High-Dimensional Statistical Learning: Introduction

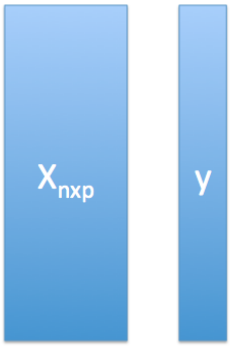
Ali Shojaie
University of Washington
http://faculty.washington.edu/ashojaie/

August 2018
Tehran, Iran

A Simple Example

- Suppose we have $n = 400$ people with diabetes for whom we have $p = 3$ serum-level measurements (LDL, HDL, GLU).
- Want to predict these peoples' disease progression after 1 year.

A Simple Example



Notation:

- n is the number of observations.
- p the number of variables/features/predictors.
- y is a n-vector containing response/outcome for each of n observations.
- X is a $n \times p$ data matrix.

Linear Regression on a Simple Example

- You can perform linear regression to develop a model to predict progression using LDL, HDL, and GLU:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

where y is our continuous measure of disease progression, X_1, X_2, X_3 are our serum-level measurements, and ϵ is a noise term.

Linear Regression on a Simple Example

- You can perform linear regression to develop a model to predict progression using LDL, HDL, and GLU:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

where y is our continuous measure of disease progression, X_1, X_2, X_3 are our serum-level measurements, and ϵ is a noise term.

- You can look at the coefficients, p-values, and t-statistics for your linear regression model in order to interpret your results.

Classical Statistics
Biological Big Data
Supervised and Unsupervised Learning

Linear Regression on a Simple Example

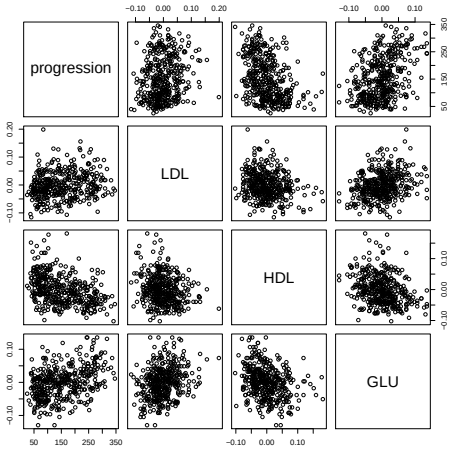
- You can perform linear regression to develop a model to predict progression using LDL, HDL, and GLU:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

where y is our continuous measure of disease progression, X_1, X_2, X_3 are our serum-level measurements, and ϵ is a noise term.

- You can look at the coefficients, p-values, and t-statistics for your linear regression model in order to interpret your results.
- You learned everything (or most of what) you need to analyze this data set in Classical Statistics!

A Relationship Between the Variables?



Linear Model Output

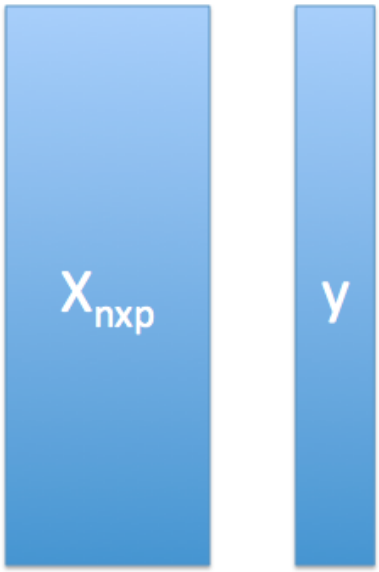
	Estimate	Std. Error	T-Stat	P-Value
Intercept	152.928	3.385	45.178	$<$ 2e-16 ***
LDL	77.057	75.701	1.018	0.309
HDL	-487.574	75.605	-6.449	3.28e-10 ***
GLU	477.604	76.643	6.232	1.18e-09 ***

$$\text{progression\_measure} \approx 152.9 + 77.1 \times \text{LDL} - 487.6 \times \text{HDL} + 477.6 \times \text{GLU}.$$

Low-Dimensional Versus High-Dimensional

- The data set that we just saw is low-dimensional: $n \gg p$.
- Lots of the data sets coming out of modern biological techniques are high-dimensional: $n \approx p$ or $n \ll p$.
- This poses statistical challenges! Classical Statistics no longer applies.

Low Dimensional



High Dimensional

What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2

What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.

Classical Statistics
Biological Big Data
Supervised and Unsupervised Learning

What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.
- If we include too many predictors, we will overfit the data.
- Overfitting: Model looks great on the data used to develop it, but will perform very poorly on future observations.

What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.
- If we include too many predictors, we will overfit the data.
- Overfitting: Model looks great on the data used to develop it, but will perform very poorly on future observations.
- When $p \approx n$ or $p > n$, overfitting is guaranteed unless we are very careful.

Why Does Dimensionality Matter?

- Classical statistical techniques, such as linear regression, *cannot* be applied.
- Even very simple tasks, like identifying variables that are associated with a response, must be done with care.
- High risks of overfitting, false positives, and more.

Why Does Dimensionality Matter?

- Classical statistical techniques, such as linear regression, *cannot* be applied.
- Even very simple tasks, like identifying variables that are associated with a response, must be done with care.
- High risks of overfitting, false positives, and more.

This course: Statistical machine learning tools for big — mostly high-dimensional — data.

Supervised and Unsupervised Learning

- Statistical machine learning can be divided into two main areas: supervised and unsupervised.

Supervised and Unsupervised Learning

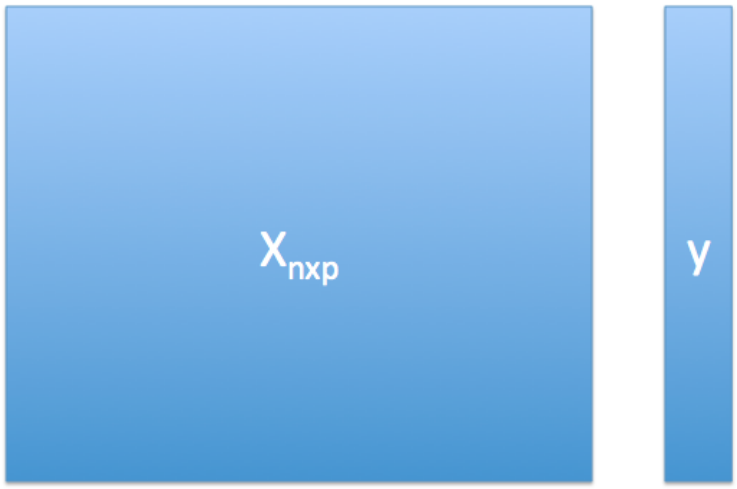
- Statistical machine learning can be divided into two main areas: supervised and unsupervised.
- Supervised Learning: Use a data set X to predict or detect association with a response y.
  - Regression
  - Classification
  - Hypothesis Testing

Classical Statistics
Biological Big Data
Supervised and Unsupervised Learning

Supervised and Unsupervised Learning

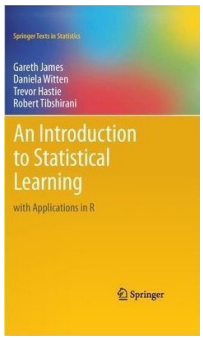
- Statistical machine learning can be divided into two main areas: supervised and unsupervised.
- Supervised Learning: Use a data set X to predict or detect association with a response y.
  - Regression
  - Classification
  - Hypothesis Testing
- Unsupervised Learning: Discover the signal in X, or detect associations within X.
  - Dimension Reduction
  - Clustering

Supervised Learning



Unsupervised Learning

"Course Textbook" . . . with applications in R



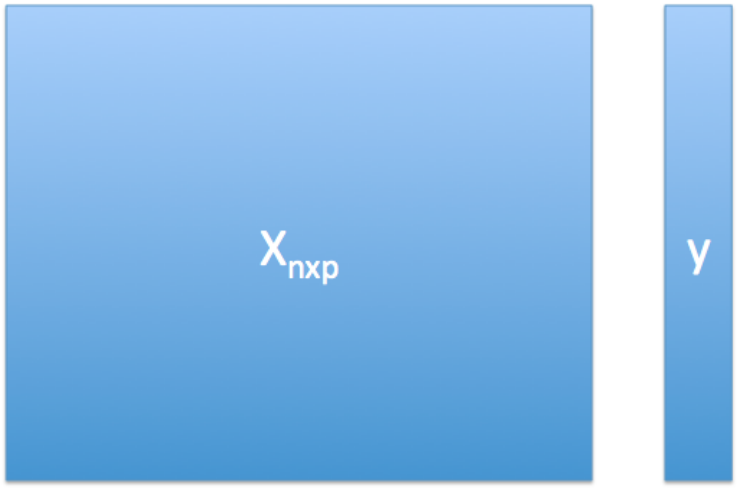
- Available for (free!) download from www.statlearning.com.
- An accessible introduction to statistical machine learning, with an R lab at the end of each chapter.
- We will go through some of these R labs.
- To learn more, go through them on your own!

High-Dimensional Statistical Learning: Bias-Variance Tradeoff and the Test Error

Ali Shojaie
University of Washington
http://faculty.washington.edu/ashojaie/

August 2018
Tehran, Iran

Supervised Learning



Regression Versus Classification

Regression Versus Classification

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

3 / 48

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.

3 / 48

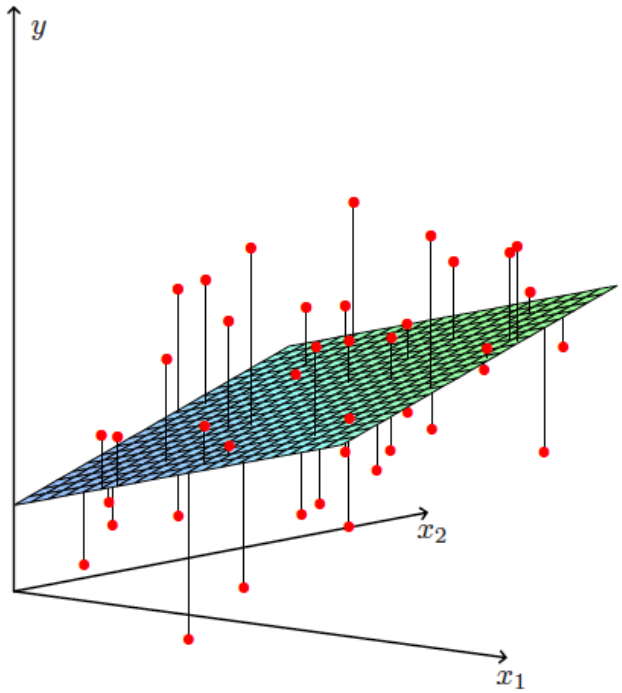
Linear Models

- 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

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

- 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.
- 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 many ways we could fit the model

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

- Most common approach in classical statistics is least squares:

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

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

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

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

is as small as possible, or in other words, 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.

Let's Try Out Least Squares in R!

Chapter 3 R lab
www.statlearning.com

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.

