BIOST 546: Machine Learning for Biomedical Big Data

Ali Shojaie

Lecture 2: Linear Regression for Big Data
Spring 2017

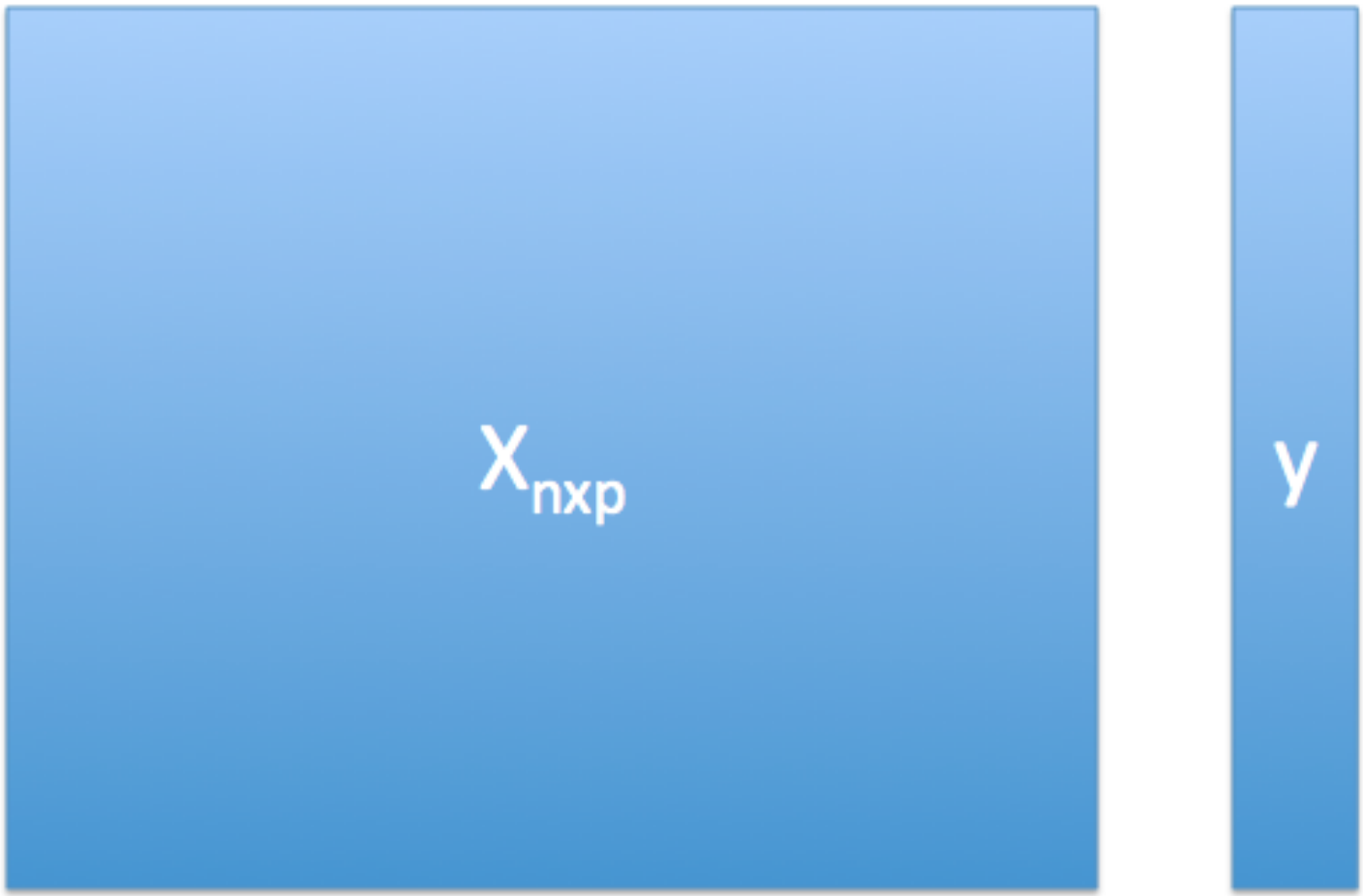
Recap

- Overview of statistical learning
- Supervised vs Unsupervised learning
- Challenges of Biomedical Big Data (briefly)

Today

- Overview of linear regression
- Overfitting
- Training and test errors
- Cross validation

Supervised Learning



Regression Versus Classification

Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size

Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size
- Classification: Predict a categorical response, such as
  - tumor versus normal tissue
  - heart disease versus no heart disease
  - subtype of glioblastoma

Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size
- Classification: Predict a categorical response, such as
  - tumor versus normal tissue
  - heart disease versus no heart disease
  - subtype of glioblastoma
- This lecture: Regression.

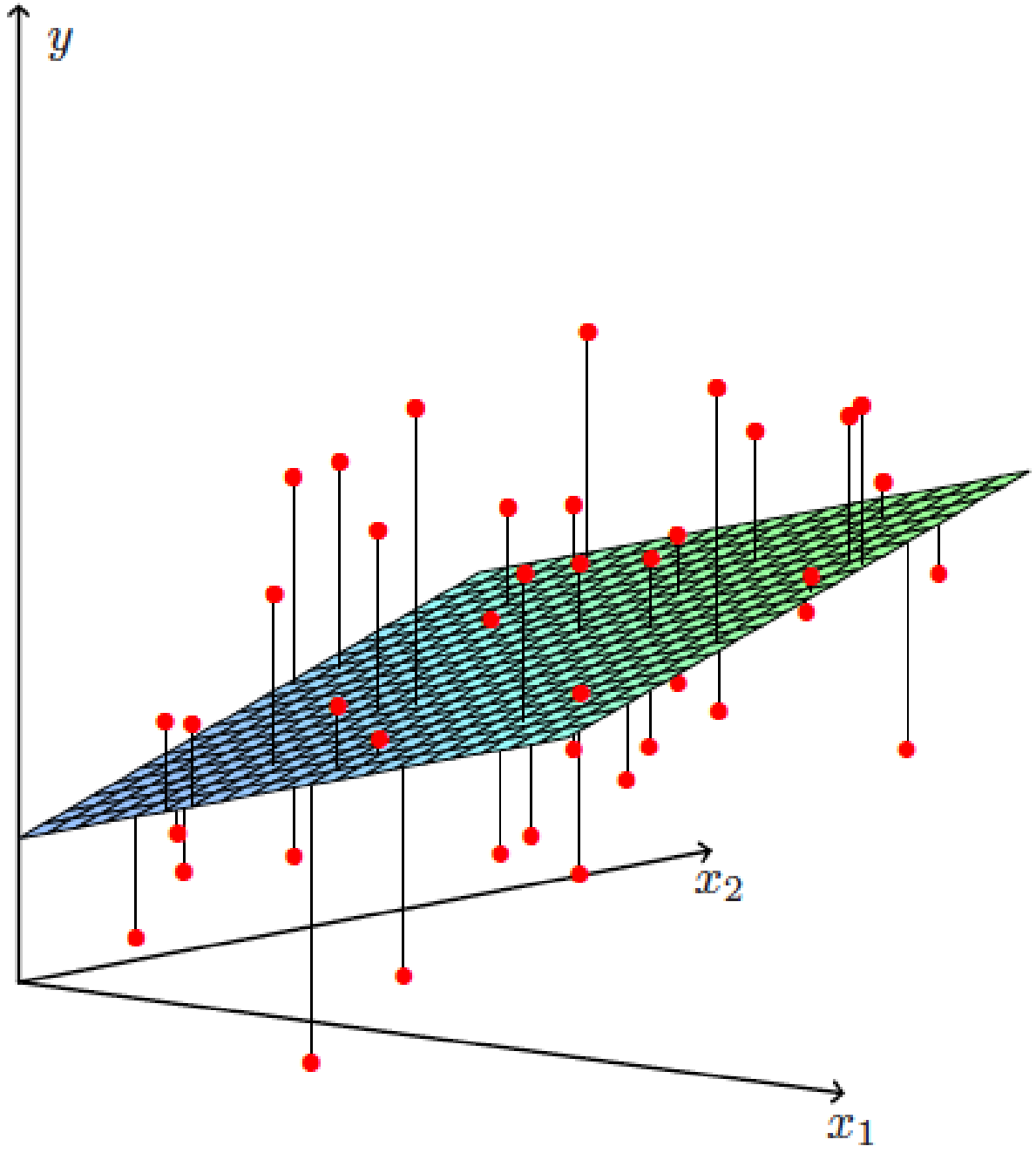
Linear Models

- We have n observations, for each of which we have p predictor measurements and a response measurement.
- Want to develop a model of the form

$$y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \varepsilon_i.$$

- Here ε_i is a noise term associated with the ith observation.
- Must estimate $\beta_0, \beta_1, \ldots, \beta_p$ – i.e. we must fit the model.

Linear Model With $p = 2$ Predictors



What Makes a Model Linear?

What Makes a Model Linear?

- A linear model is linear in the regression coefficients!

What Makes a Model Linear?

- A linear model is linear in the regression coefficients!
- This is a linear model:

$$y_i = \beta_1 \sin(X_{i1}) + \beta_2 X_{i2} X_{i3} + \varepsilon_i.$$

What Makes a Model Linear?

- A linear model is linear in the regression coefficients!
- This is a linear model:

$$y_i = \beta_1 \sin(X_{i1}) + \beta_2 X_{i2} X_{i3} + \varepsilon_i.$$

- This is not a linear model:

$$y_i = \beta_1^{X_{i1}} + \sin(\beta_2 X_{i2}) + \varepsilon_i.$$

Linear Models in Matrix Form

- For simplicity, ignore the intercept β_0.
  - Assume $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} X_{ij} = 0$; in this case, $\beta_0 = 0$.
  - Alternatively, let the first column of $\mathbf{X}$ be a column of 1's.
- In matrix form, we can write the linear model as

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon,$$

i.e.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Least Squares Regression I

- There are a lot of ways we could fit the model

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon.$$

- Most common approach in classical statistics is least squares:

$$\underset{\beta}{\text{minimize}} \left\{ \|\mathbf{y} - \mathbf{X}\beta\|^2 \right\}.$$

  Here $\|\mathbf{a}\|^2 \equiv \sum_{i=1}^{n} a_i^2$.

- We are looking for $\beta_1, \ldots, \beta_p$ such that

$$\sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \ldots + \beta_p X_{ip}))^2$$

  is as small as possible.

Connecting back to the last lecture...

- Recall from last lecture that we ideally want to find a model that minimizes $E(y - \hat{y})^2$. How does least squares relate to this?

Connecting back to the last lecture...

- Recall from last lecture that we ideally want to find a model that minimizes $E(y - \hat{y})^2$. How does least squares relate to this?
- In least squares, we're looking for coefficient estimates such that

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

is as small as possible, where $\hat{y}_i$ is the ith predicted value.

Connecting back to the last lecture...

- Recall from last lecture that we ideally want to find a model that minimizes $E(y - \hat{y})^2$. How does least squares relate to this?
- In least squares, we're looking for coefficient estimates such that

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

  is as small as possible, where $\hat{y}_i$ is the ith predicted value.
- In the last lecture we also mentioned that we want to find $\hat{f}$ such that the reducible error is minimized...

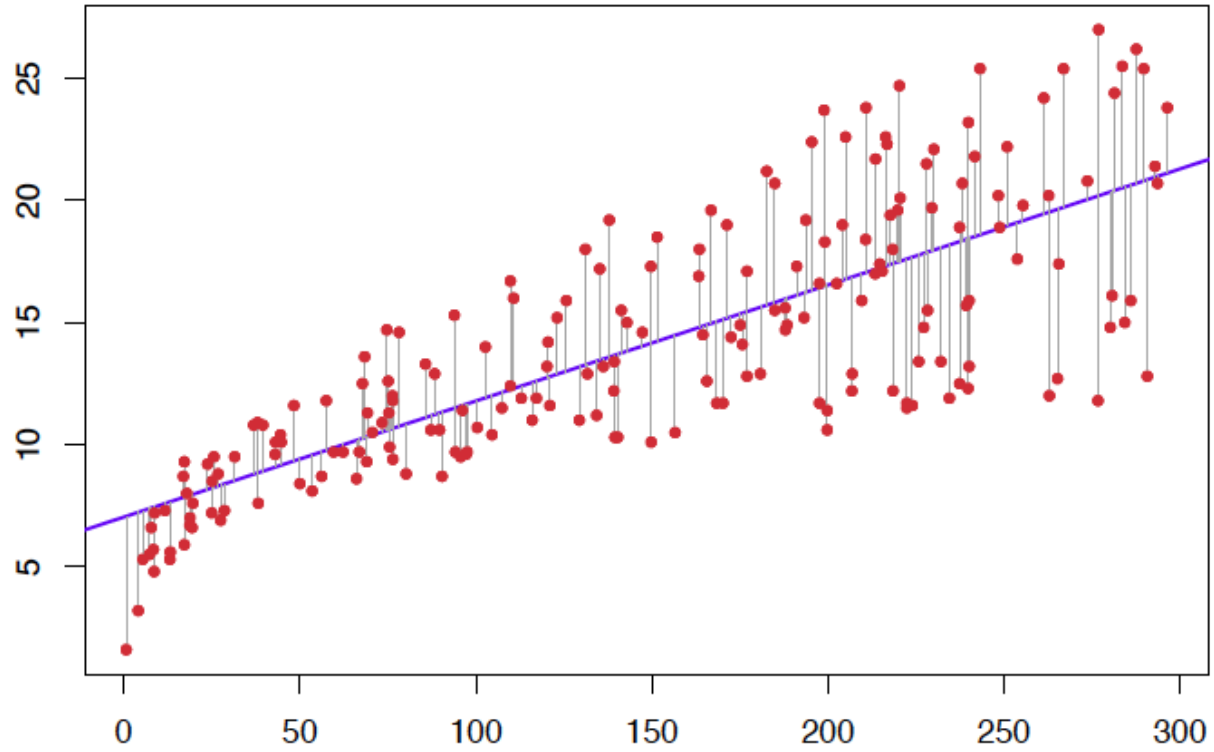
Connecting back to the last lecture...

