Unit 3
Model-Order Selection and the Bias-Variance Tradeoff

Prof. Phil Schniter

The Ohio State University

ECE 5307: Introduction to Machine Learning, Sp23

Learning objectives

- Understand the problem of model-order selection
- Visually identify underfitting and overfitting in a scatterplot
- Understand the need to partition data into training and testing subsets
- Understand the K-fold cross-validation process
  - Use it to assess the test error for a given model
  - Use it to select the model order
- Understand the concepts of bias, variance, and irreducible error
  - Know how to compute each from synthetically generated data
  - Understand the bias-variance tradeoff

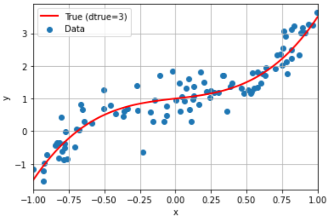
Outline

Polynomial regression

- Recall polynomial regression from last lecture
- Given data $\{(x_i, y_i)\}_{i=1}^n$, model target y as

$$y \approx \beta_0 + \beta_1 x + \cdots \beta_d x^d$$

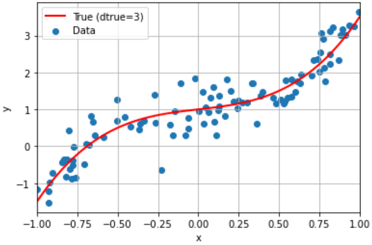
  - model parameters are $\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots, \beta_d]^\mathsf{T}$
  - d is the degree of the polynomial
  - given d, we can fit $\boldsymbol{\beta}$ using least-squares (multiple linear regression with $x_j \triangleq x^j$)
- Question: Can we select d from the data?
  - An example of "model-order selection"



Example with synthetic data

- We consider synthetic data generated using a noisy polynomial model
- $\{x_i\}$: 40 samples uniformly distributed in interval $[-1, 1]$
- $\{y_i\}$: generated as $y_i = f(x_i) + \epsilon_i$
  - $f(x) = \beta_0 + \beta_1 x + \cdots \beta_d x^d$ with $d = 3$ for some "true" coefficients $\{\beta_j\}_{j=0}^d$
  - noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, independent over i
- Synthetic data is useful for analysis and experimentation!
  - We know the "ground truth"
  - Thus we can measure the performance of various predictors

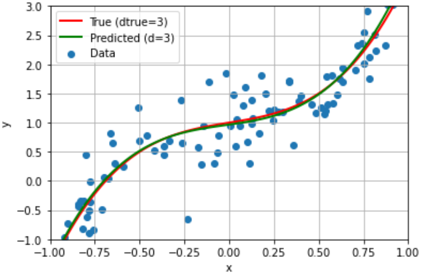
```
# Import useful polynomial library
import numpy.polynomial.polynomial as poly

# True model parameters
beta = np.array([1,0.5,0,2])   # coefficients
wstd = 0.4                     # noise std
dtrue = len(beta)-1            # true poly degree

# Independent data
nsamp = 100
xdat = np.random.uniform(-1,1,nsamp)

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



Fitting with the true model order

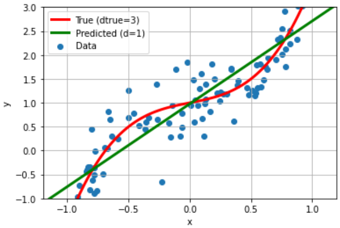
- Could implement via linear_model.LinearRegression by constructing features $x_{ij} = x_i^j$ for $j = 1 \ldots d$ and $i = 1 \ldots n$
- Shortcut: numpy.polynomial package
- In any case, we need to choose d, the polynomial order for our model
- First, let's see what happens if $d = 3$, the true polynomial order
  - The LS fit looks very good!

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

# Plot true and prediction model
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,label='True (dtrue=3)')
plt.plot(xp,yp_hat,'g-',linewidth=2,label='Predicted (d=3)')

# Plot data
plt.scatter(xdat,ydat,label='Data')
plt.legend(loc='upper left')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
```