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

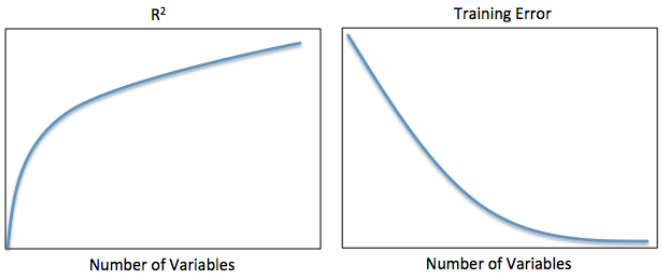
- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features to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

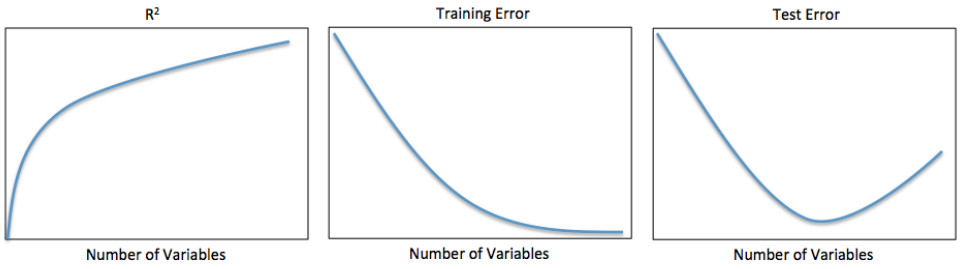
$$\hat{y}_{test} = \hat{\beta}_1 X_{test,1} + \ldots + \hat{\beta}_p X_{test,p},$$

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...

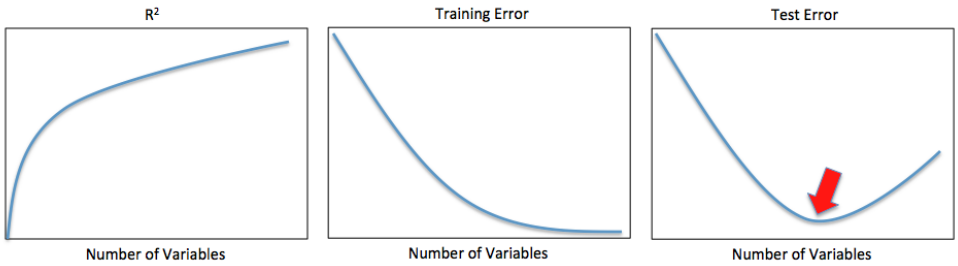


... the training error decreases and the R^2 increases!

But the test error might not!

Training Error versus Test Error

As we add more variables into the model...

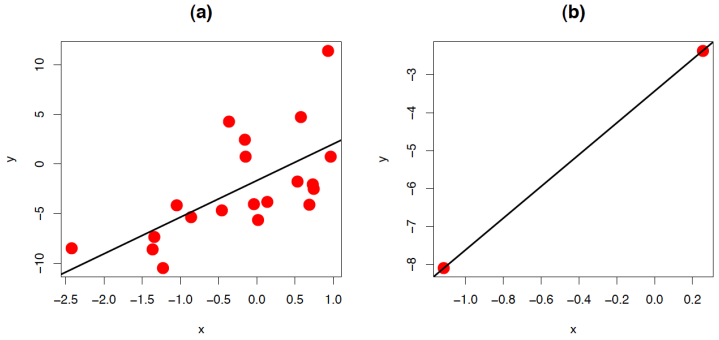


... the training error decreases and the R^2 increases!

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.

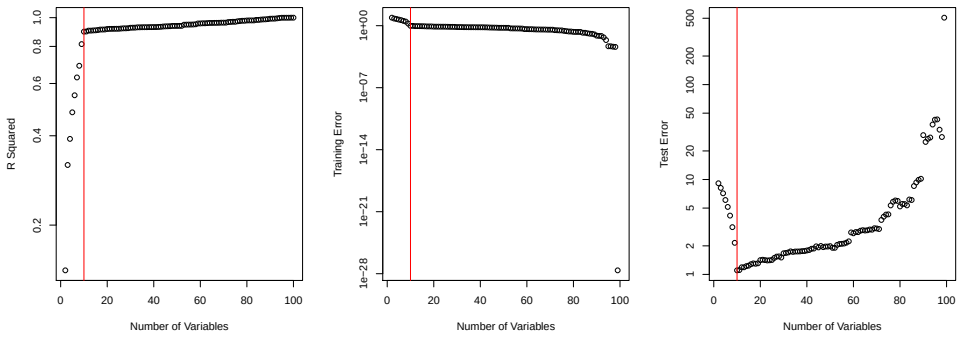
Model Complexity, Training Error, and Test Error

- 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='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")
```

A Simulated Example



- 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:

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

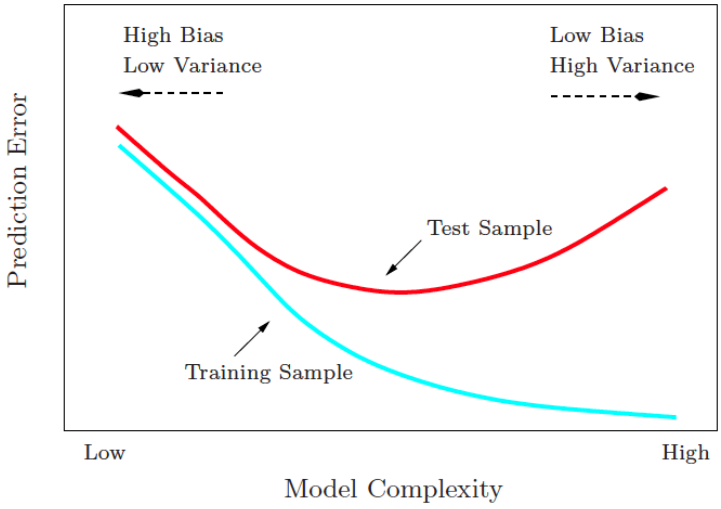
Bias and Variance

- 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.
- In the high-dimensional 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 the high-dimensional 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

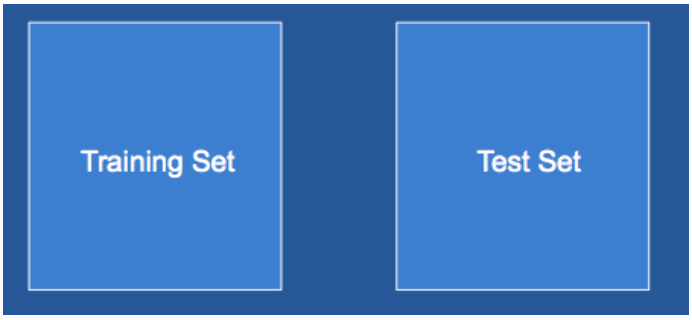
- Fitting an overly complex model – a model that has too much variance – is known as overfitting.
- In the high-dimensional setting, when $p \gg n$, we must work hard not to overfit the data.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.
-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

Training Set Versus Test Set

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

Training Set Versus Test Set

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



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

Training Set Versus Test Set

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

Q: Can there ever, under any circumstance, be sample overlap between the training and test sets?
A: Never!

Training Set Versus Test Set

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

Training Set Versus 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!

Training Set And Test Set

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.

Training Set And Test Set

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?

26 / 48

Choosing Between Several Models

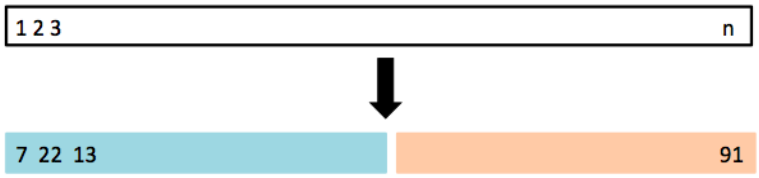
- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!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

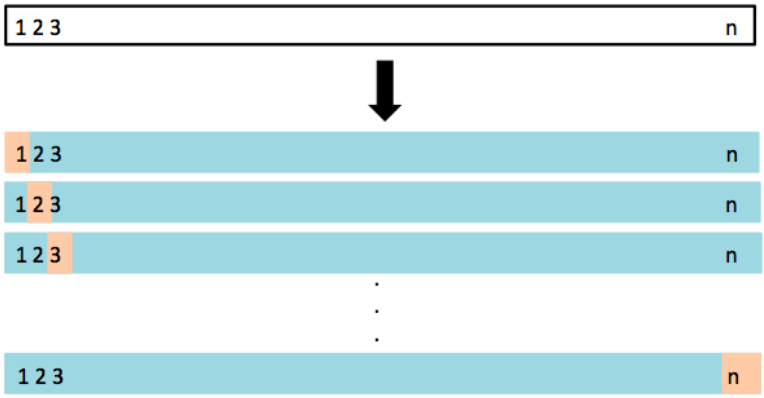
For a given model, we perform the following procedure:

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 test 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

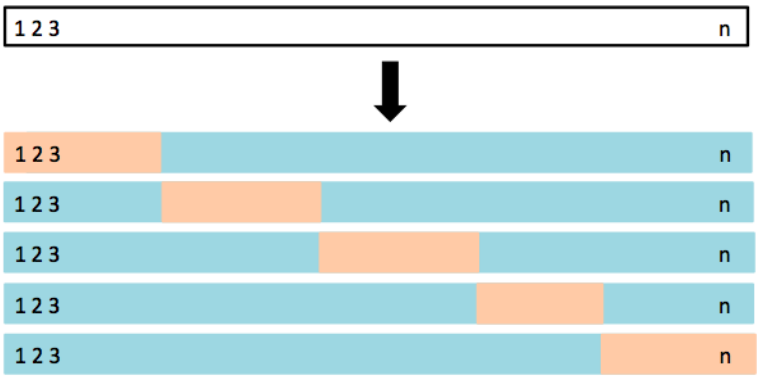
For a given model, we perform the following procedure:

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:

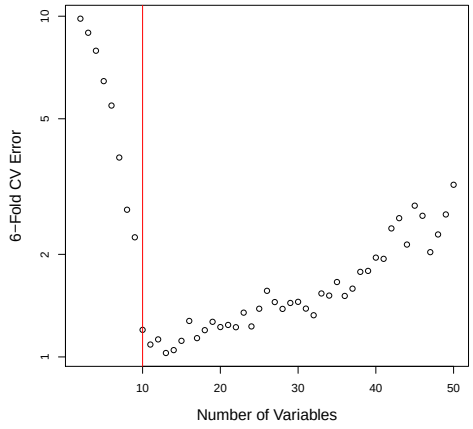
1. Split the n observations into K equally-sized folds.
2. For $k = 1, \ldots, K$:
   a. Fit the model using the observations not in the kth fold.
   b. Let e_k denote the test error for the observations in the kth fold.
3. Calculate $\sum_{k=1}^{K} e_k$, the total CV error.

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

Let's Try Out Cross-Validation in R!

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

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.

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 in High Dimensions

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

Regression in High Dimensions

- We usually cannot perform least squares regression to fit a model in the high-dimensional setting, 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.

44 / 48

Regression in High Dimensions

- We usually cannot perform least squares regression to fit a model in the high-dimensional setting, 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 in High Dimensions

- We usually cannot perform least squares regression to fit a model in the high-dimensional setting, 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 in High Dimensions

- We usually cannot perform least squares regression to fit a model in the high-dimensional setting, 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.

Regression in High Dimensions

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

Regression in High Dimensions

- We usually cannot perform least squares regression to fit a model in the high-dimensional setting, 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 in High Dimensions

- We usually cannot perform least squares regression to fit a model in the high-dimensional setting, 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.

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

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!

Wise Words

In high-dimensional data analysis, 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?

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

High-Dimensional Statistical Learning: Regression

Ali Shojaie
University of Washington
http://faculty.washington.edu/ashojaie/

August 2018
Tehran, Iran

Linear Models in High Dimensions

- When p is large, least squares regression will lead to very low training error but terrible test error.

Linear Models in High Dimensions

- When p is large, least squares regression will lead to very low training error but terrible test error.
- We will now see some approaches for fitting linear models in high dimensions, $p \gg n$.
- These approaches also work well when $p \approx n$ or $n > p$.

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes measured using a microarray.

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes measured using a microarray.
  - protein levels.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes measured using a microarray.
  - protein levels.
  - mutations in genes potentially implicated in breast cancer.

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes measured using a microarray.
  - protein levels.
  - mutations in genes potentially implicated in breast cancer.
- How can we develop a model with low test error in this setting?

Remember

- We have n training observations.
- Our goal is to get a model that will perform well on future test observations.
- We'll incur some bias in order to reduce variance.

Variable Pre-Selection

The simplest approach for fitting a model in high dimensions:

1. Choose a small set of variables, say the q variables that are most correlated with the response, where $q < n$ and $q < p$.
2. Use least squares to fit a model predicting y using only these q variables.

This approach is simple and straightforward.

Variable Pre-Selection in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
cors <- cor(xtr,ytr)
whichers <- which(abs(cors)>.2)
mod <- lm(ytr~xtr[,whichers])
print(summary(mod))
```

