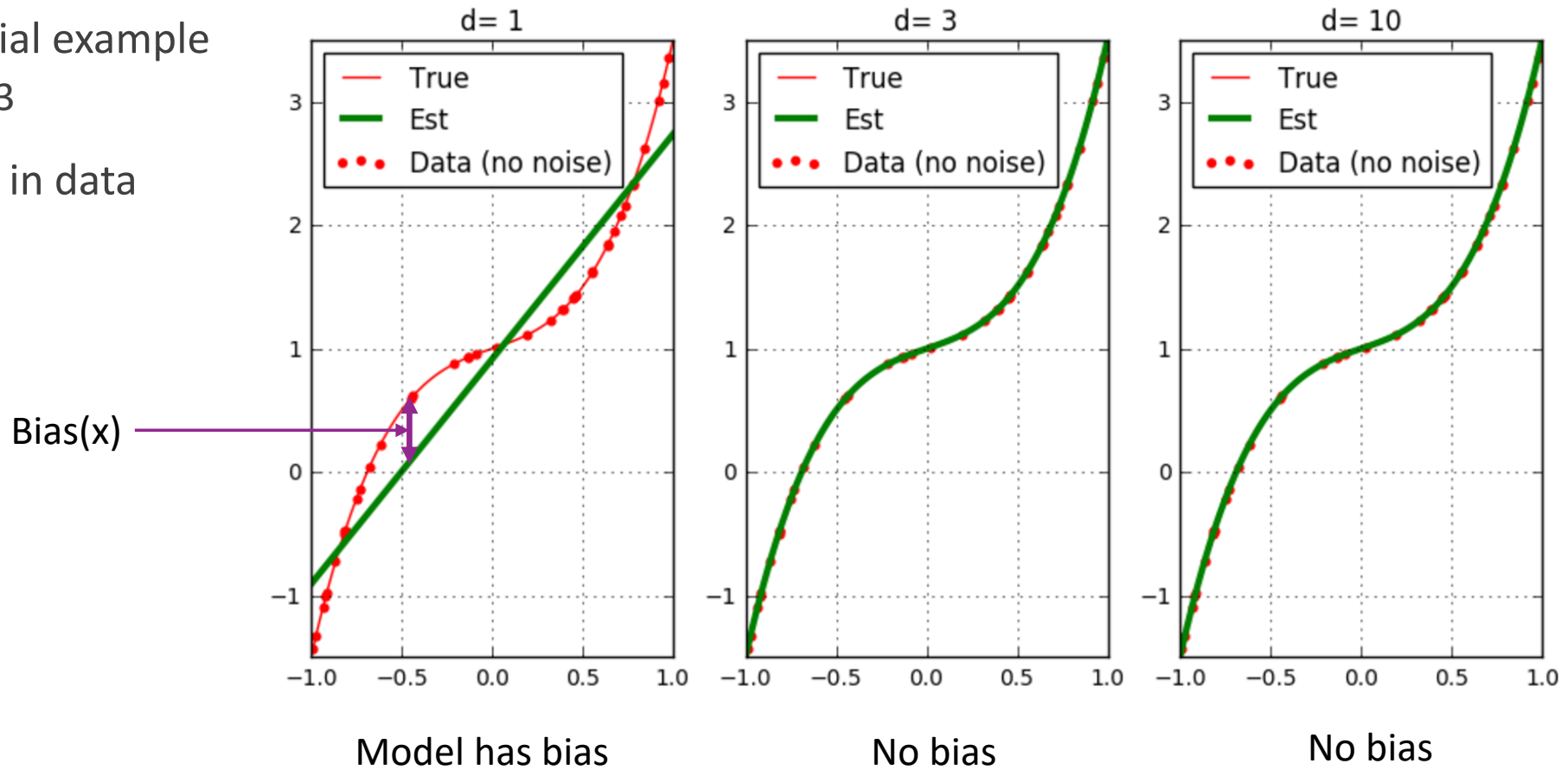
- Measures the difference between true and estimated relation in absence of noise
- Previous analysis shows:
 - Bias is small when true function is close to model class
 - When there is no under-modeling, $Bias(\mathbf{x}_{test}) = 0$ and true parameter can be found if there are sufficient training data.

