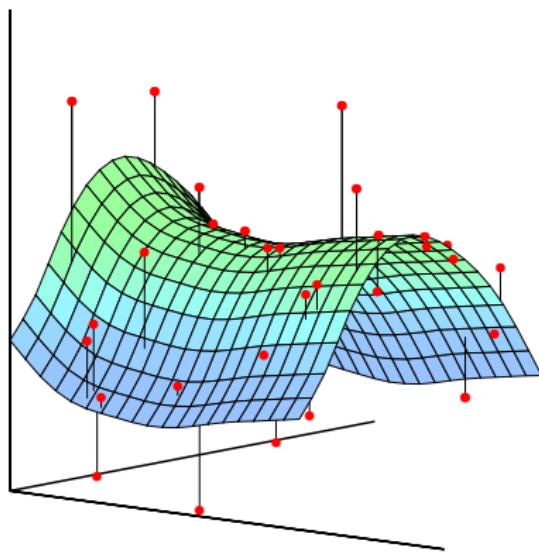


# Statistical Learning



*Trevor Hastie and Robert Tibshirani*

## Statistics in the news

### How IBM built Watson, its *Jeopardy*-playing supercomputer by Dawn Kawamoto DailyFinance

02/08/2011



**Learning from its mistakes** According to David Ferrucci (PI of Watson DeepQA technology for IBM Research), Watson's software is wired for more than handling natural language processing.

*“It’s machine learning allows the computer to become smarter as it tries to answer questions — and to learn as it gets them right or wrong.”*

## For Today's Graduate, Just One Word: Statistics

By STEVE LOHR  
Published: August 5, 2009

MOUNTAIN VIEW, Calif. — At Harvard, Carrie Grimes majored in anthropology and archaeology and ventured to places like Honduras, where she studied Mayan settlement patterns by mapping where artifacts were found. But she was drawn to what she calls “all the computer and math stuff” that was part of the job.

[Enlarge This Image](#)



Thor Swift for The New York Times

Carrie Grimes, senior staff engineer at Google, uses statistical analysis of data to help improve the company's search engine.

### Multimedia



“People think of field archaeology as Indiana Jones, but much of what you really do is data analysis,” she said.

Now Ms. Grimes does a different kind of digging. She works at [Google](#), where she uses statistical analysis of mounds of data to come up with ways to improve its search engine.

Ms. Grimes is an Internet-age statistician, one of many who are changing the image of the profession as a place for dromish number nerds. They are finding themselves increasingly in demand — and even cool.

“I keep saying that the sexy job in the next 10 years will be statisticians,” said Hal Varian, chief economist at Google. “And I’m not kidding.”

SIGN IN TO  
RECOMMEND

SIGN IN TO  
E-MAIL

PRINT

REPRINTS

SHARE

ARTICLE TOOLS  
SPONSORED BY  
**Adam**  
NOW PLAYING  
IN SELECT THEATERS

## QUOTE OF THE DAY, NEW YORK TIMES, AUGUST 5, 2009

“I keep saying that the sexy job in the next 10 years will be statisticians. And I’m not kidding.”  
— HAL VARIAN, chief economist at Google.



# FiveThirtyEight

Nate Silver's Political Calculus

**90.9%**

+13.5 since Oct. 30

Chance of  
Winning

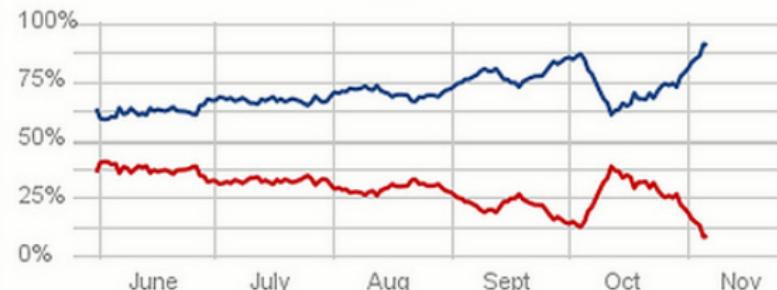
**9.1%**

-13.5 since Oct. 30



Click to [LOOK INSIDE!](#)

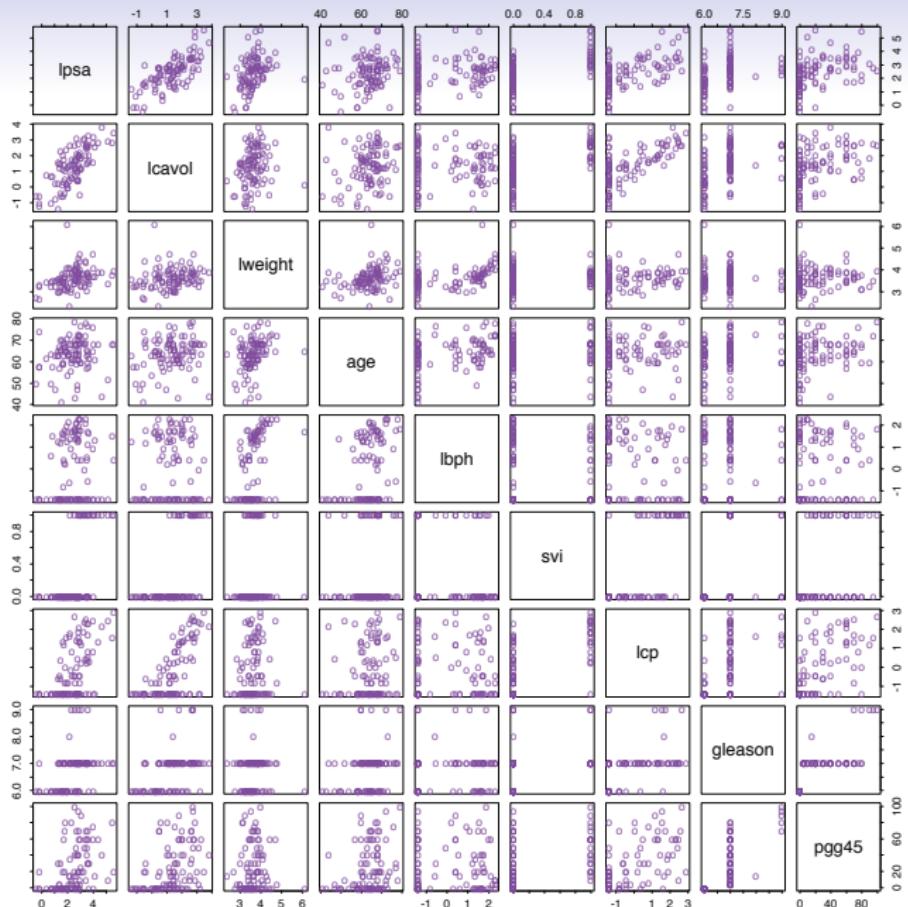
the signal and the noise  
and the noise and the  
noise and the noise and the  
noise and the noise and the  
why so many and predictions fail—but some don't the  
and the noise and the noise and the  
**nate silver noise**  
noise and the no



# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.

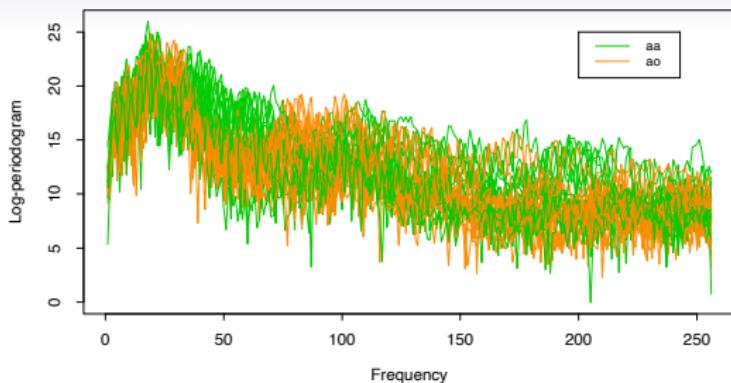




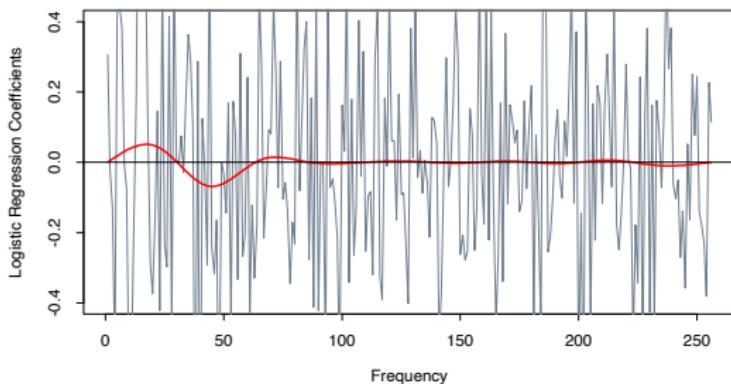
# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.

Phoneme Examples

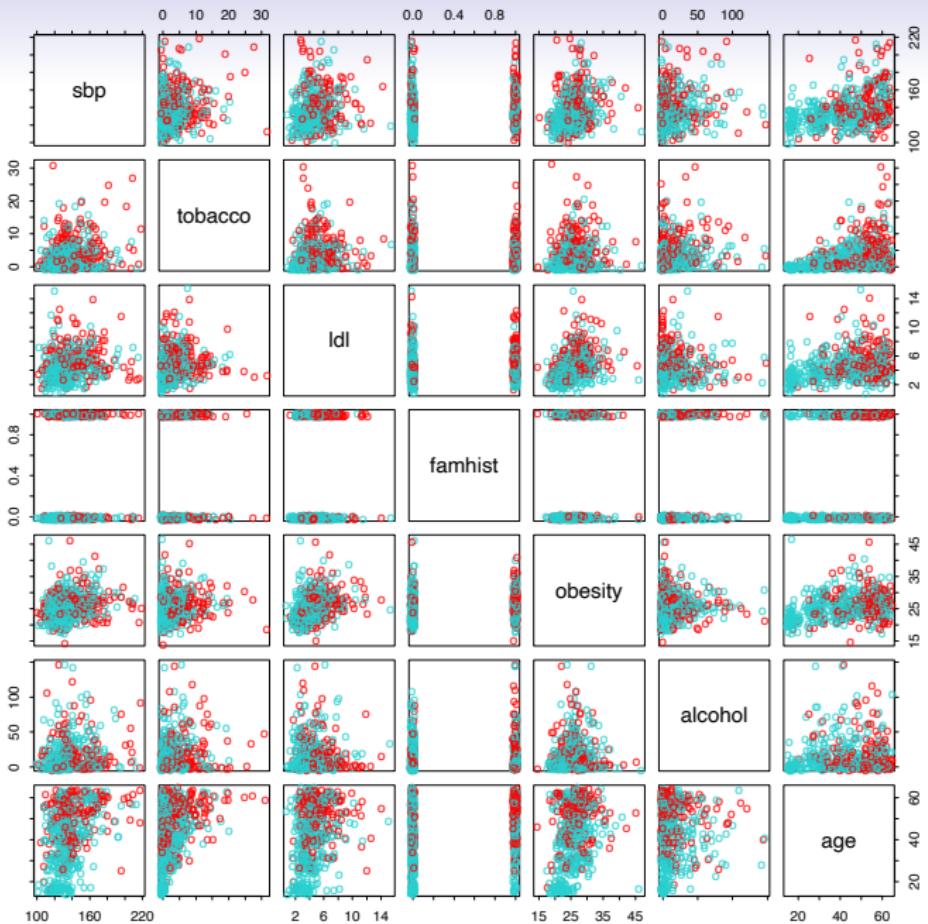


Phoneme Classification: Raw and Restricted Logistic Regression



# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.



# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.

## Spam Detection

- data from 4601 emails sent to an individual (named George, at HP labs, before 2000). Each is labeled as *spam* or *email*.
- goal: build a customized spam filter.
- input features: relative frequencies of 57 of the most commonly occurring words and punctuation marks in these email messages.

	george	you	hp	free	!	edu	remove
spam	0.00	2.26	0.02	0.52	0.51	0.01	0.28
email	1.27	1.27	0.90	0.07	0.11	0.29	0.01

*Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between **spam** and **email**.*

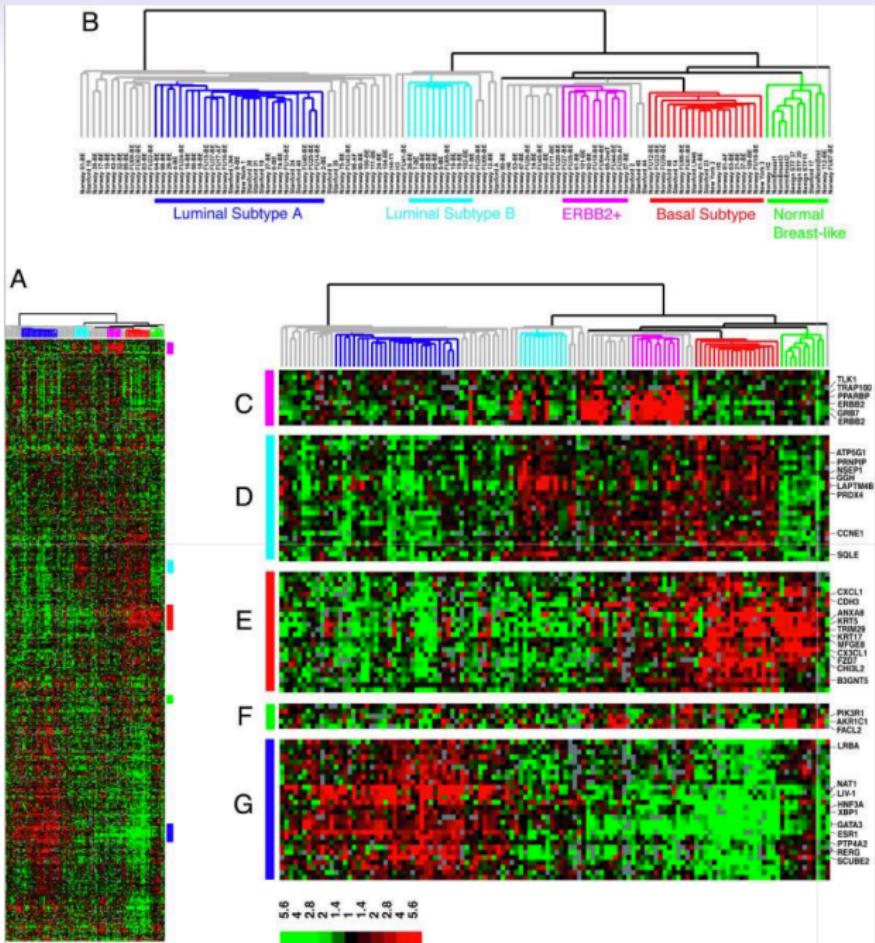
# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

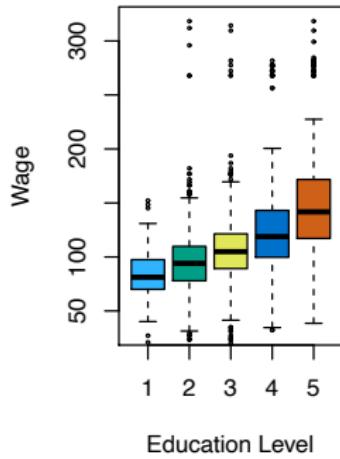
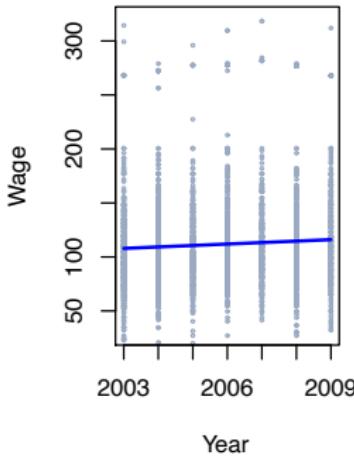
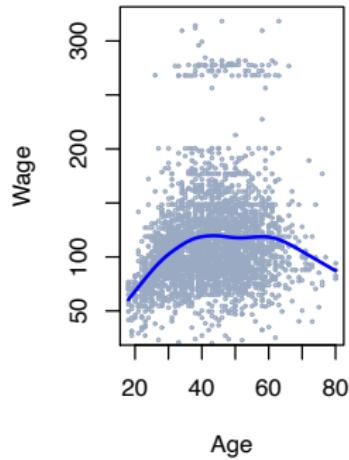
# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.



# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.

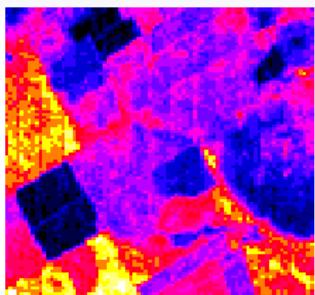


Income survey data for males from the central Atlantic region of the USA in 2009.

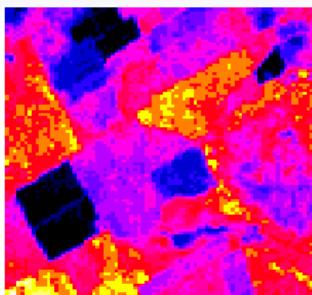
# Statistical Learning Problems

- Identify the risk factors for prostate cancer.
- Classify a recorded phoneme based on a log-periodogram.
- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements.
- Customize an email spam detection system.
- Identify the numbers in a handwritten zip code.
- Classify a tissue sample into one of several cancer classes, based on a gene expression profile.
- Establish the relationship between salary and demographic variables in population survey data.
- Classify the pixels in a LANDSAT image, by usage.

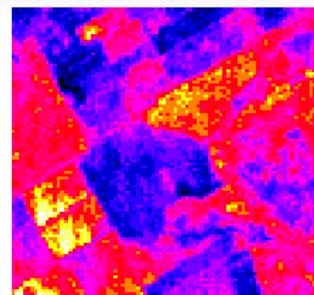
Spectral Band 1



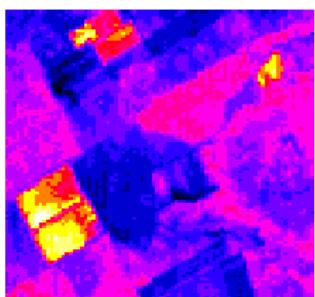
Spectral Band 2



Spectral Band 3



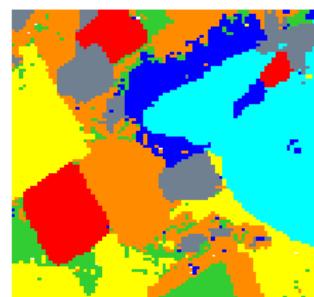
Spectral Band 4



Land Usage



Predicted Land Usage



$Usage \in \{red\ soil, cotton, vegetation\ stubble, mixture, gray\ soil, damp\ gray\ soil\}$

# The Supervised Learning Problem

*Starting point:*

- Outcome measurement  $Y$  (also called dependent variable, response, target).
- Vector of  $p$  predictor measurements  $X$  (also called inputs, regressors, covariates, features, independent variables).
- In the *regression problem*,  $Y$  is quantitative (e.g price, blood pressure).
- In the *classification problem*,  $Y$  takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample).
- We have training data  $(x_1, y_1), \dots, (x_N, y_N)$ . These are observations (examples, instances) of these measurements.

# Objectives

On the basis of the training data we would like to:

- Accurately predict unseen test cases.
- Understand which inputs affect the outcome, and how.
- Assess the quality of our predictions and inferences.

# Philosophy

- It is important to understand the ideas behind the various techniques, in order to know how and when to use them.
- One has to understand the simpler methods first, in order to grasp the more sophisticated ones.
- It is important to accurately assess the performance of a method, to know how well or how badly it is working [simpler methods often perform as well as fancier ones!]
- This is an exciting research area, having important applications in science, industry and finance.
- Statistical learning is a fundamental ingredient in the training of a modern *data scientist*.

# Unsupervised learning

- No outcome variable, just a set of predictors (features) measured on a set of samples.
- objective is more fuzzy — find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation.
- difficult to know how well you are doing.
- different from supervised learning, but can be useful as a pre-processing step for supervised learning.

## The Netflix prize

- competition started in October 2006. Training data is ratings for 18,000 movies by 400,000 Netflix customers, each rating between 1 and 5.
- training data is very sparse— about 98% missing.
- objective is to predict the rating for a set of 1 million customer-movie pairs that are missing in the training data.
- Netflix's original algorithm achieved a root MSE of 0.953. The first team to achieve a 10% improvement wins one million dollars.
- is this a supervised or unsupervised problem?

# Netflix Prize

**COMPLETED**[Home](#) | [Rules](#) | [Leaderboard](#) | [Update](#)

## Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top   leaders.

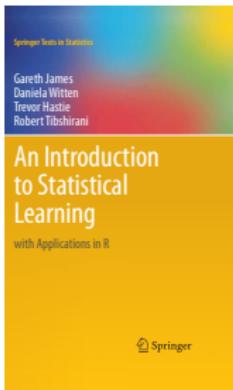
Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
<b>Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos</b>				
1	<a href="#">BellKor's Pragmatic Chaos</a>	0.8567	10.06	2009-07-26 18:18:28
2	<a href="#">The Ensemble</a>	0.8567	10.06	2009-07-26 18:38:22
3	<a href="#">Grand Prize Team</a>	0.8582	9.90	2009-07-10 21:24:40
4	<a href="#">Opera Solutions and Vandelay United</a>	0.8588	9.84	2009-07-10 01:12:31
5	<a href="#">Vandelay Industries !</a>	0.8591	9.81	2009-07-10 00:32:20
6	<a href="#">PragmaticTheory</a>	0.8594	9.77	2009-06-24 12:06:56
7	<a href="#">BellKor in BigChaos</a>	0.8601	9.70	2009-05-13 08:14:09
8	<a href="#">Dace</a>	0.8612	9.59	2009-07-24 17:18:43
9	<a href="#">Feeds2</a>	0.8622	9.48	2009-07-12 13:11:51
10	<a href="#">BigChaos</a>	0.8623	9.47	2009-04-07 12:33:59
11	<a href="#">Opera Solutions</a>	0.8623	9.47	2009-07-24 00:34:07
12	<a href="#">BellKor</a>	0.8624	9.46	2009-07-26 17:19:11

BellKor's Pragmatic Chaos wins, beating The Ensemble by a narrow margin.