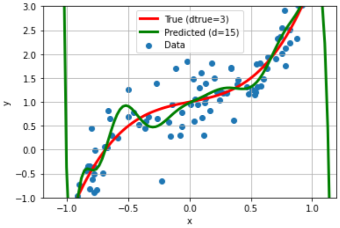


Fitting with the wrong model order

$d = 1$: "underfitting"

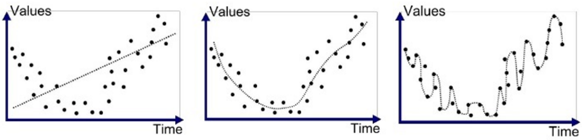


$d = 15$: "overfitting"



another illustration:

https://medium.com/greyatom



Underfitted   Good Fit/Robust   Overfitted

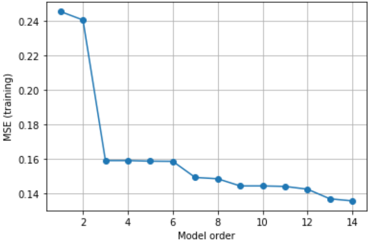
Is there a way to estimate the true d from the data $\{(x_i, y_i)\}_{i=1}^n$?

Select the model-order that minimizes training MSE?

Simple idea:

- For each hypothesized model order d:
  - Compute LS coefficients $\widehat{\boldsymbol{\beta}} \in \mathbb{R}^{d+1}$
  - Compute $\text{MSE}_{\textsf{train}}(d) \triangleq \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{\beta}}\|^2$

  Finally, pick the d that minimizes $\text{MSE}_{\textsf{train}}$
- Does this work?
  - $\text{MSE}_{\textsf{train}}(d)$ decreases with d
  - Suggests to choose d as large as possible
  - Leads to "overfitting"
- Why does this happen?



Overfitting

This is why we can't use training MSE to select the model order:

- Notice that we are choosing among a nested set of models

$$\begin{aligned} \boldsymbol{\beta} &= [\beta_0, \beta_1, 0, 0, 0, \dots]^\mathsf{T} && \text{when } d=1 \\ \boldsymbol{\beta} &= [\beta_0, \beta_1, \beta_2, 0, 0, \dots]^\mathsf{T} && \text{when } d=2 \\ \boldsymbol{\beta} &= [\beta_0, \beta_1, \beta_2, \beta_3, 0, \dots]^\mathsf{T} && \text{when } d=3 \\ &\vdots && \vdots \end{aligned}$$

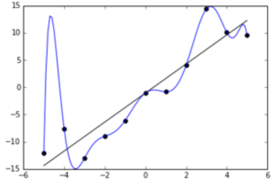
  so that each model is more capable than the previous model

- The LS *training MSE gets no worse* as the model becomes more capable
  - Minimizing the training MSE leads to choosing the largest d

- When $d \geq n-1$, the least-squares $\text{MSE}_{\textsf{train}}(d) = 0$

$$\frac{1}{n}\|\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{\beta}}\|^2 = 0$$

- When d is large, $\widehat{y}_i$ tries to fit the training noise ϵ_i
  - This is the main characteristic of overfitting!



https://en.wikipedia.org/wiki/Overfitting

Outline

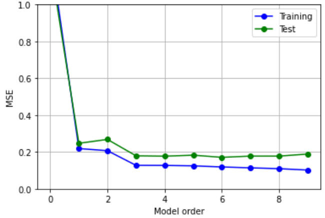
- Motivating Example: Polynomial Degree Selection
- Cross-validation
- The Bias-Variance Tradeoff
- From Model-Order Selection to Feature Selection

To prevent overfitting, use cross-validation

Main idea:

Evaluate performance on "test data" that is independent of the training data

- Simplest version: Partition total dataset into two subsets, know as "folds":
  1. $n_{\textsf{train}}$ training samples: $\{(\boldsymbol{x}_{\textsf{train},i}, y_{\textsf{train},i})\}_{i=1}^{n_{\textsf{train}}}$
  2. $n_{\textsf{test}} = n - n_{\textsf{train}}$ test samples: $\{(\boldsymbol{x}_{\textsf{test},i}, y_{\textsf{test},i})\}_{i=1}^{n_{\textsf{test}}}$
- Then, for each hypothesized model-order d:
  - Compute LS coefficients $\widehat{\boldsymbol{\beta}}$ from training data
  - Predict the test targets: $\widehat{y}_{\textsf{test},i} = [1\ \boldsymbol{x}_{\textsf{test},i}^{\mathsf{T}}]\widehat{\boldsymbol{\beta}}$
  - Compute $\mathrm{MSE}_{\textsf{test}}(d) \triangleq \frac{1}{n_{\textsf{test}}} \sum_{i=1}^{n_{\textsf{test}}} (y_{\textsf{test},i} - \widehat{y}_{\textsf{test},i})^2$
- Finally, choose the d that minimizes $\mathrm{MSE}_{\textsf{test}}(d)$ ...better but not perfect



Suggests model-order $d = 6$

K-fold cross-validation

Previously we considered splitting into training & test. More sophisticated approaches:

- K-fold cross validation (CV)
  - Partition (shuffled!) data into K folds
  - Train using $K-1$ folds, test using 1 fold
  - Repeat for each of K possible test folds
  - Typically use $K = 5$ or 10
  - Expensive: requires K parameter fits
  - Good approx of true performance!
- Leave-one-out cross validation (LOOCV)
  - Extreme case where $K = n$
    (each test fold contains 1 sample!)
  - Very expensive unless n is small

https://medium.com/@sebastiannorena

Implementing K-fold cross-validation with sklearn

- Nested for-loop approach to CV:
  - Loop over $k = 1, \ldots, K$ folds
  - Loop over $d = 1, \ldots, D$ model-orders
  - Compute test $\text{MSE}_{d,k}$ for each order d & fold k
  - Average test MSE across K folds to get $\overline{\text{MSE}}_d \triangleq \frac{1}{K} \sum_{k=1}^{K} \text{MSE}_{d,k}$
  - Choose d giving smallest $\overline{\text{MSE}}_d$
- Can use sklearn's KFold method to generate index sets for the folds!

```
# Create a k-fold object
k = 10
kfo = sklearn.model_selection.KFold(n_splits=k,shuffle=True)

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

MSEts = np.zeros((nd,k))

# Loop over the folds
for isplit, Ind in enumerate(kfo.split(xdat)): # enumerate r

    # Get the training data in the split
    Itr, Its = Ind
    #kfo.split( ) produced Ind, which contains a pair of ind
    xtr = xdat[Itr]
    ytr = ydat[Itr]
    xts = xdat[Its]
    yts = ydat[Its]

    # Loop over the model order
    for it, d in enumerate(dtest):

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

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

Confidence intervals for K-fold cross-validation

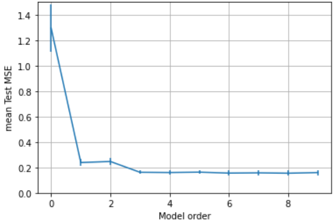
- Problem: $\overline{\text{MSE}}_d = \frac{1}{K}\sum_{k=1}^{K} \text{MSE}_{d,k}$ may inaccurately estimate true MSE_d when K is small

- Can compute confidence bounds on $\overline{\text{MSE}}_d$ using the so-called standard error (SE):

$$\text{SE}_d \triangleq \frac{\text{std}(\text{MSE}_d)}{\sqrt{K}}, \text{ where}$$

$$\text{std}(\text{MSE}_d) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(\text{MSE}_{d,k} - \overline{\text{MSE}}_d)^2}$$

line shows $\overline{\text{MSE}}_d$, error bars show SE_d:

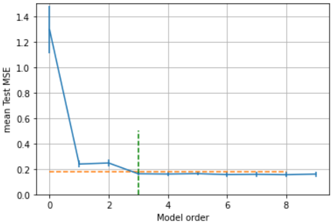


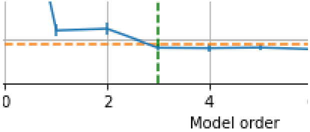
- Above, $\frac{1}{K-1}$ gives "unbiased" estimate of $\text{var}(\text{MSE}_d)$, implemented via `ddof=1` below

