Big Data Analytics

Session 5(a)

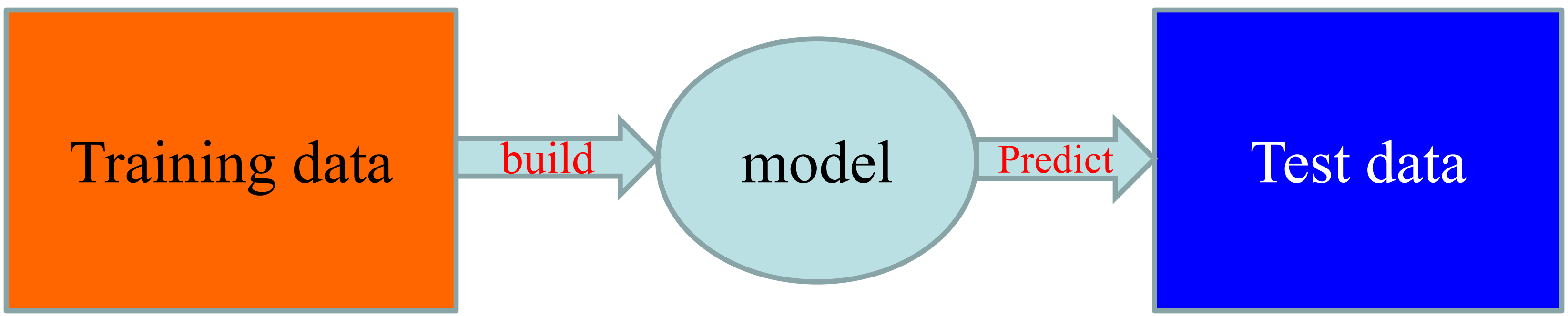
Assessing Model Accuracy

Outline

- Assessing Model Accuracy (Chapter 2.2)
  - Measuring the Quality of Fit
  - The Bias-Variance Trade-off
  - The Classification Setting

The big picture

- The general way of statistical learning



- Training data: the existing known data
- Test data: the new data that we would like to explore

Measuring Quality of Fit

- Suppose we have a regression problem.
  - Recall residual sum of squares (RSS):

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Where $\hat{y}_i$ is the prediction our method gives for the observation in our training data.

Measuring Quality of Fit

- Suppose we have a regression problem.
  - Recall residual sum of squares (RSS):

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

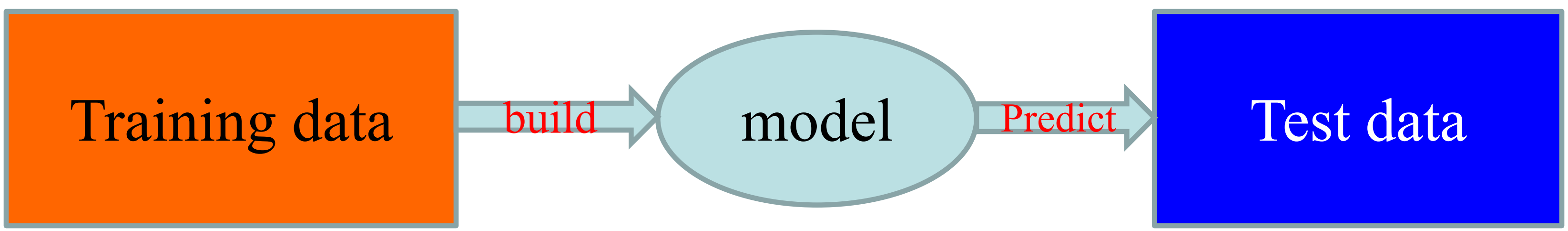
- One common measure of accuracy is the mean squared error (MSE) i.e.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{n} \mathrm{RSS}$$

- Where $\hat{y}_i$ is the prediction our method gives for the observation in our training data.

A Problem

- Our method has generally been designed to make MSE small on the training data we are looking at
  - e.g. with linear regression we choose the line such that MSE (RSS) is minimised → least squares line.



- What we really care about is how well the method works on the **test data**.