How Many Variable to Use?

- We need a way to choose q, the number of variables used in the regression model.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

How Many Variable to Use?

- We need a way to choose q, the number of variables used in the regression model.
- We want q that minimizes the test error.

7 / 45

How Many Variable to Use?

- We need a way to choose q, the number of variables used in the regression model.
- We want q that minimizes the test error.
- For a range of values of q, we can perform the validation set approach, leave-one-out cross-validation, or K-fold cross-validation in order to estimate the test error.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

How Many Variable to Use?

- We need a way to choose q, the number of variables used in the regression model.
- We want q that minimizes the test error.
- For a range of values of q, we can perform the validation set approach, leave-one-out cross-validation, or K-fold cross-validation in order to estimate the test error.
- Then choose the value of q for which the estimated test error is smallest.

Estimating the Test Error For a Given q

This is the right way to estimate the test error using the validation set approach:

1. Split the observations into a training set and a validation set.
2. Using the training set only:
   a. Identify the q variables most associated with the response.
   b. Use least squares to fit a model predicting y using those q variables.
   c. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
3. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Estimating the Test Error For a Given q

This is the wrong way to estimate the test error using the validation set approach:

1. Identify the q variables most associated with the response on the full data set.
2. Split the observations into a training set and a validation set.
3. Using the training set only:
   a. Use least squares to fit a model predicting y using those q variables.
   b. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
4. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Frequently Asked Questions

- **Q:** Does it really matter how you estimate the test error?
  A: Yes.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Frequently Asked Questions

- **Q:** Does it really matter how you estimate the test error?
  A: Yes.
- **Q:** Would anyone make such a silly mistake?
  A: Yes.

A Better Approach

- The variable pre-selection approach is simple and easy to implement — all you need is a way to calculate correlations, and software to fit a linear model using least squares.

A Better Approach

- The variable pre-selection approach is simple and easy to implement — all you need is a way to calculate correlations, and software to fit a linear model using least squares.
- But it might not work well: just because a bunch of variables are correlated with the response doesn't mean that when used together in a linear model, they will predict the response well.

A Better Approach

- The variable pre-selection approach is simple and easy to implement — all you need is a way to calculate correlations, and software to fit a linear model using least squares.
- But it might not work well: just because a bunch of variables are correlated with the response doesn't mean that when used together in a linear model, they will predict the response well.
- What we really want to do: pick the q variables that best predict the response.

Subset Selection

Several Approaches:

- Best Subset Selection: Consider all subsets of predictors
  Computational mess!
- Stepwise Regression: Greedily add/remove predictors
  Heuristic and potentially inefficient
- Modern Penalized Methods

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.
- In other words, we'd like to consider all 2^p possible models.

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.
- In other words, we'd like to consider all 2^p possible models.
- This is called best subset selection.

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.
- In other words, we'd like to consider all 2^p possible models.
- This is called best subset selection.
- Unfortunately, this is computationally intractable:
  - When $p = 3$, $2^p = 8$.
  - When $p = 6$, $2^p = 64$.
  - When $p = 250$, there are $2^{250} \approx 10^{80}$ possible models. According to www.universetoday.com, this is around the number of atoms in the known universe.
  - Not feasible to consider so many models!

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.
- In other words, we'd like to consider all 2^p possible models.
- This is called best subset selection.
- Unfortunately, this is computationally intractable:
  - When $p = 3$, $2^p = 8$.
  - When $p = 6$, $2^p = 64$.
  - When $p = 250$, there are $2^{250} \approx 10^{80}$ possible models. According to www.universetoday.com, this is around the number of atoms in the known universe.
  - Not feasible to consider so many models!
- Need an efficient way to sift through all of these models: forward stepwise regression.

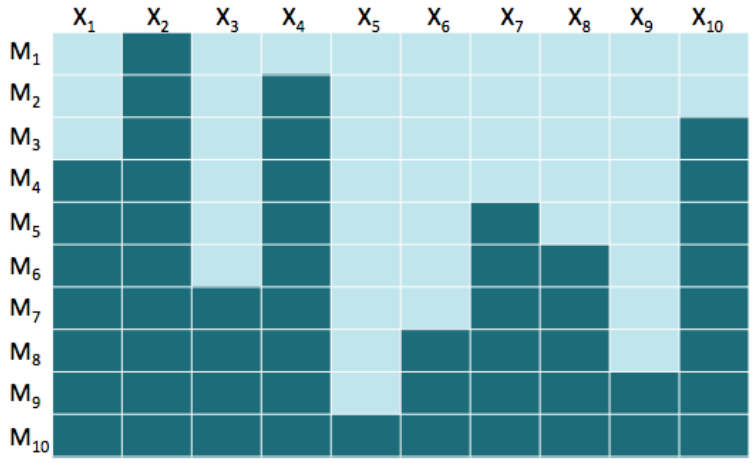
Forward Stepwise Regression

1. Use least squares to fit p univariate regression models, and select the predictor corresponding to the best model (according to e.g. training set MSE).
2. Use least squares to fit $p - 1$ models containing that one predictor, and each of the $p - 1$ other predictors. Select the predictors in the best two-variable model.
3. Now use least squares to fit $p - 2$ models containing those two predictors, and each of the $p - 2$ other predictors. Select the predictors in the best three-variable model.
4. And so on....

This gives us a nested set of models, containing the predictors

$$\mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \mathcal{M}_3 \subseteq \ldots.$$