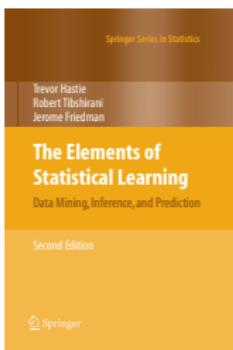
# Statistical Learning versus Machine Learning

- Machine learning arose as a subfield of Artificial Intelligence.
- Statistical learning arose as a subfield of Statistics.
- *There is much overlap* — both fields focus on supervised and unsupervised problems:
  - Machine learning has a greater emphasis on *large scale* applications and *prediction accuracy*.
  - Statistical learning emphasizes *models* and their interpretability, and *precision* and *uncertainty*.
- But the distinction has become more and more blurred, and there is a great deal of “cross-fertilization”.
- Machine learning has the upper hand in *Marketing!*

## *Course Texts*

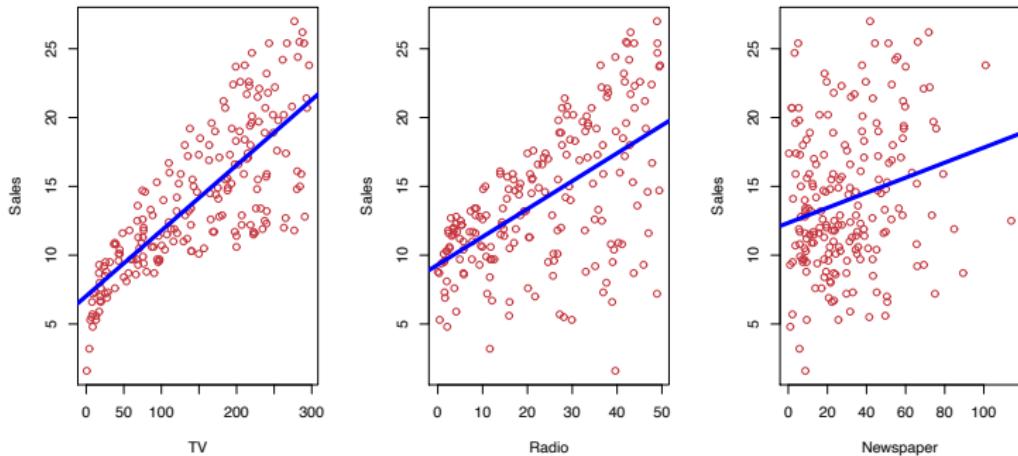


The course will cover most of the material in this Springer book (ISLR) published in 2013, which the instructors coauthored with Gareth James and Daniela Witten. Each chapter ends with an R lab, in which examples are developed. By January 1st, 2014, an electronic version of this book will be available for free from the instructors' websites.



This Springer book (ESL) is more mathematically advanced than ISLR; the second edition was published in 2009, and coauthored by the instructors and Jerome Friedman. It covers a broader range of topics. The book is available from Springer and Amazon, a free electronic version is available from the instructors' websites.

# What is Statistical Learning?



Shown are **Sales** vs **TV**, **Radio** and **Newspaper**, with a blue linear-regression line fit separately to each.

Can we predict **Sales** using these three?

Perhaps we can do better using a model

$$\text{Sales} \approx f(\text{TV}, \text{Radio}, \text{Newspaper})$$

## Notation

Here **Sales** is a *response* or *target* that we wish to predict. We generically refer to the response as  $Y$ .

**TV** is a *feature*, or *input*, or *predictor*; we name it  $X_1$ .

Likewise name **Radio** as  $X_2$ , and so on.

We can refer to the *input vector* collectively as

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

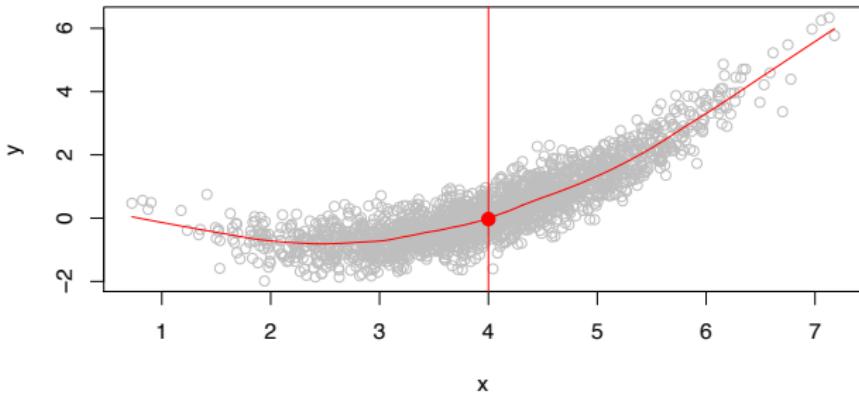
Now we write our model as

$$Y = f(X) + \epsilon$$

where  $\epsilon$  captures measurement errors and other discrepancies.

## What is $f(X)$ good for?

- With a good  $f$  we can make predictions of  $Y$  at new points  $X = x$ .
- We can understand which components of  $X = (X_1, X_2, \dots, X_p)$  are important in explaining  $Y$ , and which are irrelevant. e.g. **Seniority** and **Years of Education** have a big impact on **Income**, but **Marital Status** typically does not.
- Depending on the complexity of  $f$ , we may be able to understand how each component  $X_j$  of  $X$  affects  $Y$ .



Is there an ideal  $f(X)$ ? In particular, what is a good value for  $f(X)$  at any selected value of  $X$ , say  $X = 4$ ? There can be many  $Y$  values at  $X = 4$ . A good value is

$$f(4) = E(Y|X = 4)$$

$E(Y|X = 4)$  means *expected value* (average) of  $Y$  given  $X = 4$ .

This ideal  $f(x) = E(Y|X = x)$  is called the *regression function*.

## The regression function $f(x)$

- Is also defined for vector  $X$ ; e.g.

$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

## The regression function $f(x)$

- Is also defined for vector  $X$ ; e.g.  
$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$
- Is the *ideal* or *optimal* predictor of  $Y$  with regard to mean-squared prediction error:  $f(x) = E(Y|X = x)$  is the function that minimizes  $E[(Y - g(X))^2|X = x]$  over all functions  $g$  at all points  $X = x$ .

## The regression function $f(x)$

- Is also defined for vector  $X$ ; e.g.  
$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$
- Is the *ideal* or *optimal* predictor of  $Y$  with regard to mean-squared prediction error:  $f(x) = E(Y|X = x)$  is the function that minimizes  $E[(Y - g(X))^2|X = x]$  over all functions  $g$  at all points  $X = x$ .
- $\epsilon = Y - f(x)$  is the *irreducible* error — i.e. even if we knew  $f(x)$ , we would still make errors in prediction, since at each  $X = x$  there is typically a distribution of possible  $Y$  values.

## The regression function $f(x)$

- Is also defined for vector  $X$ ; e.g.  
$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$
- Is the *ideal* or *optimal* predictor of  $Y$  with regard to mean-squared prediction error:  $f(x) = E(Y|X = x)$  is the function that minimizes  $E[(Y - g(X))^2|X = x]$  over all functions  $g$  at all points  $X = x$ .
- $\epsilon = Y - f(x)$  is the *irreducible* error — i.e. even if we knew  $f(x)$ , we would still make errors in prediction, since at each  $X = x$  there is typically a distribution of possible  $Y$  values.
- For any estimate  $\hat{f}(x)$  of  $f(x)$ , we have

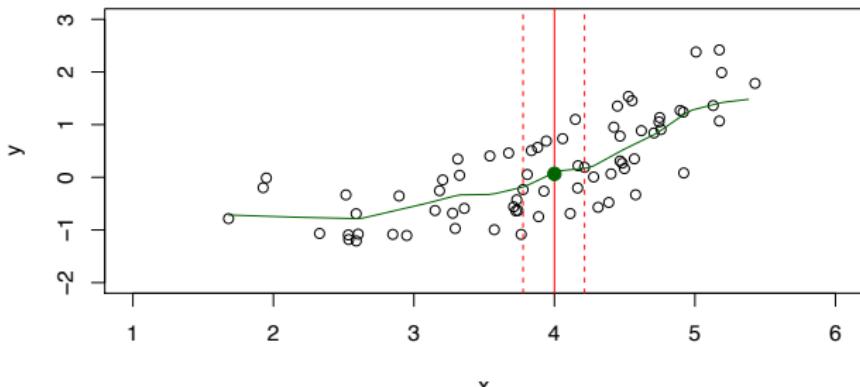
$$E[(Y - \hat{f}(X))^2|X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible}}$$

## How to estimate $f$

- Typically we have few if any data points with  $X = 4$  exactly.
- So we cannot compute  $E(Y|X = x)$ !
- Relax the definition and let

$$\hat{f}(x) = \text{Ave}(Y|X \in \mathcal{N}(x))$$

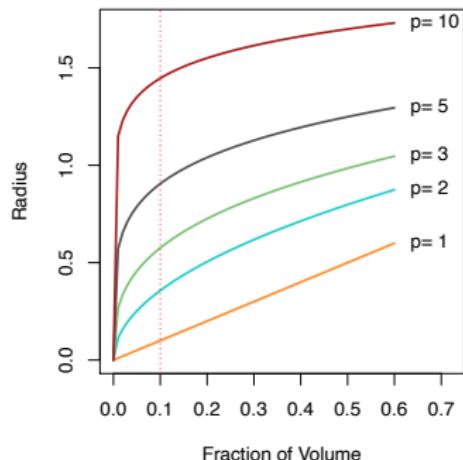
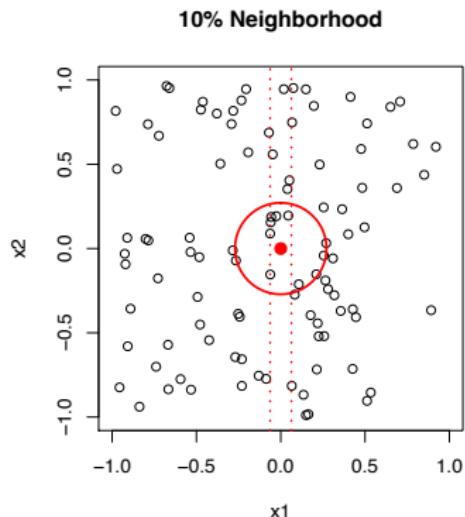
where  $\mathcal{N}(x)$  is some *neighborhood* of  $x$ .



- Nearest neighbor averaging can be pretty good for small  $p$ 
  - i.e.  $p \leq 4$  and large-ish  $N$ .
- We will discuss smoother versions, such as kernel and spline smoothing later in the course.

- Nearest neighbor averaging can be pretty good for small  $p$ 
  - i.e.  $p \leq 4$  and large-ish  $N$ .
- We will discuss smoother versions, such as kernel and spline smoothing later in the course.
- Nearest neighbor methods can be *lousy* when  $p$  is large.  
Reason: the *curse of dimensionality*. Nearest neighbors tend to be far away in high dimensions.
  - We need to get a reasonable fraction of the  $N$  values of  $y_i$  to average to bring the variance down—e.g. 10%.
  - A 10% neighborhood in high dimensions need no longer be local, so we lose the spirit of estimating  $E(Y|X = x)$  by local averaging.

# The curse of dimensionality



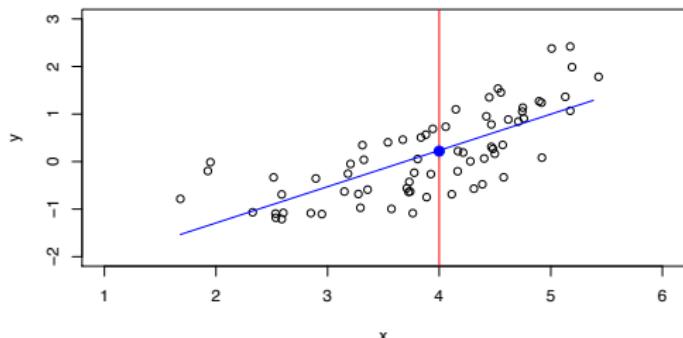
## Parametric and structured models

The *linear* model is an important example of a parametric model:

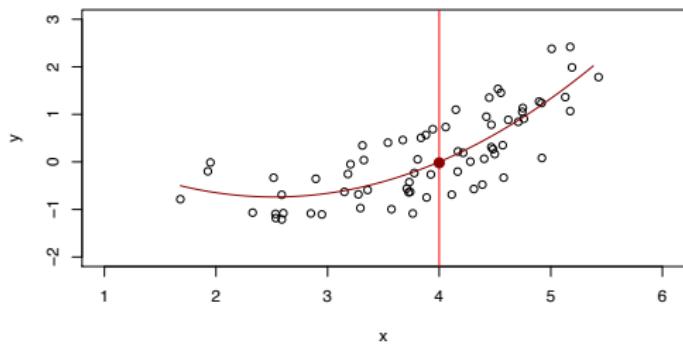
$$f_L(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p.$$

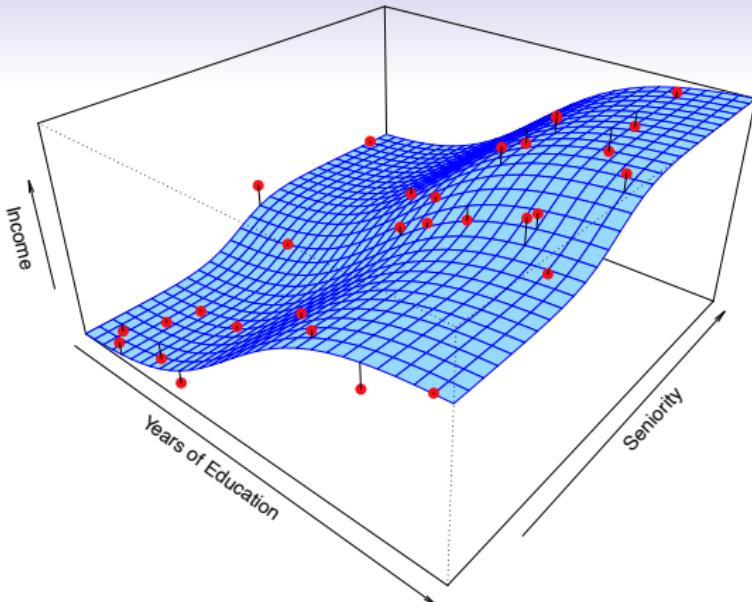
- A linear model is specified in terms of  $p + 1$  parameters  $\beta_0, \beta_1, \dots, \beta_p$ .
- We estimate the parameters by fitting the model to training data.
- Although it is *almost never correct*, a linear model often serves as a good and interpretable approximation to the unknown true function  $f(X)$ .

A linear model  $\hat{f}_L(X) = \hat{\beta}_0 + \hat{\beta}_1 X$  gives a reasonable fit here



A quadratic model  $\hat{f}_Q(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$  fits slightly better.

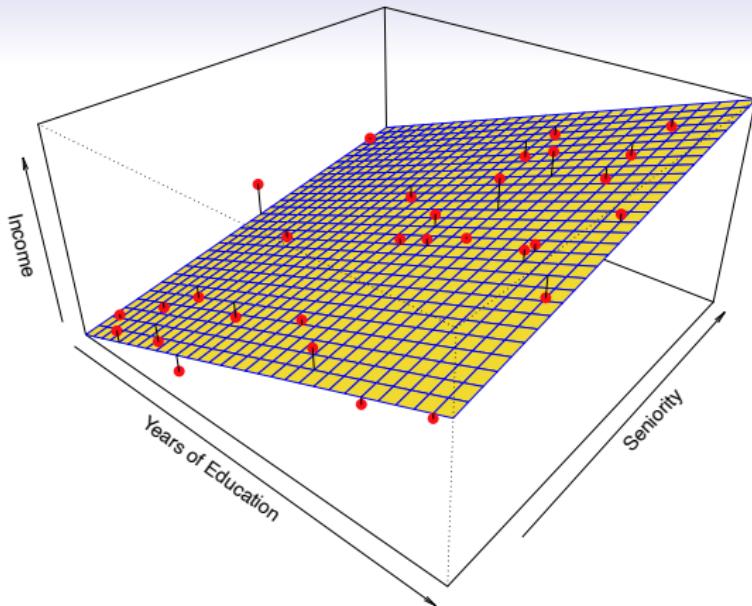




Simulated example. Red points are simulated values for `income` from the model

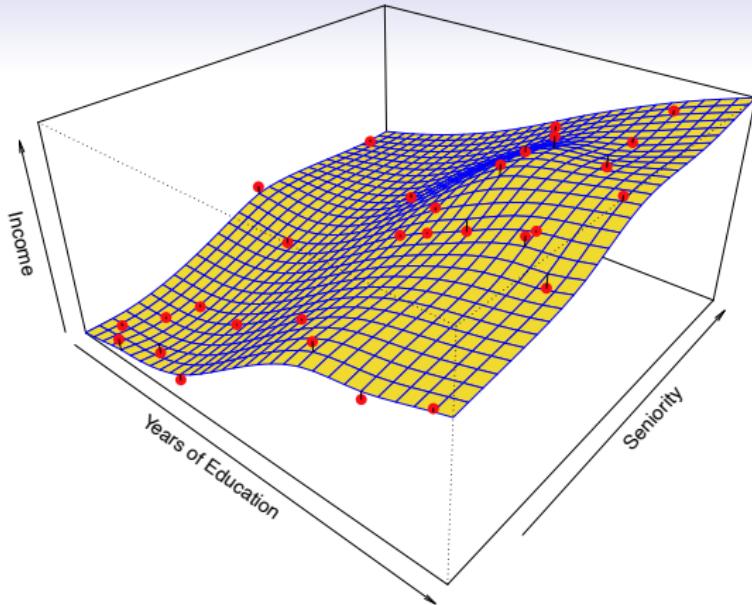
$$\text{income} = f(\text{education}, \text{seniority}) + \epsilon$$

$f$  is the blue surface.

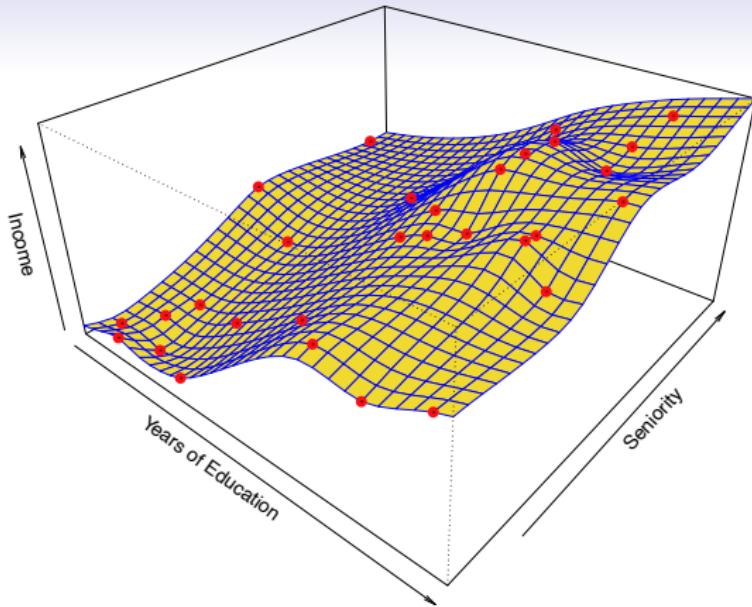


Linear regression model fit to the simulated data.

$$\hat{f}_L(\text{education}, \text{seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \text{education} + \hat{\beta}_2 \times \text{seniority}$$



More flexible regression model  $\hat{f}_S(\text{education}, \text{seniority})$  fit to the simulated data. Here we use a technique called a *thin-plate spline* to fit a flexible surface. We control the roughness of the fit (chapter 7).



Even more flexible spline regression model  
 $\hat{f}_S(\text{education}, \text{seniority})$  fit to the simulated data. Here the fitted model makes no errors on the training data! Also known as *overfitting*.