Bias Visualized

- Polynomial example

 - $d_{true} = 3$

- No noise in data



Analysis with Noise (Advanced)

□ Now assume noise: $y = f_0(x) + \epsilon, \epsilon \sim N(0, \sigma_\epsilon^2)$

□ Get training data: $(x_i, y_i), i = 1, \dots, n$

□ Fit parameter:

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^n (y_i - f(x_i, \beta))^2$$

- $\hat{\beta}$ will be random. Depends on particular noise realization for the selected training samples.
- Results may not be a good estimate for the true function!

□ Take a new test point x_{test} : Error $y(x_{test}) - f(x_{test}, \hat{\beta})$ may be large!

□ **Solution: Multiple trials, each using a different training set.**

- Compute mean and variance of estimated function $f(x_{test}, \hat{\beta})$ over $\hat{\beta}$ from different training sets
- **Bias**: Difference of true function from mean estimate
- **Variance**: Variance of estimate around its mean

Bias-Variance Formula (Advanced)

- Consider test point \mathbf{x}_{test} with noise.

Observed value $y(\mathbf{x}_{test}) = f_0(\mathbf{x}_{test}) + \varepsilon$, Predicted value: $f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})$

$$\begin{aligned}MSE(\mathbf{x}_{test}) &:= E[y(\mathbf{x}_{test}) - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 \\&= E[f_0(\mathbf{x}_{test}) + \varepsilon - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 = E[f_0(\mathbf{x}_{test}) - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 + \sigma_\varepsilon^2 \\E[f_0(\mathbf{x}_{test}) - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 &= E[f_0(\mathbf{x}_{test}) - E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})] + E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})] - f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]^2 \\&= (f_0(\mathbf{x}_{test}) - E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})])^2 + E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}}) - E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]]^2\end{aligned}$$

- Bias:** $Bias(\mathbf{x}_{test}) := f_0(\mathbf{x}_{test}) - E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]$

- Variance:** $Var(\mathbf{x}_{test}) := E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}}) - E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})]]^2$

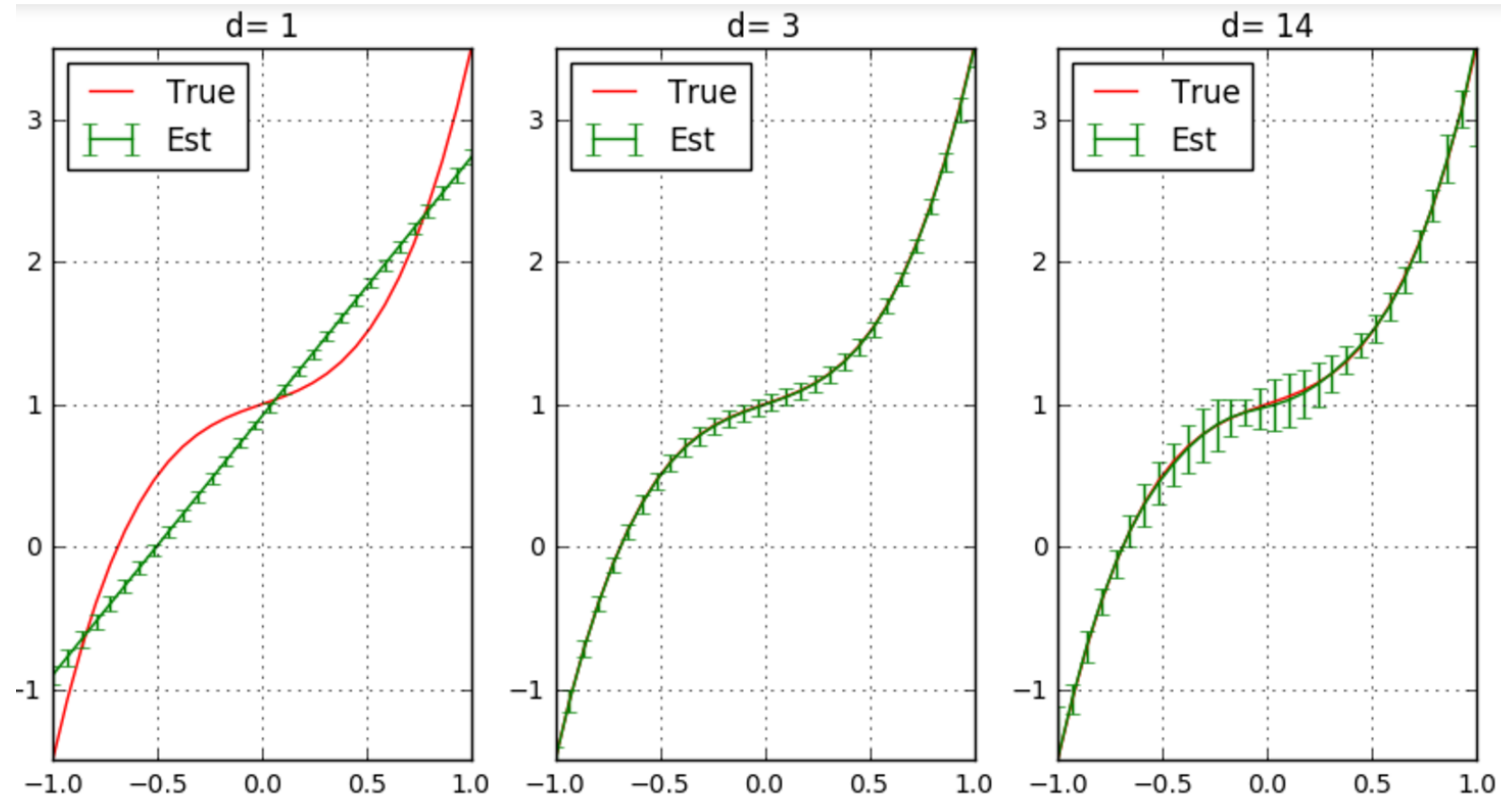
- Bias-Variance formula:** $MSE(\mathbf{x}_{test}) = Bias(\mathbf{x}_{test})^2 + Var(\mathbf{x}_{test}) + \sigma_\varepsilon^2$ (irreducible error)