```
MSE_mean = np.mean(MSEts,axis=1) #note mean is taken over
MSE_se = np.std(MSEts,axis=1,ddof=1)/np.sqrt(k)
plt.errorbar(dtest, MSE_mean, yerr=MSE_se, fmt='-')
plt.ylim(0,1.5)
plt.xlabel('Model order')
plt.ylabel('mean Test MSE')
plt.grid()
```

The one-standard-error rule

- Previously, said to choose d minimizing $\overline{\mathrm{MSE}}_d$
  - But this sometimes overfits true model-order!
- Better approach: one-standard-error (OSE) rule
  - Use *simplest* model giving $\overline{\mathrm{MSE}}_d$ within one SE of minimum $\overline{\mathrm{MSE}}$
- Detailed procedure:
  - Set $d_{\min} = \arg\min_d \overline{\mathrm{MSE}}_d$
  - Set $\overline{\mathrm{MSE}}_{\textsf{tgt}} = \overline{\mathrm{MSE}}_{d_{\min}} + \mathrm{SE}_{d_{\min}}$
  - Find smallest d such that $\overline{\mathrm{MSE}}_d \leq \overline{\mathrm{MSE}}_{\textsf{tgt}}$
- In example on right: $d_{\min} = 8$, but OSE selects $d = 3$, which is the true model-order





Training, test, and validation data

- Sometimes you will see <u>three</u> folds of data...
  1. Training data: Used to train the model
     - used during design
  2. Test data: Used to tune the model hyperparameters (e.g., model order)
     - used during design
  3. Validation data: Used to estimate model performance on unseen data
     - used only *after* your design is finalized
     - Withheld from the contestants in ML contests (e.g., kaggle)
- In many cases, the definitions of "test" and "validation" are swapped!
  - Always make sure you know the intended meaning
  - In this course, we'll use the definitions above
  - In this unit, we will consider only training & test data

Outline

Statistical learning theory

- With degree-d polynomial regression, we saw that
  - choosing d too small causes underfitting
  - choosing d too large causes overfitting
  - d can be optimized by minimizing the sample $\text{MSE}_{\textsf{test}}$ through cross-validation
- But this is just one special case of a more general concept:
  - models that are too simple cause underfitting
  - models that are too complex cause overfitting
  - model complexity can be optimized by minimizing the *statistical* mean-squared error, $\text{MSE}_{\widehat{y}}$
- From a theoretical perspective, we will see that ...
  - analyzing $\text{MSE}_{\widehat{y}}$ leads to the bias-variance equation
  - minimizing $\text{MSE}_{\widehat{y}}$ involves a tradeoff between bias and variance

Random Variables

- The concept of randomness is useful when we want to describe what *might* happen, or what *tends* to happen, in machine-learning experiments
- A random variable (RV) can generate different possible values, each with a prescribed probability
  - The values generated by a RV are called realizations
  - There are two types of RVs: discrete and continuous
- A discrete RV "A" takes on values from a countable set $\{a^{(1)}, a^{(2)}, ..., a^{(K)}\}$
  - It's described by its probability mass function (pmf) $\boldsymbol{p}_A = [p_{A,1}, \dots, p_{A,K}]^\mathsf{T}$
