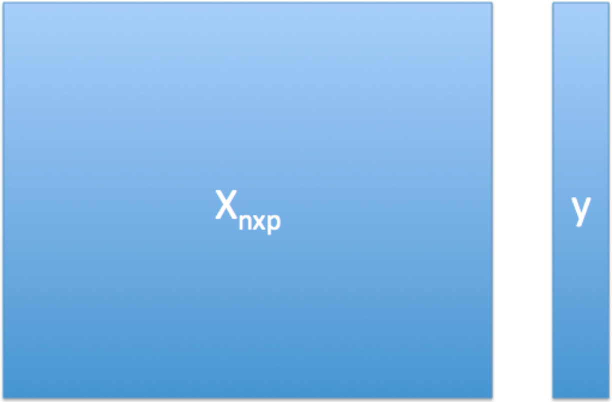
Supervised Learning: Regression, Part I

Noah Simon & Ali Shojaie

Jul 31-Aug, 2023
Summer Institute in Statistics for Big Data
University of Washington

Supervised Learning: Regression



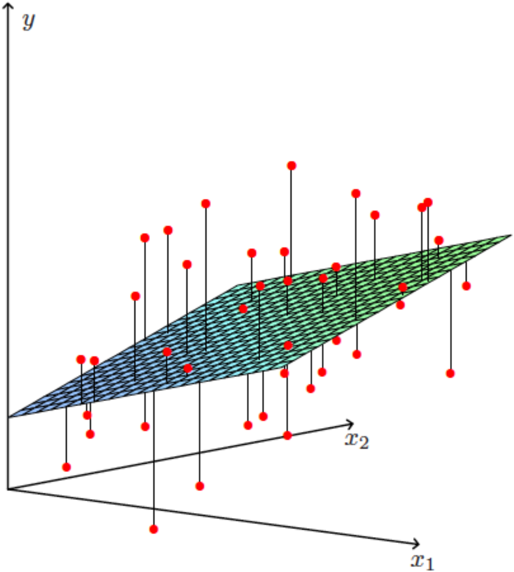
Linear Models

- 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

$$y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \epsilon_i.$$

- 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:

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

What Makes a Model Linear?

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

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

- This is not a linear model:

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

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.

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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

i.e.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}.$$

Least Squares Regression

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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

- Most common approach in classical statistics is least squares:

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

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

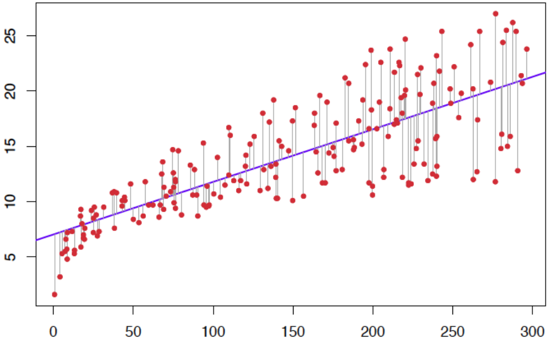
Least Squares Regression

- There are a lot of ways we could fit the model
$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$
- Most common approach in classical statistics is least squares:
$$\underset{\boldsymbol{\beta}}{\text{minimize}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \right\},$$
where $\|\mathbf{a}\|^2 \equiv \sum_{i=1}^n a_i^2$.
- This means we are looking for $\beta_1, \ldots, \beta_p$ such that
$$\sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2$$
is as small as possible.

Least Squares Regression

- There are a lot of ways we could fit the model
$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$
- Most common approach in classical statistics is least squares:
$$\underset{\boldsymbol{\beta}}{\text{minimize}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \right\},$$
where $\|\mathbf{a}\|^2 \equiv \sum_{i=1}^{n} a_i^2$.
- This means we are looking for $\beta_1, \ldots, \beta_p$ such that
$$\sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2$$
is as small as possible.
- Equivalently, 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.

Least Squares



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

Purple line minimizes 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

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

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

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

- 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

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

- 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):

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

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

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

- 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):

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

- 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

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

- 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):

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

- 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.

Least Squares as More Variables are Included in the Model

- 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 in the Model

- 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 in the Model

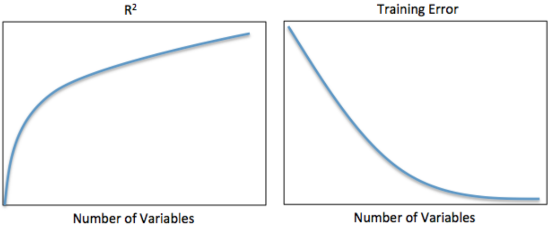
- Training error and R^2 are *not* good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features 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 in the Model

- 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!
  - Want to predict the survival time of a new patient who walks into the clinic!
  - Want to diagnose cancer for a patient not used in training!
  - 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!
  - Want to predict the survival time of a new patient who walks into the clinic!
  - Want to diagnose cancer for a patient not used in training!
  - Want to predict risk of diabetes for a patient who wasn't used to fit the model!
- What we really care about:

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

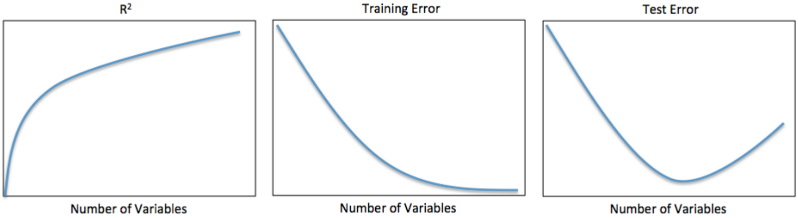
  where

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

  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...