## Some trade-offs

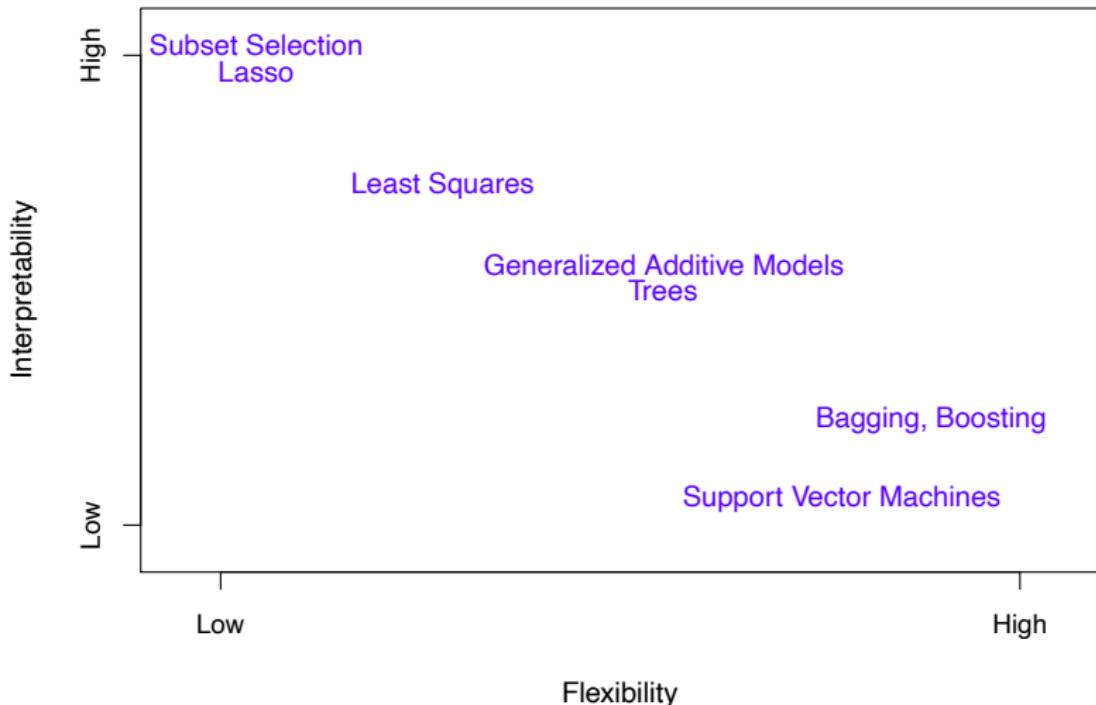
- Prediction accuracy versus interpretability.
  - Linear models are easy to interpret; thin-plate splines are not.

## Some trade-offs

- Prediction accuracy versus interpretability.
  - Linear models are easy to interpret; thin-plate splines are not.
- Good fit versus over-fit or under-fit.
  - How do we know when the fit is just right?

## Some trade-offs

- Prediction accuracy versus interpretability.
  - Linear models are easy to interpret; thin-plate splines are not.
- Good fit versus over-fit or under-fit.
  - How do we know when the fit is just right?
- Parsimony versus black-box.
  - We often prefer a simpler model involving fewer variables over a black-box predictor involving them all.



## Assessing Model Accuracy

Suppose we fit a model  $\hat{f}(x)$  to some training data  $\mathbf{Tr} = \{x_i, y_i\}_1^N$ , and we wish to see how well it performs.

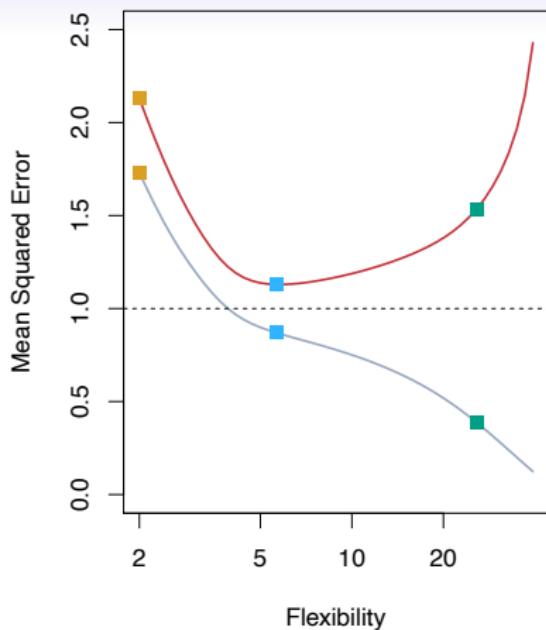
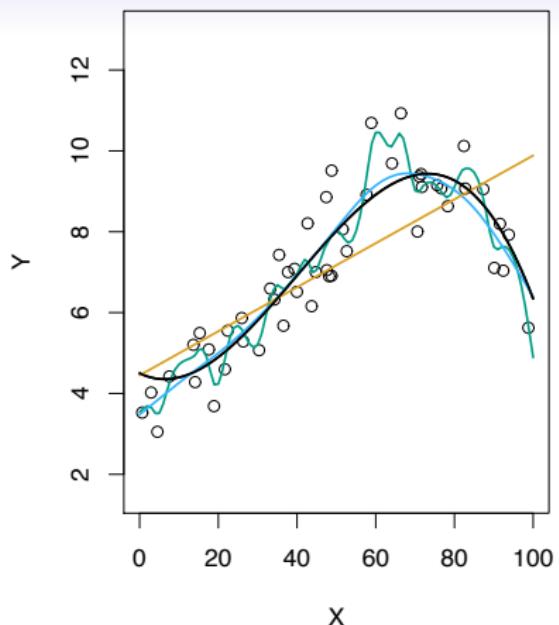
- We could compute the average squared prediction error over  $\mathbf{Tr}$ :

$$\text{MSE}_{\mathbf{Tr}} = \text{Ave}_{i \in \mathbf{Tr}} [y_i - \hat{f}(x_i)]^2$$

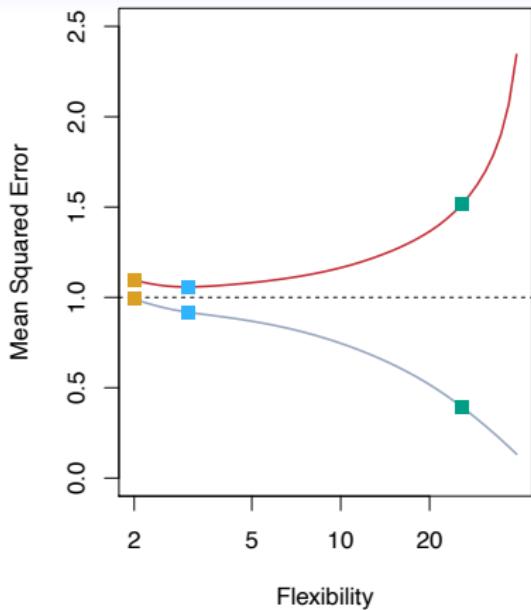
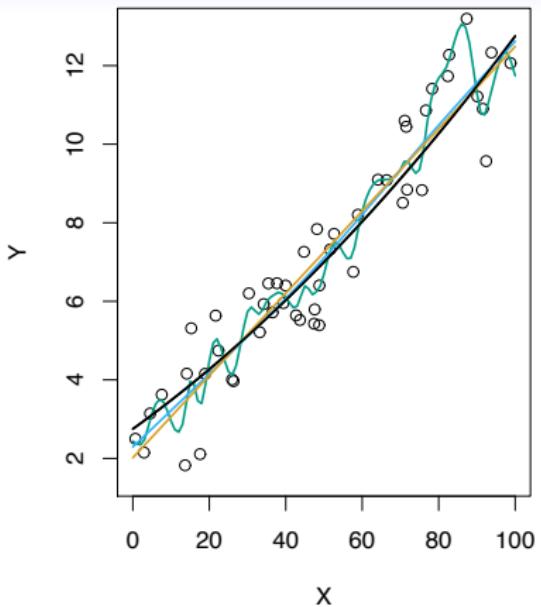
This may be biased toward more overfit models.

- Instead we should, if possible, compute it using fresh *test* data  $\mathbf{Te} = \{x_i, y_i\}_1^M$ :

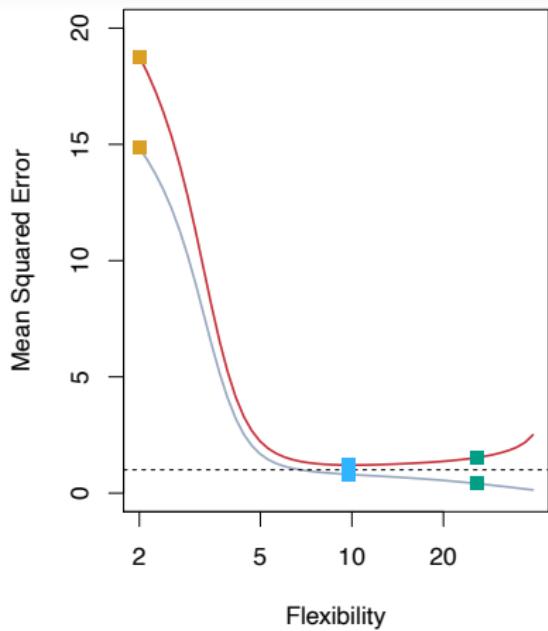
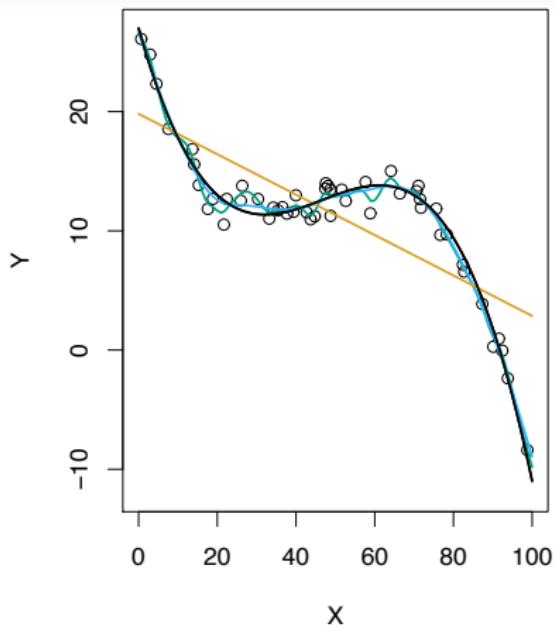
$$\text{MSE}_{\mathbf{Te}} = \text{Ave}_{i \in \mathbf{Te}} [y_i - \hat{f}(x_i)]^2$$



Black curve is truth. Red curve on right is  $MSE_{Te}$ , grey curve is  $MSE_{Tr}$ . Orange, blue and green curves/squares correspond to fits of different flexibility.



Here the truth is smoother, so the smoother fit and linear model do really well.



Here the truth is wiggly and the noise is low, so the more flexible fits do the best.

## Bias-Variance Trade-off

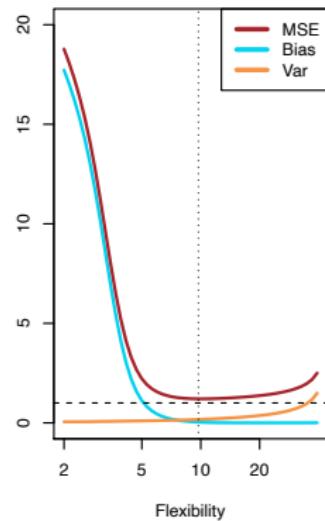
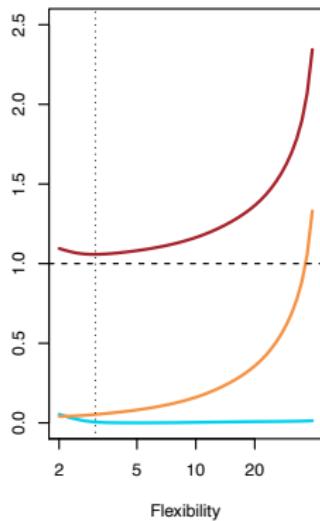
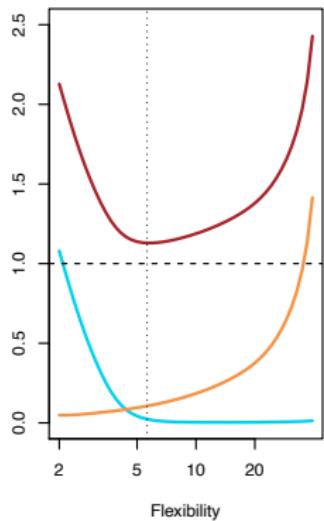
Suppose we have fit a model  $\hat{f}(x)$  to some training data  $\text{Tr}$ , and let  $(x_0, y_0)$  be a test observation drawn from the population. If the true model is  $Y = f(X) + \epsilon$  (with  $f(x) = E(Y|X = x)$ ), then

$$E \left( y_0 - \hat{f}(x_0) \right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

The expectation averages over the variability of  $y_0$  as well as the variability in  $\text{Tr}$ . Note that  $\text{Bias}(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$ .

Typically as the *flexibility* of  $\hat{f}$  increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off*.

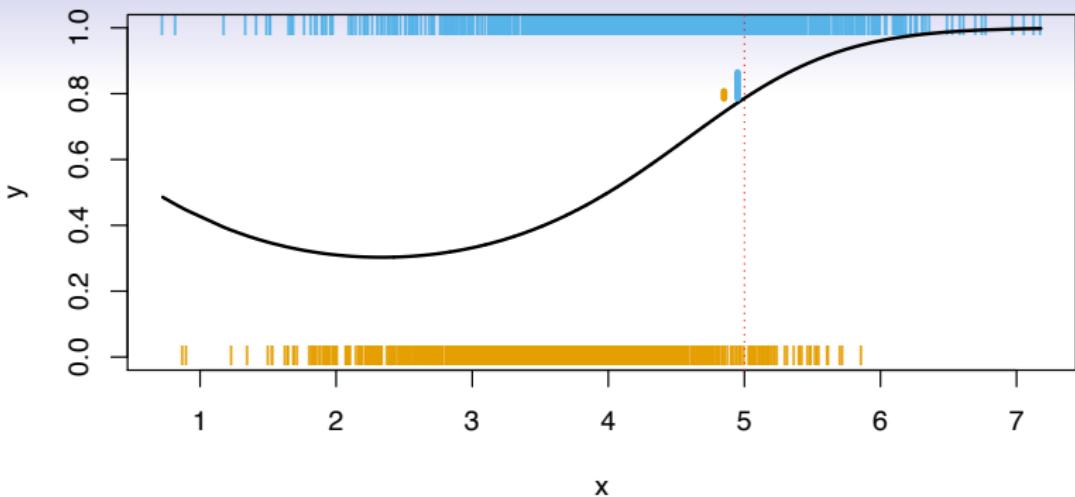
# Bias-variance trade-off for the three examples



# Classification Problems

Here the response variable  $Y$  is *qualitative* — e.g. email is one of  $\mathcal{C} = \{\text{spam}, \text{ham}\}$  ( $\text{ham}$ =good email), digit class is one of  $\mathcal{C} = \{0, 1, \dots, 9\}$ . Our goals are to:

- Build a classifier  $C(X)$  that assigns a class label from  $\mathcal{C}$  to a future unlabeled observation  $X$ .
- Assess the uncertainty in each classification
- Understand the roles of the different predictors among  $X = (X_1, X_2, \dots, X_p)$ .

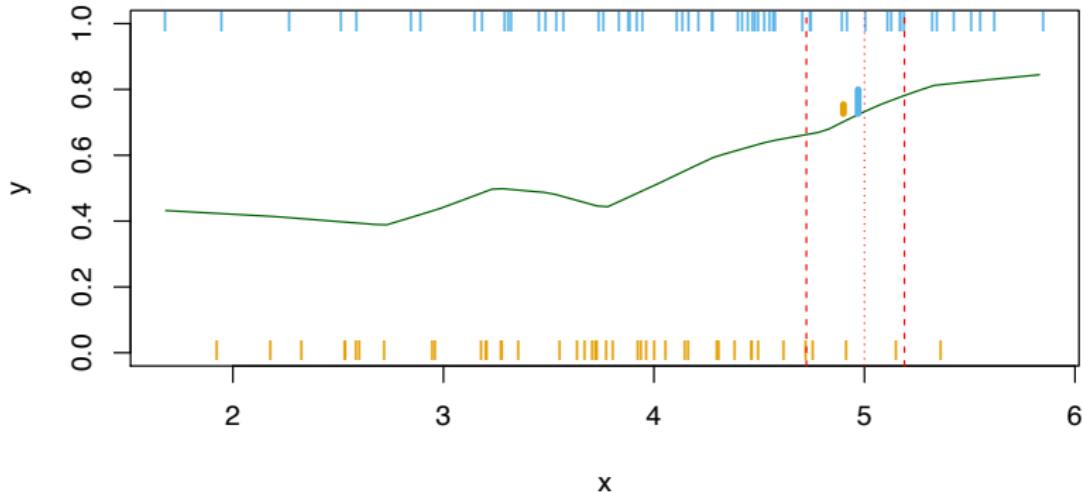


Is there an ideal  $C(X)$ ? Suppose the  $K$  elements in  $\mathcal{C}$  are numbered  $1, 2, \dots, K$ . Let

$$p_k(x) = \Pr(Y = k | X = x), \quad k = 1, 2, \dots, K.$$

These are the *conditional class probabilities* at  $x$ ; e.g. see little barplot at  $x = 5$ . Then the *Bayes optimal* classifier at  $x$  is

$$C(x) = j \text{ if } p_j(x) = \max\{p_1(x), p_2(x), \dots, p_K(x)\}$$



Nearest-neighbor averaging can be used as before.

Also breaks down as dimension grows. However, the impact on  $\hat{C}(x)$  is less than on  $\hat{p}_k(x)$ ,  $k = 1, \dots, K$ .

## Classification: some details

- Typically we measure the performance of  $\hat{C}(x)$  using the misclassification error rate:

$$\text{Err}_{\text{Te}} = \text{Ave}_{i \in \text{Te}} I[y_i \neq \hat{C}(x_i)]$$

- The Bayes classifier (using the true  $p_k(x)$ ) has smallest error (in the population).

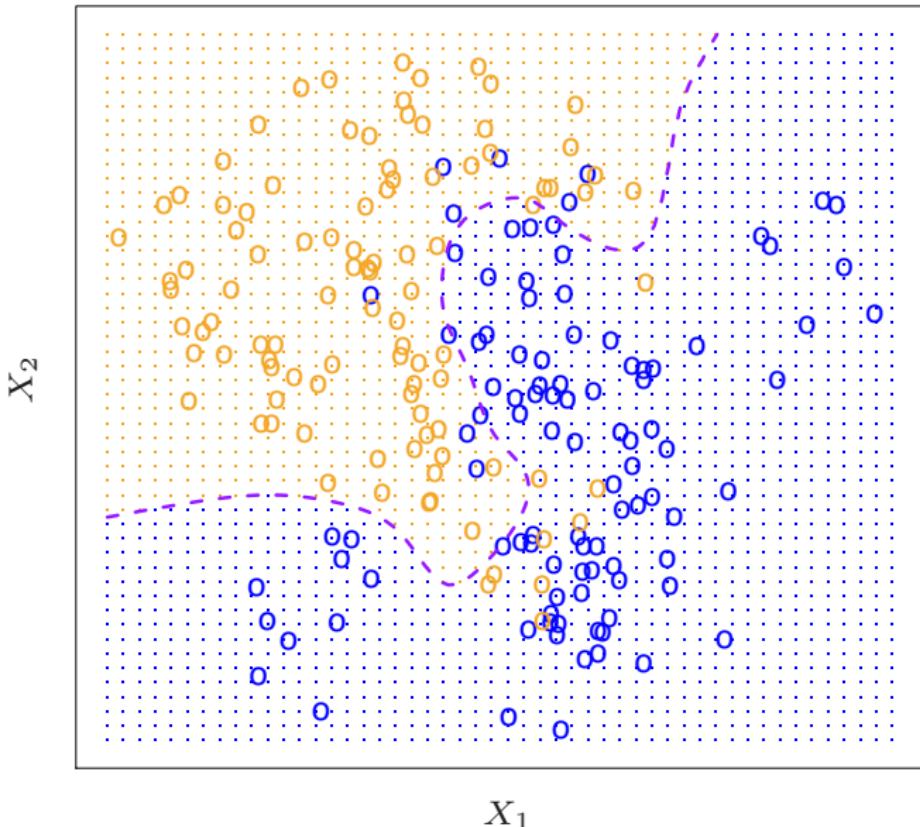
## Classification: some details

- Typically we measure the performance of  $\hat{C}(x)$  using the misclassification error rate:

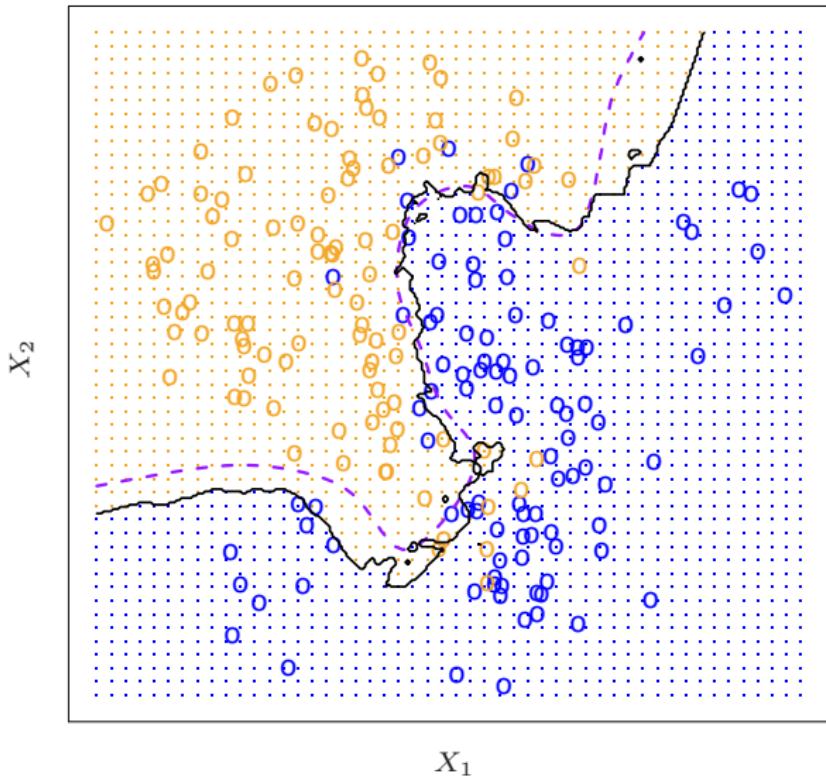
$$\text{Err}_{\text{Te}} = \text{Ave}_{i \in \text{Te}} I[y_i \neq \hat{C}(x_i)]$$

- The Bayes classifier (using the true  $p_k(x)$ ) has smallest error (in the population).
- Support-vector machines build structured models for  $C(x)$ .
- We will also build structured models for representing the  $p_k(x)$ . e.g. Logistic regression, generalized additive models.