- Note that expectation is taken over $\hat{\boldsymbol{\beta}}$ derived from different training samples

- Further averaging over test samples: $MSE = Bias^2 + Var + \sigma_\varepsilon^2$ (irreducible error)

Bias and Variance Illustrated

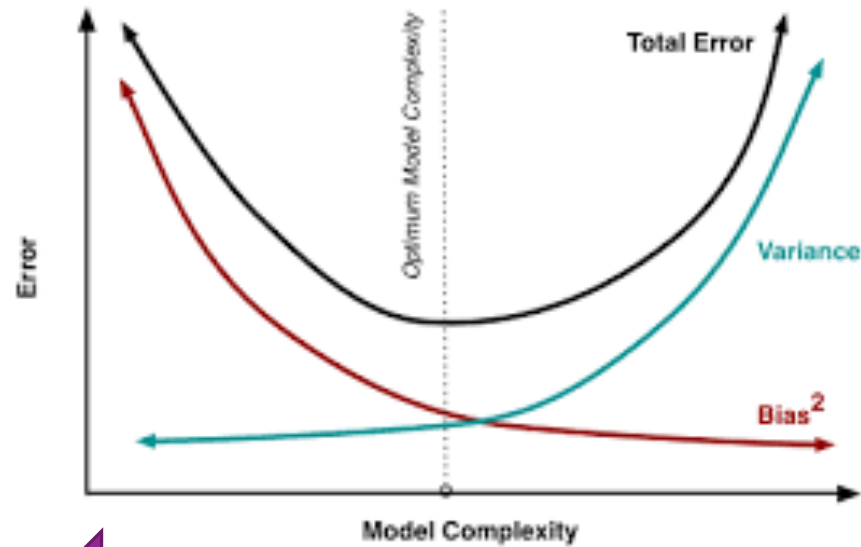
- Polynomial ex
- Mean and std dev of estimated functions
- 100 trials
- Solid line: mean estimate among all trials
- Error bars: 1 STD



Low variance,
High bias

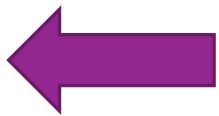
High variance,
Zero bias

Bias-Variance Tradeoff



- Optimal model order depends on:
 - Amount of samples available
 - Underlying complexity of the relation

Floor is the noise variance



Simpler models
Less parameters
Under-fitting



Richer models
More parameters
Over-fitting

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

$$E[f(\mathbf{x}_{test}, \hat{\boldsymbol{\beta}})] = f_0(\mathbf{x}_{test}).$$

□ Result 3: For $n \gg p$ and 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

□ See textbook for proof: [Hastie] Hastie, Tibshirani, Friedman, The elements of statistical learning.