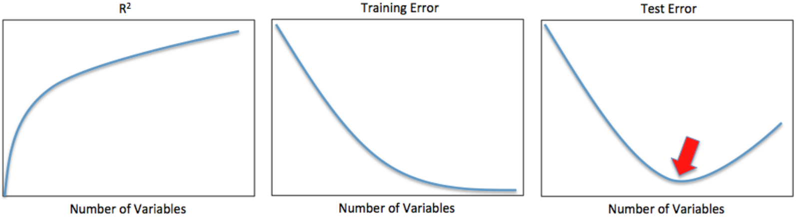


... the training error decreases and the R^2 increases!

But the test error might not!

Training Error versus Test Error

As we add more variables into the model...

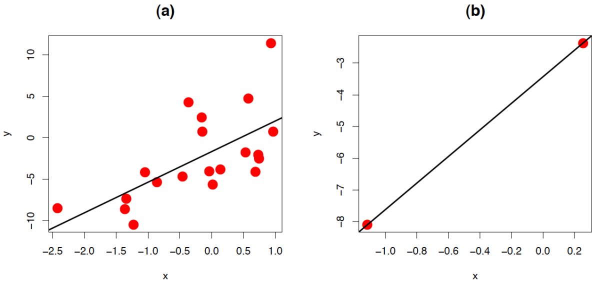


... the training error decreases and the R^2 increases!

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.

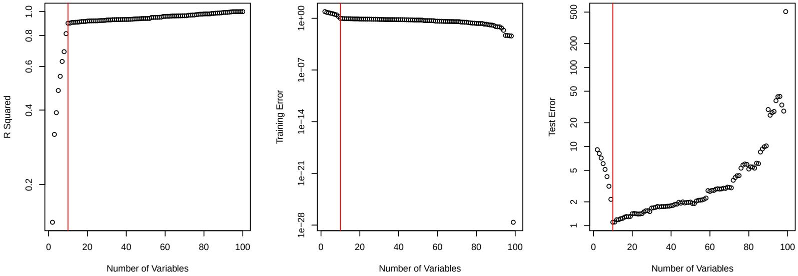
Model Complexity, Training Error, and Test Error

- 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='Number of Variables', ylab="R Squared", log="y")
abline(v=10,col="red")
plot(2:100,trainerr, xlab='Number of Variables', ylab="Training Error",log="y")
abline(v=10,col="red")
plot(2:100,testerr, xlab='Number 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:

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

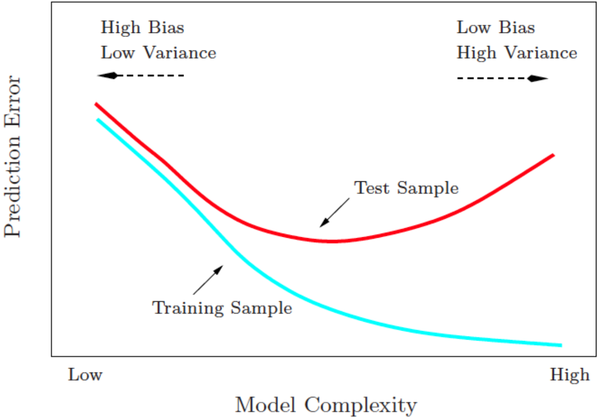
Bias and Variance

- 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



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.
- When $p \approx n$ or $p \gg n$, must work hard to avoid overfitting.

Overfitting

- Fitting an overly complex model — a model that has too much variance — is known as overfitting.
- When $p \approx n$ or $p \gg n$, must work hard to avoid overfitting.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.

Overfitting

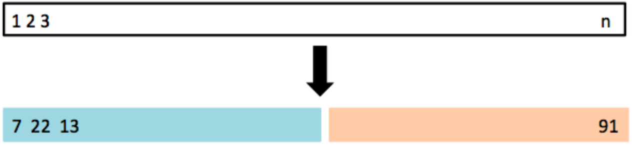
- Fitting an overly complex model — a model that has too much variance — is known as overfitting.
- When $p \approx n$ or $p \gg n$, must work hard to avoid overfitting.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.
- How can we estimate the test error?
  1. The validation set approach.
  2. Leave-one-out cross-validation.
  3. K-fold cross-validation.