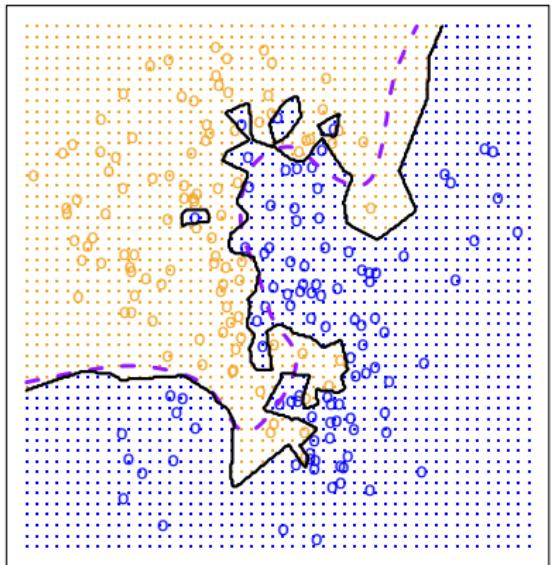
## Example: K-nearest neighbors in two dimensions



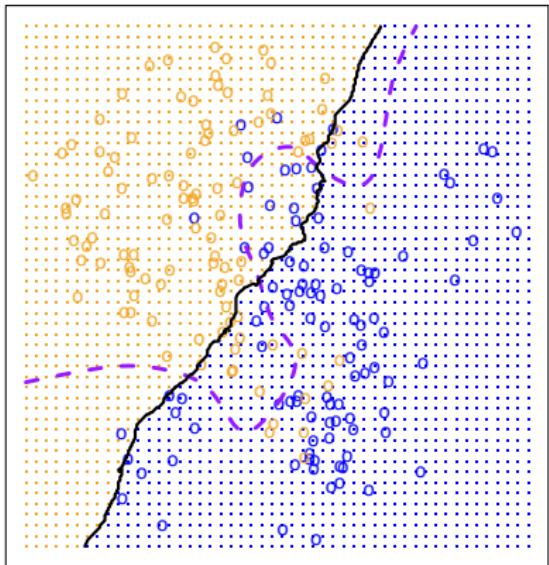
KNN: K=10

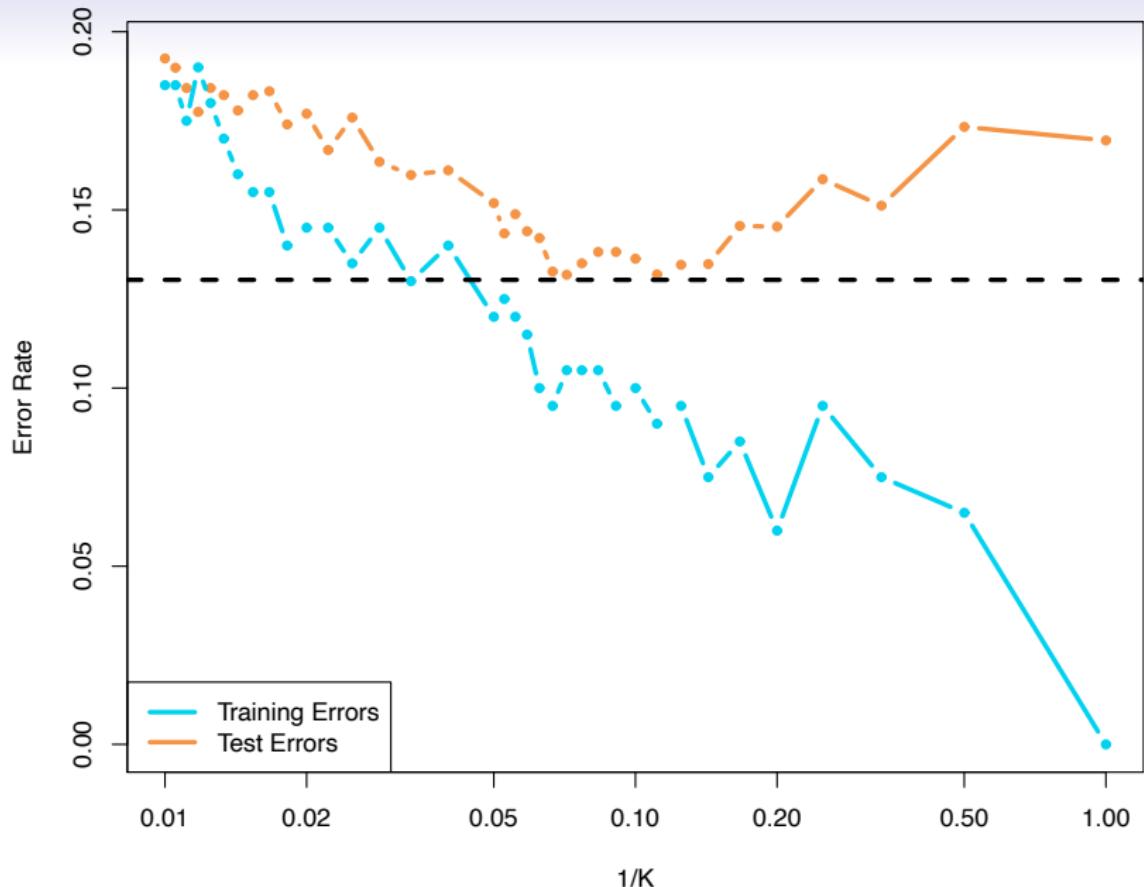


KNN: K=1



KNN: K=100



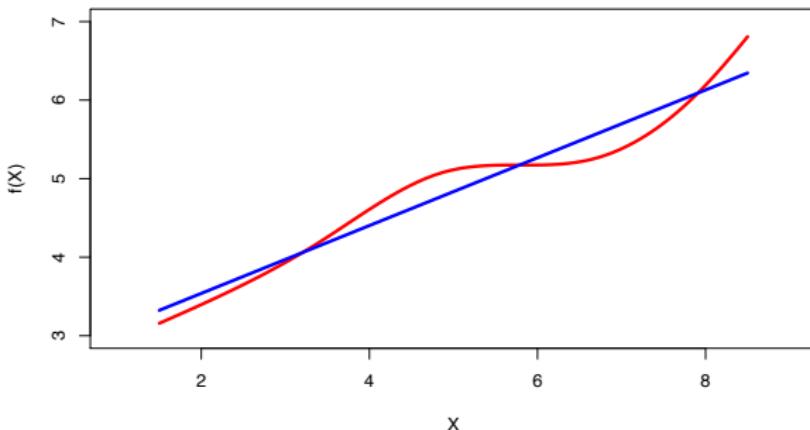


## Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of  $Y$  on  $X_1, X_2, \dots, X_p$  is linear.

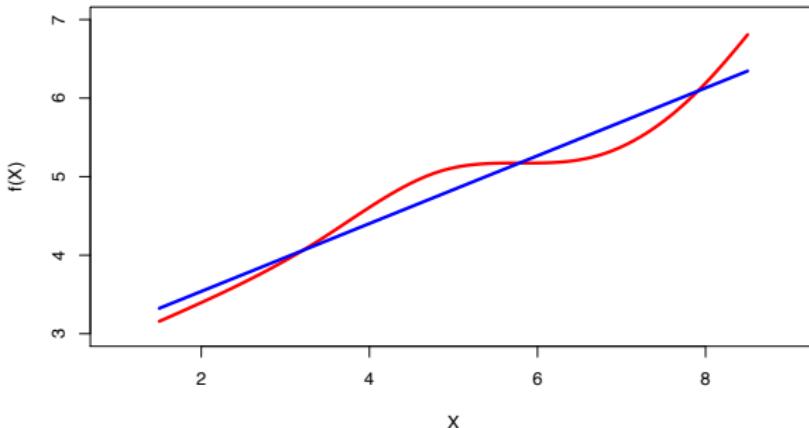
# Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of  $Y$  on  $X_1, X_2, \dots, X_p$  is linear.
- True regression functions are never linear!



## Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of  $Y$  on  $X_1, X_2, \dots, X_p$  is linear.
- True regression functions are never linear!



- although it may seem overly simplistic, linear regression is extremely useful both conceptually and practically.

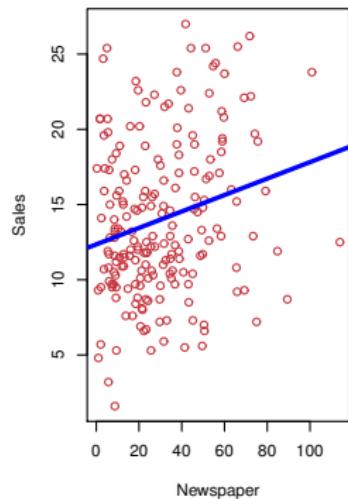
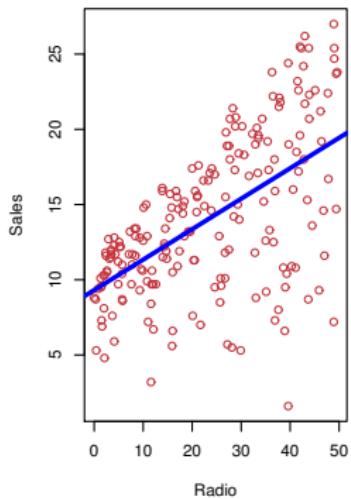
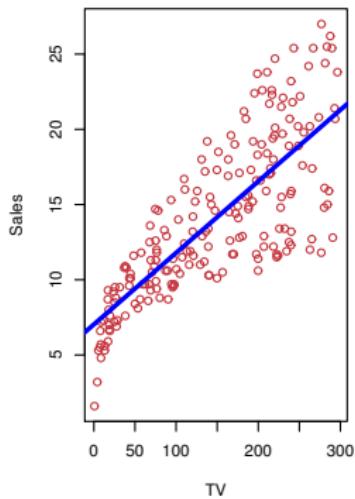
## Linear regression for the advertising data

Consider the advertising data shown on the next slide.

Questions we might ask:

- Is there a relationship between advertising budget and sales?
- How strong is the relationship between advertising budget and sales?
- Which media contribute to sales?
- How accurately can we predict future sales?
- Is the relationship linear?
- Is there synergy among the advertising media?

# Advertising data



# Simple linear regression using a single predictor $X$ .

- We assume a model

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

where  $\beta_0$  and  $\beta_1$  are two unknown constants that represent the *intercept* and *slope*, also known as *coefficients* or *parameters*, and  $\epsilon$  is the error term.

- Given some estimates  $\hat{\beta}_0$  and  $\hat{\beta}_1$  for the model coefficients, we predict future sales using

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

where  $\hat{y}$  indicates a prediction of  $Y$  on the basis of  $X = x$ . The *hat* symbol denotes an estimated value.

## Estimation of the parameters by least squares

- Let  $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$  be the prediction for  $Y$  based on the  $i$ th value of  $X$ . Then  $e_i = y_i - \hat{y}_i$  represents the  $i$ th *residual*

## Estimation of the parameters by least squares

- Let  $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$  be the prediction for  $Y$  based on the  $i$ th value of  $X$ . Then  $e_i = y_i - \hat{y}_i$  represents the  $i$ th *residual*
- We define the *residual sum of squares* (RSS) as

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2,$$

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

## Estimation of the parameters by least squares

- Let  $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$  be the prediction for  $Y$  based on the  $i$ th value of  $X$ . Then  $e_i = y_i - \hat{y}_i$  represents the  $i$ th *residual*
- We define the *residual sum of squares* (RSS) as

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2,$$

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

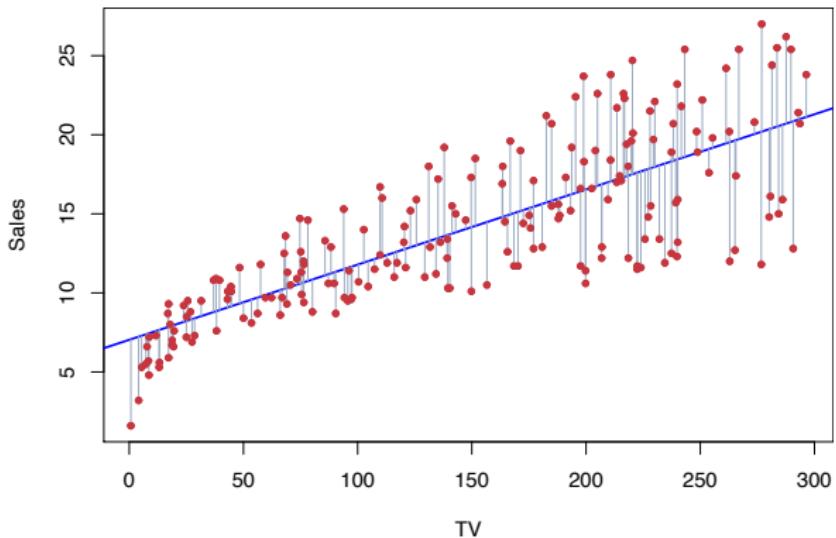
- The least squares approach chooses  $\hat{\beta}_0$  and  $\hat{\beta}_1$  to minimize the RSS. The minimizing values can be shown to be

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

where  $\bar{y} \equiv \frac{1}{n} \sum_{i=1}^n y_i$  and  $\bar{x} \equiv \frac{1}{n} \sum_{i=1}^n x_i$  are the sample means.

## Example: advertising data



The least squares fit for the regression of **sales** onto **TV**.  
In this case a linear fit captures the essence of the relationship,  
although it is somewhat deficient in the left of the plot.

## Assessing the Accuracy of the Coefficient Estimates

- The standard error of an estimator reflects how it varies under repeated sampling. We have

$$\text{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \text{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[ \frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right],$$

where  $\sigma^2 = \text{Var}(\epsilon)$

# Assessing the Accuracy of the Coefficient Estimates

- The standard error of an estimator reflects how it varies under repeated sampling. We have

$$\text{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \text{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[ \frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right],$$

where  $\sigma^2 = \text{Var}(\epsilon)$

- These standard errors can be used to compute *confidence intervals*. A 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter. It has the form

$$\hat{\beta}_1 \pm 2 \cdot \text{SE}(\hat{\beta}_1).$$

## Confidence intervals — continued

That is, there is approximately a 95% chance that the interval

$$\left[ \hat{\beta}_1 - 2 \cdot \text{SE}(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot \text{SE}(\hat{\beta}_1) \right]$$

will contain the true value of  $\beta_1$  (under a scenario where we got repeated samples like the present sample)

## Confidence intervals — continued

That is, there is approximately a 95% chance that the interval

$$\left[ \hat{\beta}_1 - 2 \cdot \text{SE}(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot \text{SE}(\hat{\beta}_1) \right]$$

will contain the true value of  $\beta_1$  (under a scenario where we got repeated samples like the present sample)

For the advertising data, the 95% confidence interval for  $\beta_1$  is  
[0.042, 0.053]

## Hypothesis testing

- Standard errors can also be used to perform *hypothesis tests* on the coefficients. The most common hypothesis test involves testing the *null hypothesis* of

$H_0$  : There is no relationship between  $X$  and  $Y$

versus the *alternative hypothesis*

$H_A$  : There is some relationship between  $X$  and  $Y$ .

## Hypothesis testing

- Standard errors can also be used to perform *hypothesis tests* on the coefficients. The most common hypothesis test involves testing the *null hypothesis* of

$H_0$  : There is no relationship between  $X$  and  $Y$

versus the *alternative hypothesis*

$H_A$  : There is some relationship between  $X$  and  $Y$ .

- Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

versus

$$H_A : \beta_1 \neq 0,$$

since if  $\beta_1 = 0$  then the model reduces to  $Y = \beta_0 + \epsilon$ , and  $X$  is not associated with  $Y$ .

## Hypothesis testing — continued

- To test the null hypothesis, we compute a *t-statistic*, given by

$$t = \frac{\hat{\beta}_1 - 0}{\text{SE}(\hat{\beta}_1)},$$

- This will have a *t*-distribution with  $n - 2$  degrees of freedom, assuming  $\beta_1 = 0$ .
- Using statistical software, it is easy to compute the probability of observing any value equal to  $|t|$  or larger. We call this probability the *p-value*.

## Results for the advertising data

	Coefficient	Std. Error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

## Assessing the Overall Accuracy of the Model

- We compute the *Residual Standard Error*

$$\text{RSE} = \sqrt{\frac{1}{n-2} \text{RSS}} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

where the *residual sum-of-squares* is  $\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ .

## Assessing the Overall Accuracy of the Model

- We compute the *Residual Standard Error*

$$\text{RSE} = \sqrt{\frac{1}{n-2} \text{RSS}} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

where the *residual sum-of-squares* is  $\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ .

- *R-squared* or fraction of variance explained is

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where  $\text{TSS} = \sum_{i=1}^n (y_i - \bar{y})^2$  is the *total sum of squares*.

## Assessing the Overall Accuracy of the Model

- We compute the *Residual Standard Error*

$$\text{RSE} = \sqrt{\frac{1}{n-2} \text{RSS}} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

where the *residual sum-of-squares* is  $\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ .

- *R-squared* or fraction of variance explained is

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where  $\text{TSS} = \sum_{i=1}^n (y_i - \bar{y})^2$  is the *total sum of squares*.

- It can be shown that in this simple linear regression setting that  $R^2 = r^2$ , where  $r$  is the correlation between  $X$  and  $Y$ :

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}.$$

## Advertising data results

Quantity	Value
Residual Standard Error	3.26
$R^2$	0.612
F-statistic	312.1

# Multiple Linear Regression

- Here our model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

- We interpret  $\beta_j$  as the *average* effect on  $Y$  of a one unit increase in  $X_j$ , *holding all other predictors fixed*. In the advertising example, the model becomes

$$\text{sales} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times \text{newspaper} + \epsilon.$$

# Interpreting regression coefficients

- The ideal scenario is when the predictors are uncorrelated
  - a *balanced design*:
    - Each coefficient can be estimated and tested separately.
    - Interpretations such as “*a unit change in  $X_j$  is associated with a  $\beta_j$  change in  $Y$ , while all the other variables stay fixed*”, are possible.
- Correlations amongst predictors cause problems:
  - The variance of all coefficients tends to increase, sometimes dramatically
  - Interpretations become hazardous — when  $X_j$  changes, everything else changes.
- *Claims of causality* should be avoided for observational data.

# The woes of (interpreting) regression coefficients

*“Data Analysis and Regression” Mosteller and Tukey 1977*

- a regression coefficient  $\beta_j$  estimates the expected change in  $Y$  per unit change in  $X_j$ , *with all other predictors held fixed*. But predictors usually change together!

# The woes of (interpreting) regression coefficients

*“Data Analysis and Regression” Mosteller and Tukey 1977*

- a regression coefficient  $\beta_j$  estimates the expected change in  $Y$  per unit change in  $X_j$ , *with all other predictors held fixed*. But predictors usually change together!
- Example:  $Y$  total amount of change in your pocket;  
 $X_1 = \#$  of coins;  $X_2 = \#$  of pennies, nickels and dimes. By itself, regression coefficient of  $Y$  on  $X_2$  will be  $> 0$ . But how about with  $X_1$  in model?

# The woes of (interpreting) regression coefficients

*“Data Analysis and Regression” Mosteller and Tukey 1977*

- a regression coefficient  $\beta_j$  estimates the expected change in  $Y$  per unit change in  $X_j$ , *with all other predictors held fixed*. But predictors usually change together!
- Example:  $Y$  total amount of change in your pocket;  $X_1 = \#$  of coins;  $X_2 = \#$  of pennies, nickels and dimes. By itself, regression coefficient of  $Y$  on  $X_2$  will be  $> 0$ . But how about with  $X_1$  in model?
- $Y$  = number of tackles by a football player in a season;  $W$  and  $H$  are his weight and height. Fitted regression model is  $\hat{Y} = b_0 + .50W - .10H$ . How do we interpret  $\hat{\beta}_2 < 0$ ?