Outline

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

❑ Bias and variance

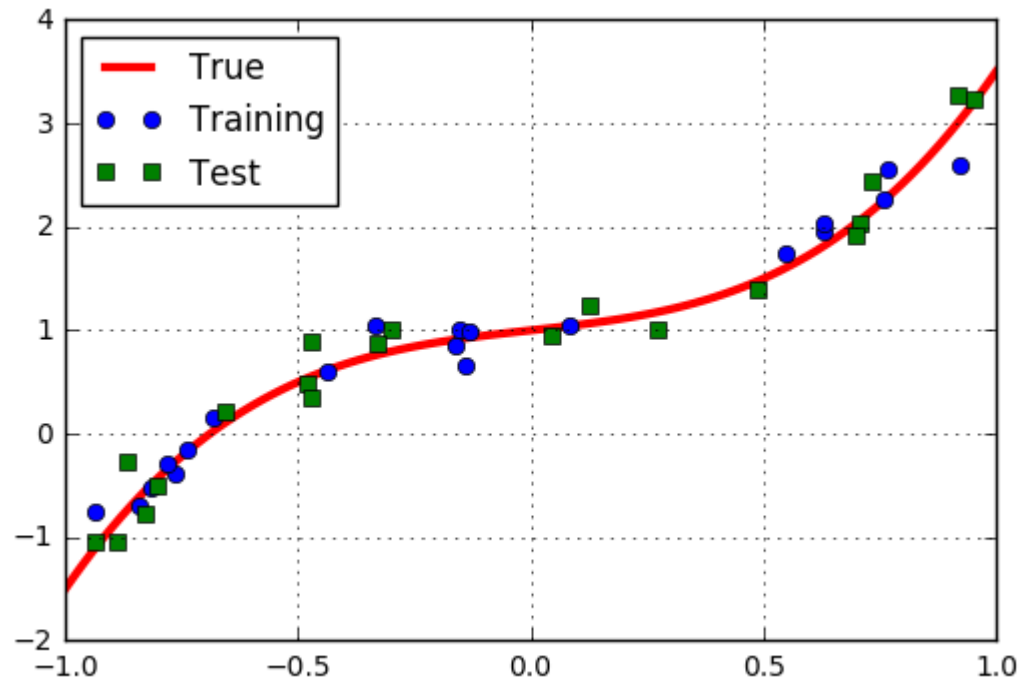
❑ Cross-validation

Cross Validation

- ❑ Concept: Need to fit on test data independent of training data
- ❑ Divide data into two sets:
 - N_{train} training samples, N_{test} test samples
- ❑ For each model order, p , learn parameters $\hat{\beta}$ from training samples
- ❑ Measure RSS on test samples.
$$RSS_{test}(p) = \sum_{i \in \text{test}} (\hat{y}_i - y_i)^2$$
- ❑ Select model order p that minimizes $RSS_{test}(p)$

Polynomial Example: Training Test Split

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



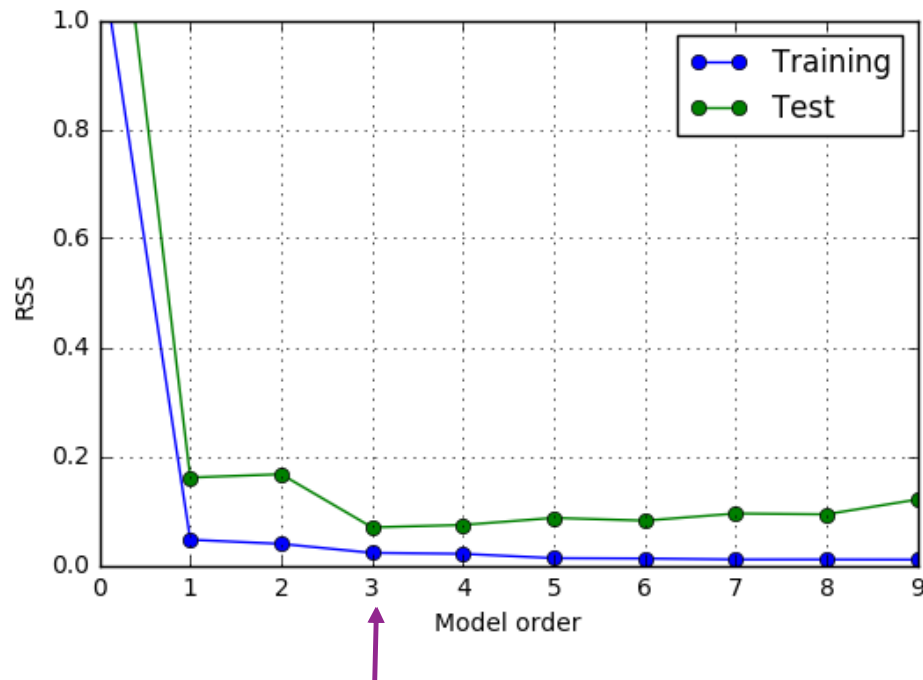
```
# Number of samples for training and test
ntr = nsamp // 2
nts = nsamp - ntr

# 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')
```