Forward Stepwise Regression With $p = 10$



Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(leaps)
out <- regsubsets(xtr,ytr,nvmax=30,method="forward")
print(summary(out))
print(coef(out,1:10))
```

Which Value of q is Best?

- This procedure traces out a set of models, containing between 1 and p variables.
- The qth model contains q variables, given by the set $\mathcal{M}_q$.
- **Q:** Which value of q is best?

Which Value of q is Best?

- This procedure traces out a set of models, containing between 1 and p variables.
- The qth model contains q variables, given by the set $\mathcal{M}_q$.
- **Q:** Which value of q is best?
  A: *The one that minimizes the test error!*
- We can select the value of q using cross-validation or the validation set approach.

Drawback of Forward Stepwise Selection

- Forward stepwise selection isn't guaranteed to give you the best model containing q variables.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Drawback of Forward Stepwise Selection

- Forward stepwise selection isn't guaranteed to give you the best model containing q variables.
- To get the best model with q variables, you'd need to consider every possible one; computationally intractable.

Drawback of Forward Stepwise Selection

- Forward stepwise selection isn't guaranteed to give you the best model containing q variables.
- To get the best model with q variables, you'd need to consider every possible one; computationally intractable.
- For instance, suppose that the best model with one variable is

$$y = \beta_3 X_3 + \epsilon$$

and the best model with two variables is

$$y = \beta_4 X_4 + \beta_8 X_8 + \epsilon.$$

Then forward stepwise selection will not identify the best two-variable model.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Drawback of Forward Stepwise Selection

- Forward stepwise selection isn't guaranteed to give you the best model containing q variables.
- To get the best model with q variables, you'd need to consider every possible one; computationally intractable.
- For instance, suppose that the best model with one variable is

$$y = \beta_3 X_3 + \epsilon$$

and the best model with two variables is

$$y = \beta_4 X_4 + \beta_8 X_8 + \epsilon.$$

Then forward stepwise selection will not identify the best two-variable model.

- **Q:** Does this really happen in practice?
  A: Yes.

How To Do Forward Stepwise?

Wrong: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the model with best performance on the validation set. Then, refit the model (using those q variables) on the full data set.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

How To Do Forward Stepwise?

Wrong: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the model with best performance on the validation set. Then, refit the model (using those q variables) on the full data set.

Right: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the value of q corresponding to the best-performing model on the validation set. Then, perform forward stepwise selection in order to obtain a q-variable model on the full data set.

How To Do Forward Stepwise?

Wrong: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the model with best performance on the validation set. Then, refit the model (using those q variables) on the full data set.

Right: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the value of q corresponding to the best-performing model on the validation set. Then, perform forward stepwise selection in order to obtain a q-variable model on the full data set.

Bottom Line: We estimate the test error in order to choose the correct level of **model complexity**. Then we refit the model on the full data set.

Let's Try It Out in R!

Chapter 6 R Lab, Part 1
www.statlearning.com

Ridge Regression and the Lasso

- Best subset, and stepwise regression control model complexity by using subsets of the predictors.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Ridge Regression and the Lasso

- Best subset, and stepwise regression control model complexity by using subsets of the predictors.
- Ridge regression and the lasso instead control model complexity by using an alternative to least squares, by shrinking the regression coefficients.

Ridge Regression and the Lasso

- Best subset, and stepwise regression control model complexity by using subsets of the predictors.
- Ridge regression and the lasso instead control model complexity by using an alternative to least squares, by shrinking the regression coefficients.
- This is known as regularization or penalization.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Ridge Regression and the Lasso

- Best subset, and stepwise regression control model complexity by using subsets of the predictors.
- Ridge regression and the lasso instead control model complexity by using an alternative to least squares, by shrinking the regression coefficients.
- This is known as regularization or penalization.
- Hot area in statistical machine learning today.

Crazy Coefficients

- When $p > n$, some of the variables are highly correlated.

Crazy Coefficients

- When $p > n$, some of the variables are highly correlated.
- Why does correlation matter?
  - Suppose that X_1 and X_2 are highly correlated with each other... assume $X_1 = X_2$ for the sake of argument.
  - And suppose that the least squares model is

$$\hat{y} = X_1 - 2X_2 + 3X_3.$$

  - Then this is also a least squares model:

$$\hat{y} = 100000001X_1 - 100000002X_2 + 3X_3.$$

Crazy Coefficients

- When $p > n$, some of the variables are highly correlated.
- Why does correlation matter?
  - Suppose that X_1 and X_2 are highly correlated with each other... assume $X_1 = X_2$ for the sake of argument.
  - And suppose that the least squares model is

$$\hat{y} = X_1 - 2X_2 + 3X_3.$$

  - Then this is also a least squares model:

$$\hat{y} = 100000001X_1 - 100000002X_2 + 3X_3.$$

- Bottom Line: When there are too many variables, the least squares coefficients can get crazy!
- This craziness is directly responsible for poor test error.
- It amounts to too much model complexity.

A Solution: Don't Let the Coefficients Get Too Crazy

- Recall that least squares involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2.$$

A Solution: Don't Let the Coefficients Get Too Crazy

- Recall that least squares involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2.$$

- Ridge regression involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j \beta_j^2.$$

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

A Solution: Don't Let the Coefficients Get Too Crazy

- Recall that least squares involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2.$$

- Ridge regression involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j \beta_j^2.$$

- Equivalently, find $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

subject to the constraint that

$$\sum_{j=1}^{p} \beta_j^2 \leq s.$$

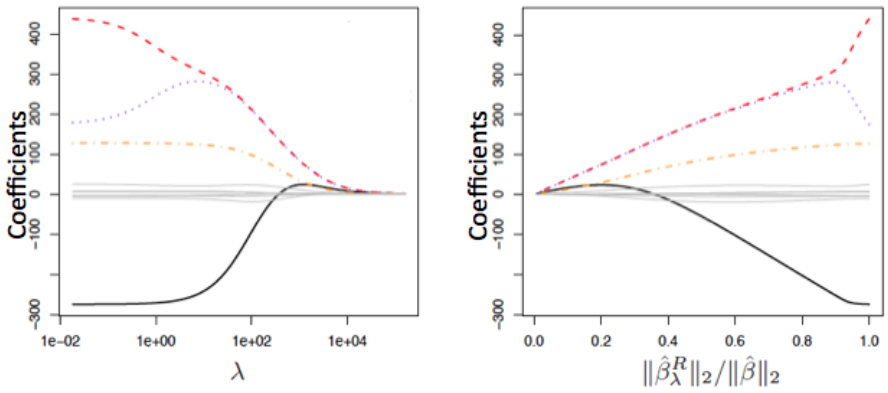
Ridge Regression

- Ridge regression coefficient estimates minimize

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j \beta_j^2.$$

- Here λ is a nonnegative tuning parameter that shrinks the coefficient estimates.
- When $\lambda = 0$, then ridge regression is just the same as least squares.
- As λ increases, then $\sum_{j=1}^{p} (\hat{\beta}_{\lambda,j}^R)^2$ decreases – i.e. coefficients become shrunken towards zero.
- When $\lambda = \infty$, $\hat{\boldsymbol{\beta}}_\lambda^R = 0$.

Ridge Regression As λ Varies



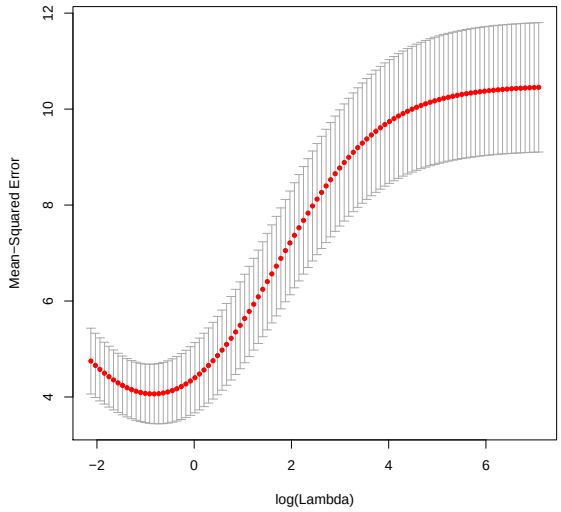
Ridge Regression In Practice

- Perform ridge regression for a very fine grid of λ values.
- Use cross-validation or the validation set approach to select the optimal value of λ – that is, the best level of model complexity.
- Perform ridge on the full data set, using that value of λ.

Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(glmnet)
cv.out <- cv.glmnet(xtr,ytr,alpha=0,nfolds=5)
print(cv.out$cvm)
plot(cv.out)
cat("CV Errors", cv.out$cvm,fill=TRUE)
cat("Lambda with smallest CV Error",
cv.out$lambda[which.min(cv.out$cvm)],fill=TRUE)
cat("Coefficients", as.numeric(coef(cv.out)),fill=TRUE)
cat("Number of Zero Coefficients",
sum(abs(coef(cv.out))<1e-8),fill=TRUE)
```

R Output



Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.

Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.
- Often want a simpler model involving a subset of the features.

29 / 45

Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.
- Often want a simpler model involving a subset of the features.
- The lasso involves performing a little tweak to ridge regression so that the resulting model contains mostly zeros.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.
- Often want a simpler model involving a subset of the features.
- The lasso involves performing a little tweak to ridge regression so that the resulting model contains mostly zeros.
- In other words, the resulting model is sparse. We say that the lasso performs feature selection.

Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.
- Often want a simpler model involving a subset of the features.
- The lasso involves performing a little tweak to ridge regression so that the resulting model contains mostly zeros.
- In other words, the resulting model is sparse. We say that the lasso performs feature selection.
- The lasso is a very active area of research interest in the statistical community!

The Lasso

- The lasso involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j |\beta_j|.$$

The Lasso

- The lasso involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j |\beta_j|.$$

- Equivalently, find $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

subject to the constraint that

$$\sum_{j=1}^{p} |\beta_j| \leq s.$$

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

The Lasso

- The lasso involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j |\beta_j|.$$

- Equivalently, find $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

subject to the constraint that

$$\sum_{j=1}^{p} |\beta_j| \leq s.$$

- So lasso is just like ridge, except that β_j^2 has been replaced with $|\beta_j|$.

The Lasso

- Lasso is a lot like ridge:

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

The Lasso

- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.

The Lasso

- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.
  - When $\lambda = 0$, we get least squares.

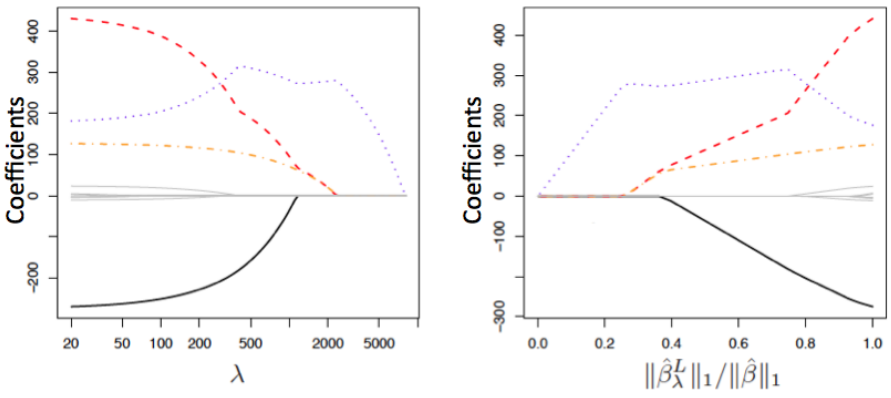
The Lasso

- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.
  - When $\lambda = 0$, we get least squares.
  - When λ is very large, we get $\hat{\beta}^L_\lambda = 0$.

The Lasso

- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.
  - When $\lambda = 0$, we get least squares.
  - When λ is very large, we get $\hat{\beta}^L_\lambda = 0$.
- But unlike ridge, lasso will give some coefficients exactly equal to zero for intermediate values of λ!

Lasso As λ Varies



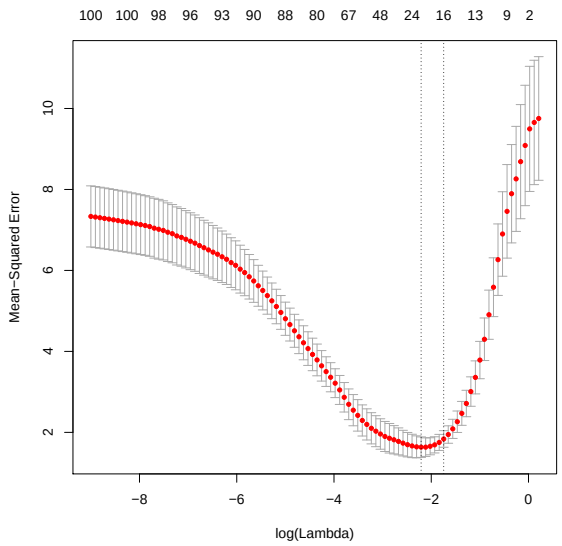
Lasso In Practice

- Perform lasso for a very fine grid of λ values.
- Use cross-validation or the validation set approach to select the optimal value of λ – that is, the best level of model complexity.
- Perform the lasso on the full data set, using that value of λ.