## Two quotes by famous Statisticians

*“Essentially, all models are wrong, but some are useful”*

George Box

## Two quotes by famous Statisticians

*“Essentially, all models are wrong, but some are useful”*

George Box

*“The only way to find out what will happen when a complex system is disturbed is to disturb the system, not merely to observe it passively”*

Fred Mosteller and John Tukey, paraphrasing George Box

## Estimation and Prediction for Multiple Regression

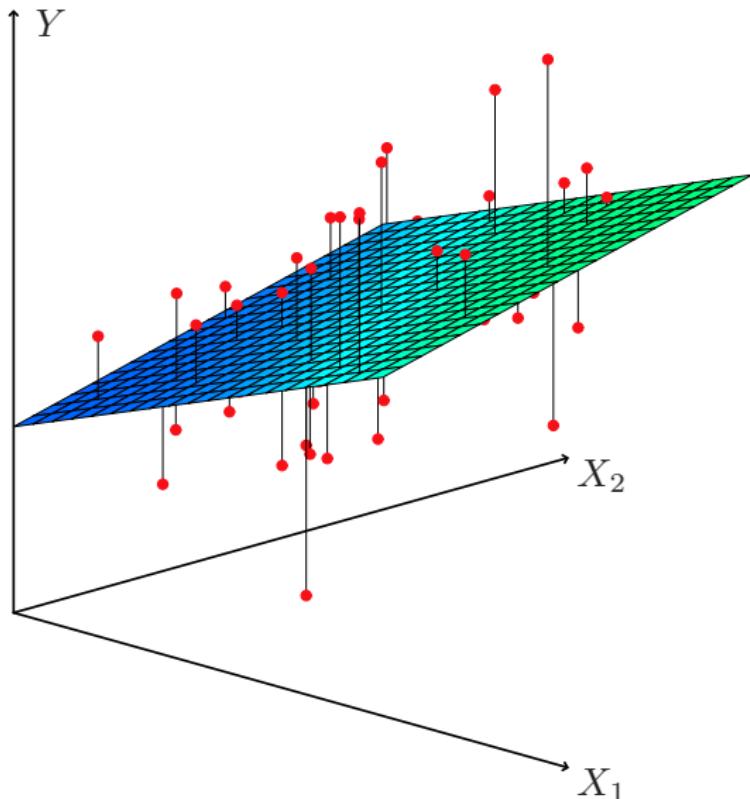
- Given estimates  $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ , we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$

- We estimate  $\beta_0, \beta_1, \dots, \beta_p$  as the values that minimize the sum of squared residuals

$$\begin{aligned}\text{RSS} &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2.\end{aligned}$$

This is done using standard statistical software. The values  $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$  that minimize RSS are the multiple least squares regression coefficient estimates.



## Results for advertising data

	Coefficient	Std. Error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

	Correlations:			
	TV	radio	newspaper	sales
TV	1.0000	0.0548	0.0567	0.7822
radio		1.0000	0.3541	0.5762
newspaper			1.0000	0.2283
sales				1.0000

## Some important questions

1. *Is at least one of the predictors  $X_1, X_2, \dots, X_p$  useful in predicting the response?*

## Some important questions

1. *Is at least one of the predictors  $X_1, X_2, \dots, X_p$  useful in predicting the response?*
2. *Do all the predictors help to explain  $Y$ , or is only a subset of the predictors useful?*

## Some important questions

1. *Is at least one of the predictors  $X_1, X_2, \dots, X_p$  useful in predicting the response?*
2. *Do all the predictors help to explain  $Y$ , or is only a subset of the predictors useful?*
3. *How well does the model fit the data?*

## Some important questions

1. *Is at least one of the predictors  $X_1, X_2, \dots, X_p$  useful in predicting the response?*
2. *Do all the predictors help to explain  $Y$ , or is only a subset of the predictors useful?*
3. *How well does the model fit the data?*
4. *Given a set of predictor values, what response value should we predict, and how accurate is our prediction?*

## Is at least one predictor useful?

For the first question, we can use the F-statistic

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n - p - 1)} \sim F_{p,n-p-1}$$

Quantity	Value
Residual Standard Error	1.69
$R^2$	0.897
F-statistic	570

## Deciding on the important variables

- The most direct approach is called *all subsets* or *best subsets* regression: we compute the least squares fit for all possible subsets and then choose between them based on some criterion that balances training error with model size.

## Deciding on the important variables

- The most direct approach is called *all subsets* or *best subsets* regression: we compute the least squares fit for all possible subsets and then choose between them based on some criterion that balances training error with model size.
- However we often can't examine all possible models, since they are  $2^p$  of them; for example when  $p = 40$  there are over a billion models!

Instead we need an automated approach that searches through a subset of them. We discuss two commonly used approaches next.

## Forward selection

- Begin with the *null model* — a model that contains an intercept but no predictors.
- Fit  $p$  simple linear regressions and add to the null model the variable that results in the lowest RSS.
- Add to that model the variable that results in the lowest RSS amongst all two-variable models.
- Continue until some stopping rule is satisfied, for example when all remaining variables have a p-value above some threshold.

## Backward selection

- Start with all variables in the model.
- Remove the variable with the largest p-value — that is, the variable that is the least statistically significant.
- The new  $(p - 1)$ -variable model is fit, and the variable with the largest p-value is removed.
- Continue until a stopping rule is reached. For instance, we may stop when all remaining variables have a significant p-value defined by some significance threshold.

## Model selection — continued

- Later we discuss more systematic criteria for choosing an “optimal” member in the path of models produced by forward or backward stepwise selection.
- These include *Mallow’s  $C_p$* , *Akaike information criterion (AIC)*, *Bayesian information criterion (BIC)*, *adjusted  $R^2$*  and *Cross-validation (CV)*.

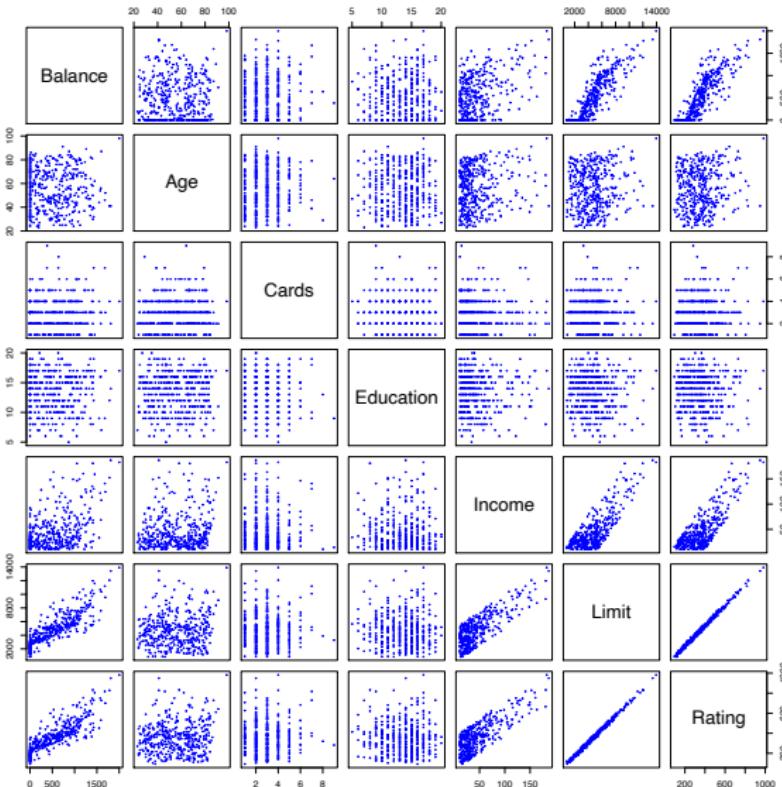
# Other Considerations in the Regression Model

## *Qualitative Predictors*

- Some predictors are not *quantitative* but are *qualitative*, taking a discrete set of values.
- These are also called *categorical* predictors or *factor variables*.
- See for example the scatterplot matrix of the credit card data in the next slide.

In addition to the 7 quantitative variables shown, there are four qualitative variables: **gender**, **student** (student status), **status** (marital status), and **ethnicity** (Caucasian, African American (AA) or Asian).

# Credit Card Data



## Qualitative Predictors — continued

Example: investigate differences in credit card balance between males and females, ignoring the other variables. We create a new variable

$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ 0 & \text{if } i\text{th person is male} \end{cases}$$

Resulting model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is female} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is male.} \end{cases}$$

Intrepretation?

## Credit card data — continued

Results for gender model:

	Coefficient	Std. Error	t-statistic	p-value
Intercept	509.80	33.13	15.389	< 0.0001
gender [Female]	19.73	46.05	0.429	0.6690

## Qualitative predictors with more than two levels

- With more than two levels, we create additional dummy variables. For example, for the **ethnicity** variable we create two dummy variables. The first could be

$$x_{i1} = \begin{cases} 1 & \text{if } i\text{th person is Asian} \\ 0 & \text{if } i\text{th person is not Asian,} \end{cases}$$

and the second could be

$$x_{i2} = \begin{cases} 1 & \text{if } i\text{th person is Caucasian} \\ 0 & \text{if } i\text{th person is not Caucasian.} \end{cases}$$

## Qualitative predictors with more than two levels — continued.

- Then both of these variables can be used in the regression equation, in order to obtain the model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is Asian} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i\text{th person is Caucasian} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is AA.} \end{cases}$$

- There will always be one fewer dummy variable than the number of levels. The level with no dummy variable — African American in this example — is known as the *baseline*.

## Results for ethnicity

	Coefficient	Std. Error	t-statistic	p-value
Intercept	531.00	46.32	11.464	< 0.0001
ethnicity[Asian]	-18.69	65.02	-0.287	0.7740
ethnicity[Caucasian]	-12.50	56.68	-0.221	0.8260

## Extensions of the Linear Model

Removing the additive assumption: *interactions* and *nonlinearity*

### *Interactions:*

- In our previous analysis of the **Advertising** data, we assumed that the effect on **sales** of increasing one advertising medium is independent of the amount spent on the other media.
- For example, the linear model

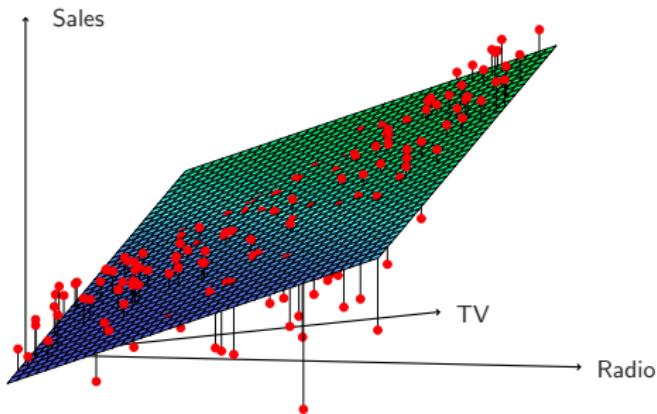
$$\widehat{\text{sales}} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times \text{newspaper}$$

states that the average effect on **sales** of a one-unit increase in **TV** is always  $\beta_1$ , regardless of the amount spent on **radio**.

## Interactions — continued

- But suppose that spending money on radio advertising actually increases the effectiveness of TV advertising, so that the slope term for **TV** should increase as **radio** increases.
- In this situation, given a fixed budget of \$100,000, spending half on **radio** and half on **TV** may increase **sales** more than allocating the entire amount to either **TV** or to **radio**.
- In marketing, this is known as a *synergy* effect, and in statistics it is referred to as an *interaction* effect.

# Interaction in the Advertising data?



When levels of either **TV** or **radio** are low, then the true **sales** are lower than predicted by the linear model.  
But when advertising is split between the two media, then the model tends to underestimate **sales**.

## Modelling interactions — Advertising data

Model takes the form

$$\begin{aligned}\text{sales} &= \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times (\text{radio} \times \text{TV}) + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 \times \text{radio}) \times \text{TV} + \beta_2 \times \text{radio} + \epsilon.\end{aligned}$$

Results:

	Coefficient	Std. Error	t-statistic	p-value
Intercept	6.7502	0.248	27.23	< 0.0001
TV	0.0191	0.002	12.70	< 0.0001
radio	0.0289	0.009	3.24	0.0014
TV × radio	0.0011	0.000	20.73	< 0.0001

## Interpretation

- The results in this table suggests that interactions are important.
- The p-value for the interaction term  $\text{TV} \times \text{radio}$  is extremely low, indicating that there is strong evidence for  $H_A : \beta_3 \neq 0$ .
- The  $R^2$  for the interaction model is 96.8%, compared to only 89.7% for the model that predicts **sales** using **TV** and **radio** without an interaction term.

## Interpretation — continued

- This means that  $(96.8 - 89.7)/(100 - 89.7) = 69\%$  of the variability in **sales** that remains after fitting the additive model has been explained by the interaction term.
- The coefficient estimates in the table suggest that an increase in TV advertising of \$1,000 is associated with increased sales of
$$(\hat{\beta}_1 + \hat{\beta}_3 \times \text{radio}) \times 1000 = 19 + 1.1 \times \text{radio} \text{ units.}$$
- An increase in radio advertising of \$1,000 will be associated with an increase in sales of
$$(\hat{\beta}_2 + \hat{\beta}_3 \times \text{TV}) \times 1000 = 29 + 1.1 \times \text{TV} \text{ units.}$$

# Hierarchy

- Sometimes it is the case that an interaction term has a very small p-value, but the associated main effects (in this case, **TV** and **radio**) do not.
- The *hierarchy principle*:

*If we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant.*

## Hierarchy — continued

- The rationale for this principle is that interactions are hard to interpret in a model without main effects — their meaning is changed.
- Specifically, the interaction terms also contain main effects, if the model has no main effect terms.

## Interactions between qualitative and quantitative variables

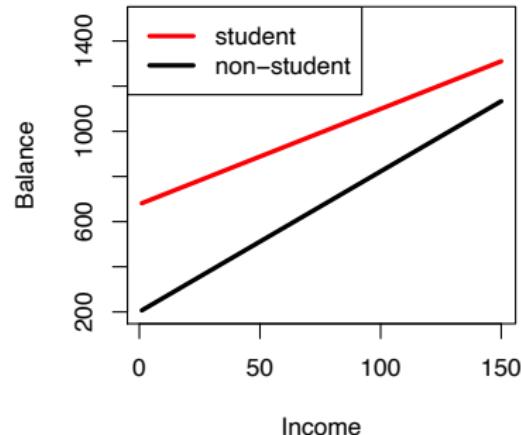
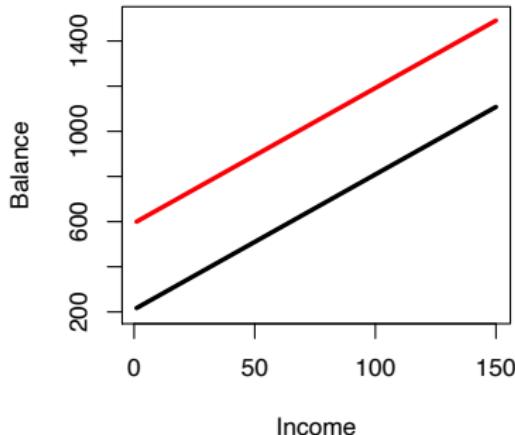
Consider the **Credit** data set, and suppose that we wish to predict **balance** using **income** (quantitative) and **student** (qualitative).

Without an interaction term, the model takes the form

$$\begin{aligned}\text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 & \text{if } i\text{th person is a student} \\ 0 & \text{if } i\text{th person is not a student} \end{cases} \\ &= \beta_1 \times \text{income}_i + \begin{cases} \beta_0 + \beta_2 & \text{if } i\text{th person is a student} \\ \beta_0 & \text{if } i\text{th person is not a student.} \end{cases}\end{aligned}$$

With interactions, it takes the form

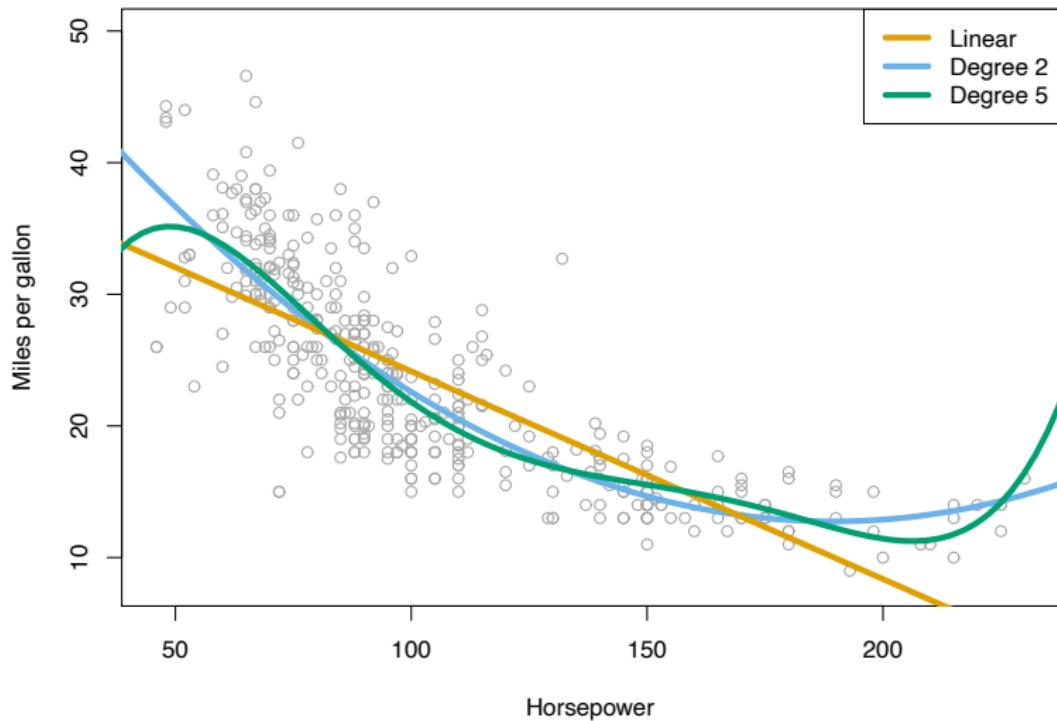
$$\begin{aligned}\text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 + \beta_3 \times \text{income}_i & \text{if student} \\ 0 & \text{if not student} \end{cases} \\ &= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \text{income}_i & \text{if student} \\ \beta_0 + \beta_1 \times \text{income}_i & \text{if not student} \end{cases}\end{aligned}$$



Credit data; Left: no interaction between `income` and `student`.  
Right: with an interaction term between `income` and `student`.

# Non-linear effects of predictors

polynomial regression on **Auto** data



The figure suggests that

$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower} + \beta_2 \times \text{horsepower}^2 + \epsilon$$

may provide a better fit.

	Coefficient	Std. Error	t-statistic	p-value
Intercept	56.9001	1.8004	31.6	< 0.0001
horsepower	-0.4662	0.0311	-15.0	< 0.0001
horsepower <sup>2</sup>	0.0012	0.0001	10.1	< 0.0001

## What we did not cover

Outliers

Non-constant variance of error terms

High leverage points

Collinearity

See text Section 3.33

# Generalizations of the Linear Model

In much of the rest of this course, we discuss methods that expand the scope of linear models and how they are fit:

- *Classification problems*: logistic regression, support vector machines
- *Non-linearity*: kernel smoothing, splines and generalized additive models; nearest neighbor methods.
- *Interactions*: Tree-based methods, bagging, random forests and boosting (these also capture non-linearities)
- *Regularized fitting*: Ridge regression and lasso

# Classification

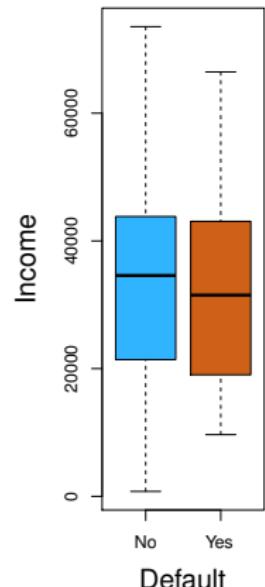
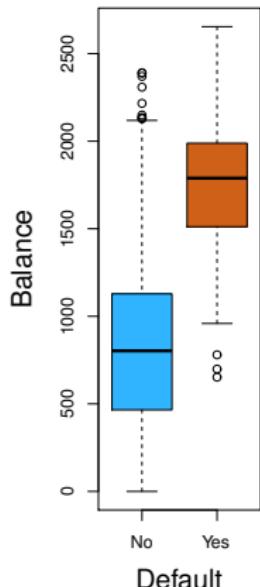
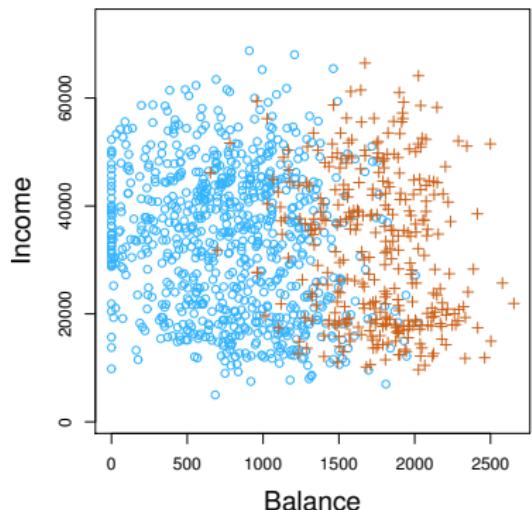
- Qualitative variables take values in an unordered set  $\mathcal{C}$ , such as:  
 $\text{eye color} \in \{\text{brown, blue, green}\}$   
 $\text{email} \in \{\text{spam, ham}\}.$
- Given a feature vector  $X$  and a qualitative response  $Y$  taking values in the set  $\mathcal{C}$ , the classification task is to build a function  $C(X)$  that takes as input the feature vector  $X$  and predicts its value for  $Y$ ; i.e.  $C(X) \in \mathcal{C}$ .
- Often we are more interested in estimating the *probabilities* that  $X$  belongs to each category in  $\mathcal{C}$ .

# Classification

- Qualitative variables take values in an unordered set  $\mathcal{C}$ , such as:  
 $\text{eye color} \in \{\text{brown, blue, green}\}$   
 $\text{email} \in \{\text{spam, ham}\}.$
- Given a feature vector  $X$  and a qualitative response  $Y$  taking values in the set  $\mathcal{C}$ , the classification task is to build a function  $C(X)$  that takes as input the feature vector  $X$  and predicts its value for  $Y$ ; i.e.  $C(X) \in \mathcal{C}$ .
- Often we are more interested in estimating the *probabilities* that  $X$  belongs to each category in  $\mathcal{C}$ .