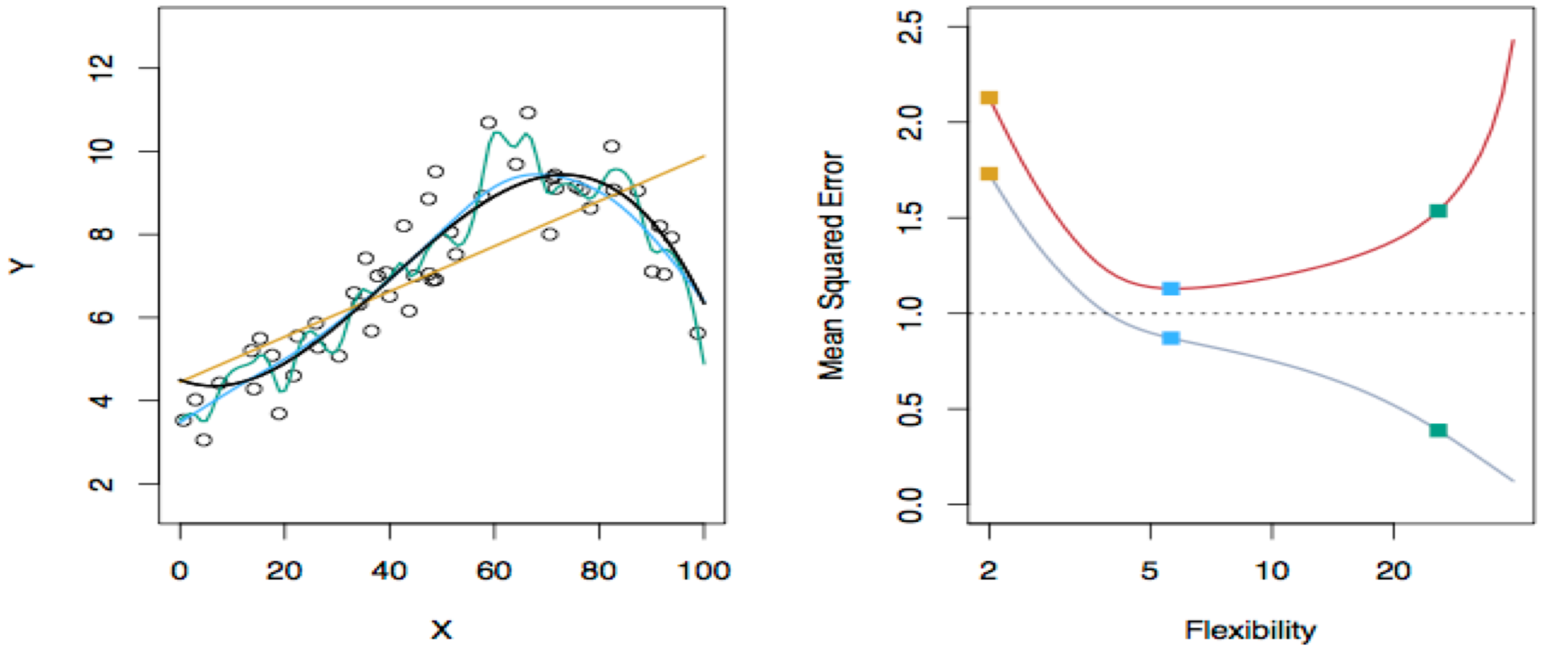
- There is no guarantee that the method with the smallest training MSE will have the smallest test MSE.

Training vs. Test MSE's

- In general,
  the more flexible a method is,
  the lower its training MSE will be
  i.e. it will "fit" or explain the training data very well.

- However, the test MSE may in fact be higher for a more flexible method than for a simple approach like linear regression.

Examples with Different Levels of Flexibility



<u>LEFT</u>
Black: Truth
Orange: Linear Estimate
Blue: smoothing spline
Green: smoothing spline (more flexible)

<u>RIGHT</u>
RED: Test MSE
Grey: Training MSE
Dashed: Minimum possible test MSE (irreducible error)

Bias/Variance Tradeoff

- The previous graph of test versus training MSE's illustrates a very important tradeoff that governs the choice of statistical learning methods.

- There are always two competing forces that govern the choice of learning method i.e. bias and variance.

Bias of Learning Methods

- Bias refers to the error that is introduced by modeling a real life problem (that is usually extremely complicated) by a much simpler problem.

- For example, linear regression assumes that there is a linear relationship between Y and X.