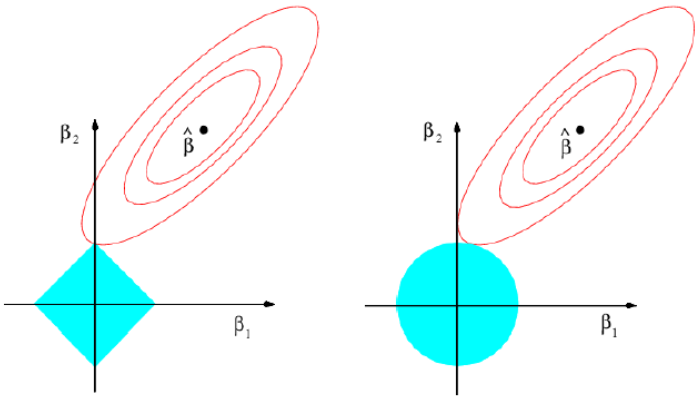
Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(glmnet)
cv.out <- cv.glmnet(xtr,ytr,alpha=1,nfolds=5)
print(cv.out$cvm)
plot(cv.out)
cat("CV Errors", cv.out$cvm,fill=TRUE)
cat("Lambda with smallest CV Error",
cv.out$lambda[which.min(cv.out$cvm)],fill=TRUE)
cat("Coefficients", as.numeric(coef(cv.out)),fill=TRUE)
cat("Number of Zero Coefficients",sum(abs(coef(cv.out))<1e-8),
fill=TRUE)
```

R Output



Ridge and Lasso: A Geometric Interpretation



Pros/Cons of Each Approach

Approach	Simplicity?*	Sparsity?**	Predictions?***
Pre-Selection	Good	Yes	So-So
Forward Stepwise	Good	Yes	So-So
Ridge	Medium	No	Great
Lasso	Bad	Yes	Great

* How simple is this model-fitting procedure? If you were stranded on a desert island with pretty limited statistical software, could you fit this model?
** Does this approach perform feature selection, i.e. is the resulting model sparse?
*** How good are the predictions resulting from this model?

No "Best" Approach

- There is no "best" approach to regression in high dimensions.

No "Best" Approach

- There is no "best" approach to regression in high dimensions.
- Some approaches will work better than others. For instance:
  - Lasso will work well if it's really true that just a few features are associated with the response.
  - Ridge will do better if all of the features are associated with the response.

Variable Pre-Selection
Subset Selection
Ridge Regression
Lasso Regression

No "Best" Approach

- There is no "best" approach to regression in high dimensions.
- Some approaches will work better than others. For instance:
  - Lasso will work well if it's really true that just a few features are associated with the response.
  - Ridge will do better if all of the features are associated with the response.
- If somebody tells you that one approach is "best"... then they are mistaken. Politely contradict them.
- While no approach is "best", some approaches are wrong (e.g.: there is a wrong way to do cross-validation)!

Let's Try It Out in R!

Chapter 6 R Lab, Part 2
www.statlearning.com

Making Linear Regression Less Linear

What if the relationship isn't linear?

$$y = 3\sin(x) + \epsilon$$
$$y = 2e^x + \epsilon$$
$$y = 3x^2 + 2x + 1 + \epsilon$$

If we know the functional form we can still use "linear regression"

Making Linear Regression Less Linear

$y = 3\sin(x) + \epsilon$:

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \begin{pmatrix} sin(x) \end{pmatrix}$$

$y = 3x^2 + 2x + 1 + \epsilon$:

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \left(\begin{array}{c|c} x & x^2 \end{array}\right)$$

Making Linear Regression Less Linear

What if we don't know the right functional form?

Use a flexible basis expansion:

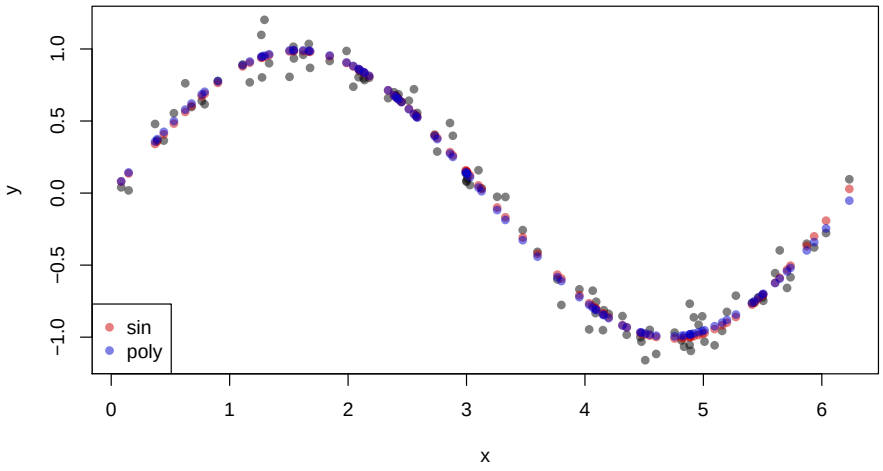
- polynomial basis

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \left(\begin{array}{c|c|c|c} x & x^2 & \cdots & x^k \end{array}\right)$$

- hockey-stick (/spline) basis

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \left(\begin{array}{c|c|c|c} x & (x-t_1)_+ & \cdots & (x-t_k)_+ \end{array}\right)$$

Making Linear Regression Less Linear



Making Linear Regression Less Linear

For high dimensional problems, expand each variable

$$\left(\begin{array}{c|c|c|c} x_1 & x_2 & \cdots & x_p \end{array}\right) \rightarrow \left(\begin{array}{c|ccc|ccc|c|ccc} x_1 & \cdots & x_1^k & x_2 & \cdots & x_2^k & \cdots & x_p & \cdots & x_p^k \end{array}\right)$$

and use the *Lasso* on this expanded problem.

---

k must be small ($\sim$ 5ish)

Spline basis generally outperforms *polynomial*

Bottom Line

Much more important than what model you fit is how you fit it.

- Was cross-validation performed properly?
- Did you select a model (or level of model complexity) based on an estimate of test error?

High-Dimensional Statistical Learning: Classification

Ali Shojaie
University of Washington
http://faculty.washington.edu/ashojaie/

August 2018
Tehran, Iran

Classification

- Regression involves predicting a continuous-valued response, like tumor size.

Classification

- Regression involves predicting a continuous-valued response, like tumor size.
- Classification involves predicting a categorical response:
  - Cancer versus Normal
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Classification

- Regression involves predicting a continuous-valued response, like tumor size.
- Classification involves predicting a categorical response:
  - Cancer versus Normal
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3
- Classification problems tend to occur very frequently in the analysis of high-dimensional data.

2 / 50

Classification

- Regression involves predicting a continuous-valued response, like tumor size.
- Classification involves predicting a categorical response:
  - Cancer versus Normal
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3
- Classification problems tend to occur very frequently in the analysis of high-dimensional data.
- Just like regression,
  - Classification cannot be blindly performed in high-dimensions because you will get zero training error but awful test error;
  - Properly estimating the test error is crucial; and
  - There are a few tricks to extend classical classification approaches to high-dimensions, which we have already seen in the regression context!

Classification

- There are many approaches out there for performing classification.
- We will discuss a few, logistic regression, K-nearest neighbors, discriminant analysis and support vector machines.

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.
- For simplicity, suppose $y \in \{0, 1\}$: a two-class classification problem.

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.
- For simplicity, suppose $y \in \{0, 1\}$: a two-class classification problem.
- The simple linear model

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

doesn't make sense for classification.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.
- For simplicity, suppose $y \in \{0, 1\}$: a two-class classification problem.
- The simple linear model

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

doesn't make sense for classification.

- Instead, the logistic regression model is

$$P(y = 1|X) = \frac{\exp(X^T\beta)}{1 + \exp(X^T\beta)}.$$

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.
- For simplicity, suppose $y \in \{0, 1\}$: a two-class classification problem.
- The simple linear model

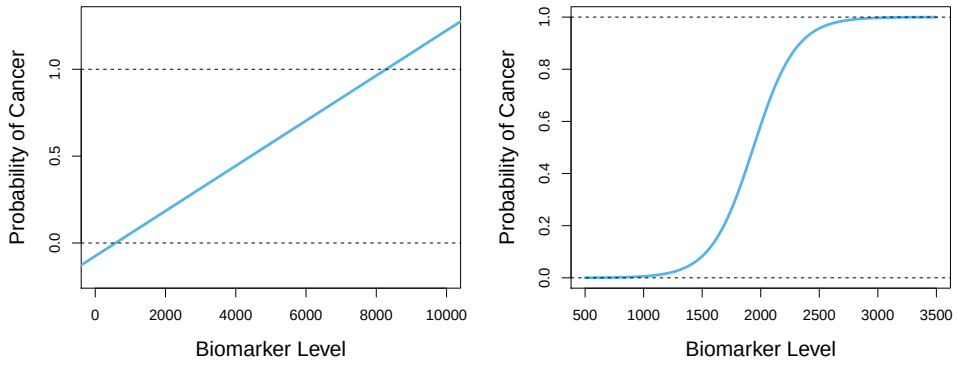
$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

  doesn't make sense for classification.
- Instead, the logistic regression model is

$$P(y = 1|X) = \frac{\exp(X^T\beta)}{1 + \exp(X^T\beta)}.$$

- We usually fit this model using maximum likelihood — like least squares, but for logistic regression.

Why Not Linear Regression?



- Left: linear regression.
- Right: logistic regression.

Ways to Extend Logistic Regression to High Dimensions

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Ways to Extend Logistic Regression to High Dimensions

1. Variable Pre-Selection

6 / 50

Ways to Extend Logistic Regression to High Dimensions

1. Variable Pre-Selection
2. Forward Stepwise Logistic Regression

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Ways to Extend Logistic Regression to High Dimensions

1. Variable Pre-Selection
2. Forward Stepwise Logistic Regression
3. Ridge Logistic Regression

Ways to Extend Logistic Regression to High Dimensions

1. Variable Pre-Selection
2. Forward Stepwise Logistic Regression
3. Ridge Logistic Regression
4. Lasso Logistic Regression

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Ways to Extend Logistic Regression to High Dimensions

1. Variable Pre-Selection
2. Forward Stepwise Logistic Regression
3. Ridge Logistic Regression
4. Lasso Logistic Regression

How to decide which approach is best, and which tuning parameter value to use for each approach? Cross-validation or validation set approach.

What is an appropriate validation measure?

For classification without a probability or score:

- Misclassification rate:

$$\frac{\#\text{test samples misclassified}}{\text{total \# of test samples}}$$

What is an appropriate validation measure?

For probablistic classification

- Can still use misclassification rate.
- Like in continuous regression could use SSE:

$$\sum_{i \in \text{test}} (y_i - \hat{p}_i)^2$$

- Often preferable to use "predictive [log]likelihood":

$$-\log \left[\prod_{i \in \text{test}} \hat{p}_i^{y_i} (1 - \hat{p}_i)^{1-y_i} \right]$$

- Can also use ROC-curve-based metric (eg. AUC)

Remember though; all of these must be conducted on a separate validation set.

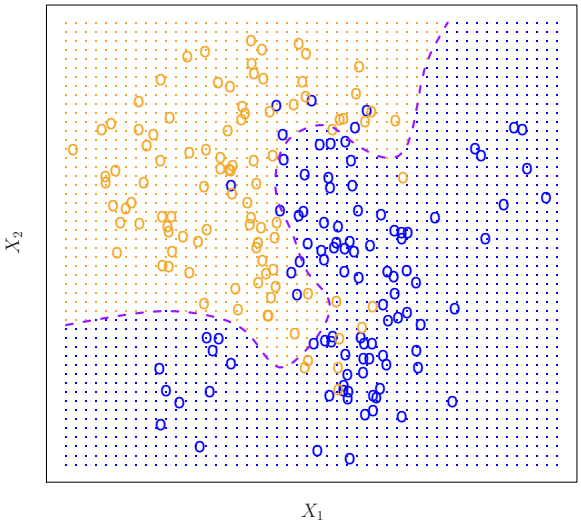
Example in R: Lasso Logistic Regression