  - Here, $p_{A,k} \triangleq \Pr\{A = a^{(k)}\}$, where $0 \leq p_{A,k} \leq 1$ and $\sum_{k=1}^{K} p_{A,k} = 1$
- A continuous RV "A" takes on an uncountable number of values from $\mathbb{R}$
  - It's described by its cumulative distribution function (cdf) $P_A(a) \triangleq \Pr\{A \leq a\}$
  - Also described by its probability density function (pdf) $p_A(\cdot) \triangleq \frac{d}{da} P_A(\cdot)$
  - Note $\Pr\{A \leq a\} = \int_{-\infty}^{a} p_A(a') \, \mathrm{d}a'$, where $p_A(a) \geq 0$ and $\int_{-\infty}^{\infty} p_A(a') \, \mathrm{d}a' = 1$
- See the document: A Primer on Probability and Expectation

Expectation

- Expectation $\mathbb{E}\{\cdot\}$ is the statistical mean of a random variable
- Formally, for any function $f(\cdot)$ and random variable A,

$$\mathbb{E}\{f(A)\} = \sum_{k=1}^{K} f(a^{(k)})\, p_{A,k} \quad \text{for a discrete RV}$$

$$\mathbb{E}\{f(A)\} = \int_{-\infty}^{\infty} f(a)\, p_A(a)\, \mathrm{d}a \quad \text{for a continuous RV}$$

  Vector-valued random variables (e.g., $\boldsymbol{a} \in \mathbb{R}^M$) can be handled similarly

- We will avoid formalities for now and focus on two key properties of $\mathbb{E}\{\cdot\}$: For any functions $f(\cdot)$ & $g(\cdot)$ and random variables A & B:
  - $\mathbb{E}\{c + d\, f(A)\} = c + d\,\mathbb{E}\{f(A)\}$ for deterministic c and d (by linearity)
  - $\mathbb{E}\{f(A)g(B)\} = \mathbb{E}\{f(A)\}\,\mathbb{E}\{g(B)\}$ for independent A and B:
    $p_{A,B}(a,b) = p_A(a)p_B(b)\ \forall a, b$

- Variance is defined as $\mathrm{var}\{A\} \triangleq \mathbb{E}\{(A - \mathbb{E}\{A\})^2\} = \mathbb{E}\{A^2\} - \mathbb{E}\{A\}^2$

Conditional expectation

- Conditional expectation $\mathbb{E}\{f(A) \,|\, B=b\}$ is the mean value of a function $f(\cdot)$ of random variable A given that some other random variable B equals b.
- Formally, for any function $f(\cdot)$ and random variables $A, B \in \mathbb{R}$,

$$\mathbb{E}\{f(A) \,|\, B\!=\!b\} = \int_{-\infty}^{\infty} f(a)\, p_{A|B}(a|b)\, \mathrm{d}a \quad \text{for conditional pdf } p_{A|B}(\cdot|\cdot)$$

- Often we will see $\mathbb{E}\{f(A) \,|\, B\}$. This is like $\mathbb{E}\{f(A) \,|\, B\!=\!b\}$ but with b replaced by random variable B. Thus, $\mathbb{E}\{f(A) \,|\, B\}$ is random
- We will frequently encounter the law of total expectation, which says that $\mathbb{E}\{A\} = \mathbb{E}\{\ \mathbb{E}\{A \,|\, B\}\ \}$.
  - This is often used to compute expectation one variable at a time.
- See the document: A Primer on Probability and Expectation

Statistical model

Setup for our theoretical analysis...

- True model: $\boxed{y = f(\boldsymbol{x}) + \epsilon}$ with $\mathbb{E}\{\epsilon\} = 0$ and $\text{var}\{\epsilon\} = \sigma^2$
  - noise ϵ is random with mean zero and variance σ^2, and independent over draws
  - feature vectors $\boldsymbol{x}$ also random, independent over draws, and independent of ϵ
  - model holds for both training $\{(\boldsymbol{x}_i, y_i, \epsilon_i)\}_{i=1}^n$ and test $(\boldsymbol{x}, y, \epsilon)$ quantities, which are independent of each other