  It is unlikely that, in real life, the relationship is exactly linear so some bias will be present.

- The more flexible/complex a method is the less bias it will generally have.

Variance of Learning Methods

- Variance refers to how much your estimate for f would change by if you had a different training data set (from the same population).

- Generally, the more flexible a method is the more variance it has.

The Trade-Off

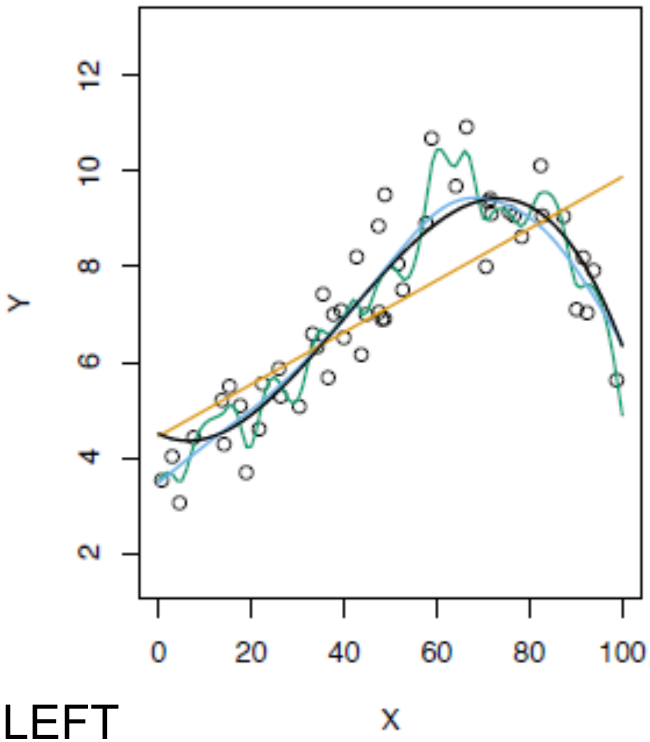
- The expected test MSE is equal to

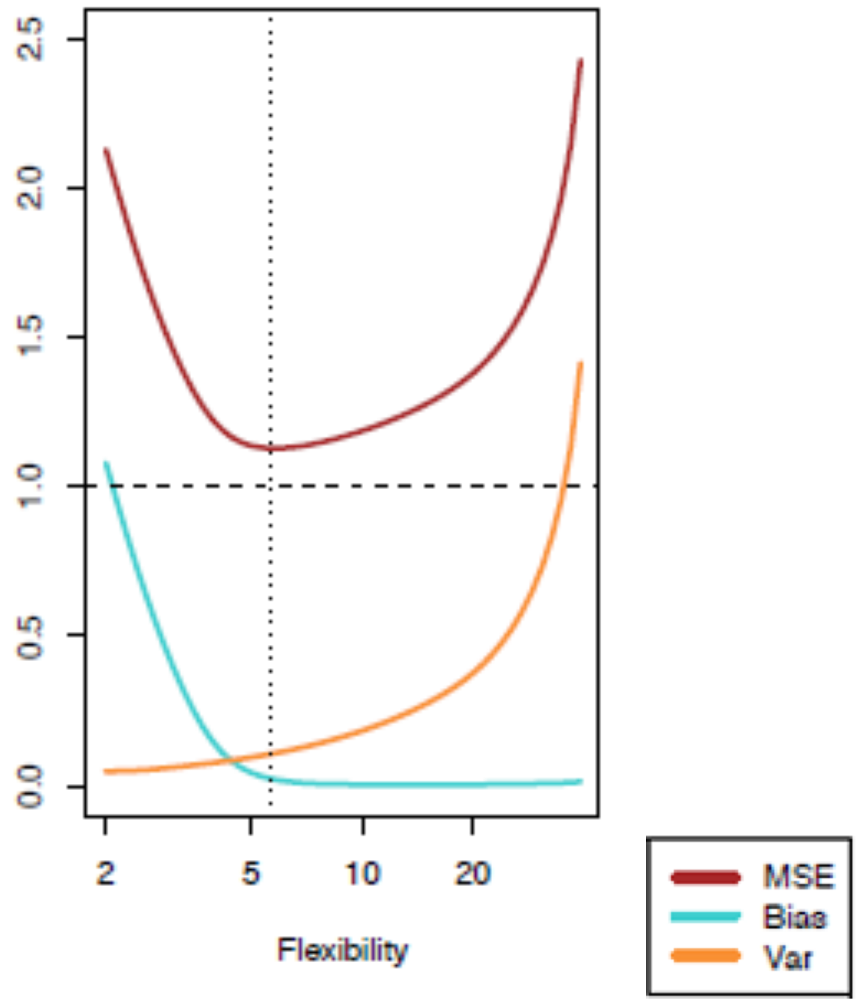
$$Expected\,Test\,MSE = Bias^2 + Var + \underbrace{\sigma^2}_{\text{Irreducible Error}}$$