```
xtr <- matrix(rnorm(1000*20),ncol=20)
beta <- c(rep(1,5),rep(0,15))
ytr <- 1*((xtr%*%beta + .5*rnorm(1000)) >= 0)
cv.out <- cv.glmnet(xtr, ytr, family="binomial", alpha=1)
plot(cv.out)
```

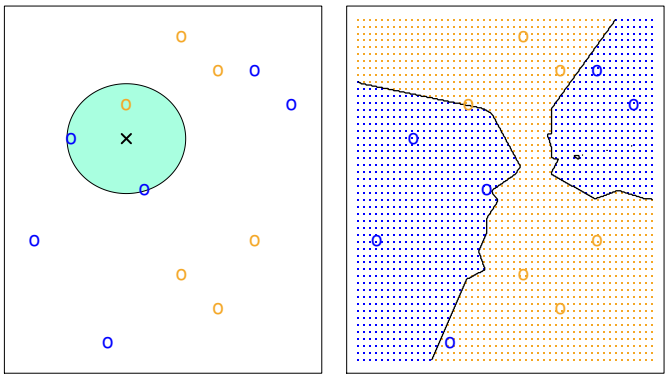

K-Nearest Neighbors

- Can I take a totally non-parametric (model-free) approach to classification?
- K-nearest neighbors:
  1. Identify the K observations whose X values are closest to the observation at which we want to make a prediction.
  2. Classify the observation of interest to the most frequent class label of those K nearest neighbors.

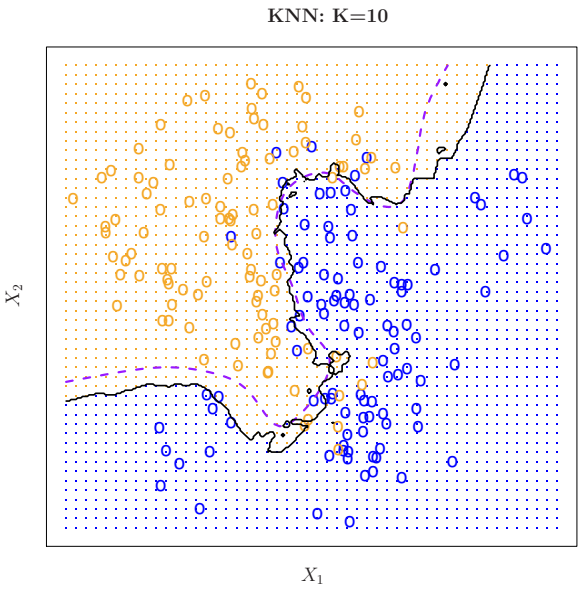
K-Nearest Neighbors



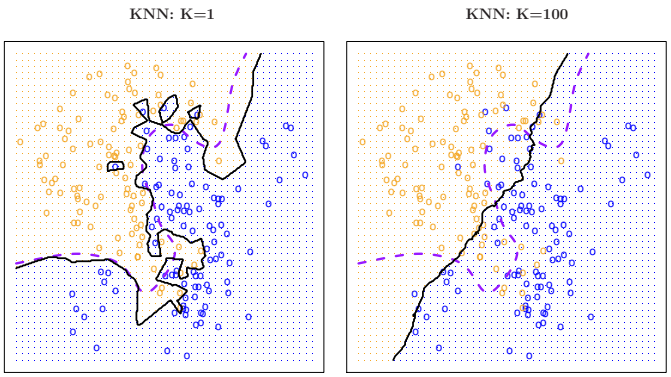
K-Nearest Neighbors



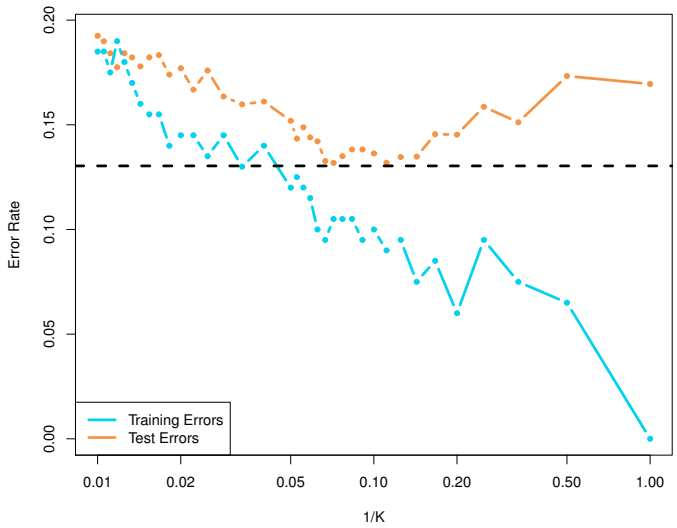
K-Nearest Neighbors



K-Nearest Neighbors



K-Nearest Neighbors



K-Nearest Neighbors

- Simple, intuitive, model-free.
- Good option when p is very small.
- Curse of dimensionality: when p is large, no neighbors are "near". All observations are close to the boundary.
- Do not use in high dimensions!

K-Nearest Neighbors

- Simple, intuitive, model-free.
- Good option when p is very small.
- Curse of dimensionality: when p is large, no neighbors are "near". All observations are close to the boundary.
- Do not use in high dimensions!
  - unless, you use it with a dimension reduction approach

Bayes-Based Classifiers

Suppose rather than knowing $P(y = j|x)$...

we have information on $f_j(x) = P(x|y = j)$, the feature distribution within each class

How do we use this to make predictions?

Bayes-Based Classifiers

Suppose rather than knowing $P(y = j|x)$...

we have information on $f_j(x) = P(x|y = j)$, the feature distribution within each class

How do we use this to make predictions?

Using Bayes Theorem:

$$P(y = j|x) = \frac{f_j(x)\pi_j}{\sum_k f_k(x)\pi_k}$$

here $\pi_k = P(y = k)$ is the prior probability of class k.

Estimating the Rule

To apply Bayes Theorem

$$P(y = j|x) = \frac{f_j(x)\pi_j}{\sum_k f_k(x)\pi_k}$$

we need

- $f_k(x)$ for $k = 1, \ldots, K$
- π_k for $k = 1, \ldots, K$

Estimating π_k

π_k is generally simple to estimate

- If your data are a random sample; then can use the sample proportion

$$\hat{\pi}_k = \frac{\#\{y_i = k\}}{n}$$

- Otherwise can use outside information (eg. historical data)

If you change population proportions; it is easy to adjust the rule.

Estimating $f_k(x)$

Estimate of $f_k(x) = P(x|y = k)$ is more difficult.

This is a density estimation problem.

The tools we discuss for this break down into 3 general categories

- flexible, non-parametric estimates
- parametric estimates
- shrunken parametric estimates

The above are ordered (more-or-less) by where they fall on bias/variance spectrum:

more flexible $\rightarrow$ less bias/more variance

Parametric $f_k(x)$ Estimate

Most well known estimator of this type is Linear/Quadratic Discriminant Analysis:

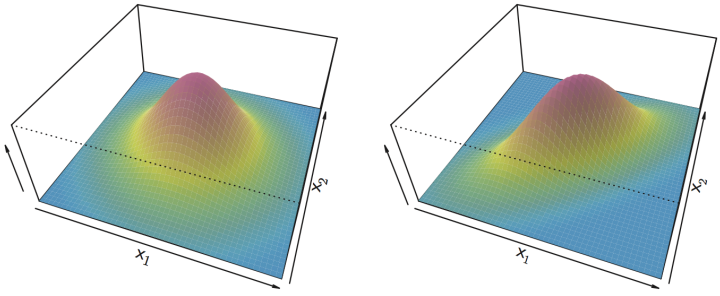
Here we assume that $f_k(x)$ is Gaussian density, $N(\mu_k, \Sigma_k)$

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Parametric $f_k(x)$ Estimate

Most well known estimator of this type is Linear/Quadratic Discriminant Analysis:

Here we assume that $f_k(x)$ is Gaussian density, $N(\mu_k, \Sigma_k)$



21 / 50

Discriminant Analysis

There are three main types of unpenalized discriminant analysis:

- Quadratic (QDA)
- Linear (LDA)
- Diagonal (DDA)

These make different assumptions on the covariance structure:

- QDA makes no assumptions
- LDA assumes a pooled variance $\Sigma = \Sigma_k$ for all k
- DDA assumes a pooled variance; and further that Σ is diagonal (i.e. no correlation among covariates!)

Discriminant Analysis

Why would we choose DDA over QDA?

Remember, *flexibility* comes at a price!

QDA will have the least bias; but has many more parameters to estimate

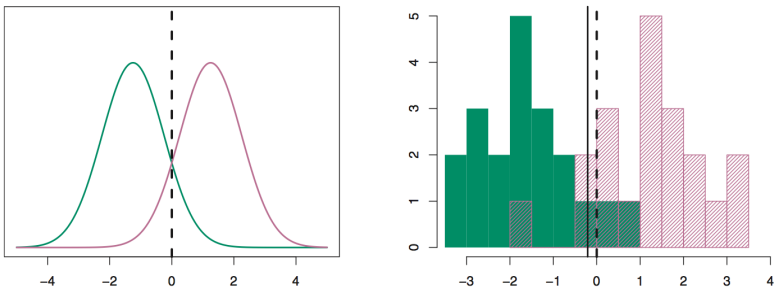
Often good estimates of the correlation don't improve classifications much

DDA takes into account the scale of each feature, but trades a bit of bias for potentially a large reduction in variance

LDA for $p = 1$

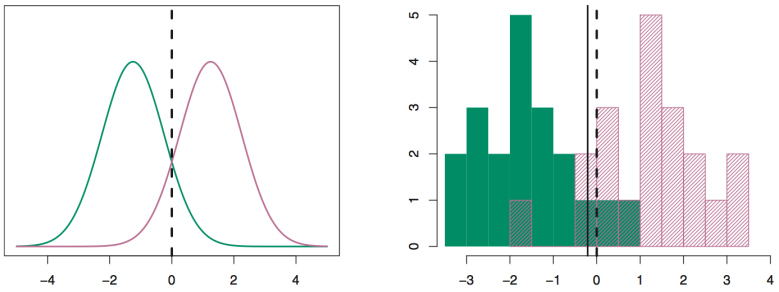
Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

LDA for $p = 1$



24 / 50

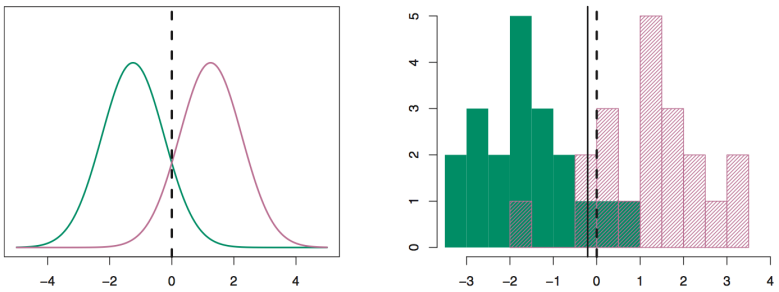
LDA for $p = 1$



- To make this work, we need to estimate the parameters. The ML estimates are given by $\hat{\pi}_k = n_k/n$ and

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

LDA for $p = 1$



- To make this work, we need to estimate the parameters. The ML estimates are given by $\hat{\pi}_k = n_k/n$ and

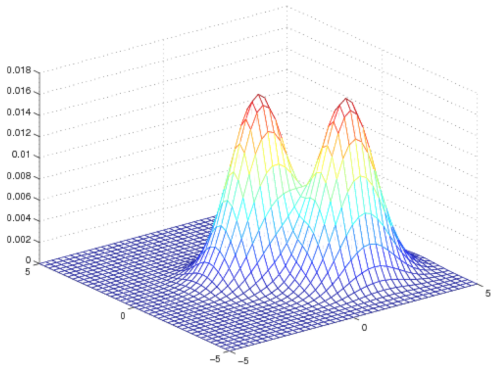
$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

- The picture is very similar if $K > 2$...or if $p > 1$

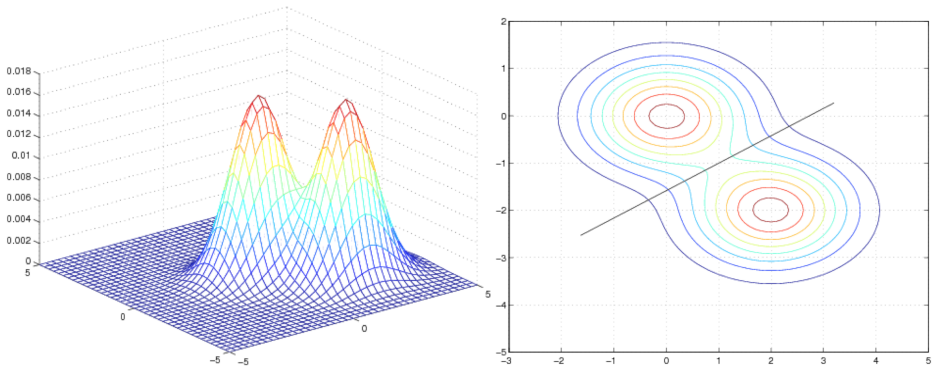
LDA for $p > 1$

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

LDA for $p > 1$

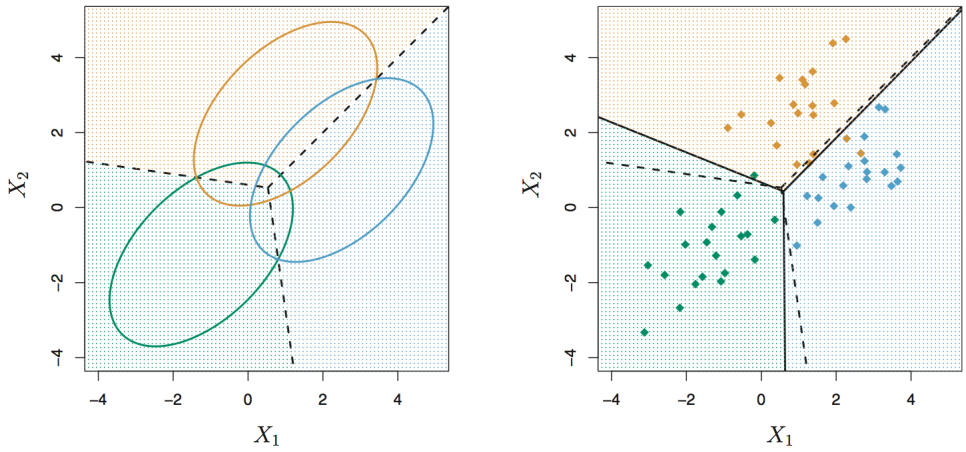


LDA for $p > 1$



LDA for $p > 1$

LDA for $p > 1$

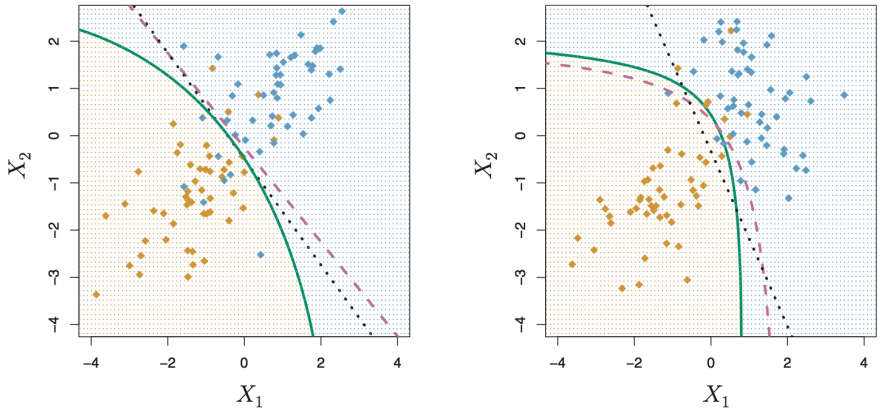


QDA vs LDA

The level-curves for each class look identical with LDA;

QDA allows for different classes to have differently shaped ellipsoids...

Results in non-linear decision boundaries (quadratic in fact)



DDA

For DDA...

- level curves are spheres (not ellipsoids).
- decision boundaries are still linear
- sometimes called *naive bayes* (that doesn't mean it's bad though!)
- with $\pi_k = \frac{1}{K}$ for all k, and equal variances (ie. $\Sigma = \sigma I$); this is just the *nearest centroid* classifier