- Prediction model: $\boxed{\widehat{y} = \widehat{f}(\boldsymbol{x}; \widehat{\boldsymbol{\beta}})}$ for some $\widehat{f}$ and trained coefficients $\widehat{\boldsymbol{\beta}}$
  - $\widehat{\boldsymbol{\beta}}$ was designed from training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and thus is random
  - $\{\boldsymbol{x}, \epsilon, \widehat{\boldsymbol{\beta}}\}$ are mutually independent
- Mean-squared error on $\widehat{y}$ for a given $\boldsymbol{x}$: $\boxed{\text{MSE}_{\widehat{y}}(\boldsymbol{x}) \triangleq \mathbb{E}\left\{(y - \widehat{y})^2 \mid \boldsymbol{x}\right\}}$
  - this expectation averages over ϵ (in y) and $\widehat{\boldsymbol{\beta}}$ (in $\widehat{y}$), but holds $\boldsymbol{x}$ fixed

Writing MSE in terms of bias and variance

We now analyze the statistical MSE for fixed test features $\boldsymbol{x}$:

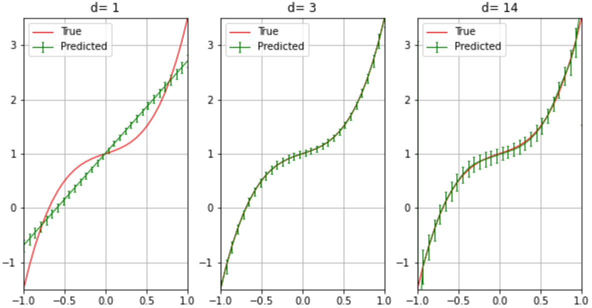
$$
\begin{aligned}
\text{MSE}_{\widehat{y}}(\boldsymbol{x}) &\triangleq \mathbb{E}\left\{(y-\widehat{y})^2 \,\middle|\, \boldsymbol{x}\right\} = \mathbb{E}\left\{\left(\epsilon + f(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})\right)^2 \,\middle|\, \boldsymbol{x}\right\} \\
&= \mathbb{E}\left\{\epsilon^2 + 2\epsilon\big(f(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})\big) + \big(f(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})\big)^2 \,\middle|\, \boldsymbol{x}\right\} \qquad \swarrow \ \text{via linearity and independence of } \epsilon \ \&\ \widehat{\beta} \\
&= \mathbb{E}\{\epsilon^2 \,|\, \boldsymbol{x}\} + 2\cancelto{0}{\mathbb{E}\{\epsilon \,|\, \boldsymbol{x}\}}\,\mathbb{E}\left\{f(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}}) \,\middle|\, \boldsymbol{x}\right\} + \mathbb{E}\left\{\big(f(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})\big)^2 \,\middle|\, \boldsymbol{x}\right\} \\
&= \sigma^2 + \mathbb{E}\left\{\big(f(\boldsymbol{x}) - \mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}] + \mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}] - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})\big)^2 \,\middle|\, \boldsymbol{x}\right\} \\
&= \sigma^2 + \mathbb{E}\left\{\big(f(\boldsymbol{x}) - \mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}]\big)^2 \,\middle|\, \boldsymbol{x}\right\} + \mathbb{E}\left\{\big(\mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}] - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})\big)^2 \,\middle|\, \boldsymbol{x}\right\} \\
&\qquad + 2\big(f(\boldsymbol{x}) - \mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}]\big)\cancelto{0}{\mathbb{E}\left\{\mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}] - \widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}}) \,\middle|\, \boldsymbol{x}\right\}} \\
&= \underbrace{\sigma^2}_{\text{irreducible error}} + \Big(\underbrace{f(\boldsymbol{x}) - \mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}]}_{\text{bias}_{\widehat{y}}(\boldsymbol{x}) \,\triangleq\, \mathbb{E}\{y-\widehat{y} \,|\, \boldsymbol{x}\}}\Big)^2 + \underbrace{\mathbb{E}\left\{\big(\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}}) - \mathbb{E}[\widehat{f}(\boldsymbol{x};\widehat{\boldsymbol{\beta}})|\boldsymbol{x}]\big)^2 \,\middle|\, \boldsymbol{x}\right\}}_{\text{var}\{\widehat{y} \,|\, \boldsymbol{x}\}}