- Recall from last lecture that we ideally want to find a model that minimizes $E(y - \hat{y})^2$. How does least squares relate to this?
- In least squares, we're looking for coefficient estimates such that

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

  is as small as possible, where $\hat{y}_i$ is the ith predicted value.
- In the last lecture we also mentioned that we want to find $\hat{f}$ such that the reducible error is minimized...
- In linear regression, we "assume" that f is linear, and so the problem reduces to finding the best combination of $\beta_1, \ldots, \beta_p$

Least Squares



- Horizontal axis: predictor
- Vertical axis: response
- Red dots: observations
- Blue line: least squares line

Blue line is positioned to minimize the sum of squared lengths of the gray lines.

Least Squares Regression

- When we fit a model, we use a training set of observations.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}.$$

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.
- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}))^2.$$

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.
- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}))^2.$$

- The training error is closely related to the R^2 for a linear model – that is, the proportion of variance explained.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.
- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}))^2.$$

- The training error is closely related to the R^2 for a linear model – that is, the proportion of variance explained.
- Big $R^2 \Leftrightarrow$ Small Training Error.

Accessing the Accuracy of the Estimated Coefficients

- Recall that we assume that $y = f(X_1, \ldots, X_p) + \varepsilon$ for some unknown function f

Accessing the Accuracy of the Estimated Coefficients

- Recall that we assume that $y = f(X_1, \ldots, X_p) + \varepsilon$ for some unknown function f
- As we discussed before, don't know the form of $f(X_1, \ldots, X_p)$, and we can only make an assumption about it

Accessing the Accuracy of the Estimated Coefficients

- Recall that we assume that $y = f(X_1, \ldots, X_p) + \varepsilon$ for some unknown function f
- As we discussed before, don't know the form of $f(X_1, \ldots, X_p)$, and we can only make an assumption about it
- When using linear regression, we assume that $f(X_1, \ldots, X_p) = X_1\beta_1 + \ldots + X_p\beta_p$

Accessing the Accuracy of the Estimated Coefficients

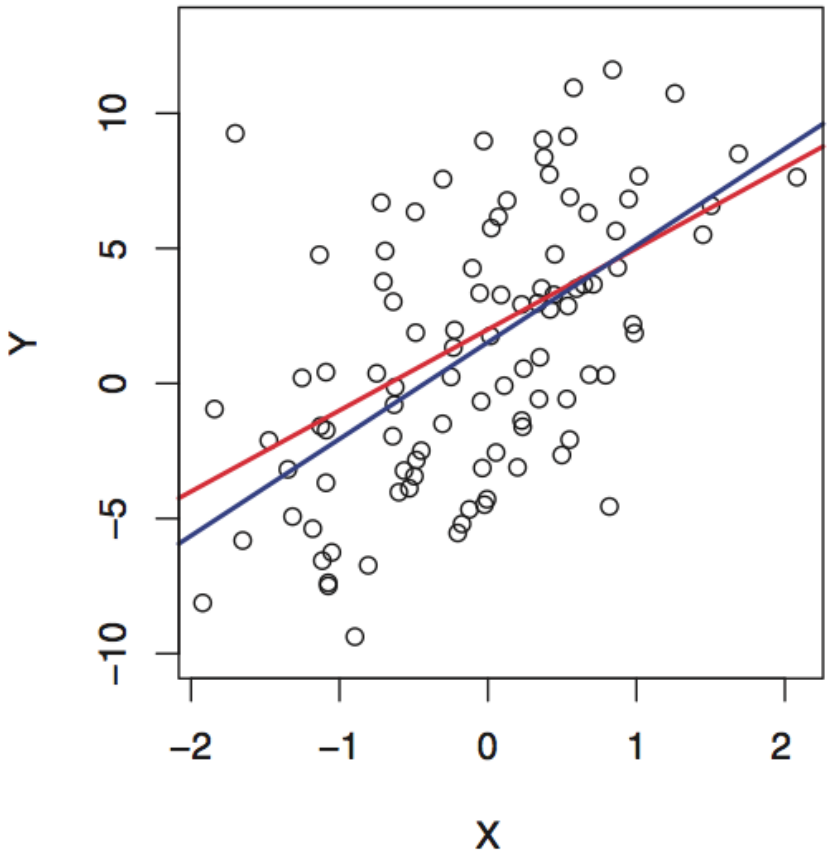
- Recall that we assume that $y = f(X_1, \ldots, X_p) + \varepsilon$ for some unknown function f
- As we discussed before, don't know the form of $f(X_1, \ldots, X_p)$, and we can only make an assumption about it
- When using linear regression, we assume that $f(X_1, \ldots, X_p) = X_1\beta_1 + \ldots + X_p\beta_p$
- If this assumption is true, then the least squares estimate is *unbiased*. However, even if not, then it is still useful...

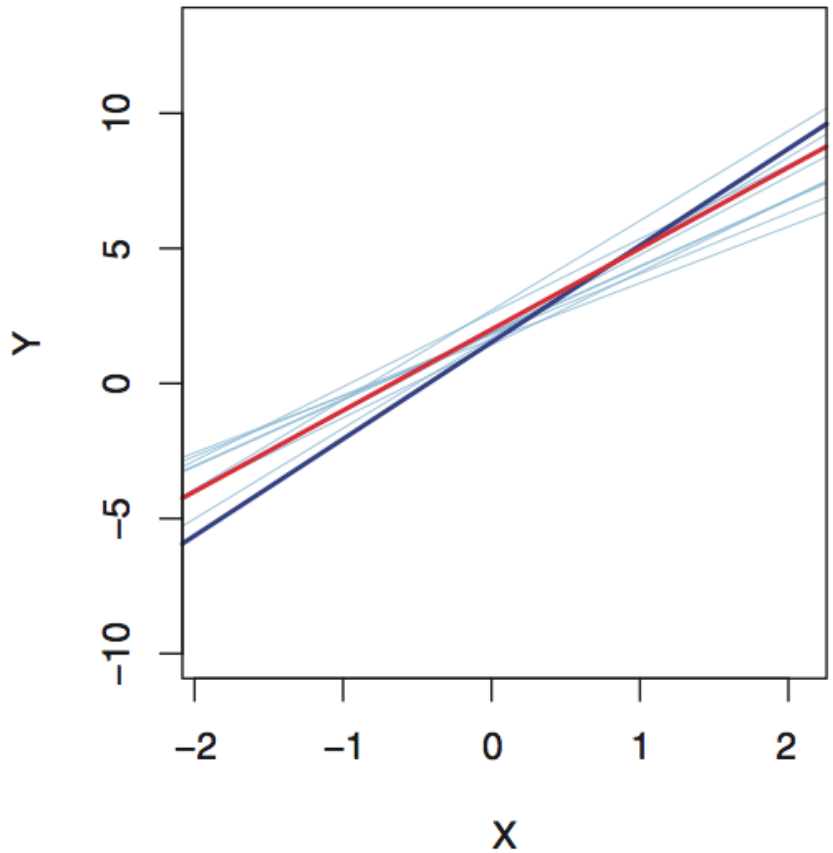
Accessing the Accuracy of the Estimated Coefficients

Here's an example for simple linear regression

Accessing the Accuracy of the Estimated Coefficients

Here's an example for simple linear regression





Inference on Model Parameters

- Since we are not really interested in the coefficient estimates based on our single sample, the question is: “what can we say about the population-level parameters based on our estimated coefficients”?

Inference on Model Parameters

- Since we are not really interested in the coefficient estimates based on our single sample, the question is: “what can we say about the population-level parameters based on our estimated coefficients”?
  - is there really an association between shoe size and run time?
  - do height, weight and shoe size provide any information about run time?

Inference on Model Parameters

- Since we are not really interested in the coefficient estimates based on our single sample, the question is: “what can we say about the population-level parameters based on our estimated coefficients”?
  - is there really an association between shoe size and run time?
  - do height, weight and shoe size provide any information about run time?
- For linear regression, when $p \ll n$, we can use a t-statistic of the form

$$t = \frac{\hat{\beta}}{SE(\hat{\beta})}$$

to test whether

$$H_0 : \beta = 0 \qquad \text{vs} \qquad H_a : \beta \neq 0$$

Inference on Model Parameters

- Since we are not really interested in the coefficient estimates based on our single sample, the question is: “what can we say about the population-level parameters based on our estimated coefficients”?
  - is there really an association between shoe size and run time?
  - do height, weight and shoe size provide any information about run time?
- For linear regression, when $p \ll n$, we can use a t-statistic of the form

$$t = \frac{\hat{\beta}}{SE(\hat{\beta})}$$

to test whether

$$H_0 : \beta = 0 \qquad \text{vs} \qquad H_a : \beta \neq 0$$

- More generally, we can use an F-test for the whole model, or subset of parameters

Inference on Model Parameters

- Since we are not really interested in the coefficient estimates based on our single sample, the question is: "what can we say about the population-level parameters based on our estimated coefficients"?
  - is there really an association between shoe size and run time?
  - do height, weight and shoe size provide any information about run time?
- For linear regression, when $p \ll n$, we can use a t-statistic of the form

$$t = \frac{\hat{\beta}}{SE(\hat{\beta})}$$

to test whether

$$H_0 : \beta = 0 \qquad \text{vs} \qquad H_a : \beta \neq 0$$

- More generally, we can use an F-test for the whole model, or subset of parameters
- These are all standard and covered in intro stat classes if $p \ll n$ ...

Inference on Model Parameters

- Since we are not really interested in the coefficient estimates based on our single sample, the question is: "what can we say about the population-level parameters based on our estimated coefficients"?
  - is there really an association between shoe size and run time?
  - do height, weight and shoe size provide any information about run time?
- For linear regression, when $p \ll n$, we can use a t-statistic of the form

$$t = \frac{\hat{\beta}}{SE(\hat{\beta})}$$

to test whether

$$H_0 : \beta = 0 \qquad \text{vs} \qquad H_a : \beta \neq 0$$

- More generally, we can use an F-test for the whole model, or subset of parameters
- These are all standard and covered in intro stat classes if $p \ll n$ ... but *very difficult in high-dimensional settings*!!

Least Squares as More Variables are Included

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.

Least Squares as More Variables are Included

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.

Least Squares as More Variables are Included

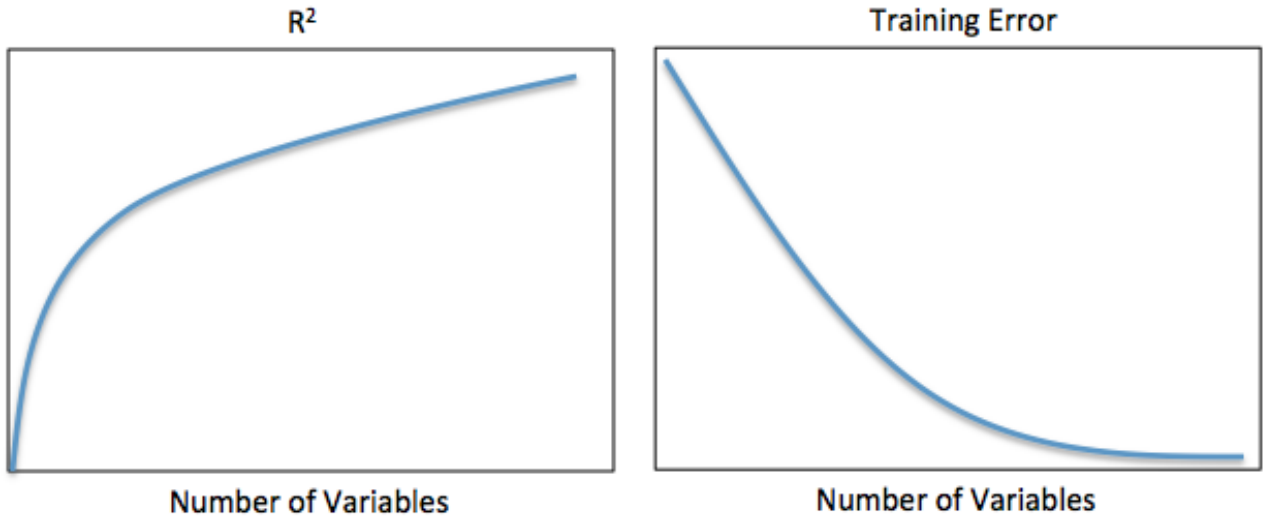
- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features to use in developing a model, then I could surely come up with a regression model that fits the training data perfectly! Unfortunately, this model wouldn't capture the true signal in the data.