-DA vs logistic regression

Discriminant Analysis model can actually be rewritten as multinomial logistic models:

Beginning with

$$P(y = j|x) = \frac{f_j(x)\pi_j}{\sum_k f_k(x)\pi_k}$$

and

$$f_k(x) \propto \exp\left[-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)\right]$$

substituting and simplifying we get

$$P(y = j|x) = \frac{e^{\eta_j}}{\sum_k e^{\eta_k}}$$

-DA vs logistic regression

$$P(y = j|x) = \frac{e^{\eta_j}}{\sum_k e^{\eta_k}}$$

where

$$\eta_k = \beta_0 + x^\top \beta + x^\top \Sigma_k^{-1} x$$

This is just a multinomial logistic model with quadratic terms and interactions.

In particular for LDA (where $\Sigma_k = \Sigma$ is pooled) we have cancellation and get

$$\eta_k = \beta_0 + x^\top \beta$$

Simply a linear logistic model.

Shrunken Parametric Estimates

Sometimes the optimal bias/variance tradeoff is between two parametric classes.

For example: We may not have the data to estimate completely different covariance matrices for each class (i.e. QDA); but we may not want to use identical covariance matrices.

In this case we can take a weighted combination of our estimates. This is called regularized discriminant analysis.

This is a type of *shrunken parametric estimator*.

Regularized Discriminant Analysis

For shrinking between QDA/LDA we use:

$$\hat{\Sigma}_k^{RDA} = \lambda \hat{\Sigma}^{LDA} + (1-\lambda)\hat{\Sigma}_k^{QDA}$$

For shrinking between LDA and Naive Bayes we use

$$\hat{\Sigma}^{RDA} = \lambda \hat{\Sigma}^{LDA} + (1-\lambda)\hat{\Sigma}^{NB}$$

λ is a tuning parameter, and is generally selected via CV

DA in High Dimensions

All of the Discriminant Analysis techniques discussed so far use all the features.

For high dimensional problems this will lead to over-fitting

One popular solution is to shrink each class-mean estimate $\hat{\mu}_k$ towards the overall mean $\hat{\mu}$ using element-wise soft-thresholding

This method is called Nearest Shrunken Centroids (though it should probably more appropriately be "nearest shrunken DDA")

Regularized DA Methods

- Recall that in PAM, $\hat{\mu}_{jk}$'s are soft thresholded towards the common mean $\hat{\mu}_{j.}$

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Regularized DA Methods

- Recall that in PAM, $\hat{\mu}_{jk}$'s are soft thresholded towards the common mean $\hat{\mu}_{j.}$
- Alternatively, μ_{jk}'s can be penalized
  - towards zero, using a lasso penalty

$$\sum_j \sum_k |\mu_{jk}|,$$

Regularized DA Methods

- Recall that in PAM, $\hat{\mu}_{jk}$'s are soft thresholded towards the common mean $\hat{\mu}_j$.
- Alternatively, μ_{jk}'s can be penalized
  - towards zero, using a lasso penalty

$$\sum_j \sum_k |\mu_{jk}|,$$

  - or towards each other, using a fused lasso penalty

$$\sum_j \sum_{k,k'} |\mu_{jk} - \mu_{jk'}|.$$

Both of these are implemented in R-package `penalizedLDA`.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Regularized DA Methods

- Recall that in PAM, $\hat{\mu}_{jk}$'s are soft thresholded towards the common mean $\hat{\mu}_j$.
- Alternatively, μ_{jk}'s can be penalized
  - towards zero, using a lasso penalty

$$\sum_j \sum_k |\mu_{jk}|,$$

  - or towards each other, using a fused lasso penalty

$$\sum_j \sum_{k,k'} |\mu_{jk} - \mu_{jk'}|.$$

Both of these are implemented in R-package `penalizedLDA`.

- Another option, which is especially helpful when using QDA is to penalize the covariance matrices Σ_k (or their inverses).

Let's Try It Out in R!

Chapter 4 R Lab
www.statlearning.com

Support Vector Machines

- Developed in around 1995.

Support Vector Machines

- Developed in around 1995.
- Touted as "overcoming the curse of dimensionality."

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Support Vector Machines

- Developed in around 1995.
- Touted as "overcoming the curse of dimensionality."
- Does not (automatically) overcome the curse of dimensionality!

Support Vector Machines

- Developed in around 1995.
- Touted as "overcoming the curse of dimensionality."
- Does not (automatically) overcome the curse of dimensionality!
- Fundamentally and numerically very similar to logistic regression.

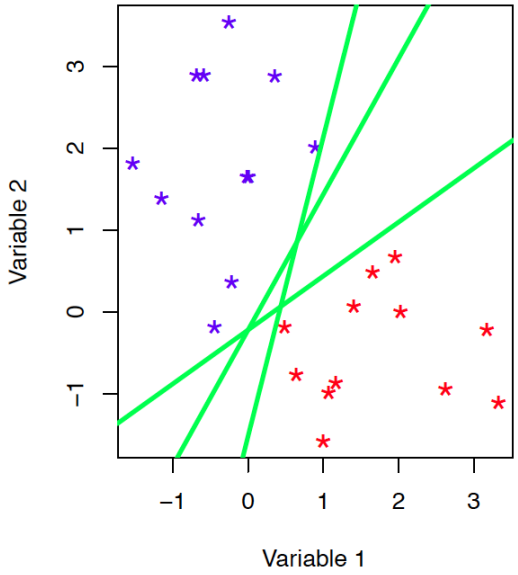
Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Support Vector Machines

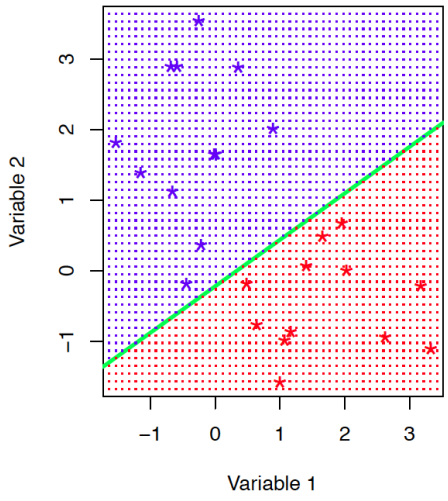
- Developed in around 1995.
- Touted as "overcoming the curse of dimensionality."
- Does not (automatically) overcome the curse of dimensionality!
- Fundamentally and numerically very similar to logistic regression.
- But, it is a nice idea.

36 / 50

Separating Hyperplane

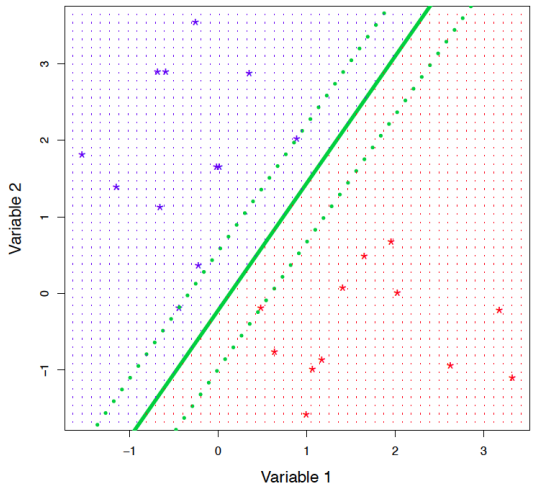


Classification Via a Separating Hyperplane



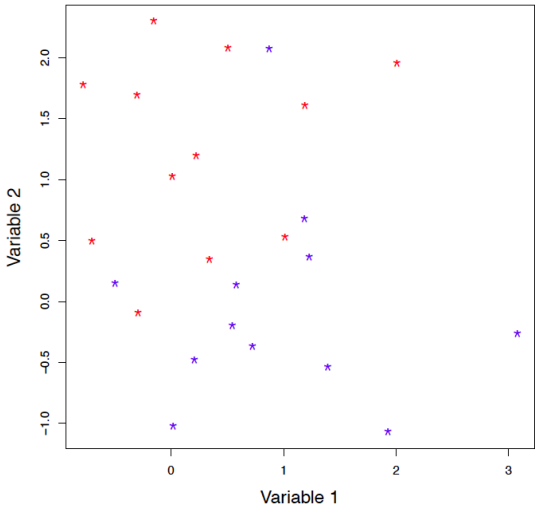
Blue class if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > c$; red class otherwise.

Maximal Separating Hyperplane

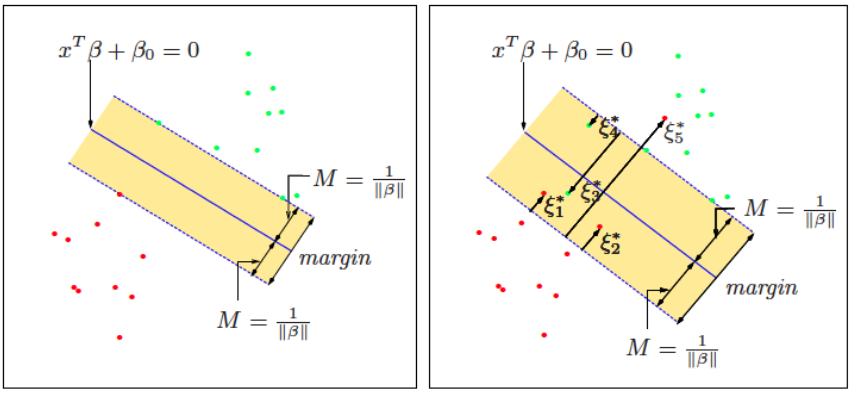


Note that only a few observations are on the margin: these are the support vectors.

What if There is No Separating Hyperplane?



Support Vector Classifier: Allow for Violations



Support Vector Machine

- The support vector machine is just like the support vector classifier, but it elegantly allows for non-linear expansions of the variables: "non-linear kernels".

Support Vector Machine

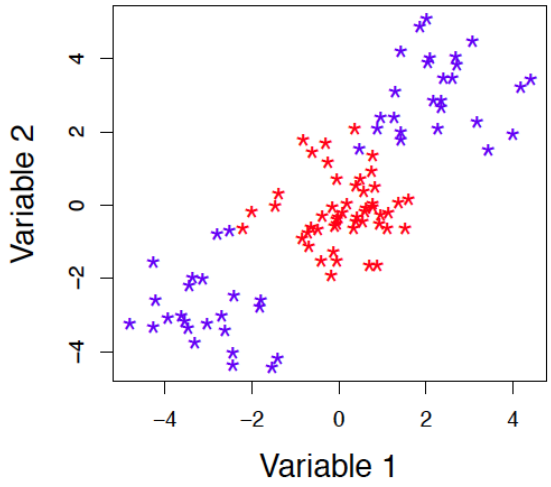
- The support vector machine is just like the support vector classifier, but it elegantly allows for non-linear expansions of the variables: "non-linear kernels".