\end{aligned}
$$

We can go one step further and take the mean of $\text{MSE}_{\widehat{y}}(\boldsymbol{x})$ over random $\boldsymbol{x}$:

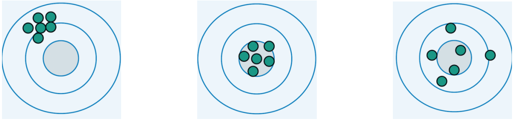
$$\text{MSE}_{\widehat{y}} \triangleq \mathbb{E}\left\{\text{MSE}_{\widehat{y}}(\boldsymbol{x})\right\} = \sigma^2 + \mathbb{E}\{\text{bias}_{\widehat{y}}(\boldsymbol{x})^2\} + \mathbb{E}\{\text{var}\{\widehat{y} \,|\, \boldsymbol{x}\}\}$$

A bias-variance experiment for polynomial models

- Polynomial demo
- Red curve: $f(x) = \mathbb{E}\{y|x\}$
- Solid green curves: $\mathbb{E}\{\widehat{y}\,|\,x\}$
  - $\text{bias}_{\widehat{y}}(x)$ is gap between red & green curves at x
- Green error-bars: $\sqrt{\text{var}\{\widehat{y}\,|\,x\}}$
- The $\mathbb{E}\{.\}$ and $\text{var}\{.\}$ are approximated by sample-averaging 100 independent train/test experiments



high bias, low variance | no bias, medium variance | no bias, high variance



A bias-variance analysis of LS linear regression

- Consider noisy linear training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$:
  - $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ with $f(\boldsymbol{x}_i) = \beta_0 + \sum_{j=1}^{d_{\sf true}} \beta_j x_{ij}$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

  and the d-term linear regression model:
  - $\widehat{y} = \widehat{f}(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}) = \widehat{\beta}_0 + \sum_{j=1}^{d} \widehat{\beta}_j x_j$ with LS weights $\widehat{\boldsymbol{\beta}}$

- Result 1: If $n < d+1$, then $\widehat{\boldsymbol{\beta}}$ is not unique, so LS solution undefined

- Result 2: If $n \geq d+1$ and $d < d_{\sf true}$, then $\widehat{y}$ will be biased due to underfitting

- Result 3: If $n \geq d+1$ and $d \geq d_{\sf true}$, then $\widehat{y}$ is unbiased, i.e.,

$$\boxed{\text{bias}_{\widehat{y}}(\boldsymbol{x}) = \mathbb{E}\{\widehat{y} - y \,|\, \boldsymbol{x}\} = 0}$$

- Result 4: If $n \gg d$ and $d \geq d_{\sf true}$ and $\boldsymbol{x}$ has same distribution as $\{\boldsymbol{x}_i\}$,

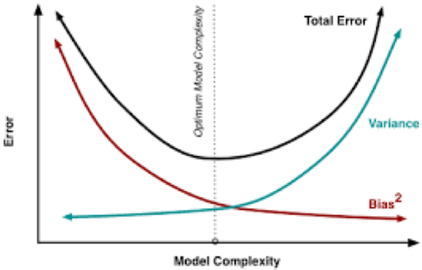
$$\boxed{\mathbb{E}\{\text{var}\{\widehat{y} \,|\, \boldsymbol{x}\}\} \approx \frac{d+1}{n}\sigma^2} \quad \text{so} \quad \begin{cases} \text{variance increases linearly with \# model parameters} \\ \text{variance decreases inversely with \# training samples} \end{cases}$$

Details in handout "Bias and Variance Analysis of Multiple Linear Regression"

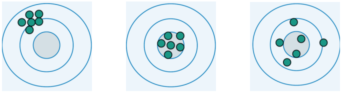
The bias-variance tradeoff for general regression

$$\mathrm{MSE}_{\widehat{y}} = \sigma^2 + \mathbb{E}\{\mathrm{bias}_{\widehat{y}}(\boldsymbol{x})^2\} + \mathbb{E}\{\mathrm{var}\{\widehat{y} \,|\, \boldsymbol{x}\}\}$$

- We saw two examples of how $\mathrm{MSE}_{\widehat{y}}$ changes with model complexity:
  - polynomials: polynomial degree d