Least Squares as More Variables are Included

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features to use in developing a model, then I could surely come up with a regression model that fits the training data perfectly! Unfortunately, this model wouldn't capture the true signal in the data.
- We really care about the model's performance on test observations – observations not used to fit the model.

The Problem

As we add more variables into the model...



... the training error decreases and the R^2 increases!

Why is this a Problem?

- We really care about the model's performance on observations not used to fit the model!
  - We want a model that will predict the survival time of a new patient who walks into the clinic!
  - We want a model that can be used to diagnose cancer for a patient not used in model training!
  - We want to predict risk of diabetes for a patient who wasn't used to fit the model!

Why is this a Problem?

- We really care about the model's performance on observations not used to fit the model!
  - We want a model that will predict the survival time of a new patient who walks into the clinic!
  - We want a model that can be used to diagnose cancer for a patient not used in model training!
  - We want to predict risk of diabetes for a patient who wasn't used to fit the model!
- What we really care about:

$$(y_{test} - \hat{y}_{test})^2,$$

where

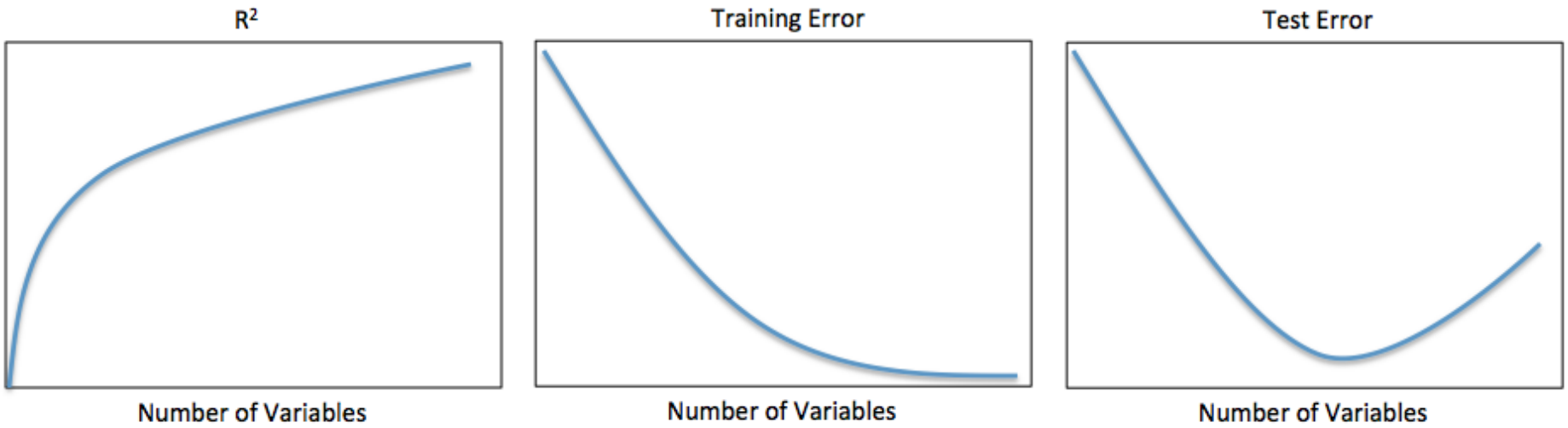
$$\hat{y}_{test} = \hat{\beta}_1 X_{test,1} + \ldots + \hat{\beta}_p X_{test,p},$$

and (X_{test}, y_{test}) was not used to train the model.

- The test error is the average of $(y_{test} - \hat{y}_{test})^2$ over a bunch of test observations.

Training Error versus Test Error

As we add more variables into the model...

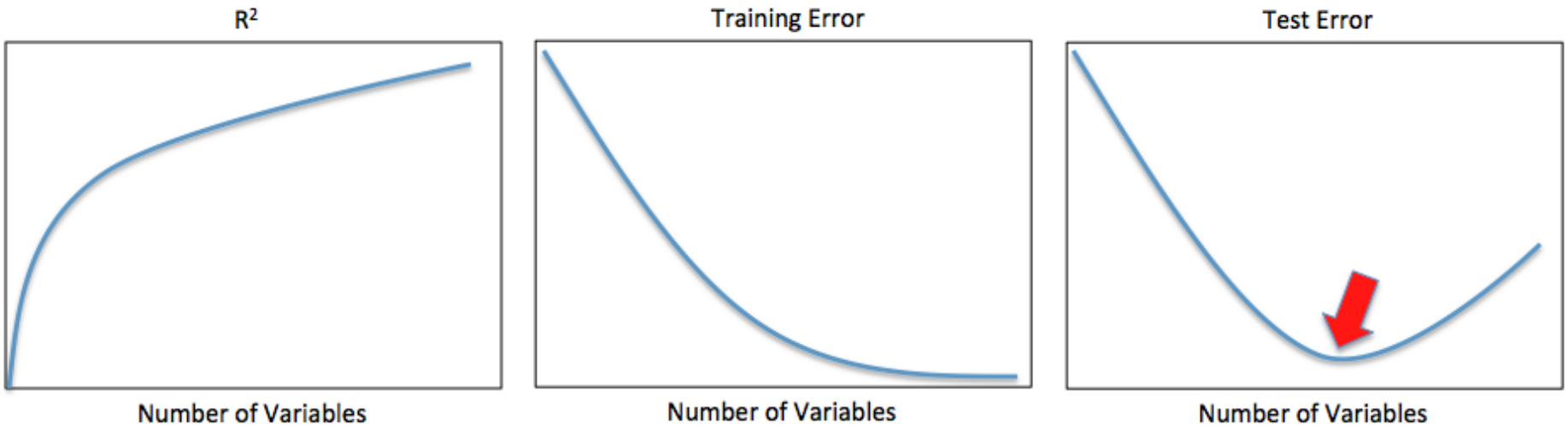


... the training error decreases and the R^2 increases!

But the test error might not!

Training Error versus Test Error

As we add more variables into the model...

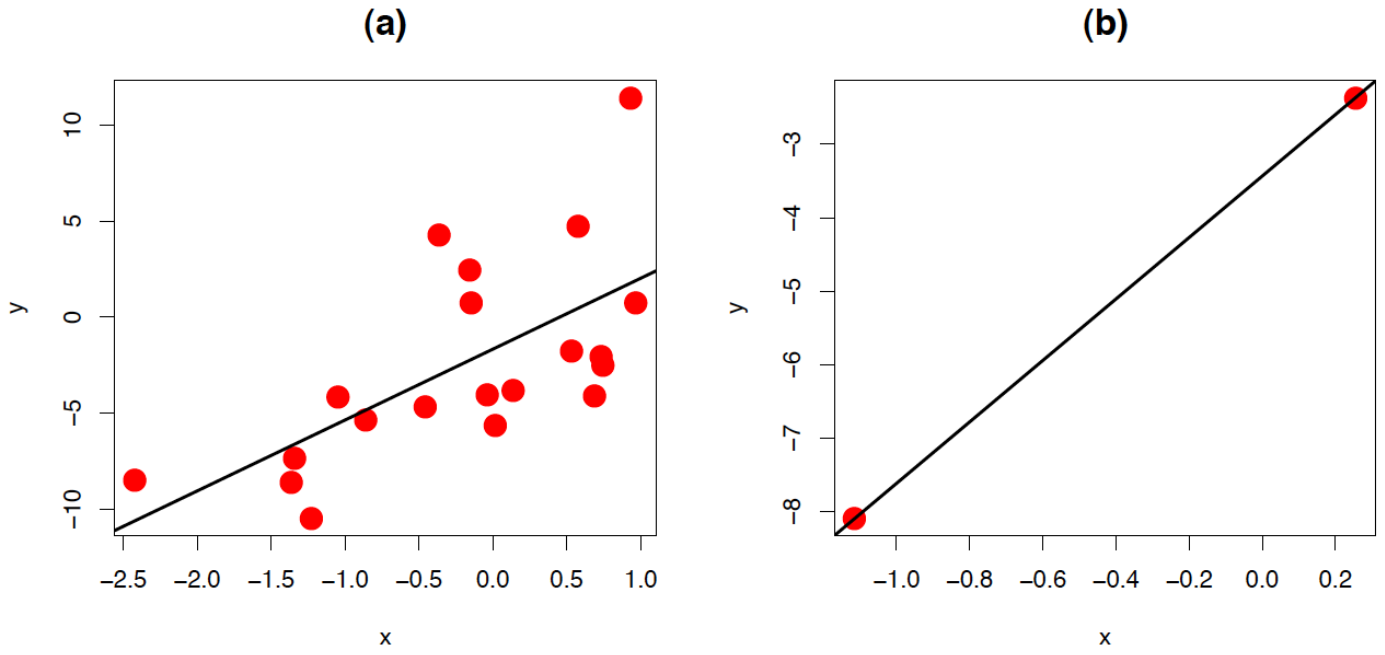


... the training error decreases and the R^2 increases!

But the test error might not!

Why the Number of Variables Matters

- Linear regression will have a very low training error if p is large relative to n.
- A simple example:



- When $n \leq p$, you can always get a perfect model fit to the training data!
- But the test error will be awful.

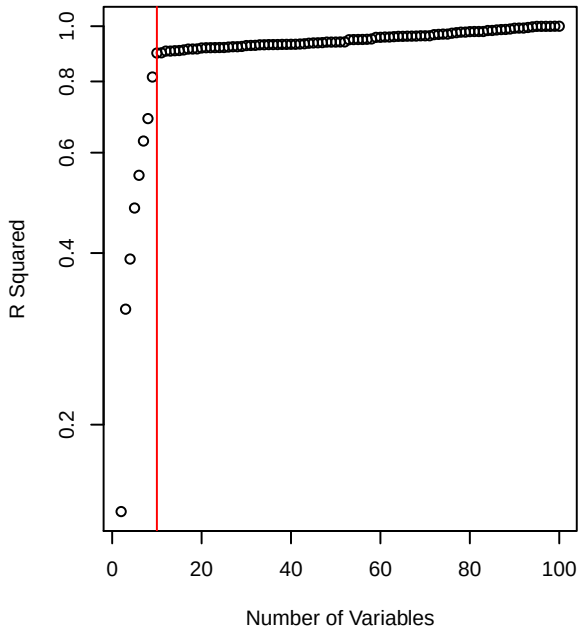
Model Complexity, Training Error, and Test Error

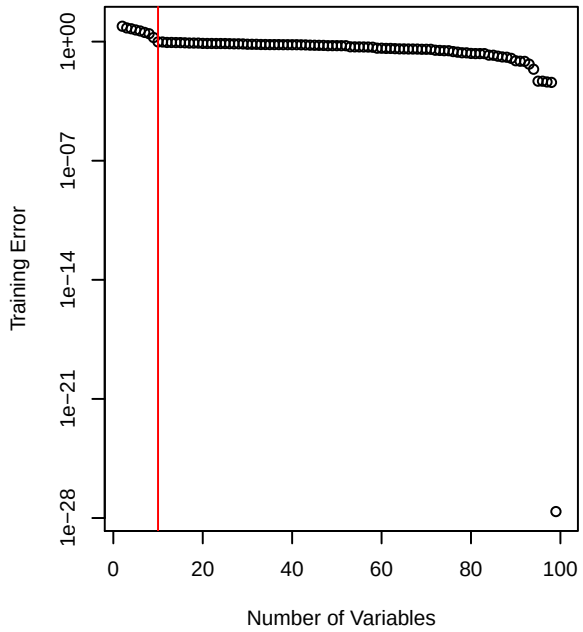
- In this course, we will consider various types of models.
- We will be very concerned with model complexity: e.g. the number of variables used to fit a model.
- As we fit more complex models – e.g. models with more variables – the training error will always decrease.
- But the test error might not.
- As we will see, the number of variables in the model is not the only – or even the best – way to quantify model complexity.