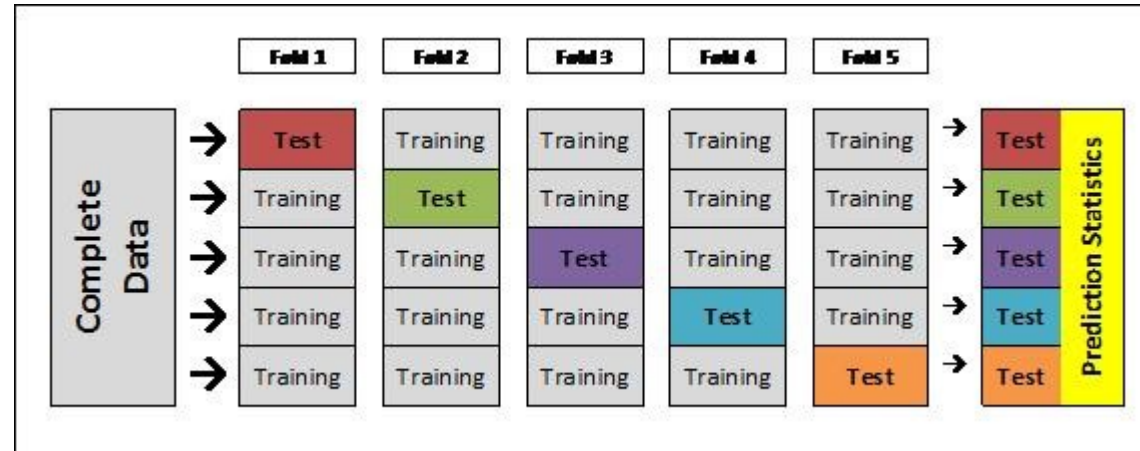
Problems with Simple Train/Test Split

- ❑ 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
- ($= \text{Bias}^2 + \text{Variance} + \text{irreducible error}$)