For example, it is more valuable to have an estimate of the probability that an insurance claim is fraudulent, than a classification fraudulent or not.

## Example: Credit Card Default



## Can we use Linear Regression?

Suppose for the **Default** classification task that we code

$$Y = \begin{cases} 0 & \text{if No} \\ 1 & \text{if Yes.} \end{cases}$$

Can we simply perform a linear regression of  $Y$  on  $X$  and classify as **Yes** if  $\hat{Y} > 0.5$ ?

## Can we use Linear Regression?

Suppose for the **Default** classification task that we code

$$Y = \begin{cases} 0 & \text{if No} \\ 1 & \text{if Yes.} \end{cases}$$

Can we simply perform a linear regression of  $Y$  on  $X$  and classify as **Yes** if  $\hat{Y} > 0.5$ ?

- In this case of a binary outcome, linear regression does a good job as a classifier, and is equivalent to *linear discriminant analysis* which we discuss later.
- Since in the population  $E(Y|X = x) = \Pr(Y = 1|X = x)$ , we might think that regression is perfect for this task.

## Can we use Linear Regression?

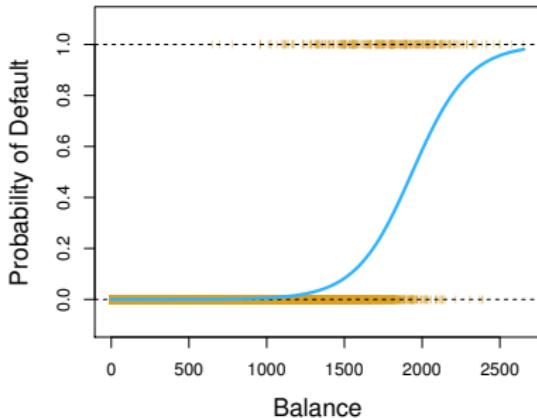
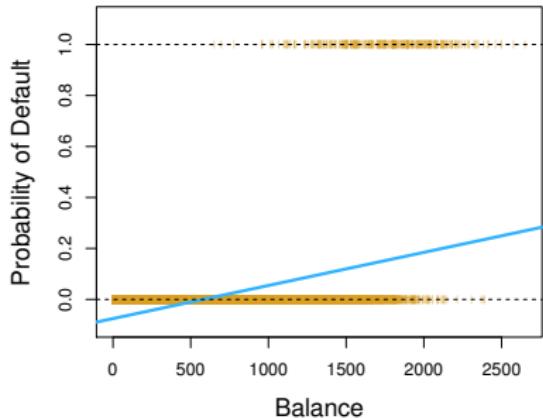
Suppose for the **Default** classification task that we code

$$Y = \begin{cases} 0 & \text{if No} \\ 1 & \text{if Yes.} \end{cases}$$

Can we simply perform a linear regression of  $Y$  on  $X$  and classify as **Yes** if  $\hat{Y} > 0.5$ ?

- In this case of a binary outcome, linear regression does a good job as a classifier, and is equivalent to *linear discriminant analysis* which we discuss later.
- Since in the population  $E(Y|X = x) = \Pr(Y = 1|X = x)$ , we might think that regression is perfect for this task.
- However, *linear* regression might produce probabilities less than zero or bigger than one. *Logistic regression* is more appropriate.

# Linear versus Logistic Regression



The orange marks indicate the response  $Y$ , either 0 or 1. Linear regression does not estimate  $\Pr(Y = 1|X)$  well. Logistic regression seems well suited to the task.

## Linear Regression continued

Now suppose we have a response variable with three possible values. A patient presents at the emergency room, and we must classify them according to their symptoms.

$$Y = \begin{cases} 1 & \text{if } \texttt{stroke}; \\ 2 & \text{if } \texttt{drug overdose}; \\ 3 & \text{if } \texttt{epileptic seizure}. \end{cases}$$

This coding suggests an ordering, and in fact implies that the difference between **stroke** and **drug overdose** is the same as between **drug overdose** and **epileptic seizure**.

## Linear Regression continued

Now suppose we have a response variable with three possible values. A patient presents at the emergency room, and we must classify them according to their symptoms.

$$Y = \begin{cases} 1 & \text{if } \texttt{stroke}; \\ 2 & \text{if } \texttt{drug overdose}; \\ 3 & \text{if } \texttt{epileptic seizure}. \end{cases}$$

This coding suggests an ordering, and in fact implies that the difference between **stroke** and **drug overdose** is the same as between **drug overdose** and **epileptic seizure**.

Linear regression is not appropriate here.

*Multiclass Logistic Regression* or *Discriminant Analysis* are more appropriate.

## Logistic Regression

Let's write  $p(X) = \Pr(Y = 1|X)$  for short and consider using **balance** to predict **default**. Logistic regression uses the form

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}.$$

( $e \approx 2.71828$  is a mathematical constant [Euler's number.])

It is easy to see that no matter what values  $\beta_0$ ,  $\beta_1$  or  $X$  take,  $p(X)$  will have values between 0 and 1.

## Logistic Regression

Let's write  $p(X) = \Pr(Y = 1|X)$  for short and consider using **balance** to predict **default**. Logistic regression uses the form

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}.$$

( $e \approx 2.71828$  is a mathematical constant [Euler's number].)

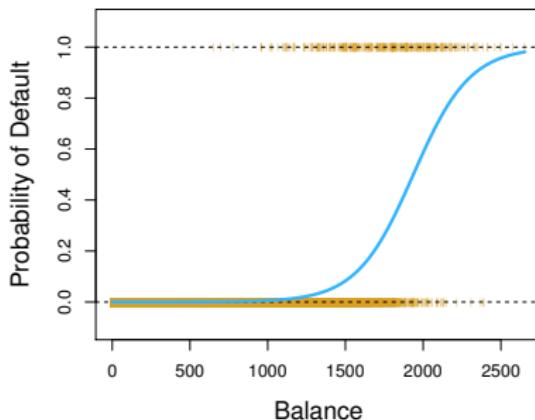
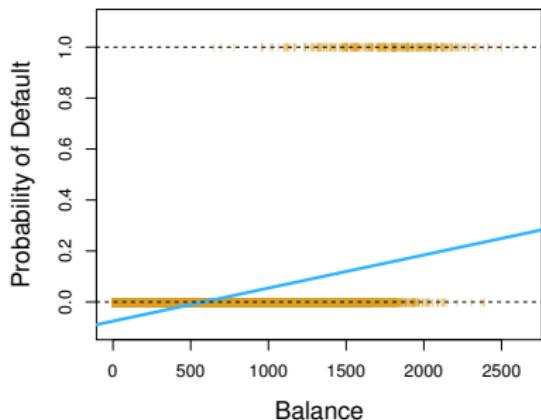
It is easy to see that no matter what values  $\beta_0$ ,  $\beta_1$  or  $X$  take,  $p(X)$  will have values between 0 and 1.

A bit of rearrangement gives

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X.$$

This monotone transformation is called the *log odds* or *logit* transformation of  $p(X)$ . (by log we mean *natural log*: ln.)

# Linear versus Logistic Regression



Logistic regression ensures that our estimate for  $p(X)$  lies between 0 and 1.

## Maximum Likelihood

We use maximum likelihood to estimate the parameters.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

This *likelihood* gives the probability of the observed zeros and ones in the data. We pick  $\beta_0$  and  $\beta_1$  to maximize the likelihood of the observed data.

## Maximum Likelihood

We use maximum likelihood to estimate the parameters.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

This *likelihood* gives the probability of the observed zeros and ones in the data. We pick  $\beta_0$  and  $\beta_1$  to maximize the likelihood of the observed data.

Most statistical packages can fit linear logistic regression models by maximum likelihood. In **R** we use the **glm** function.

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.6513	0.3612	-29.5	< 0.0001
balance	0.0055	0.0002	24.9	< 0.0001

## Making Predictions

What is our estimated probability of **default** for someone with a balance of \$1000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1000}}{1 + e^{-10.6513 + 0.0055 \times 1000}} = 0.006$$

## Making Predictions

What is our estimated probability of **default** for someone with a balance of \$1000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1000}}{1 + e^{-10.6513 + 0.0055 \times 1000}} = 0.006$$

With a balance of \$2000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 2000}}{1 + e^{-10.6513 + 0.0055 \times 2000}} = 0.586$$

Lets do it again, using **student** as the predictor.

	Coefficient	Std. Error	Z-statistic	P-value
<b>Intercept</b>	-3.5041	0.0707	-49.55	< 0.0001
<b>student [Yes]</b>	0.4049	0.1150	3.52	0.0004

Lets do it again, using **student** as the predictor.

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-3.5041	0.0707	-49.55	< 0.0001
student [Yes]	0.4049	0.1150	3.52	0.0004

$$\widehat{\Pr}(\text{default=Yes} | \text{student=Yes}) = \frac{e^{-3.5041 + 0.4049 \times 1}}{1 + e^{-3.5041 + 0.4049 \times 1}} = 0.0431,$$

$$\widehat{\Pr}(\text{default=Yes} | \text{student=No}) = \frac{e^{-3.5041 + 0.4049 \times 0}}{1 + e^{-3.5041 + 0.4049 \times 0}} = 0.0292.$$

## Logistic regression with several variables

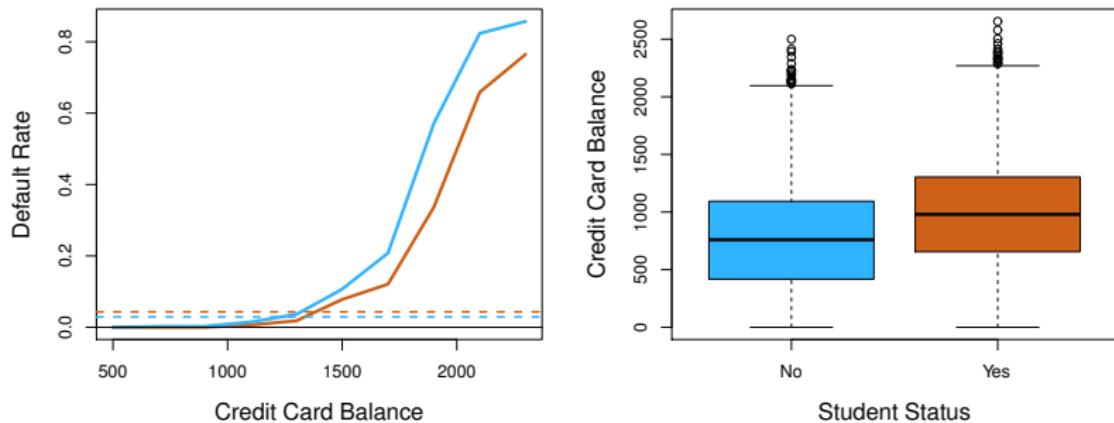
$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}$$

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.8690	0.4923	-22.08	< 0.0001
balance	0.0057	0.0002	24.74	< 0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	-0.6468	0.2362	-2.74	0.0062

Why is coefficient for **student** negative, while it was positive before?

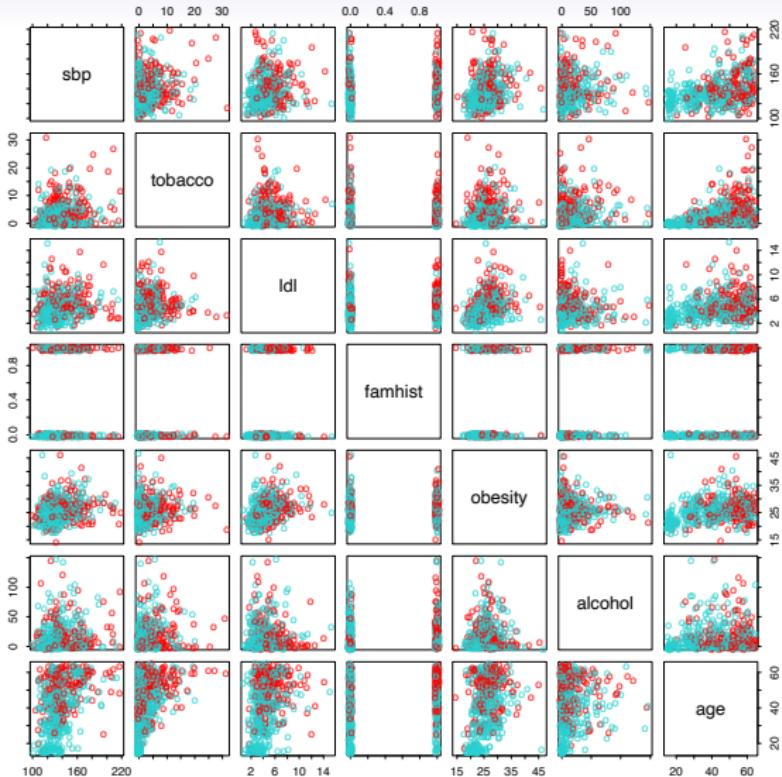
# Confounding



- Students tend to have higher balances than non-students, so their marginal default rate is higher than for non-students.
- But for each level of balance, students default less than non-students.
- Multiple logistic regression can tease this out.

## Example: South African Heart Disease

- 160 cases of MI (myocardial infarction) and 302 controls (all male in age range 15-64), from Western Cape, South Africa in early 80s.
- Overall prevalence very high in this region: 5.1%.
- Measurements on seven predictors (risk factors), shown in scatterplot matrix.
- Goal is to identify relative strengths and directions of risk factors.
- This was part of an intervention study aimed at educating the public on healthier diets.



Scatterplot matrix of the *South African Heart Disease* data. The response is color coded — The cases (MI) are red, the controls turquoise. **famhist** is a binary variable, with 1 indicating family history of MI.

```
> heartfit<-glm(chd~.,data=heart,family=binomial)
> summary(heartfit)

Call:
glm(formula = chd ~ ., family = binomial, data = heart)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.1295997  0.9641558 -4.283 1.84e-05 ***
sbp          0.0057607  0.0056326  1.023  0.30643
tobacco      0.0795256  0.0262150  3.034  0.00242 **
ldl          0.1847793  0.0574115  3.219  0.00129 **
famhistPresent 0.9391855  0.2248691  4.177 2.96e-05 ***
obesity      -0.0345434  0.0291053 -1.187  0.23529
alcohol       0.0006065  0.0044550  0.136  0.89171
age           0.0425412  0.0101749  4.181 2.90e-05 ***

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11    on 461    degrees of freedom
Residual deviance: 483.17    on 454    degrees of freedom
AIC: 499.17
```

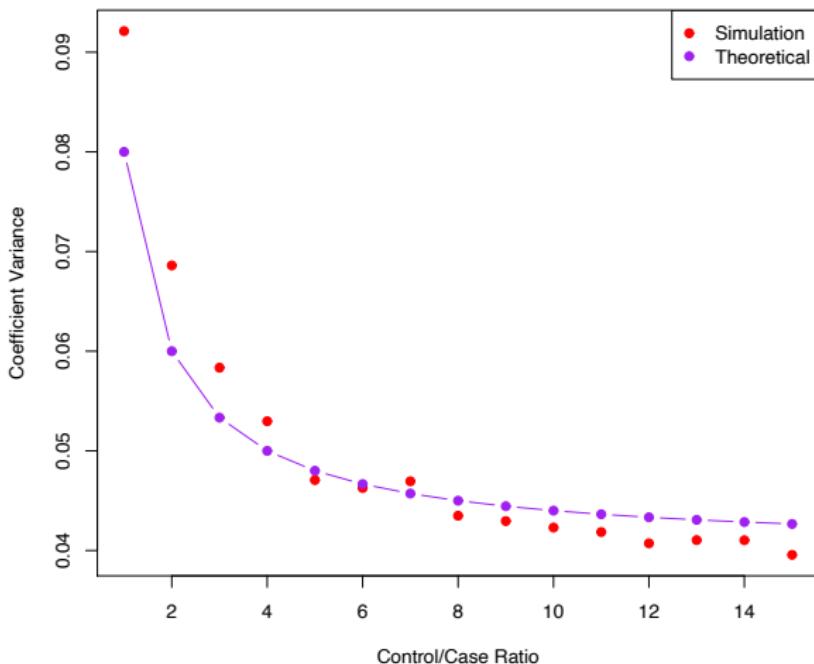
## Case-control sampling and logistic regression

- In South African data, there are 160 cases, 302 controls —  $\tilde{\pi} = 0.35$  are cases. Yet the prevalence of MI in this region is  $\pi = 0.05$ .
- With case-control samples, we can estimate the regression parameters  $\beta_j$  accurately (if our model is correct); the constant term  $\beta_0$  is incorrect.
- We can correct the estimated intercept by a simple transformation

$$\hat{\beta}_0^* = \hat{\beta}_0 + \log \frac{\pi}{1 - \pi} - \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

- Often cases are rare and we take them all; up to five times that number of controls is sufficient. See next frame

# Diminishing returns in unbalanced binary data



Sampling more controls than cases reduces the variance of the parameter estimates. But after a ratio of about 5 to 1 the variance reduction flattens out.

## Logistic regression with more than two classes

So far we have discussed logistic regression with two classes. It is easily generalized to more than two classes. One version (used in the R package **glmnet**) has the symmetric form

$$\Pr(Y = k|X) = \frac{e^{\beta_{0k} + \beta_{1k}X_1 + \dots + \beta_{pk}X_p}}{\sum_{\ell=1}^K e^{\beta_{0\ell} + \beta_{1\ell}X_1 + \dots + \beta_{p\ell}X_p}}$$

Here there is a linear function for *each* class.

## Logistic regression with more than two classes

So far we have discussed logistic regression with two classes. It is easily generalized to more than two classes. One version (used in the R package `glmnet`) has the symmetric form

$$\Pr(Y = k|X) = \frac{e^{\beta_{0k} + \beta_{1k}X_1 + \dots + \beta_{pk}X_p}}{\sum_{\ell=1}^K e^{\beta_{0\ell} + \beta_{1\ell}X_1 + \dots + \beta_{p\ell}X_p}}$$

Here there is a linear function for *each* class.

(The *mathier* students will recognize that some cancellation is possible, and only  $K - 1$  linear functions are needed as in 2-class logistic regression.)

## Logistic regression with more than two classes

So far we have discussed logistic regression with two classes. It is easily generalized to more than two classes. One version (used in the R package `glmnet`) has the symmetric form

$$\Pr(Y = k|X) = \frac{e^{\beta_{0k} + \beta_{1k}X_1 + \dots + \beta_{pk}X_p}}{\sum_{\ell=1}^K e^{\beta_{0\ell} + \beta_{1\ell}X_1 + \dots + \beta_{p\ell}X_p}}$$

Here there is a linear function for *each* class.

(The *mathier* students will recognize that some cancellation is possible, and only  $K - 1$  linear functions are needed as in 2-class logistic regression.)

Multiclass logistic regression is also referred to as *multinomial regression*.

## Discriminant Analysis

Here the approach is to model the distribution of  $X$  in each of the classes separately, and then use *Bayes theorem* to flip things around and obtain  $\Pr(Y|X)$ .

When we use normal (Gaussian) distributions for each class, this leads to linear or quadratic discriminant analysis.

However, this approach is quite general, and other distributions can be used as well. We will focus on normal distributions.

## Bayes theorem for classification

Thomas Bayes was a famous mathematician whose name represents a big subfield of statistical and probabilistic modeling. Here we focus on a simple result, known as Bayes theorem:

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)}$$

## Bayes theorem for classification

Thomas Bayes was a famous mathematician whose name represents a big subfield of statistical and probabilistic modeling. Here we focus on a simple result, known as Bayes theorem:

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)}$$

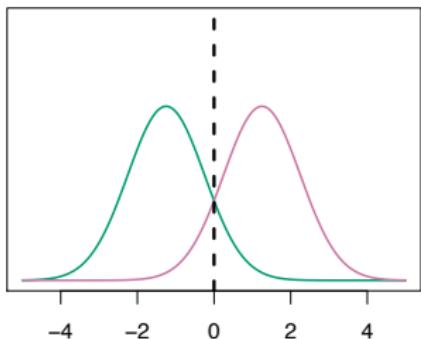
One writes this slightly differently for discriminant analysis:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}, \quad \text{where}$$

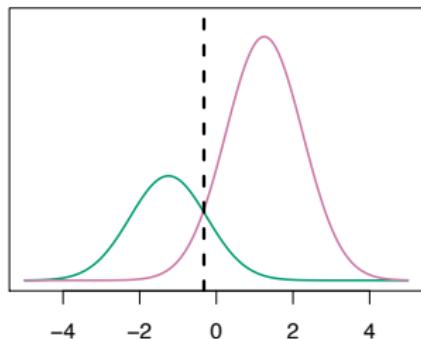
- $f_k(x) = \Pr(X = x|Y = k)$  is the *density* for  $X$  in class  $k$ . Here we will use normal densities for these, separately in each class.
- $\pi_k = \Pr(Y = k)$  is the marginal or *prior* probability for class  $k$ .