Three Ways to Estimate Test Error

1. The validation set approach.
2. Leave-one-out cross-validation.
3. K-fold cross-validation.

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

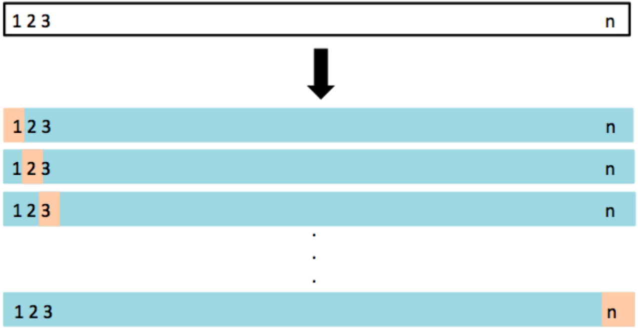
1. Split the observations into two sets of approximately equal size, a training set and a validation set.
   a. Fit the model using the training observations. Let $\hat{\boldsymbol{\beta}}_{(train)}$ denote the regression coefficient estimates.
   b. For each observation in the validation set, compute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

Three Ways to Estimate Test Error

1. The validation set approach.
2. Leave-one-out cross-validation.
3. K-fold cross-validation.

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

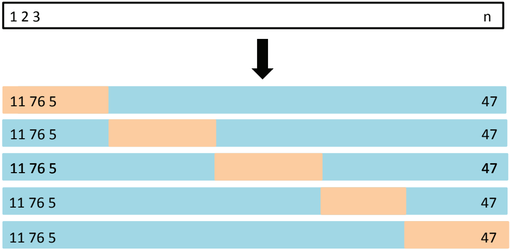
1. For $i = 1, \dots, n$:
   a. Fit the model using observations $1, \dots, i-1, i+1, \dots, n$. Let $\hat{\boldsymbol{\beta}}_{(i)}$ denote the regression coefficient estimates.
   b. Compute $e_i = (y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{(i)})^2$.
2. Calculate $\sum_{i=1}^n e_i$, the total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

Three Ways to Estimate Test Error

1. The validation set approach.
2. Leave-one-out cross-validation.
3. K-fold cross-validation.

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

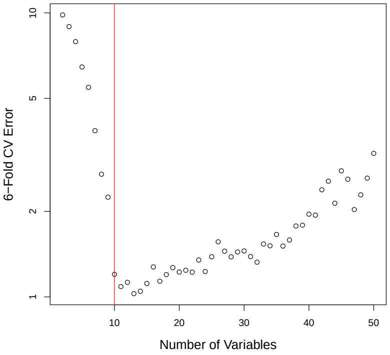
A generalization of leave-one-out cross-validation:

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.

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...

After we estimate the test error, we refit the "best" model on all of the available observations.

Let's Try Out Cross-Validation in R!

Chapter 5 R lab
First Half: Cross-Validation
www.statlearning.com

Regression with Big Data

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

Regression with Big Data

- We usually cannot perform least squares regression to fit a model with high-dimensional 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 with Big Data

- We usually cannot perform least squares regression to fit a model with high-dimensional 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 three ways to fit less complex models:
  1. Variable Pre-Selection
  2. Ridge Regression
  3. Lasso Regression

Regression with Big Data

- We usually cannot perform least squares regression to fit a model with high-dimensional 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 three ways to fit less complex models:
  1. Variable Pre-Selection
  2. Ridge Regression
  3. Lasso Regression
- These are alternatives to least squares.

Regression with Big Data

- We usually cannot perform least squares regression to fit a model with high-dimensional 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 three ways to fit less complex models:
  1. Variable Pre-Selection
  2. Ridge Regression
  3. Lasso Regression
- These are alternatives to least squares.
- Each of these approaches will allow us to choose the level of complexity — e.g. the number of variables in the model.
- Will select level of complexity using cross-validation or validation set approach.

The Fundamental Truth About High-Dimensional Data

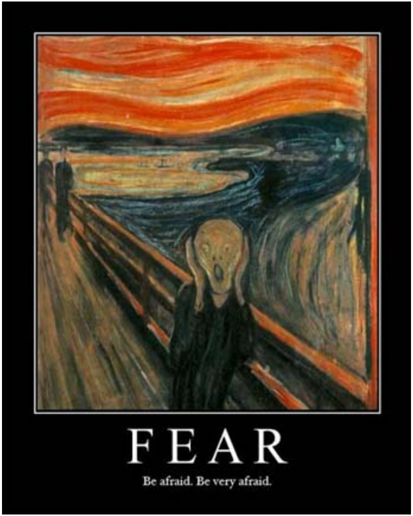
If you

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

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

Common mistakes are simple, and simple mistakes are common.

– Keith Baggerly

Before You're Done 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?

Questions to Discuss

If my data come from a time series, are there additional challenges in validation?

If my dataset has multiple measurements from each of my patients, do I need to be careful with validation?

> e.g. if I am trying to predict delirium after surgery and have patients with multiple surgeries...

If a collaborator tells me that a relationship is monotonic, but definitely not linear, does it ever make sense to use a linear model?