□ 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/>

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
RSSs = 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)
        RSSs[it,isplit] = np.mean((yhat-yts)**2)
```



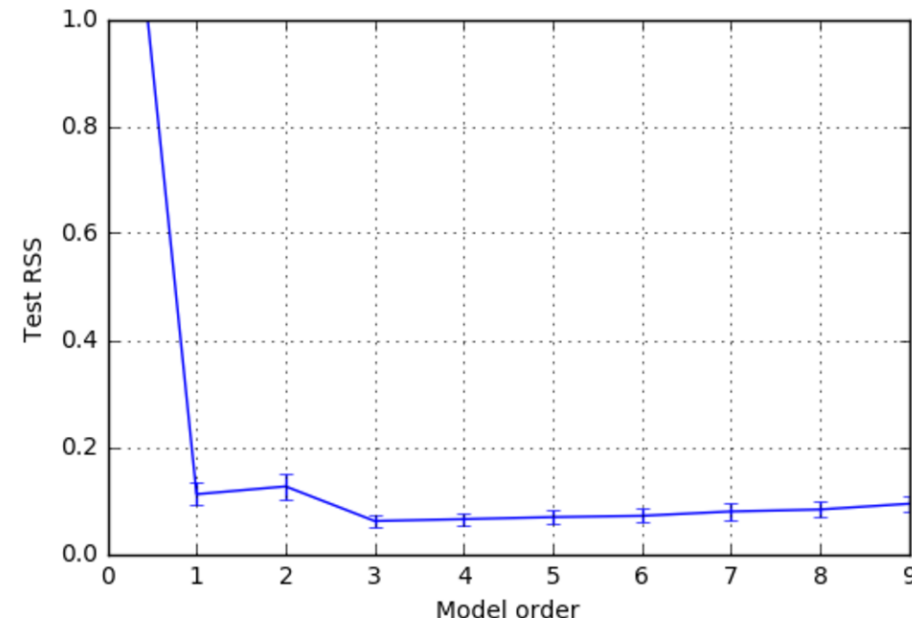
Polynomial Example CV Results

- ❑ For each model order d
 - Compute mean test RSS over K folds
 - Compute standard error (SE) of test RSS
 - $SE = \text{std dev} / \sqrt{K - 1}$
 - Mean and SE averaged over the K folds

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

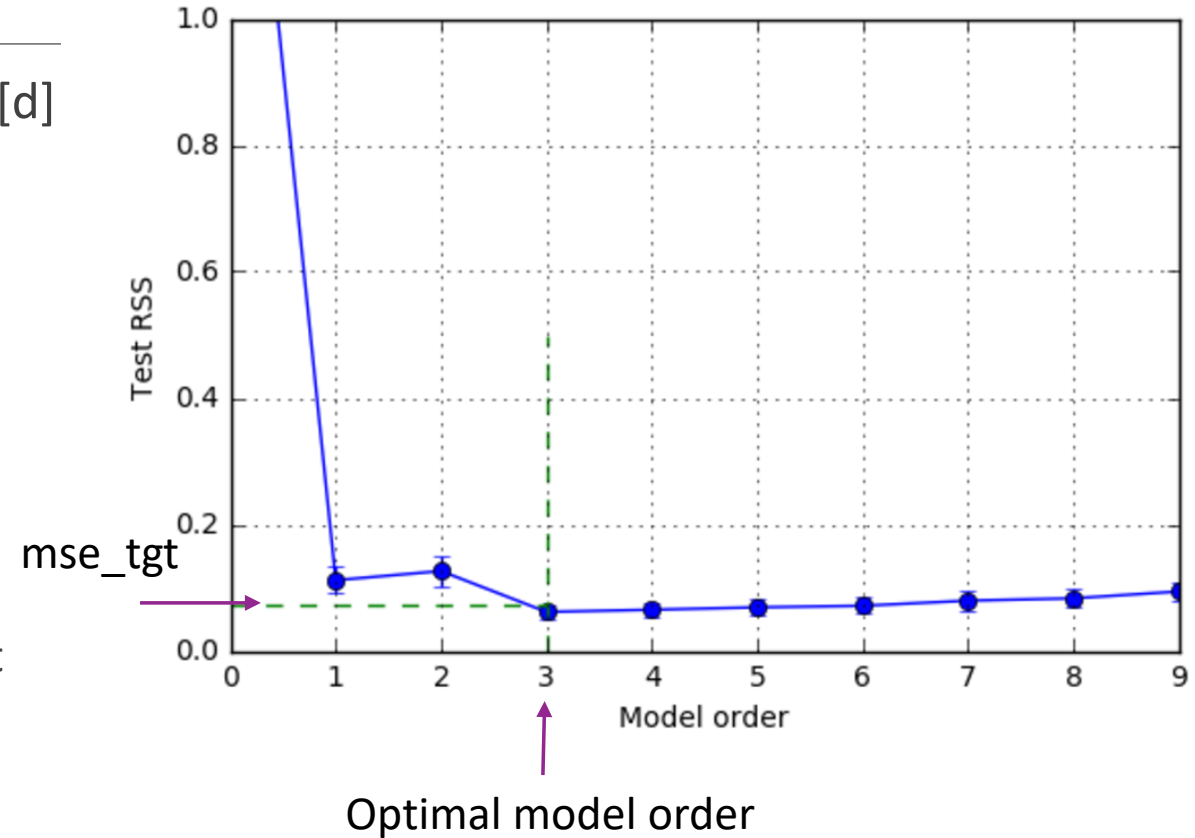
- ❑ For this example
 - Estimate model order = 3

```
RSS_mean = np.mean(RSSsts,axis=1)
RSS_std  = np.std(RSSsts,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 $\text{mse_mean}[d]$
- ❑ Problem: Often over-predicts model order
- ❑ One standard deviation rule
 - Use simplest model within one SE of minimum
- ❑ Detailed procedure:
 - Find d_0 to minimize $\text{mse_mean}[d]$
 - Set $\text{mse_tgt} = \text{mse_mean}[d_0] + \text{mse_std}[d_0]$
 - Find minimal d_{opt} s.t. $\text{mse_mean}[d_{\text{opt}}] \leq \text{mse_tgt}$



Feature selection as model selection

- ❑ So far we discussed how to select the order of a fitting polynomial as a model selection problem.
- ❑ More generally, given many features, only a subset of the features may be useful for predicting the target. How do we select the useful features?
- ❑ With linear regression, each possible feature subset corresponds to a different model, and the feature number is the model order!
- ❑ Higher feature number leads to low bias but higher variance!
- ❑ We can use the approach for polynomial order selection to solve the feature selection problem.