## Classify to the highest density

$$\pi_1=.5, \quad \pi_2=.5$$



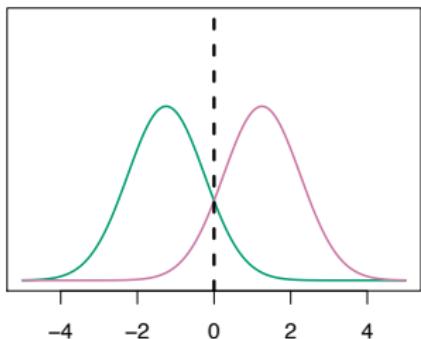
$$\pi_1=.3, \quad \pi_2=.7$$



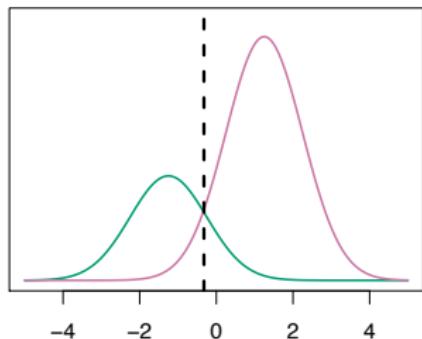
We classify a new point according to which density is highest.

## Classify to the highest density

$$\pi_1=.5, \quad \pi_2=.5$$



$$\pi_1=.3, \quad \pi_2=.7$$



We classify a new point according to which density is highest.

When the priors are different, we take them into account as well, and compare  $\pi_k f_k(x)$ . On the right, we favor the pink class — the decision boundary has shifted to the left.

## Why discriminant analysis?

- When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis does not suffer from this problem.
- If  $n$  is small and the distribution of the predictors  $X$  is approximately normal in each of the classes, the linear discriminant model is again more stable than the logistic regression model.
- Linear discriminant analysis is popular when we have more than two response classes, because it also provides low-dimensional views of the data.

## Linear Discriminant Analysis when $p = 1$

The Gaussian density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

Here  $\mu_k$  is the mean, and  $\sigma_k^2$  the variance (in class  $k$ ). We will assume that all the  $\sigma_k = \sigma$  are the same.

## Linear Discriminant Analysis when $p = 1$

The Gaussian density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

Here  $\mu_k$  is the mean, and  $\sigma_k^2$  the variance (in class  $k$ ). We will assume that all the  $\sigma_k = \sigma$  are the same.

Plugging this into Bayes formula, we get a rather complex expression for  $p_k(x) = \Pr(Y = k|X = x)$ :

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma}\right)^2}}$$

Happily, there are simplifications and cancellations.

## Discriminant functions

To classify at the value  $X = x$ , we need to see which of the  $p_k(x)$  is largest. Taking logs, and discarding terms that do not depend on  $k$ , we see that this is equivalent to assigning  $x$  to the class with the largest *discriminant score*:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Note that  $\delta_k(x)$  is a *linear* function of  $x$ .

## Discriminant functions

To classify at the value  $X = x$ , we need to see which of the  $p_k(x)$  is largest. Taking logs, and discarding terms that do not depend on  $k$ , we see that this is equivalent to assigning  $x$  to the class with the largest *discriminant score*:

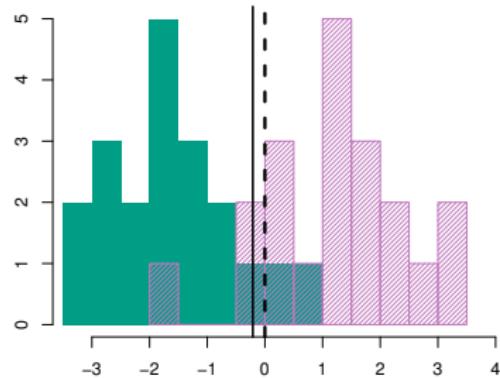
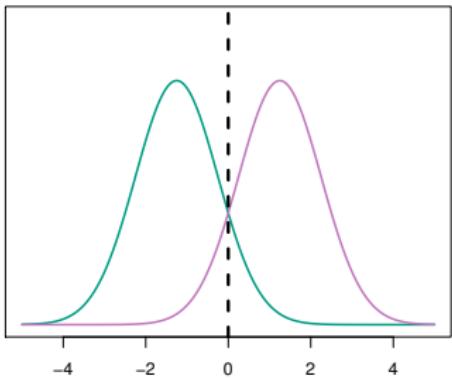
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Note that  $\delta_k(x)$  is a *linear* function of  $x$ .

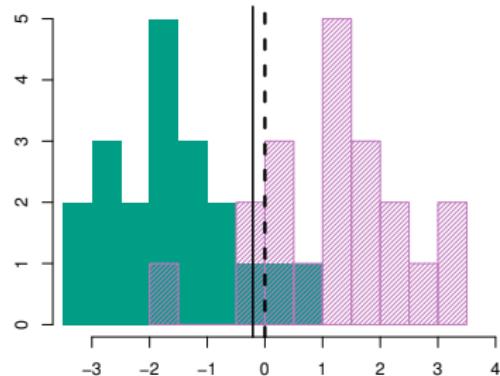
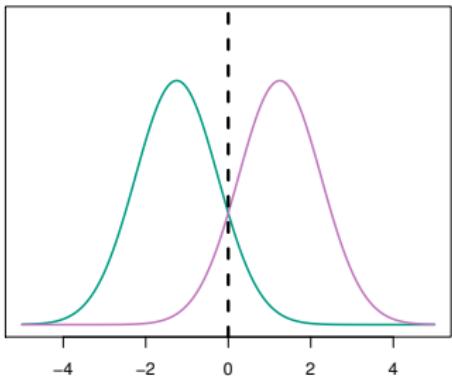
If there are  $K = 2$  classes and  $\pi_1 = \pi_2 = 0.5$ , then one can see that the *decision boundary* is at

$$x = \frac{\mu_1 + \mu_2}{2}.$$

(See if you can show this)



Example with  $\mu_1 = -1.5$ ,  $\mu_2 = 1.5$ ,  $\pi_1 = \pi_2 = 0.5$ , and  $\sigma^2 = 1$ .



Example with  $\mu_1 = -1.5$ ,  $\mu_2 = 1.5$ ,  $\pi_1 = \pi_2 = 0.5$ , and  $\sigma^2 = 1$ .

Typically we don't know these parameters; we just have the training data. In that case we simply estimate the parameters and plug them into the rule.

## Estimating the parameters

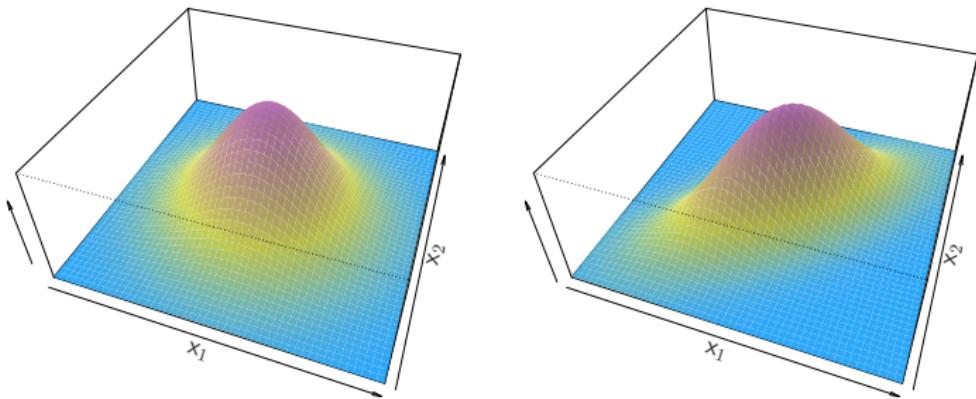
$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i: y_i=k} x_i$$

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

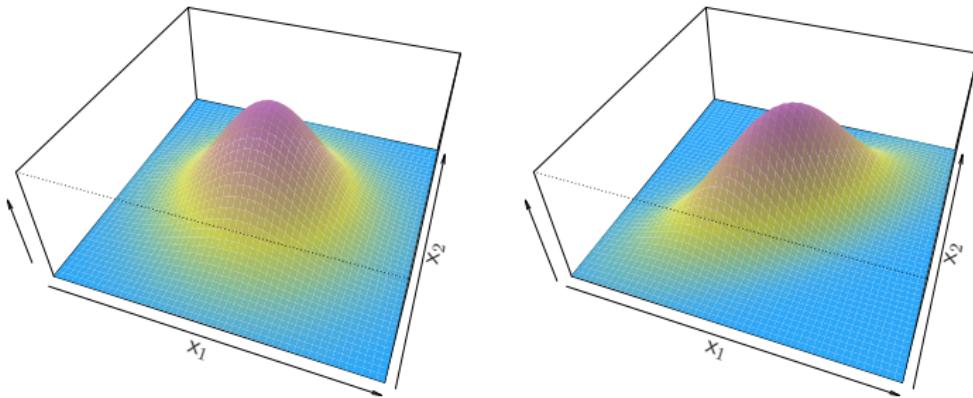
where  $\hat{\sigma}_k^2 = \frac{1}{n_k - 1} \sum_{i: y_i=k} (x_i - \hat{\mu}_k)^2$  is the usual formula for the estimated variance in the  $k$ th class.

## Linear Discriminant Analysis when $p > 1$



Density:  $f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$

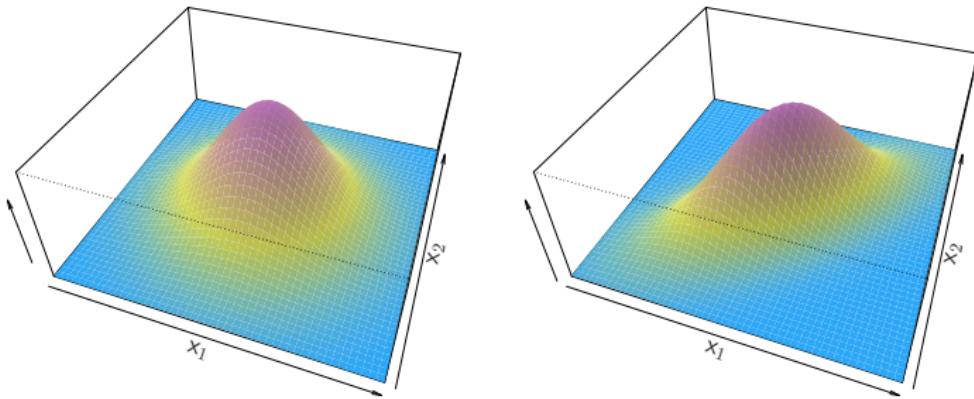
## Linear Discriminant Analysis when $p > 1$



Density:  $f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$

Discriminant function:  $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$

## Linear Discriminant Analysis when $p > 1$



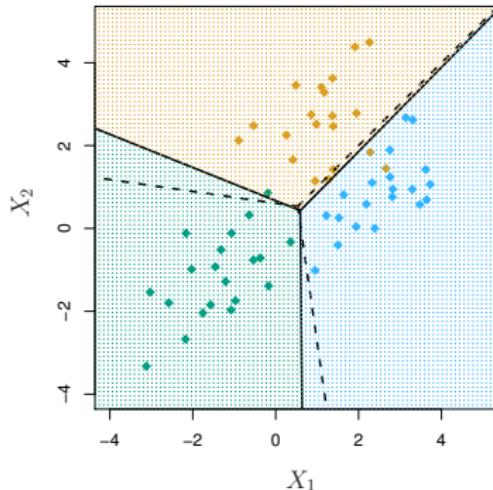
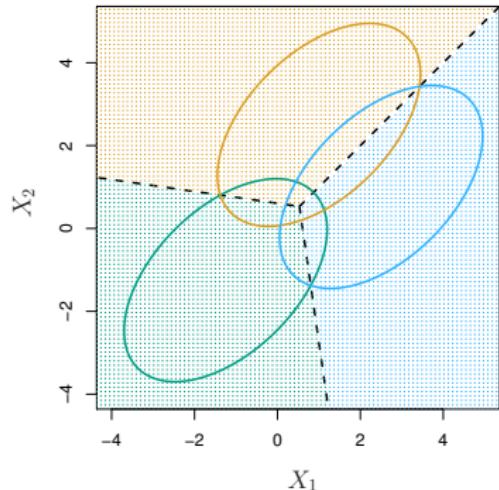
Density:  $f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$

Discriminant function:  $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$

Despite its complex form,

$$\delta_k(x) = c_{k0} + c_{k1}x_1 + c_{k2}x_2 + \dots + c_{kp}x_p — \text{a linear function.}$$

## Illustration: $p = 2$ and $K = 3$ classes



Here  $\pi_1 = \pi_2 = \pi_3 = 1/3$ .

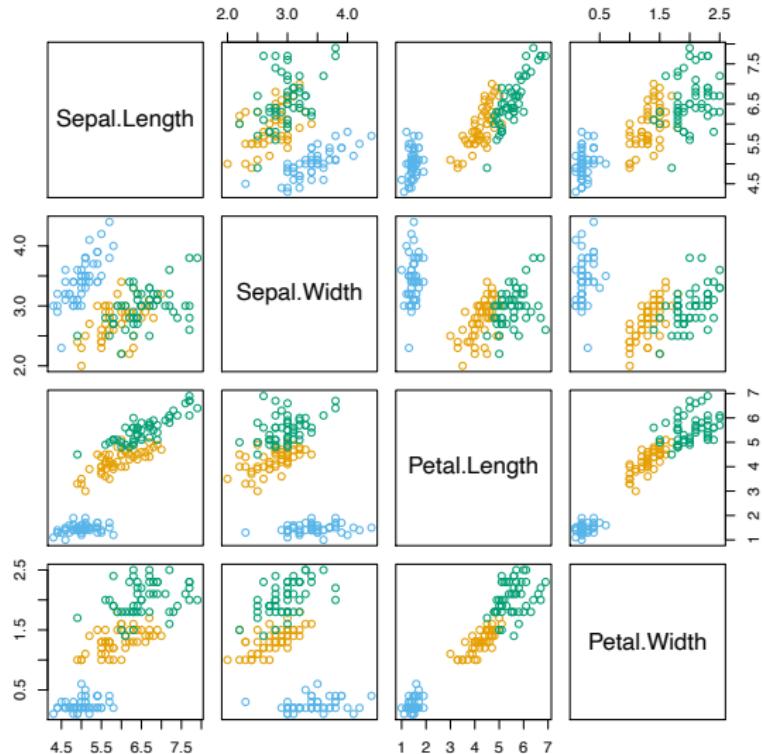
The dashed lines are known as the *Bayes decision boundaries*. Were they known, they would yield the fewest misclassification errors, among all possible classifiers.

# Fisher's Iris Data

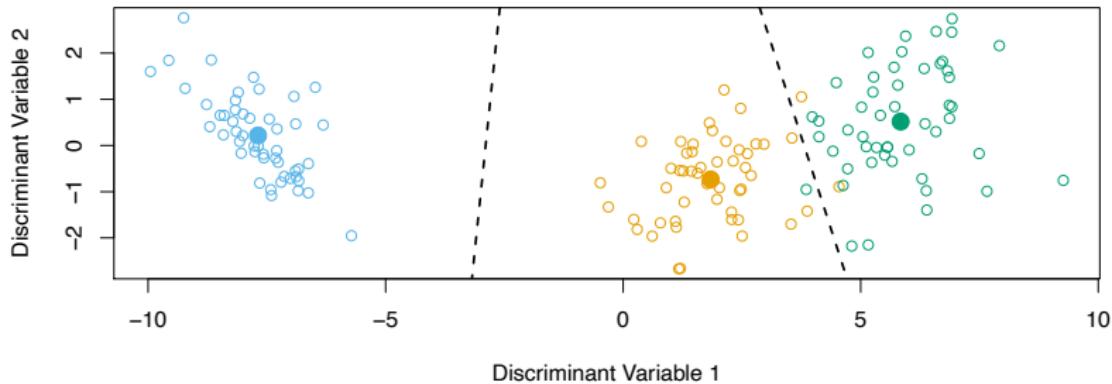
4 variables  
3 species  
50 samples/class

- Setosa
- Versicolor
- Virginica

LDA classifies all but 3 of the 150 training samples correctly.



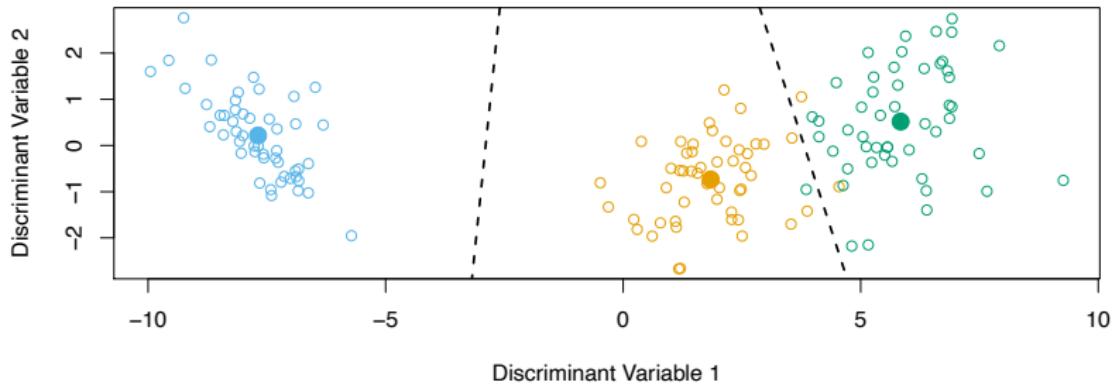
# Fisher's Discriminant Plot



When there are  $K$  classes, linear discriminant analysis can be viewed exactly in a  $K - 1$  dimensional plot.

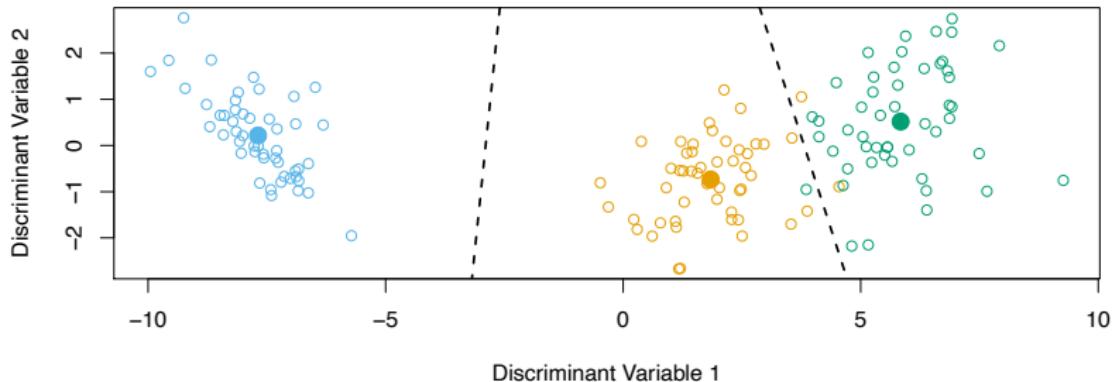
Why?

## Fisher's Discriminant Plot



When there are  $K$  classes, linear discriminant analysis can be viewed exactly in a  $K - 1$  dimensional plot.  
Why? Because it essentially classifies to the closest centroid, and they span a  $K - 1$  dimensional plane.

## Fisher's Discriminant Plot



When there are  $K$  classes, linear discriminant analysis can be viewed exactly in a  $K - 1$  dimensional plot.

Why? Because it essentially classifies to the closest centroid, and they span a  $K - 1$  dimensional plane.

Even when  $K > 3$ , we can find the “best” 2-dimensional plane for visualizing the discriminant rule.

## From $\delta_k(x)$ to probabilities

Once we have estimates  $\hat{\delta}_k(x)$ , we can turn these into estimates for class probabilities:

$$\widehat{\Pr}(Y = k | X = x) = \frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^K e^{\hat{\delta}_l(x)}}.$$

So classifying to the largest  $\hat{\delta}_k(x)$  amounts to classifying to the class for which  $\widehat{\Pr}(Y = k | X = x)$  is largest.

## From $\delta_k(x)$ to probabilities

Once we have estimates  $\hat{\delta}_k(x)$ , we can turn these into estimates for class probabilities:

$$\widehat{\Pr}(Y = k|X = x) = \frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^K e^{\hat{\delta}_l(x)}}.$$

So classifying to the largest  $\hat{\delta}_k(x)$  amounts to classifying to the class for which  $\widehat{\Pr}(Y = k|X = x)$  is largest.

When  $K = 2$ , we classify to class 2 if  $\widehat{\Pr}(Y = 2|X = x) \geq 0.5$ , else to class 1.

## LDA on Credit Data

		<i>True Default Status</i>		
		No	Yes	Total
<i>Predicted Default Status</i>	No	9644	252	9896
	Yes	23	81	104
Total		9667	333	10000

$(23 + 252)/10000$  errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting.

## LDA on Credit Data

		True Default Status		
		No	Yes	Total
Predicted Default Status	No	9644	252	9896
	Yes	23	81	104
Total		9667	333	10000

$(23 + 252)/10000$  errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since  $n = 10000$  and  $p = 2$ !

## LDA on Credit Data

		True Default Status		
		No	Yes	Total
Predicted Default Status	No	9644	252	9896
	Yes	23	81	104
Total		9667	333	10000

$(23 + 252)/10000$  errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since  $n = 10000$  and  $p = 2$ !
- If we classified to the prior — always to class **No** in this case — we would make  $333/10000$  errors, or only 3.33%.

## LDA on Credit Data

		True Default Status		
		No	Yes	Total
Predicted Default Status	No	9644	252	9896
	Yes	23	81	104
Total		9667	333	10000

$(23 + 252)/10000$  errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since  $n = 10000$  and  $p = 2$ !
- If we classified to the prior — always to class **No** in this case — we would make  $333/10000$  errors, or only 3.33%.
- Of the true **No**'s, we make  $23/9667 = 0.2\%$  errors; of the true **Yes**'s, we make  $252/333 = 75.7\%$  errors!