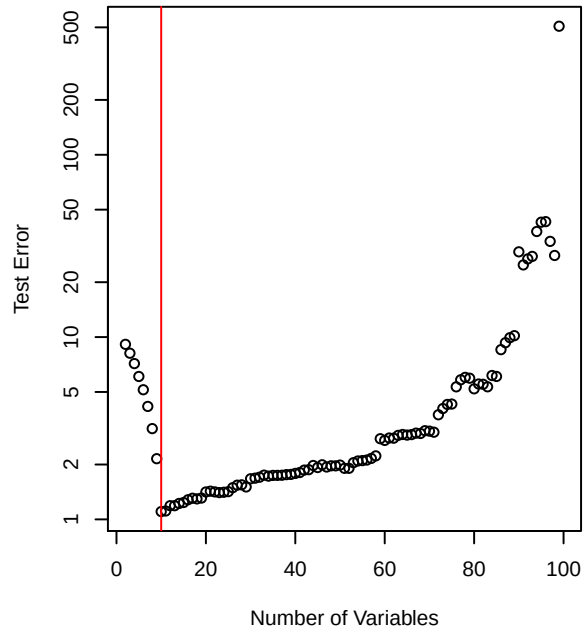
An Example In R

```
xtr <- matrix(rnorm(100*100),ncol=100)
xte <- matrix(rnorm(100000*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
yte <- xte%*%beta + rnorm(100000)
rsq <- trainerr <- testerr <-  NULL
for(i in 2:100){
mod <- lm(ytr~xtr[,1:i])
rsq <- c(rsq,summary(mod)$r.squared)
beta <- mod$coef[-1]
intercept <- mod$coef[1]
trainerr <- c(trainerr, mean((xtr[,1:i]%*%beta+intercept - ytr)^2))
testerr <- c(testerr, mean((xte[,1:i]%*%beta+intercept - yte)^2))
}
par(mfrow=c(1,3))
plot(2:100,rsq,xlab='No of Variables',ylab="R Squared",log="y")
abline(v=10,col="red")
plot(2:100,trainerr,xlab='No of Variables',ylab="Training Error",log="y")
abline(v=10,col="red")
plot(2:100,testerr,xlab='No of Variables',ylab="Test Error",log="y")
abline(v=10,col="red")
```

Output of R Code







- 1st 10 variables are related to response; remaining 90 are not.
- R^2 increases and training error decreases as more variables are added to the model.
- Test error is lowest when only signal variables in model.

Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ – the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times – will decrease.

Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ – the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times – will decrease.
- But as complexity increases, the variance of $\hat{\beta}$ – the amount by which the $\hat{\beta}$'s will differ across experiments – will increase.

Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ – the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times – will decrease.
- But as complexity increases, the variance of $\hat{\beta}$ – the amount by which the $\hat{\beta}$'s will differ across experiments – will increase.
- The test error depends on both the bias and variance:

$$\text{Test Error} = \text{Bias}^2 + \text{Variance}.$$

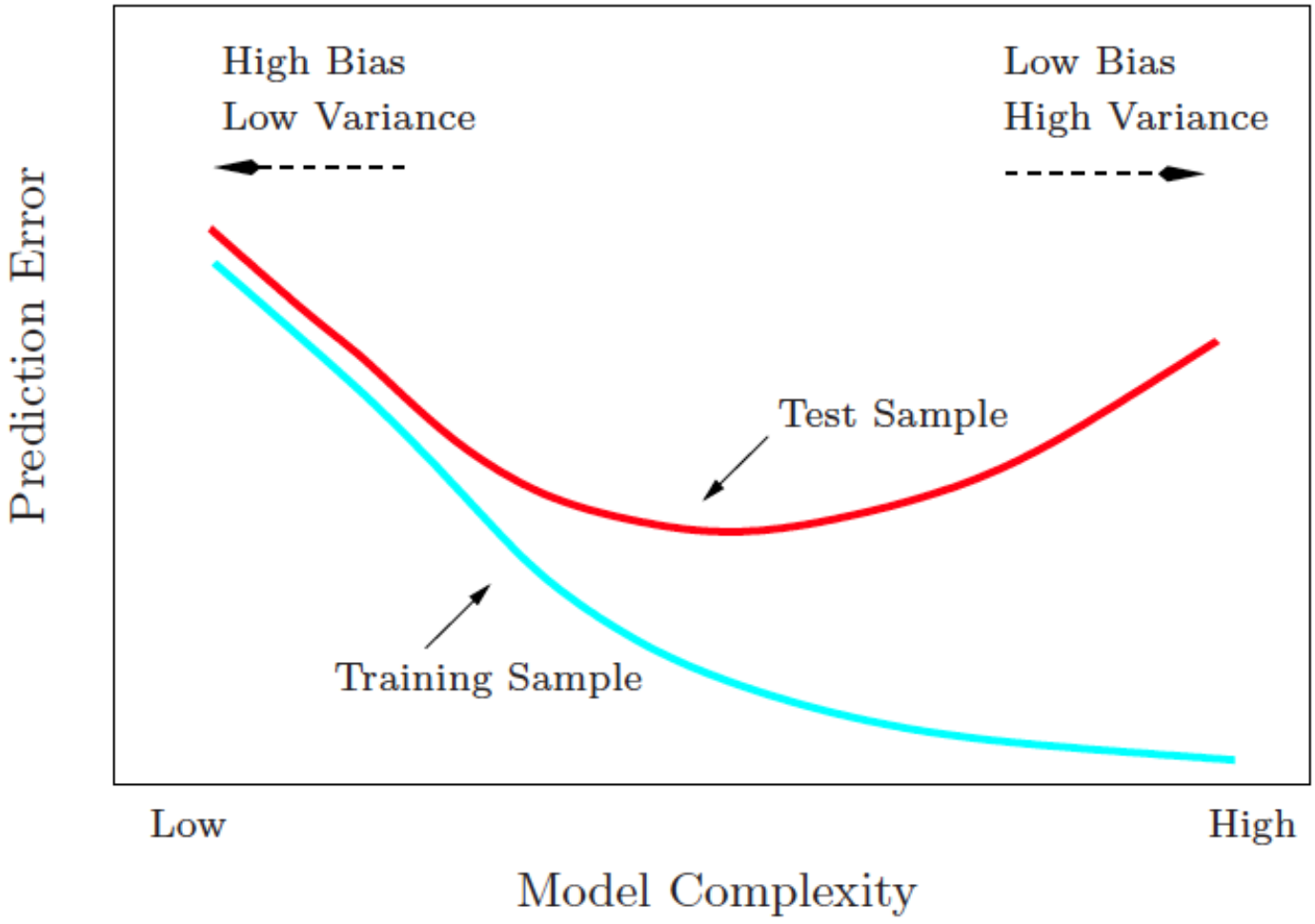
Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ – the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times – will decrease.
- But as complexity increases, the variance of $\hat{\beta}$ – the amount by which the $\hat{\beta}$'s will differ across experiments – will increase.
- The test error depends on both the bias and variance:

$$\text{Test Error} = \text{Bias}^2 + \text{Variance}.$$

- There is a bias-variance trade-off. We want a model that is sufficiently complex as to have not too much bias, but not so complex that it has too much variance.

A Really Fundamental Picture



Model complexity in low dimensions...

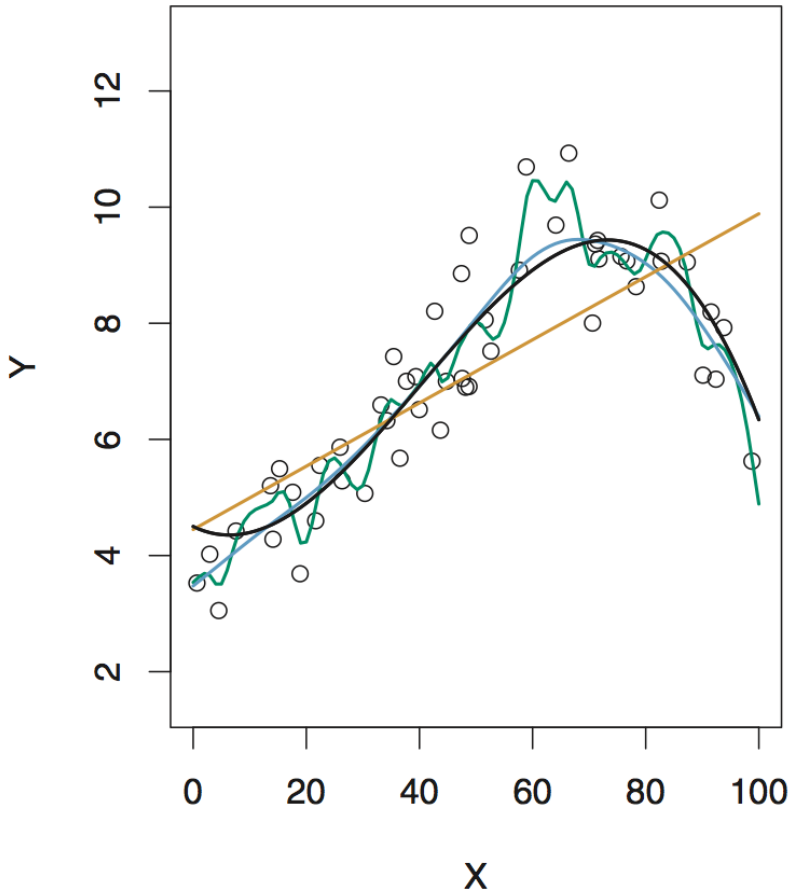
- The concepts of model complexity and bias-variance trade-off are not only for high dimensional settings

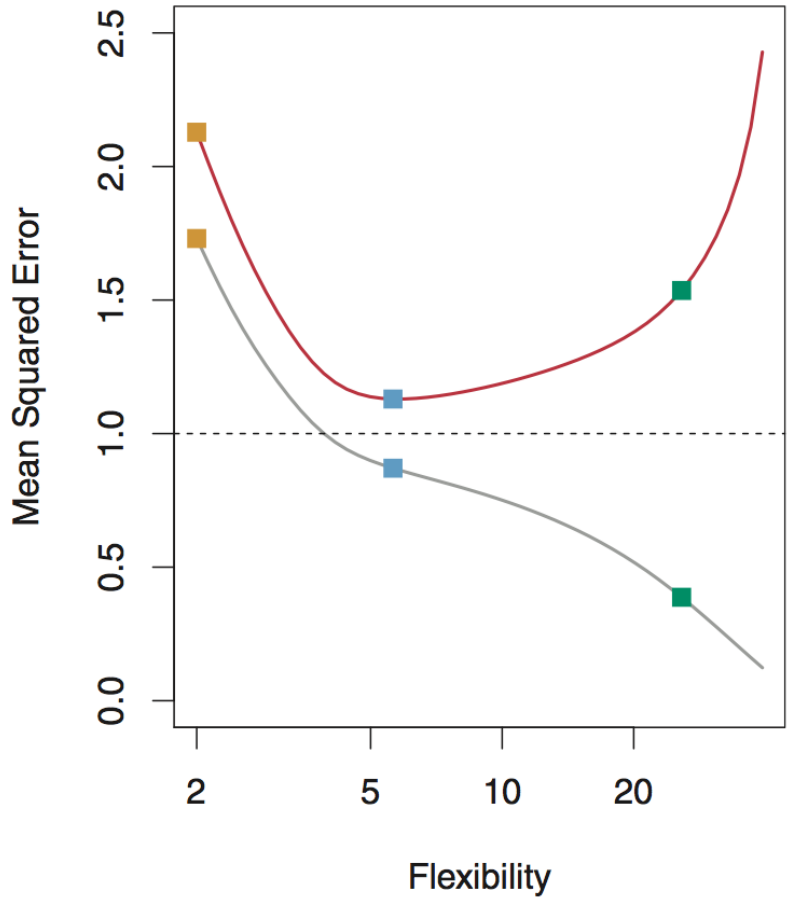
Model complexity in low dimensions...

- The concepts of model complexity and bias-variance trade-off are not only for high dimensional settings
- Even in one dimension, we can make the model highly complex to over fit the training data

Model complexity in low dimensions...