Exhaustive search for feature selection

- ❑ Suppose you want to consider feature subset of size up to p
- ❑ For all possible feature subsets of size 1 to p : use cross validation to find mean RSS mean and standard deviation for each feature subset.
- ❑ Choose the subset with the minimal RSS mean, or use the one standard error rule.

Feature selection based on correlation with target

- ❑ Exhaustive search may be infeasible when the raw feature dimension is large!
- ❑ Suboptimal approach:
 - For each candidate feature order $d \leq p$, choose d features with the highest correlation coefficients with the target
 - Use cross validation to determine the RSS mean and variance for this subset
 - Select the feature subset with minimal RSS mean or using the one standard error rule.
- ❑ Is using correlation with target a good idea?
 - Two features that are correlated could both be highly correlated with the target, but provide redundant information and ideally only one of them should be used.

Greedy feature selection

❑ Forward-Stepwise Selection

- Select one feature from all features that provides the lowest RSS with cross validation
- Select one new feature from all remaining features, so that previously chosen features plus the new feature provides the lowest RSS
- Repeat until the maximum feature number is reached, or when the RSS starts to increase

❑ Backward-Stepwise

- First use all features and find the RSS (using cross validation)
- Remove one feature and find the new RSS. Go through all possible features to remove.
- Find the one that leads to the least RSS increase. Remove this feature.
- Repeat the above, remove one from the remaining features, to find the next most important feature.

❑ Except exhaustive search, can all lead to suboptimal solution

❑ We will discuss the method of LASSO in the next lecture for feature selection.

Comparison of feature selection methods

Figure from [Hastie2008]: Hastie, Tibshirani, Friedman, The elements of statistical learning.

For more on this subject, see Sec. 3.3

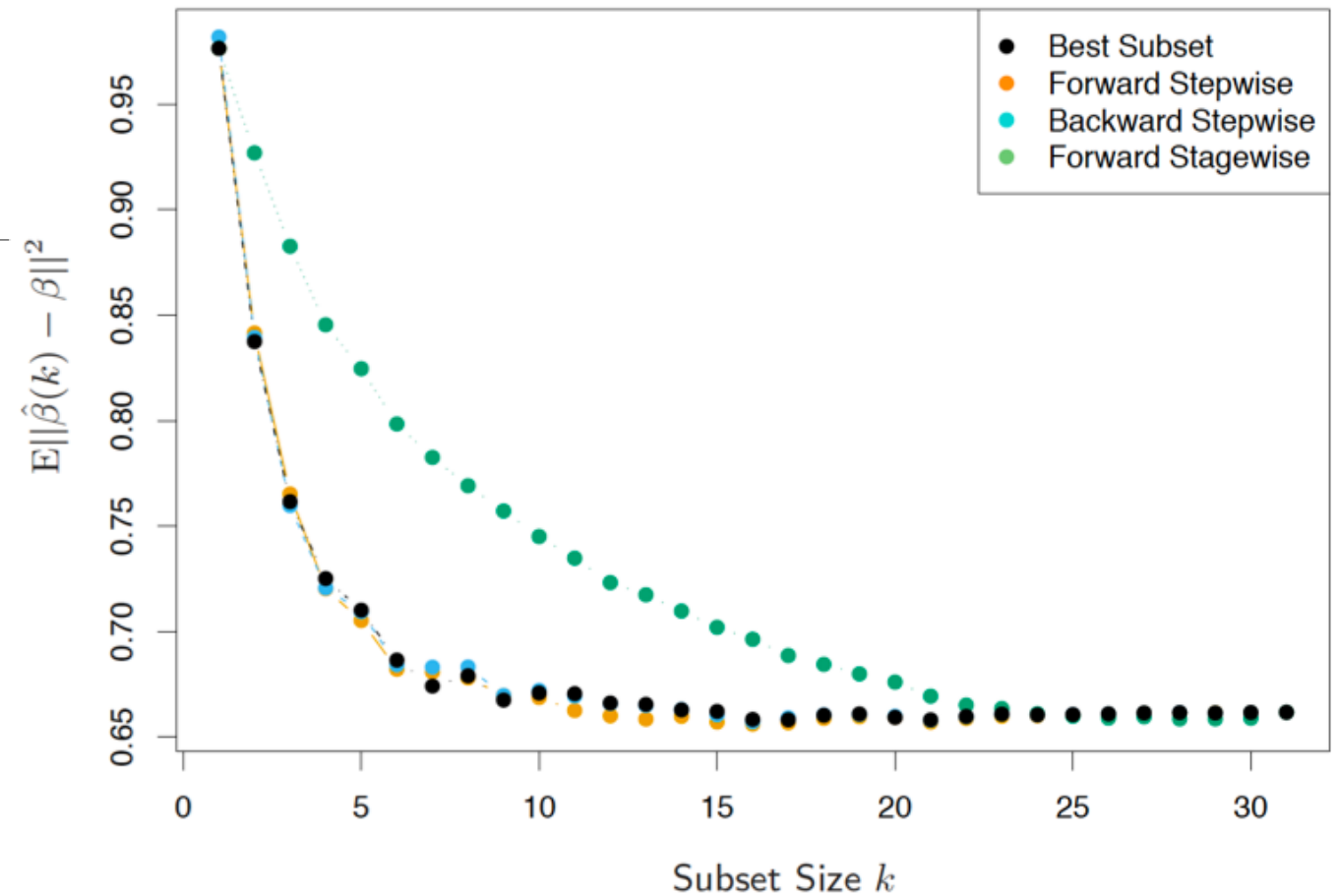
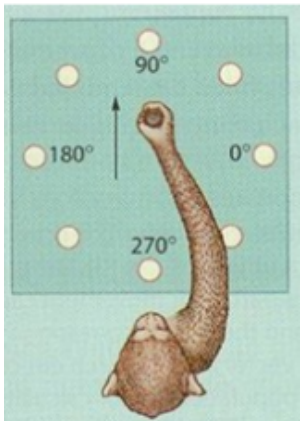