- However, linear regression, logistic regression, and other classical statistical approaches can also be applied to non-linear functions of the variables.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Support Vector Machine

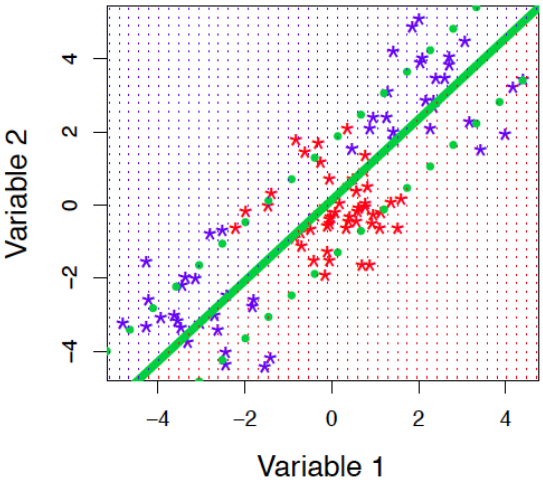
- The support vector machine is just like the support vector classifier, but it elegantly allows for non-linear expansions of the variables: "non-linear kernels".
- However, linear regression, logistic regression, and other classical statistical approaches can also be applied to non-linear functions of the variables.
- For historical reasons, SVMs are more frequently used with non-linear expansions as compared to other statistical approaches.

Non-Linear Class Structure



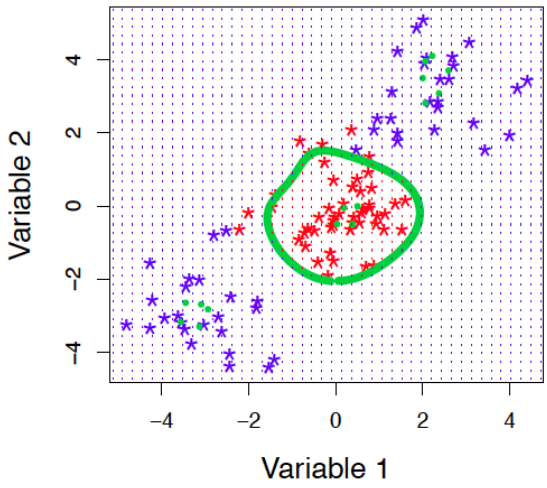
This will be hard for a linear classifier!

Try a Support Vector Classifier



Uh-oh!!

Support Vector Machine



Much Better.

Is A Non-Linear Kernel Better?

Is A Non-Linear Kernel Better?

- Yes, if the true decision boundary between the classes is non-linear, and you have enough observations (relative to the number of features) to accurately estimate the decision boundary.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

Is A Non-Linear Kernel Better?

- Yes, if the true decision boundary between the classes is non-linear, and you have enough observations (relative to the number of features) to accurately estimate the decision boundary.
- No, if you are in a very high-dimensional setting such that estimating a non-linear decision boundary is hopeless.

SVM vs Other Classification Methods

- The main difference between SVM and other classification methods (e.g. logistic regression) is the loss function used to assess the "fit":

$$\sum_{i=1}^{n} L(f(x_i), y_i)$$

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

SVM vs Other Classification Methods

- The main difference between SVM and other classification methods (e.g. logistic regression) is the loss function used to assess the "fit":

$$\sum_{i=1}^{n} L(f(x_i), y_i)$$

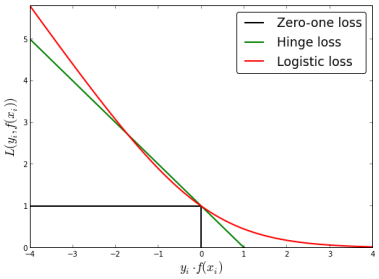
- Zero-one loss: $I(f(x_i) = y_i)$, where $I()$ is the indicator function. Not continuous, so hard to work with!!
- Hinge loss: $\max(0, 1 - f(x_i)y_i)$
- Logistic loss: $\log(1 + \exp f(x_i)y_i$

SVM vs Other Classification Methods

- The main difference between SVM and other classification methods (e.g. logistic regression) is the loss function used to assess the "fit":

$$\sum_{i=1}^{n} L(f(x_i), y_i)$$

- Zero-one loss: $I(f(x_i) = y_i)$, where $I()$ is the indicator function. Not continuous, so hard to work with!!
- Hinge loss: $\max(0, 1 - f(x_i)y_i)$
- Logistic loss: $\log(1 + \exp f(x_i)y_i$



SVM vs Logistic Regression

SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".

SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".
- The "kernel trick" makes things computationally easier, but it does not remove the danger of overfitting.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".
- The "kernel trick" makes things computationally easier, but it does not remove the danger of overfitting.
- SVM uses a non-linear kernel... but could do that with logistic or linear regression too!

SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".
- The "kernel trick" makes things computationally easier, but it does not remove the danger of overfitting.
- SVM uses a non-linear kernel... but could do that with logistic or linear regression too!
- A disadvantage of SVM (compared to, e.g. logistic regression) is that it does not provide a measure of uncertainty: cases are "classified" to belong to one of the two classes.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".
- The "kernel trick" makes things computationally easier, but it does not remove the danger of overfitting.
- SVM uses a non-linear kernel... but could do that with logistic or linear regression too!
- A disadvantage of SVM (compared to, e.g. logistic regression) is that it does not provide a measure of uncertainty: cases are "classified" to belong to one of the two classes.

48 / 50

Let's Try It Out in R!

Chapter 9 R Lab
www.statlearning.com

In High Dimensions...

In High Dimensions...

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

In High Dimensions...

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.
- This tuning parameter is like a ridge penalty, both mathematically and conceptually. The SVM decision rule involves all of the variables (the SVM problem can be written as a ridge problem but with the Hinge loss).

In High Dimensions...

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.
- This tuning parameter is like a ridge penalty, both mathematically and conceptually. The SVM decision rule involves all of the variables (the SVM problem can be written as a ridge problem but with the Hinge loss).
- Can get a sparse SVM using a lasso penalty; this yields a decision rule involving only a subset of the features.

Logistic Regression
K-Nearest Neighbors
Bayes-Based Classifiers
SVM

In High Dimensions...

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.
- This tuning parameter is like a ridge penalty, both mathematically and conceptually. The SVM decision rule involves all of the variables (the SVM problem can be written as a ridge problem but with the Hinge loss).
- Can get a sparse SVM using a lasso penalty; this yields a decision rule involving only a subset of the features.
- Logistic regression and other classical statistical approaches could be used with non-linear expansions of features. But this makes high-dimensionality issues worse.

50 / 50