- The concepts of model complexity and bias-variance trade-off are not only for high dimensional settings
- Even in one dimension, we can make the model highly complex to over fit the training data





Bias-Variance Tradeoff

Bias-Variance Tradeoff

- For general (non-linear) models we can define the MSE in a similar way

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(X_{i1}, \ldots, X_{ip}))^2$$

Bias-Variance Tradeoff

- For general (non-linear) models we can define the MSE in a similar way

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(X_{i1}, \ldots, X_{ip}))^2$$

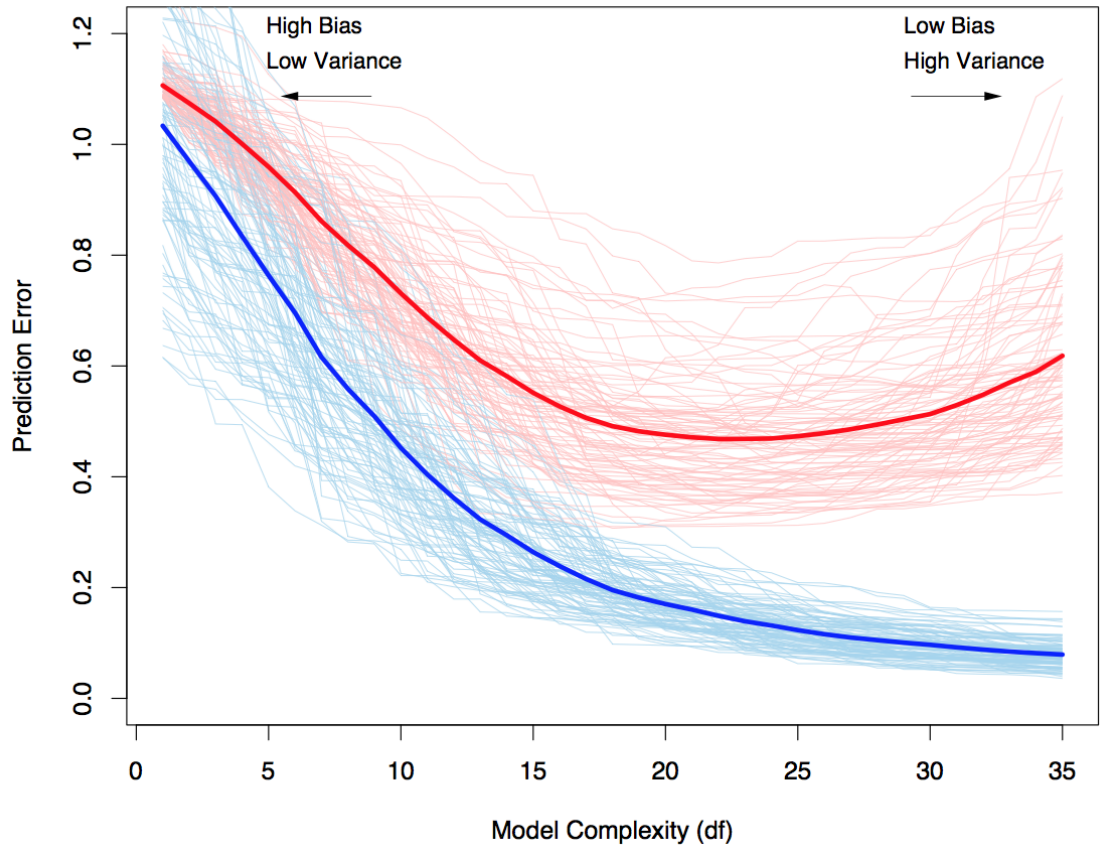
- And we can check how the complexity affects the tradeoff

Bias-Variance Tradeoff

- For general (non-linear) models we can define the MSE in a similar way

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(X_{i1}, \ldots, X_{ip}))^2$$

- And we can check how the complexity affects the tradeoff

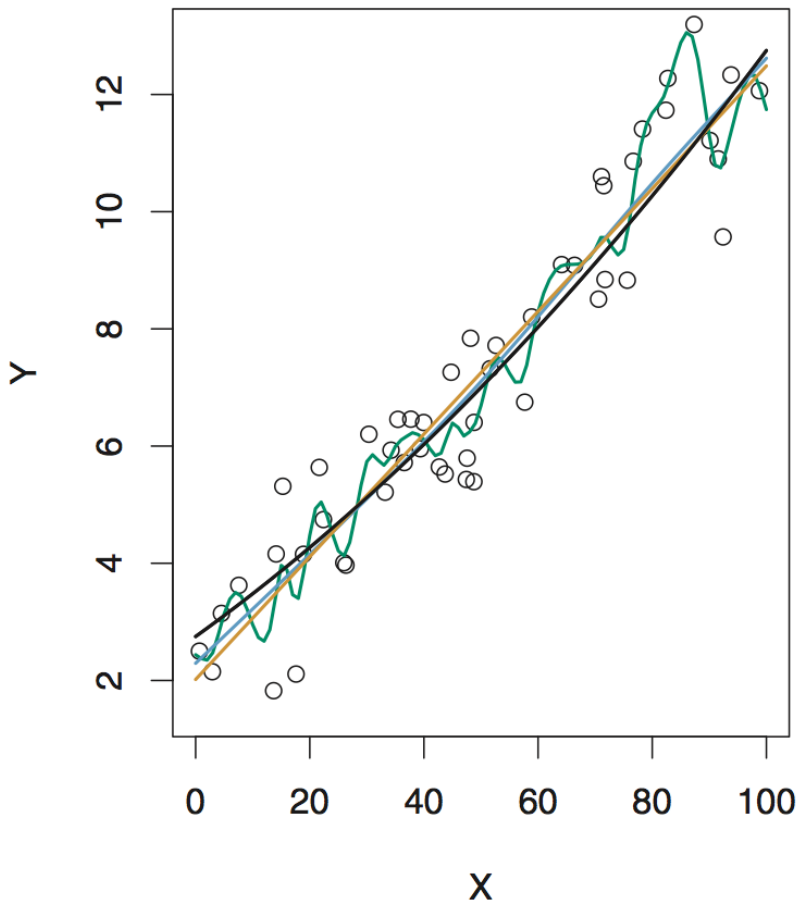


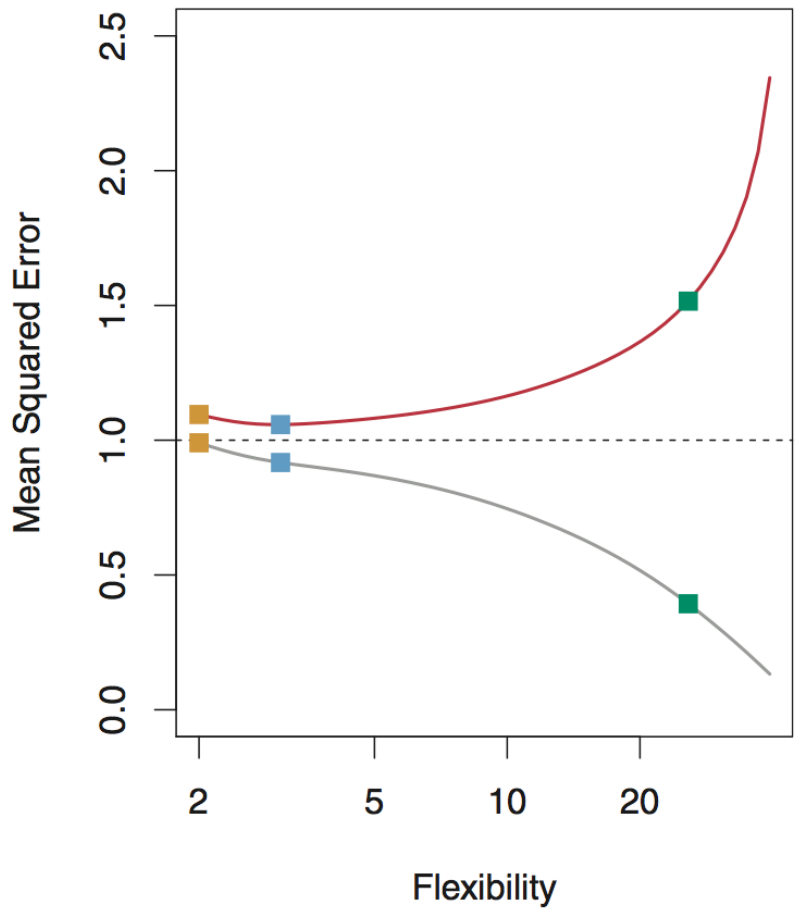
Bias-Variance Tradeoff

Of course, everything depends on the true relationships between y and X

Bias-Variance Tradeoff

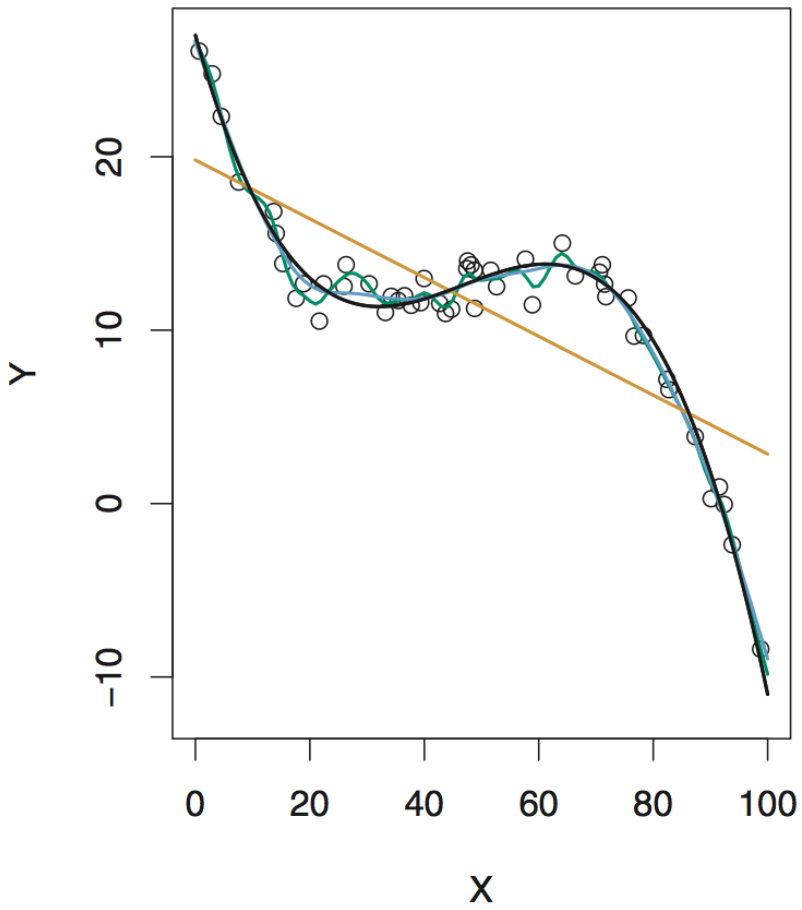
Of course, everything depends on the true relationships between y and X

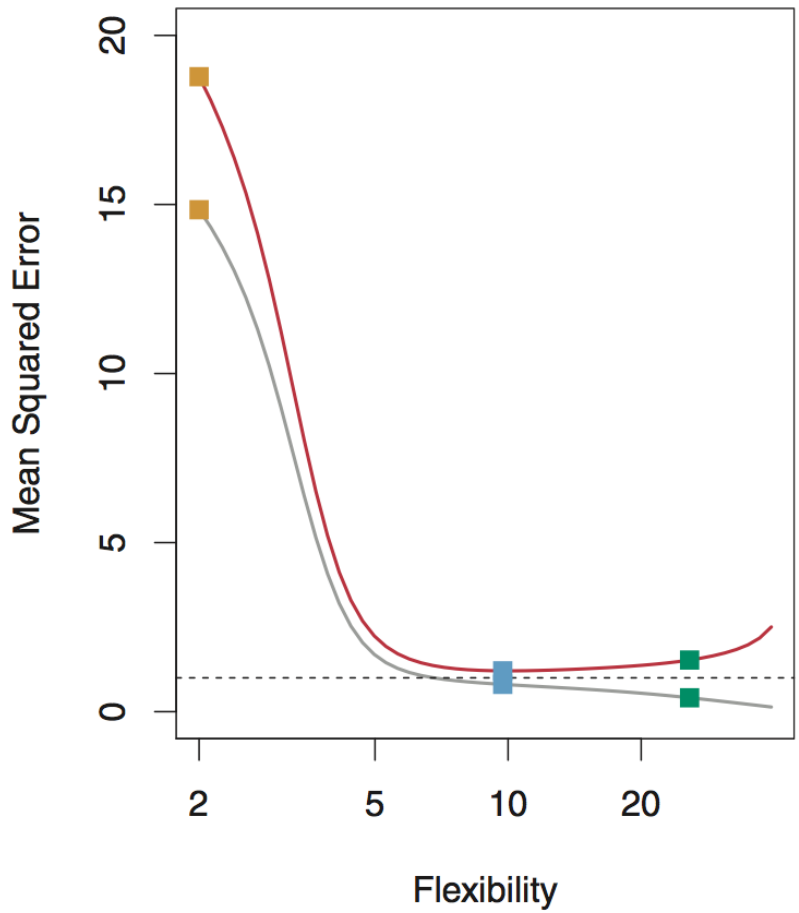




Bias-Variance Tradeoff

Of course, everything depends on the true relationships between y and X





Overfitting

Overfitting

- Fitting an overly complex model – a model that has too much variance – is known as overfitting.