Method	**Bias**	**Variance**	**Expected TestMSE**
more complex	decrease	increase	Decrease or increase?
simpler	increase	decrease	Unknown!

- It is a challenge to find a method for which both the variance and the squared bias are low.
  - This trade-off is one of the most important recurring themes in this course.

Test MSE, Bias and Variance





LEFT
Black: Truth
Orange: Linear Estimate
Blue: smoothing spline
Green: smoothing spline (more flexible)

How to calculate MSE in R?

- Consider the linear regression models
  - Recall

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - Given the dataset DS, we compute its training MSE

```
>lm.fit <- lm(y~x,data=DS)
>mean((y-predict(lm.fit,DS))^2)
```

- Try it on the Auto data set

  y: mpg -- gas mileage (miles per gallon)

  x: horsepower -- engine horsepower

  Training MSE is 23.94366

The Classification Setting

- For a regression problem, we used the MSE to assess the accuracy of the statistical learning method
- For a classification problem we can use the error rate.

Evaluation of classification models

- First, get a confusion matrix
  - Counts of test records that are correctly (or incorrectly) predicted by the classification model

Predicted Class / **Actual Class**

	Class = 1	Class = 0
Class = 1	f_{11}	f_{10}
Class = 0	f_{01}	f_{00}

f_{11} is the number of records that are actually 1 and are predicted to be 1 .

f_{10} is the number of records that are actually 1 and are predicted to be 0 .

f_{00} and f_{01} are defined similarly.

- **Then compute error rate**

$$\text{Accuracy} = \frac{\text{\# correct predictions}}{\text{total \# of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\text{\# wrong predictions}}{\text{total \# of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

An Example for Confusion Matrix

- Given the following table of 10 observations with their actual y value and predicted y value.
  - Draw your confusion matrix.
  - Calculate the accuracy rate and error rate.

Obs.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Actual value	Yes	Yes	No	No	No	No	No	No	Yes	No
Predicted value	No	Yes	Yes	Yes	No	Yes	Yes	No	No	No

An Example for Confusion Matrix

- Confusion matrix:

Predicted Class

Actual Class	Class = Yes	Class = No
Class = Yes		
Class = No		

Obs.	1	2	3	4	5	6	7	8	9	10
Actual value	Yes	Yes	No	No	No	No	No	No	Yes	No
Predicted value	No	Yes	Yes	Yes	No	Yes	Yes	No	No	No

An Example for Confusion Matrix

- Confusion matrix:

Predicted Class

Actual Class

	Class = Yes	Class = No
Class = Yes	1	2
Class = No	4	3

Accuracy = (1+3)/10=0.4
Error rate = (4+2)/10=0.6

Obs.	1	2	3	4	5	6	7	8	9	10
Actual value	Yes	Yes	No	No	No	No	No	No	Yes	No
Predicted value	No	Yes	Yes	Yes	No	Yes	Yes	No	No	No

How to Calculate Error Rate in R

- In logistic regression, calculate the training error rate
  - Building the glm.fit
  - Using glm.fit to make probability predictions
  - Set a threshold (could be 0.5, or other number) to make qualitative predictions based on the probability predictions
  - Using table() function to build a confusion matrix
  - Using mean() function to calculate the error rate
- Try it on the Default data set

Code

```
glm.fit <- glm(default~balance,data = Default, family=binomial)
dim(Default)
#[1] 10000 4

glm.probs <- predict(glm.fit, Default, type="response")
glm.pred <- rep("Yes",10000)
glm.pred[glm.probs<.5] <- "No"
table(glm.pred,default)

default
glm.pred No Yes
 No      9625 233
Yes      42 100
```