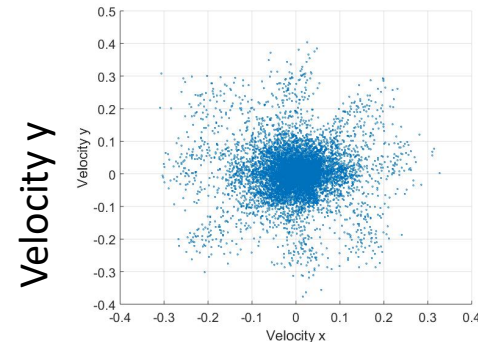
FIGURE 3.6. Comparison of four subset-selection techniques on a simulated linear regression problem $Y = X^T \beta + \varepsilon$. There are $N = 300$ observations on $p = 31$ standard Gaussian variables, with pairwise correlations all equal to 0.85. For 10 of the variables, the coefficients are drawn at random from a $N(0, 0.4)$ distribution; the rest are zero. The noise $\varepsilon \sim N(0, 6.25)$, resulting in a signal-to-noise ratio of 0.64. Results are averaged over 50 simulations. Shown is the mean-squared error of the estimated coefficient $\hat{\beta}(k)$ at each step from the true β .

Lab: Neural ECoG Data

Record arm motion

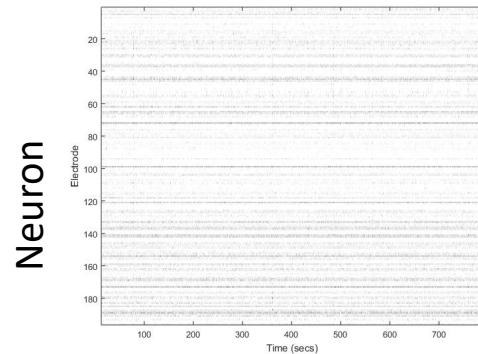
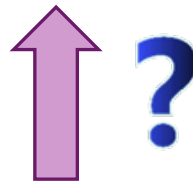


Record spikes
(ECoG in M1)



Arm
velocity

Velocity x



Spikes

Time

☐ Read a monkey's brain!

☐ Predict hand motion from electrode measurements (number of spikes in each neuron)

☐ Feature selection

- Which neuron signals predict arm motion?
- Use correlation to select candidate feature subsets

☐ Data from <https://crcns.org/>

- Open source website on neural data
- Great for projects

What you should know from this lecture

- ❑ Understand the concept of model class and model order
- ❑ 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
- ❑ Understand the concept of bias, variance and the irreducible error for a model
 - Know how to compute each from synthetically generated data
- ❑ Understand the cross-validation process
 - Use it to assess the test error for a given model
 - Use it to select an optimal model order and for feature selection