Overfitting

- Fitting an overly complex model – a model that has too much variance – is known as overfitting.
- In biomedical big data setting, when $p \gg n$, we must work hard not to overfit the data.

Overfitting

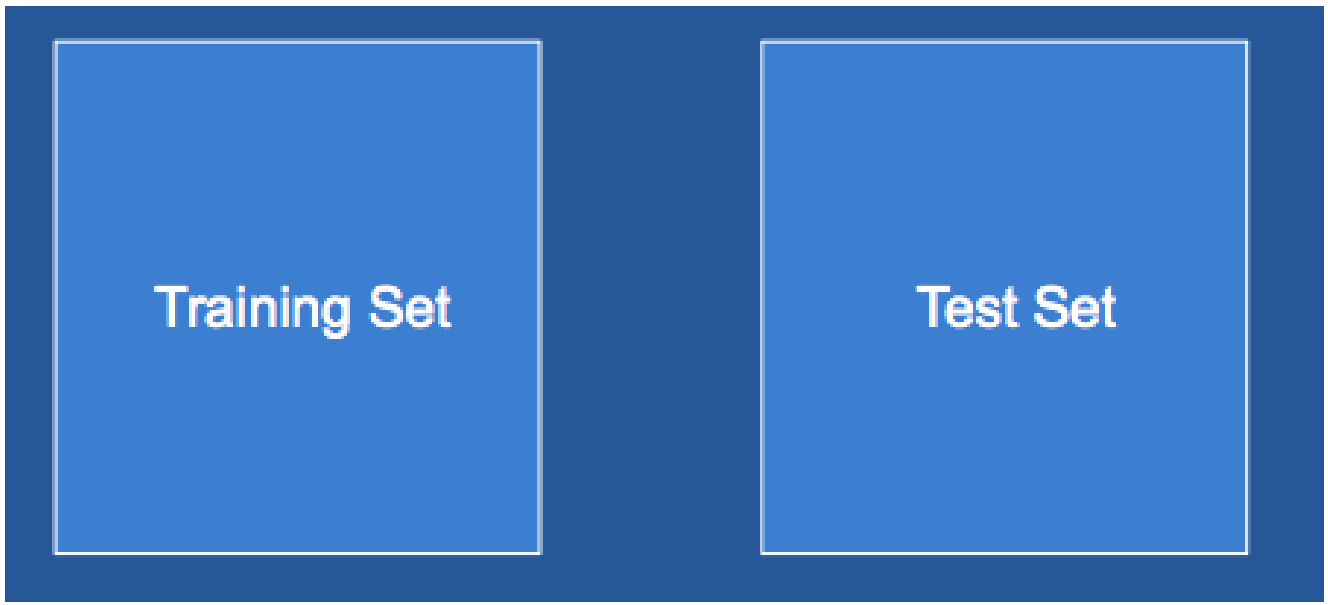
- Fitting an overly complex model – a model that has too much variance – is known as overfitting.
- In biomedical big data setting, when $p \gg n$, we must work hard not to overfit the data.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.

Overfitting

- Fitting an overly complex model – a model that has too much variance – is known as overfitting.
- In biomedical big data setting, when $p \gg n$, we must work hard not to overfit the data.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.
- How can we estimate the test error?

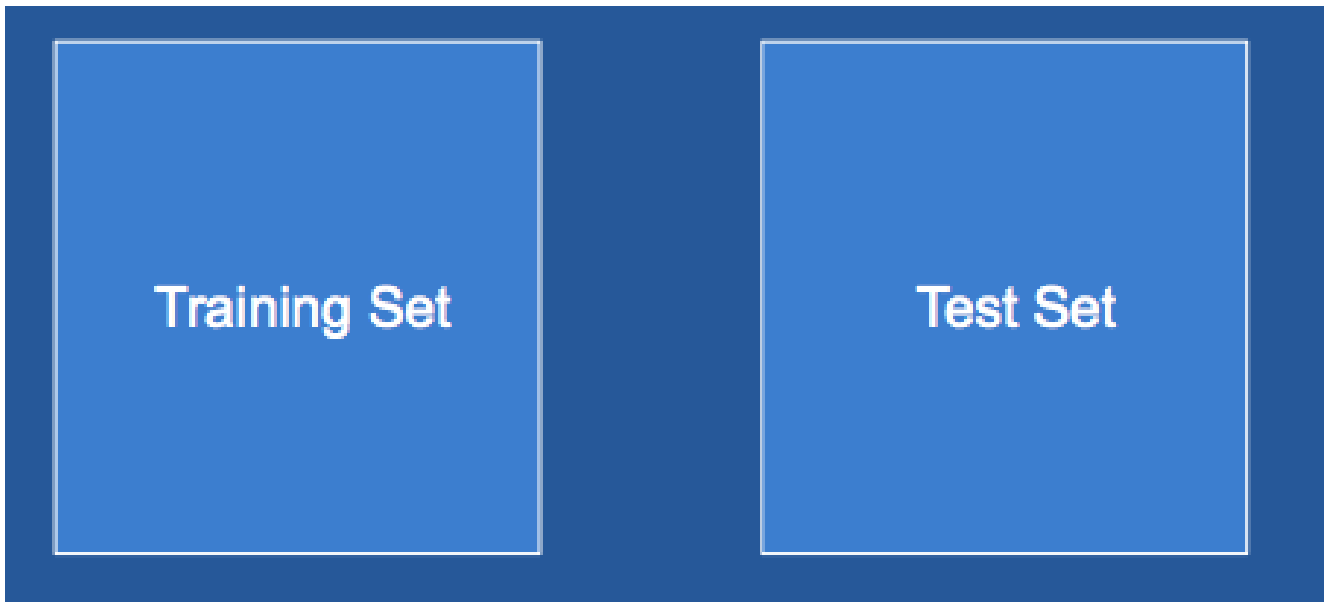
How to Estimate the Test Error?

- Split samples into *training* and *test* sets.
- Fit the model on the training set, and evaluate on test set.



How to Estimate the Test Error?

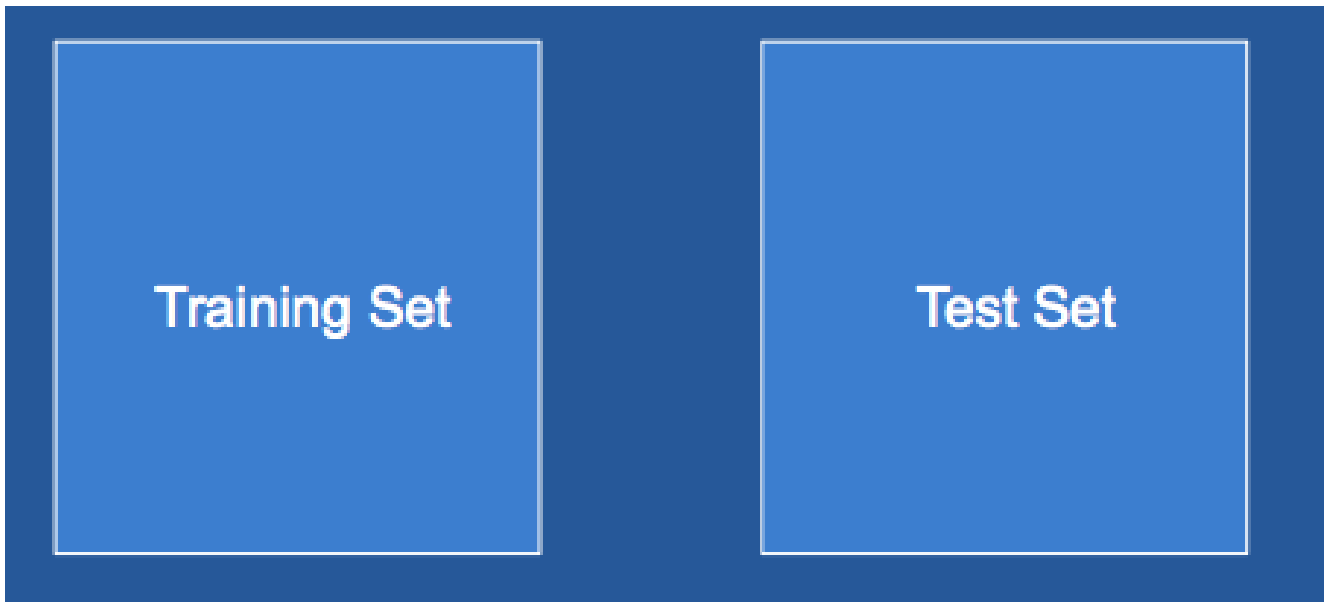
- Split samples into *training* and *test* sets.
- Fit the model on the training set, and evaluate on test set.



Q: Can there ever, under any circumstance, be sample overlap between the training and test sets?

How to Estimate the Test Error?

- Split samples into *training* and *test* sets.
- Fit the model on the training set, and evaluate on test set.



Q: Can there ever, under any circumstance, be sample overlap between the training and test sets?
A: Absolutely Not!

How to Estimate the Test Error?

- We fit a model on the training set, but we must evaluate its performance on a test set.
- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.

How to Estimate the Test Error?

- We fit a model on the training set, but we must evaluate its performance on a test set.
- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.

You can't peek at the test set until you are completely done all aspects of model-fitting on the training set!

How to Estimate the Test Error?

To get an estimate of the test error of a particular model on a future observation:

1. Split the samples into a training set and a test set.
2. Fit the model on the training set.
3. Evaluate its performance on the test set.
4. The test set error rate is an estimate of the model's performance on a future observation.

How to Estimate the Test Error?

To get an estimate of the test error of a particular model on a future observation:

1. Split the samples into a training set and a test set.
2. Fit the model on the training set.
3. Evaluate its performance on the test set.
4. The test set error rate is an estimate of the model's performance on a future observation.

But remember: no peeking!

Choosing Between Several Models

- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.

Choosing Between Several Models

- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!

Choosing Between Several Models

- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!

Choosing Between Several Models

- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?

Choosing Between Several Models

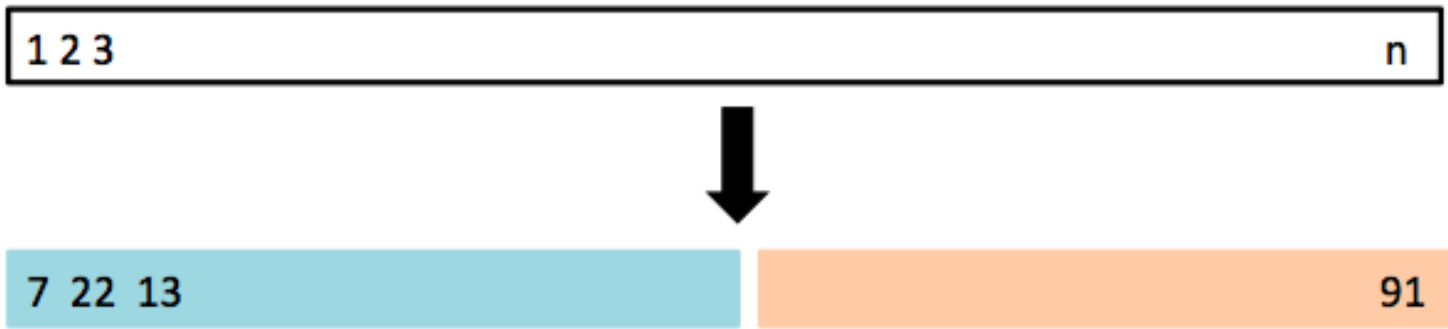
- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?
  1. The validation set approach.
  2. Leave-one-out cross-validation.
  3. K-fold cross-validation.

Choosing Between Several Models

- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?
  1. The validation set approach.
  2. Leave-one-out cross-validation.
  3. K-fold cross-validation.
- In what follows, assume that we have split the data into a training set and a test set, and the training set contains n observations.

Validation Set Approach

Split the n observations into two sets of approximately equal size. Train on one set, and evaluate performance on the other.