  - linear regression: # features d
- Similar trends hold for general models!
  - There exists a tradeoff between bias and variance
- The optimal model complexity depends on
  - the true model complexity (which affects bias)
  - the number of training samples (which affects variance)



$\Leftarrow$	$\Rightarrow$
simpler models	richer models
less parameters	more parameters
underfitting	overfitting



Example: bias and variance of the sample estimators

- Consider random variable Z with mean $\mathbb{E}\{Z\} = \mu$ and variance $\mathrm{var}\{Z\} = v$
- Say we want to estimate μ from n independent realizations $\{z_i\}_{i=1}^n$ of Z
  - It's common to use the sample mean $\overline{z} \triangleq \frac{1}{n}\sum_{i=1}^n z_i$
  - Can show that $\mathbb{E}\{\overline{z}\} = \mu$. Thus $\overline{z}$ is an unbiased estimate of μ
  - Can also show that $\mathrm{var}\{\overline{z}\} = \mathbb{E}\{(\overline{z}-\mu)^2\} = v/n$. Standard error $= \sqrt{\mathrm{var}\{\overline{z}\}}$
- Now say we want to estimate v from independent realizations $\{z_i\}_{i=1}^n$ of Z
  - First consider sample variance $s_{zz} \triangleq \frac{1}{n}\sum_{i=1}^n (z_i - \overline{z})^2$
  - Can show that $\mathbb{E}\{s_{zz}\} = \frac{n-1}{n}v$. Thus s_{zz} is a biased estimate of v
  - But $\xi_{zz} \triangleq \frac{1}{n-1}\sum_{i=1}^n (z_i - \overline{z})^2$ is an unbiased estimate of v
  - Although s_{zz} is biased, it has a lower mean-squared error. If $Z \sim \mathcal{N}(\mu, v)$ then

$$\mathbb{E}\{(s_{zz}-v)^2\} = \frac{2n+1}{n^2}v^2 \quad < \quad \frac{2}{n-1}v^2 = \mathbb{E}\{(\xi_{zz}-v)^2\}$$

Details in handout "On the Bias and Variance of the Sample Estimators"

Outline

Feature selection: A generalization of model-order selection

- So far, we discussed model-*order* selection (e.g., polynomial degree d)
  - Select between several models, each with a different complexity
  - For example, $$\begin{aligned} y &\approx \beta_0 + \beta_1 x_1 \\ y &\approx \beta_0 + \beta_1 x_1 + \beta_2 x_2 \\ y &\approx \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \end{aligned}$$
- More generally, given d total features $\{x_j\}_{j=1}^{d}$, we might wonder which subset of features works best for predicting y
  - Called "feature selection"
  - Given d features (plus intercept), there are 2^d possible subsets
- How do we choose the best subset?
  - Can use cross-validation to choose between models
  - but need to manage computational complexity...
- Discussed further in the next unit ...

Learning objectives

- Understand the problem of model-order selection
- Visually identify underfitting and overfitting in a scatterplot
- Understand the need to partition data into training and testing subsets
- Understand the K-fold cross-validation process
  - Use it to assess the test error for a given model
  - Use it to select model order
- Understand the concepts of bias, variance, and irreducible error
  - Know how to compute each from synthetically generated data
  - Understand the bias-variance tradeoff