Validation Set Approach

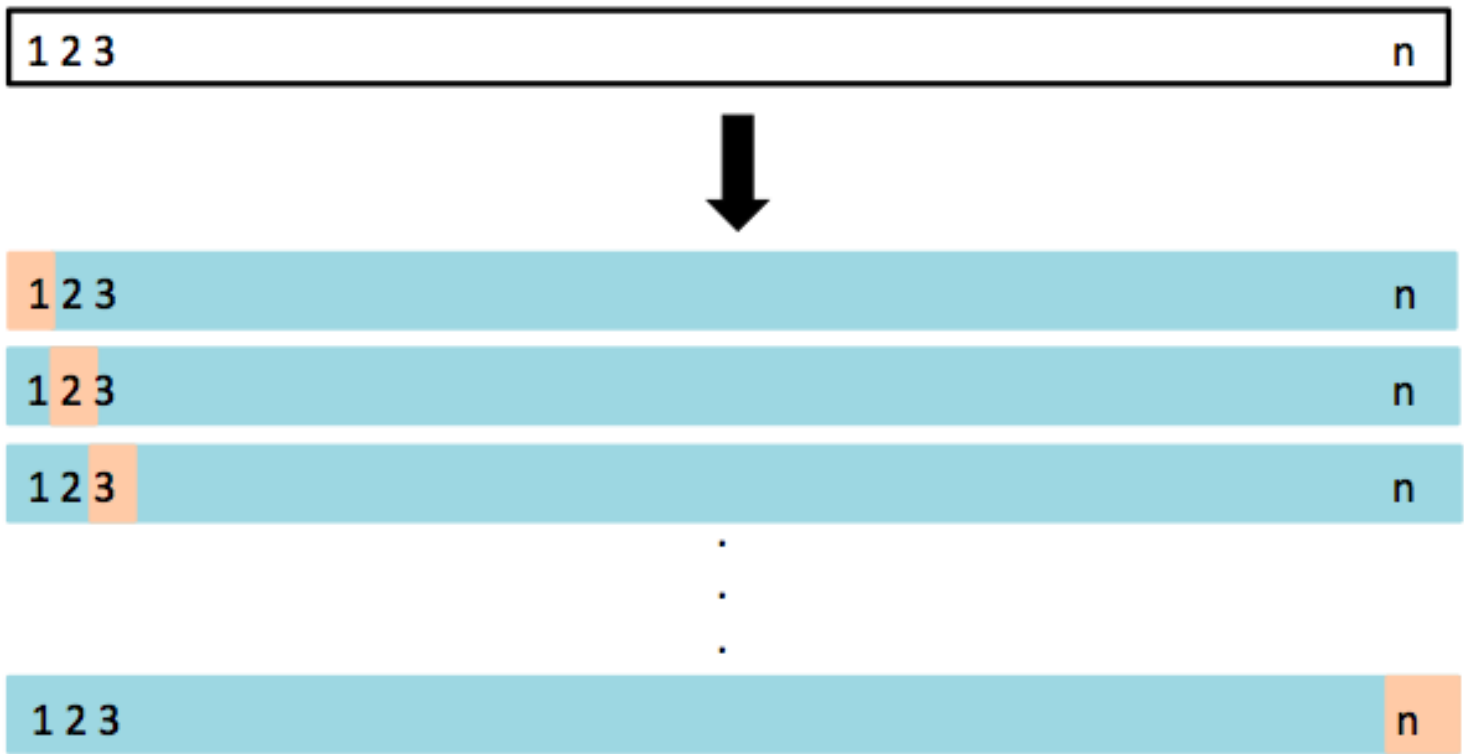
For a given model, we perform the following procedure:

1. Split the observations into two sets of approximately equal size, a training set and a validation set.
   a. Fit the model using the training observations. Let $\hat{\beta}_{(train)}$ denote the regression coefficient estimates.
   b. For each observation in the validation set, compute the error, $e_i = (y_i - \mathbf{x}_i^T \hat{\beta}_{(train)})^2$.
2. Calculate the total validation set error by summing the e_i's over all of the validation set observations.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

Leave-One-Out Cross-Validation

Fit n models, each on $n-1$ of the observations. Evaluate each model on the left-out observation.



Leave-One-Out Cross-Validation

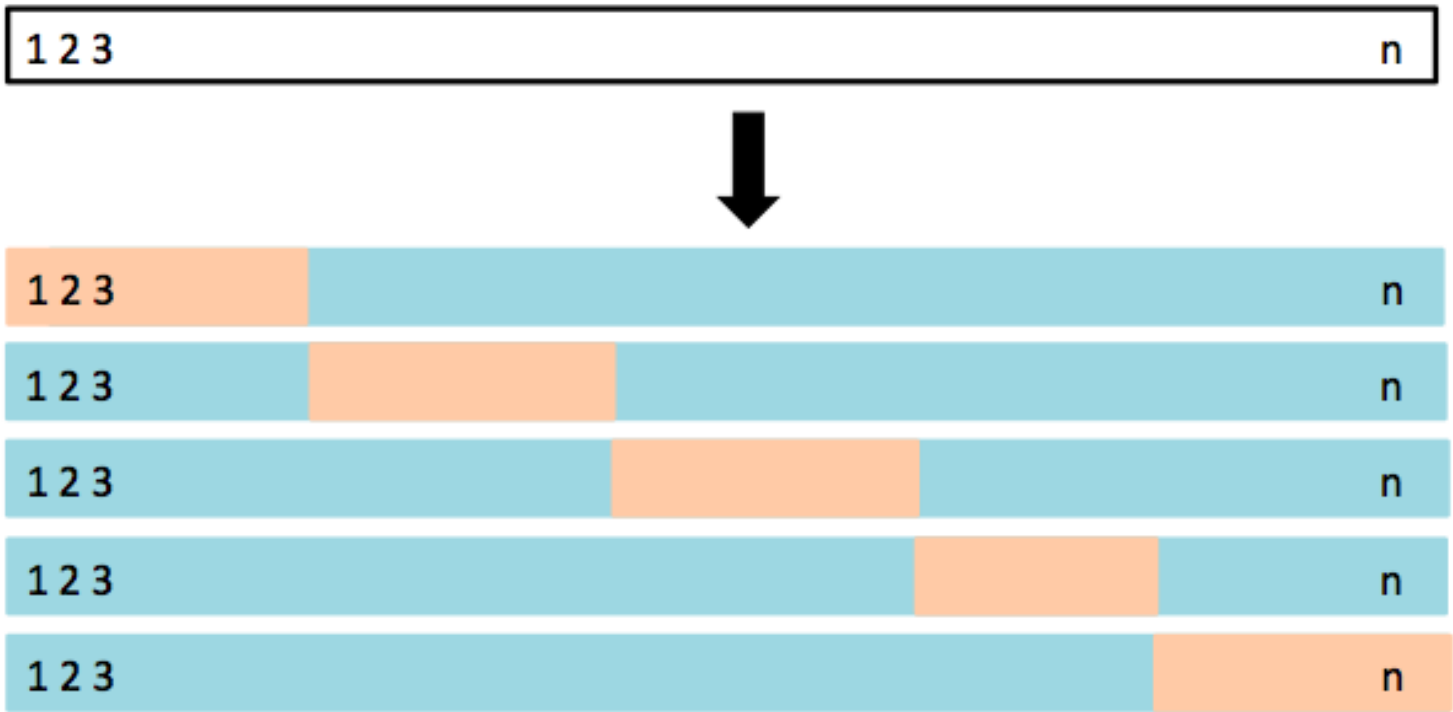
For a given model, we perform the following procedure:

1. For $i = 1, \ldots, n$:
   a. Fit the model using observations $1, \ldots, i-1, i+1, \ldots, n$. Let $\hat{\beta}_{(i)}$ denote the regression coefficient estimates.
   b. Compute the test error, $e_i = (y_i - \mathbf{x}_i^T \hat{\beta}_{(i)})^2$.
2. Calculate $\sum_{i=1}^{n} e_i$, the total CV error.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

5-Fold Cross-Validation

Split the observations into 5 sets. Repeatedly train the model on 4 sets and evaluate its performance on the 5th.



K-fold cross-validation

A generalization of leave-one-out cross-validation. For a given model, we perform the following procedure:

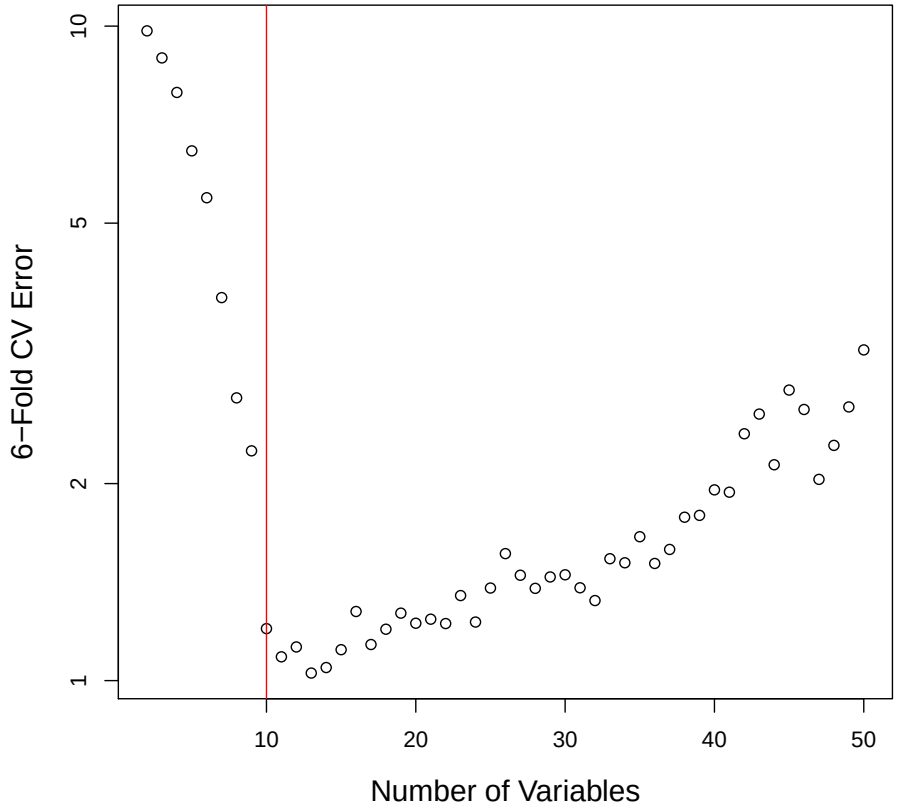
1. Split the n observations into K equally-sized folds.
2. For $k = 1, \ldots, K$:
   a. Fit the model using the observations not in the kth fold.
   b. Let e_k denote the test error for the observations in the kth fold.
3. Calculate $\sum_{k=1}^{K} e_k$, the total CV error.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

An Example In R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
cv.err <- NULL
for(i in 2:50){
dat <- data.frame(x=xtr[,1:i],y=ytr)
mod <- glm(y~.,data=dat)
cv.err <- c(cv.err, cv.glm(dat,mod,K=6)$delta[1])
}
plot(2:50, cv.err, xlab="Number of Variables",
ylab="6-Fold CV Error", log="y")
abline(v=10, col="red")
```

Output of R Code



- Six-fold CV identifies the model with just over ten predictors.
- First ten predictors contain signal, and the rest are noise.

After Estimating the Test Error on the Training Set...

After we estimate the test error using the training set, we refit the “best" model on all of the available training observations. We then evaluate this model on the test set.

Big Picture

Big Picture

Big Picture

Big Picture

Big Picture

Summary: Four-Step Procedure

1. Split observations into training set and test set.
2. Fit a bunch of models on training set, and estimate the test error, using cross-validation or validation set approach.
3. Refit the best model on the full training set.
4. Evaluate the model's performance on the test set.

Cross Validation in Practice

Cross Validation in Practice

- In practice, we don't know the true test MSE, and use CV to estimate it

Cross Validation in Practice

- In practice, we don't know the true test MSE, and use CV to estimate it
- This is either to

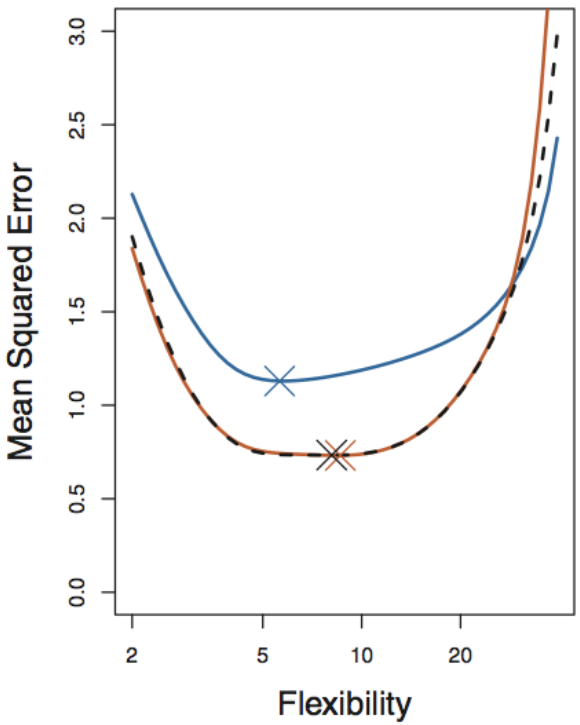
Cross Validation in Practice

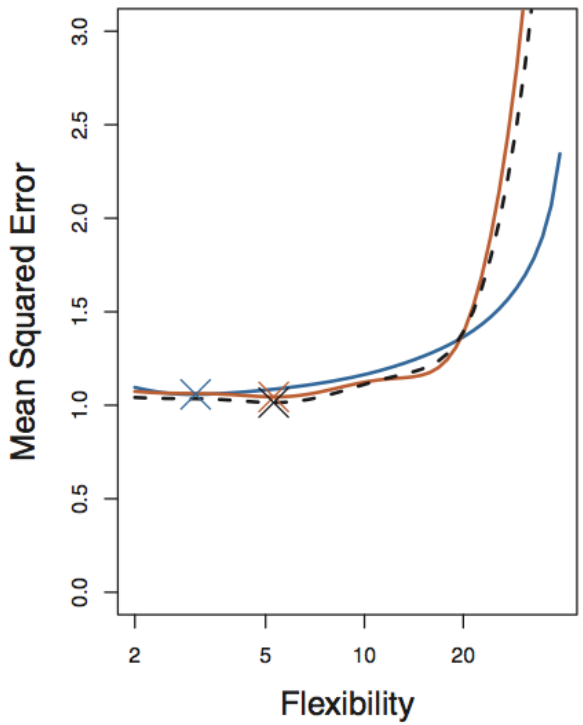
- In practice, we don't know the true test MSE, and use CV to estimate it
- This is either to
  - determine how well the model would do on independent data: in this case, the actual estimate of MSE is of interest

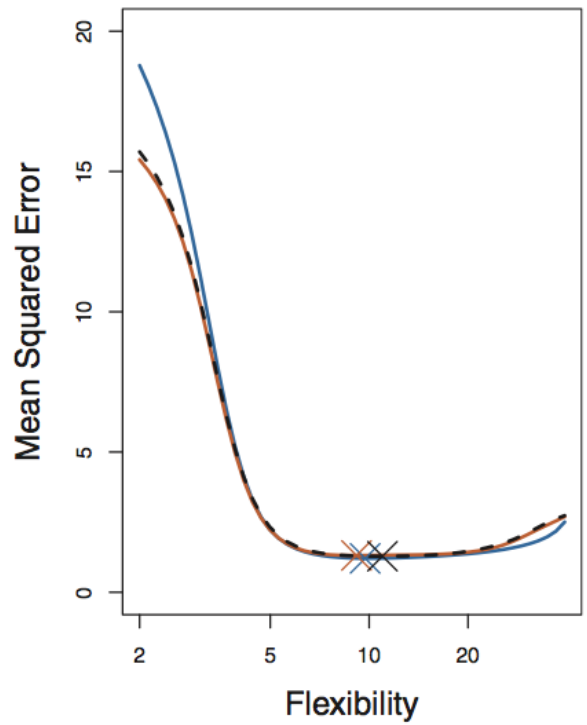
Cross Validation in Practice

- In practice, we don't know the true test MSE, and use CV to estimate it
- This is either to
  - determine how well the model would do on independent data: in this case, the actual estimate of MSE is of interest
  - decide which model is best among a set of candidate models: in this case, the actual estimate of MSE is not very important, as long as the location of the minimum MSE value is correct

Cross Validation in Practice







Cross Validation in Practice

Cross Validation in Practice

- k-fold CV is computationally less expensive than LOOCV (though for linear regression, and some other methods, LOOCV is very simple)

Cross Validation in Practice

- k-fold CV is computationally less expensive than LOOCV (though for linear regression, and some other methods, LOOCV is very simple)
- When the goal is to estimate the MSE:
  - the validation set approach can lead to overestimates of the test error rate, since only $1/2$ of observations are used to fit the model
  - LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations
  - k-fold CV is somewhere in between

Cross Validation in Practice

- k-fold CV is computationally less expensive than LOOCV (though for linear regression, and some other methods, LOOCV is very simple)
- When the goal is to estimate the MSE:
  - the validation set approach can lead to overestimates of the test error rate, since only $1/2$ of observations are used to fit the model
  - LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n-1$ observations
  - k-fold CV is somewhere in between
- However, we want to balance the bias and variance!!

Cross Validation in Practice

- k-fold CV is computationally less expensive than LOOCV (though for linear regression, and some other methods, LOOCV is very simple)
- When the goal is to estimate the MSE:
  - the validation set approach can lead to overestimates of the test error rate, since only $1/2$ of observations are used to fit the model
  - LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations
  - k-fold CV is somewhere in between
- However, we want to balance the bias and variance!!
  - LOOCV has a higher variance than k-fold CV:

Cross Validation in Practice

- k-fold CV is computationally less expensive than LOOCV (though for linear regression, and some other methods, LOOCV is very simple)
- When the goal is to estimate the MSE:
  - the validation set approach can lead to overestimates of the test error rate, since only $1/2$ of observations are used to fit the model
  - LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations
  - k-fold CV is somewhere in between
- However, we want to balance the bias and variance!!
  - LOOCV has a higher variance than k-fold CV: the outputs of models used in LOOCV are highly correlated with each other, but there is less correlation, and hence less variance in k-fold CV

Cross Validation in Practice

- k-fold CV is computationally less expensive than LOOCV (though for linear regression, and some other methods, LOOCV is very simple)
- When the goal is to estimate the MSE:
  - the validation set approach can lead to overestimates of the test error rate, since only $1/2$ of observations are used to fit the model
  - LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations
  - k-fold CV is somewhere in between
- However, we want to balance the bias and variance!!
  - LOOCV has a higher variance than k-fold CV: the outputs of models used in LOOCV are highly correlated with each other, but there is less correlation, and hence less variance in k-fold CV
- In practice k-fold CV with k between 5 to 10 often gives good results

Why All the Bother?

Q: Why do I need to have a separate test set? Why can't I just estimate test error using cross-validation or a validation set approach using all the observations, and then be done with it?

Why All the Bother?

Q: Why do I need to have a separate test set? Why can't I just estimate test error using cross-validation or a validation set approach using all the observations, and then be done with it?

A: In general, we are choosing between a whole bunch of models, and we will use the cross-validation or validation set approach to pick between these models. If we use the resulting estimate as a final estimate of test error, then this could be an extreme underestimate, because one model might give a lower estimated test error than others by chance. To avoid having an extreme underestimate of test error, we need to evaluate the "best" model obtained on an independent test set. *This is particularly important in high dimensions!!*

Regression for Biomedical Big Data

- We usually cannot use least squares to fit a regression model in high dimensional biomedical data, because we will get zero training error but a terrible test error.

Regression for Biomedical Big Data

- We usually cannot use least squares to fit a regression model in high dimensional biomedical data, because we will get zero training error but a terrible test error.
- Instead, we must fit a less complex model, e.g. a model with fewer variables.

Regression for Biomedical Big Data

- We usually cannot use least squares to fit a regression model in high dimensional biomedical data, because we will get zero training error but a terrible test error.
- Instead, we must fit a less complex model, e.g. a model with fewer variables.
- We will consider five ways to fit less complex models:
  1. Variable Pre-Selection
  2. Forward Stepwise Regression
  3. Ridge Regression
  4. Lasso Regression
  5. Principal Components Regression

Regression for Biomedical Big Data

- We usually cannot use least squares to fit a regression model in high dimensional biomedical data, because we will get zero training error but a terrible test error.
- Instead, we must fit a less complex model, e.g. a model with fewer variables.
- We will consider five ways to fit less complex models:
  1. Variable Pre-Selection
  2. Forward Stepwise Regression
  3. Ridge Regression
  4. Lasso Regression
  5. Principal Components Regression
- These are alternatives to fitting a linear model using least squares.

Regression for Biomedical Big Data

- We usually cannot use least squares to fit a regression model in high dimensional biomedical data, because we will get zero training error but a terrible test error.
- Instead, we must fit a less complex model, e.g. a model with fewer variables.
- We will consider five ways to fit less complex models:
  1. Variable Pre-Selection
  2. Forward Stepwise Regression
  3. Ridge Regression
  4. Lasso Regression
  5. Principal Components Regression
- These are alternatives to fitting a linear model using least squares.
- Each of these approaches will allow us to choose the level of complexity – e.g. the number of variables in the model.

The Fundamental Truth About High-Dimensional Data

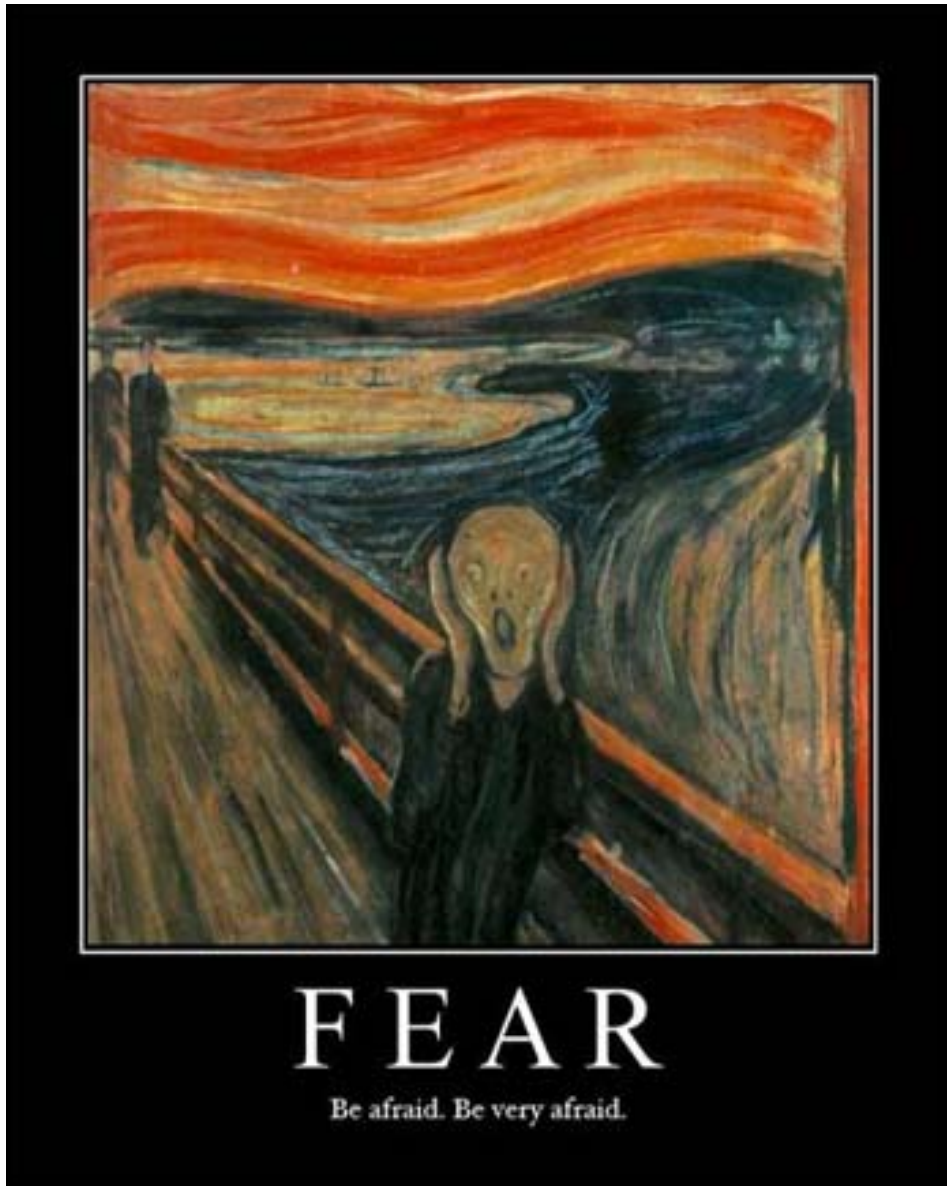
If you

- fit your model carelessly;
- do not properly estimate the test error;
- or select a model based on training set rather than test set performance;

then you will woefully overfit your training data, leading to a model that looks good on training data but will perform atrociously on future observations.

Our intuition breaks down in high dimensions, and so rigorous model-fitting is crucial.

The Curse of Dimensionality



The Curse of Dimensionality

Q: A data set with more variables is better than a data set with fewer variables, right?

The Curse of Dimensionality

Q: A data set with more variables is better than a data set with fewer variables, right?

A: Not necessarily!

Noise variables – such as genes whose expression levels are not truly associated with the response being studied – will simply increase the risk of overfitting, and the difficulty of developing an effective model that will perform well on future observations.

On the other hand, more signal variables – variables that are truly associated with the response being studied – are always useful!

Every Biostatisticians' Favorite Anecdote

A biostatistician walks into a collaborator's office with a list of genes found to be predictive of survival time in a condition of interest....

Wise Words

In high-dimensional data analysis, common mistakes are simple, and simple mistakes are common.

– Keith Baggerly

Before You're Done with Your Analysis

- Estimate the test error.
- Do a "sanity check" whenever possible.
  - "Spot-check" the variables that have the largest coefficients in the model.
  - Rewrite your code from scratch. Do you get the same answer again?

Fitting models in high-dimensions: one mistake away from disaster!