## Types of errors

**False positive rate:** The fraction of negative examples that are classified as positive — 0.2% in example.

**False negative rate:** The fraction of positive examples that are classified as negative — 75.7% in example.

We produced this table by classifying to class **Yes** if

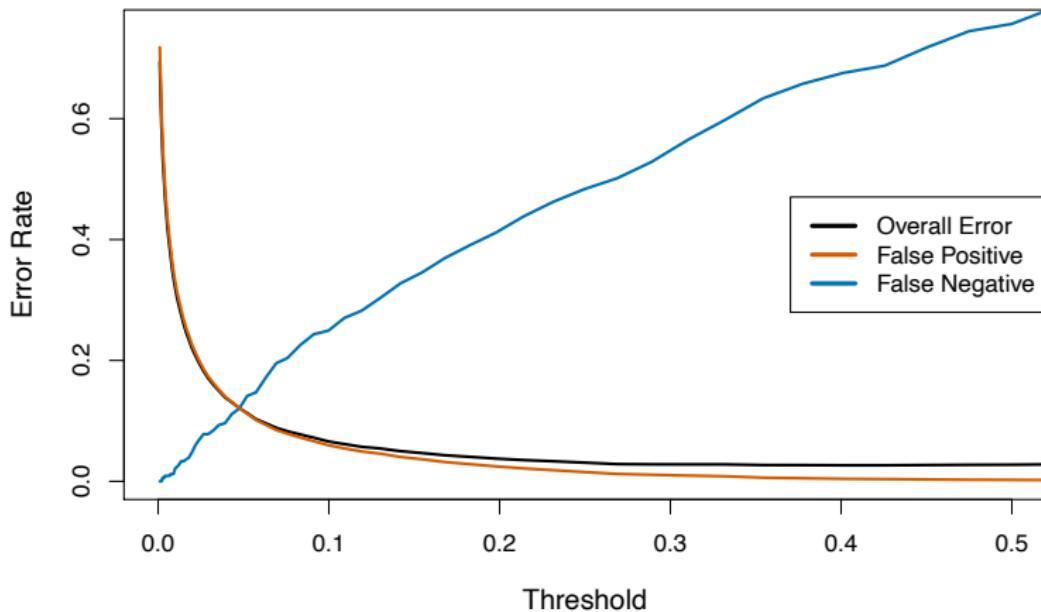
$$\widehat{\Pr}(\text{Default} = \text{Yes} | \text{Balance}, \text{Student}) \geq 0.5$$

We can change the two error rates by changing the threshold from 0.5 to some other value in  $[0, 1]$ :

$$\widehat{\Pr}(\text{Default} = \text{Yes} | \text{Balance}, \text{Student}) \geq \text{threshold},$$

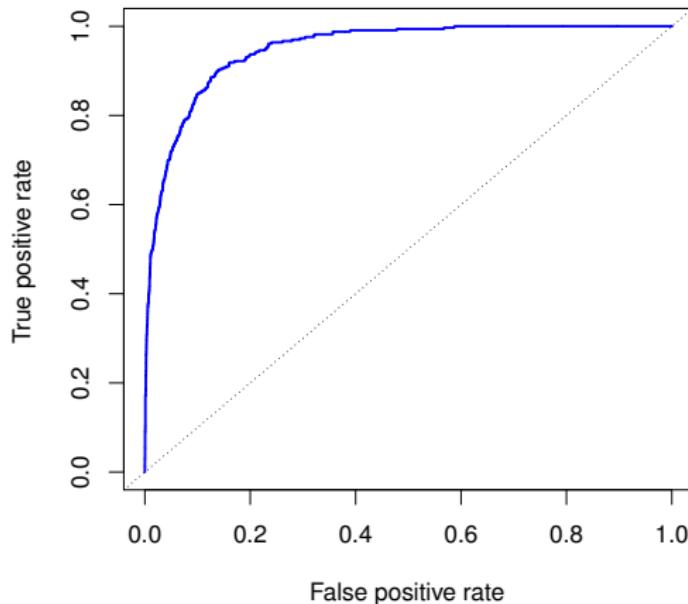
and vary *threshold*.

## Varying the *threshold*



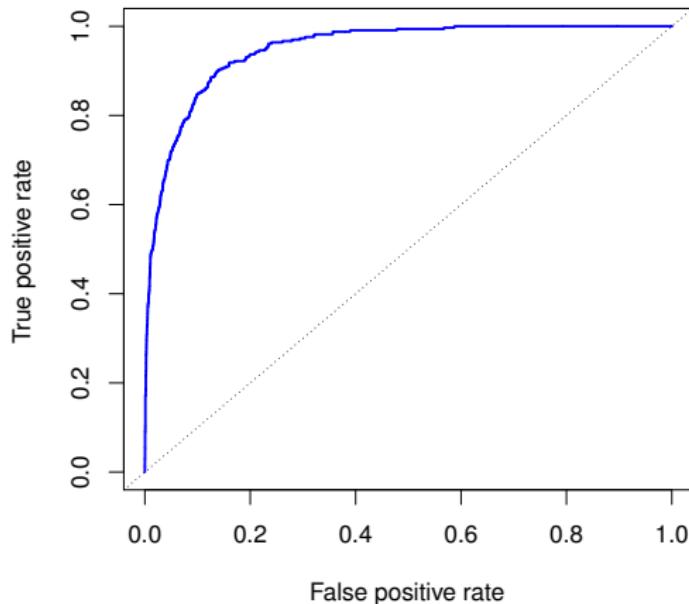
In order to reduce the false negative rate, we may want to reduce the threshold to 0.1 or less.

## ROC Curve



The *ROC plot* displays both simultaneously.

## ROC Curve



The *ROC plot* displays both simultaneously.

Sometimes we use the *AUC* or *area under the curve* to summarize the overall performance. Higher *AUC* is good.

## Other forms of Discriminant Analysis

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

When  $f_k(x)$  are Gaussian densities, with the same covariance matrix  $\Sigma$  in each class, this leads to linear discriminant analysis. By altering the forms for  $f_k(x)$ , we get different classifiers.

- With Gaussians but different  $\Sigma_k$  in each class, we get *quadratic discriminant analysis*.

## Other forms of Discriminant Analysis

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

When  $f_k(x)$  are Gaussian densities, with the same covariance matrix  $\Sigma$  in each class, this leads to linear discriminant analysis. By altering the forms for  $f_k(x)$ , we get different classifiers.

- With Gaussians but different  $\Sigma_k$  in each class, we get *quadratic discriminant analysis*.
- With  $f_k(x) = \prod_{j=1}^p f_{jk}(x_j)$  (conditional independence model) in each class we get *naive Bayes*. For Gaussian this means the  $\Sigma_k$  are diagonal.

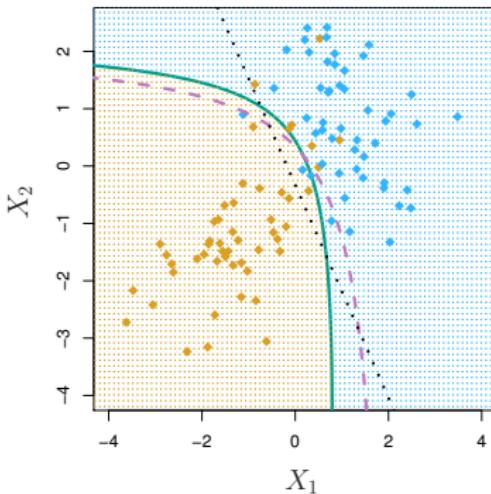
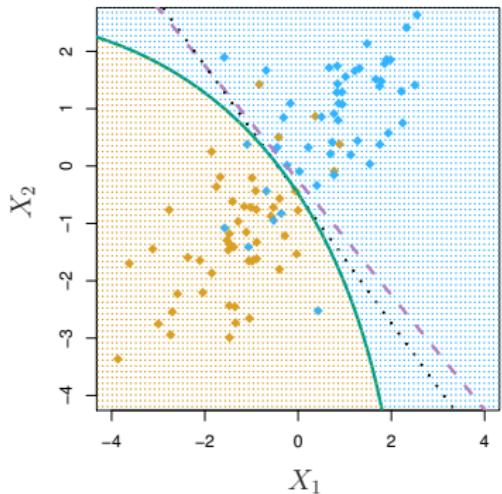
## Other forms of Discriminant Analysis

$$\Pr(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

When  $f_k(x)$  are Gaussian densities, with the same covariance matrix  $\Sigma$  in each class, this leads to linear discriminant analysis. By altering the forms for  $f_k(x)$ , we get different classifiers.

- With Gaussians but different  $\Sigma_k$  in each class, we get *quadratic discriminant analysis*.
- With  $f_k(x) = \prod_{j=1}^p f_{jk}(x_j)$  (conditional independence model) in each class we get *naive Bayes*. For Gaussian this means the  $\Sigma_k$  are diagonal.
- Many other forms, by proposing specific density models for  $f_k(x)$ , including nonparametric approaches.

# Quadratic Discriminant Analysis



$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \pi_k - \frac{1}{2} \log |\Sigma_k|$$

Because the  $\Sigma_k$  are different, the quadratic terms matter.

## Naive Bayes

Assumes features are independent in each class.

Useful when  $p$  is large, and so multivariate methods like QDA and even LDA break down.

- Gaussian naive Bayes assumes each  $\Sigma_k$  is diagonal:

$$\begin{aligned}\delta_k(x) &\propto \log \left[ \pi_k \prod_{j=1}^p f_{kj}(x_j) \right] \\ &= -\frac{1}{2} \sum_{j=1}^p \left[ \frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2} + \log \sigma_{kj}^2 \right] + \log \pi_k\end{aligned}$$

- can use for *mixed* feature vectors (qualitative and quantitative). If  $X_j$  is qualitative, replace  $f_{kj}(x_j)$  with probability mass function (histogram) over discrete categories.

Despite strong assumptions, naive Bayes often produces good classification results.

## Logistic Regression versus LDA

For a two-class problem, one can show that for LDA

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \dots + c_p x_p$$

So it has the same form as logistic regression.

The difference is in how the parameters are estimated.

## Logistic Regression versus LDA

For a two-class problem, one can show that for LDA

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \dots + c_p x_p$$

So it has the same form as logistic regression.

The difference is in how the parameters are estimated.

- Logistic regression uses the conditional likelihood based on  $\Pr(Y|X)$  (known as *discriminative learning*).

## Logistic Regression versus LDA

For a two-class problem, one can show that for LDA

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \dots + c_p x_p$$

So it has the same form as logistic regression.

The difference is in how the parameters are estimated.

- Logistic regression uses the conditional likelihood based on  $\Pr(Y|X)$  (known as *discriminative learning*).
- LDA uses the full likelihood based on  $\Pr(X, Y)$  (known as *generative learning*).

## Logistic Regression versus LDA

For a two-class problem, one can show that for LDA

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \dots + c_p x_p$$

So it has the same form as logistic regression.

The difference is in how the parameters are estimated.

- Logistic regression uses the conditional likelihood based on  $\Pr(Y|X)$  (known as *discriminative learning*).
- LDA uses the full likelihood based on  $\Pr(X, Y)$  (known as *generative learning*).
- Despite these differences, in practice the results are often very similar.

## Logistic Regression versus LDA

For a two-class problem, one can show that for LDA

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \dots + c_p x_p$$

So it has the same form as logistic regression.

The difference is in how the parameters are estimated.

- Logistic regression uses the conditional likelihood based on  $\Pr(Y|X)$  (known as *discriminative learning*).
- LDA uses the full likelihood based on  $\Pr(X, Y)$  (known as *generative learning*).
- Despite these differences, in practice the results are often very similar.

Footnote: logistic regression can also fit quadratic boundaries like QDA, by explicitly including quadratic terms in the model.

# Summary

- Logistic regression is very popular for classification, especially when  $K = 2$ .
- LDA is useful when  $n$  is small, or the classes are well separated, and Gaussian assumptions are reasonable. Also when  $K > 2$ .
- Naive Bayes is useful when  $p$  is very large.
- See Section 4.5 for some comparisons of logistic regression, LDA and KNN.

## New Topics in Chapter Four

In this section we cover three topics in Chapter 4 that are new in the second edition of ISLR:

- Multinomial logistic regression.
- Naïve Bayes classification.
- Generalized linear models.

## Multinomial Logistic Regression

Logistic regression is frequently used when the response is binary, or  $K = 2$  classes. We need a modification when there are  $K > 2$  classes. E.g. **stroke**, **drug overdose** and **epileptic seizure** for the emergency room example.

The simplest representation uses different linear functions for each class, combined with the **softmax** function to form probabilities:

$$\Pr(Y = k | X = x) = \frac{e^{\beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{kp}x_p}}{\sum_{l=1}^K e^{\beta_{l0} + \beta_{l1}x_1 + \dots + \beta_{lp}x_p}}.$$

## Multinomial Logistic Regression

Logistic regression is frequently used when the response is binary, or  $K = 2$  classes. We need a modification when there are  $K > 2$  classes. E.g. **stroke**, **drug overdose** and **epileptic seizure** for the emergency room example.

The simplest representation uses different linear functions for each class, combined with the **softmax** function to form probabilities:

$$\Pr(Y = k | X = x) = \frac{e^{\beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{kp}x_p}}{\sum_{l=1}^K e^{\beta_{l0} + \beta_{l1}x_1 + \dots + \beta_{lp}x_p}}.$$

- There is a redundancy here; we really only need  $K - 1$  functions (see the book for details).

## Multinomial Logistic Regression

Logistic regression is frequently used when the response is binary, or  $K = 2$  classes. We need a modification when there are  $K > 2$  classes. E.g. **stroke**, **drug overdose** and **epileptic seizure** for the emergency room example.

The simplest representation uses different linear functions for each class, combined with the **softmax** function to form probabilities:

$$\Pr(Y = k | X = x) = \frac{e^{\beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{kp}x_p}}{\sum_{l=1}^K e^{\beta_{l0} + \beta_{l1}x_1 + \dots + \beta_{lp}x_p}}.$$

- There is a redundancy here; we really only need  $K - 1$  functions (see the book for details).
- We fit by maximizing the **multinomial** log likelihood (cross-entropy) — a generalization of the binomial.

## Multinomial Logistic Regression

Logistic regression is frequently used when the response is binary, or  $K = 2$  classes. We need a modification when there are  $K > 2$  classes. E.g. **stroke**, **drug overdose** and **epileptic seizure** for the emergency room example.

The simplest representation uses different linear functions for each class, combined with the **softmax** function to form probabilities:

$$\Pr(Y = k | X = x) = \frac{e^{\beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{kp}x_p}}{\sum_{l=1}^K e^{\beta_{l0} + \beta_{l1}x_1 + \dots + \beta_{lp}x_p}}.$$

- There is a redundancy here; we really only need  $K - 1$  functions (see the book for details).
- We fit by maximizing the **multinomial** log likelihood (cross-entropy) — a generalization of the binomial.
- An example is given in Chapter 10 where we fit the 10-class model to the **MNIST** digit dataset.

## Generative Models and Naïve Bayes

- Logistic regression models  $\Pr(Y = k|X = x)$  directly, via the logistic function. Similarly the multinomial logistic regression uses the softmax function. These all model the *conditional distribution* of  $Y$  given  $X$ .

# Generative Models and Naïve Bayes

- Logistic regression models  $\Pr(Y = k|X = x)$  directly, via the logistic function. Similarly the multinomial logistic regression uses the softmax function. These all model the *conditional distribution* of  $Y$  given  $X$ .
- By contrast *generative models* start with the conditional distribution of  $X$  given  $Y$ , and then use *Bayes formula* to turn things around:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}.$$

- $f_k(x)$  is the density of  $X$  given  $Y = k$ ;
- $\pi_k = \Pr(Y = k)$  is the marginal probability that  $Y$  is in class  $k$ .

## Generative Models and Naïve Bayes

- Linear and quadratic discriminant analysis derive from generative models, where  $f_k(x)$  are Gaussian.

## Generative Models and Naïve Bayes

- Linear and quadratic discriminant analysis derive from generative models, where  $f_k(x)$  are Gaussian.
- Often useful if some classes are well separated — a situation where logistic regression is unstable.

## Generative Models and Naïve Bayes

- Linear and quadratic discriminant analysis derive from generative models, where  $f_k(x)$  are Gaussian.
- Often useful if some classes are well separated — a situation where logistic regression is unstable.
- Naïve Bayes assumes that the densities  $f_k(x)$  in each class *factor*:

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$$

## Generative Models and Naïve Bayes

- Linear and quadratic discriminant analysis derive from generative models, where  $f_k(x)$  are Gaussian.
- Often useful if some classes are well separated — a situation where logistic regression is unstable.
- Naïve Bayes assumes that the densities  $f_k(x)$  in each class *factor*:

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$$

- Equivalently this assumes that the features are *independent* within each class.

## Generative Models and Naïve Bayes

- Linear and quadratic discriminant analysis derive from generative models, where  $f_k(x)$  are Gaussian.
- Often useful if some classes are well separated — a situation where logistic regression is unstable.
- Naïve Bayes assumes that the densities  $f_k(x)$  in each class *factor*:

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$$

- Equivalently this assumes that the features are *independent* within each class.
- Then using Bayes formula:

$$\Pr(Y = k | X = x) = \frac{\pi_k \times f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)}{\sum_{l=1}^K \pi_l \times f_{l1}(x_1) \times f_{l2}(x_2) \times \cdots \times f_{lp}(x_p)}$$

## Naïve Bayes — Details

Why the independence assumption?

## Naïve Bayes — Details

Why the independence assumption?

- Difficult to specify and model high-dimensional densities.  
Much easier to specify one-dimensional densities.

## Naïve Bayes — Details

Why the independence assumption?

- Difficult to specify and model high-dimensional densities.  
Much easier to specify one-dimensional densities.
- Can handle *mixed* features:
  - If feature  $j$  is quantitative, can model as univariate Gaussian, for example:  $X_j|Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$ . We estimate  $\mu_{jk}$  and  $\sigma_{jk}^2$  from the data, and then plug into Gaussian density formula for  $f_{jk}(x_j)$ .

## Naïve Bayes — Details

Why the independence assumption?

- Difficult to specify and model high-dimensional densities.  
Much easier to specify one-dimensional densities.
- Can handle *mixed* features:
  - If feature  $j$  is quantitative, can model as univariate Gaussian, for example:  $X_j|Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$ . We estimate  $\mu_{jk}$  and  $\sigma_{jk}^2$  from the data, and then plug into Gaussian density formula for  $f_{jk}(x_j)$ .
  - Alternatively, can use a *histogram* estimate of the density, and directly estimate  $f_{jk}(x_j)$  by the proportion of observations in the bin into which  $x_j$  falls.

## Naïve Bayes — Details

Why the independence assumption?

- Difficult to specify and model high-dimensional densities.  
Much easier to specify one-dimensional densities.
- Can handle *mixed* features:
  - If feature  $j$  is quantitative, can model as univariate Gaussian, for example:  $X_j|Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$ . We estimate  $\mu_{jk}$  and  $\sigma_{jk}^2$  from the data, and then plug into Gaussian density formula for  $f_{jk}(x_j)$ .
  - Alternatively, can use a *histogram* estimate of the density, and directly estimate  $f_{jk}(x_j)$  by the proportion of observations in the bin into which  $x_j$  falls.
  - If feature  $j$  is qualitative, can simply model the proportion in each category. Example to follow.

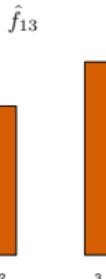
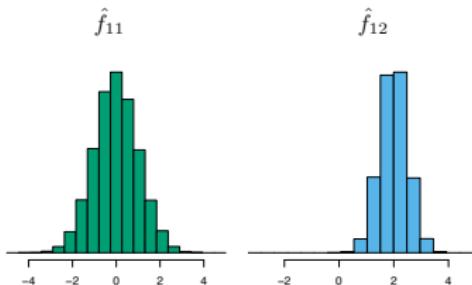
## Naïve Bayes — Details

Why the independence assumption?

- Difficult to specify and model high-dimensional densities.  
Much easier to specify one-dimensional densities.
- Can handle *mixed* features:
  - If feature  $j$  is quantitative, can model as univariate Gaussian, for example:  $X_j|Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$ . We estimate  $\mu_{jk}$  and  $\sigma_{jk}^2$  from the data, and then plug into Gaussian density formula for  $f_{jk}(x_j)$ .
  - Alternatively, can use a *histogram* estimate of the density, and directly estimate  $f_{jk}(x_j)$  by the proportion of observations in the bin into which  $x_j$  falls.
  - If feature  $j$  is qualitative, can simply model the proportion in each category. Example to follow.
- Somewhat unrealistic but extremely useful in many cases.  
Despite its simplicity, often shows good classification performance due to reduced variance.

# Naïve Bayes — Toy Example

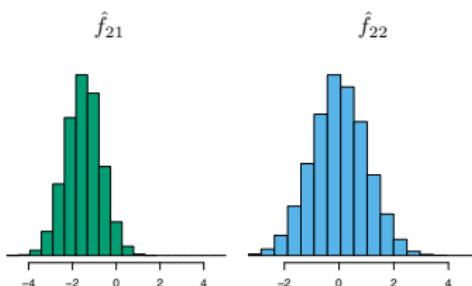
Density estimates for class k=1



$$x^* = (.4, 1.5, 1)$$

$$\hat{\pi}_1 = \hat{\pi}_2 = 0.5$$

Density estimates for class k=2



$$\hat{f}_{11}(0.4) = 0.368$$

$$\hat{f}_{12}(1.5) = 0.484$$

$$\hat{f}_{13}(1) = 0.226$$

$$\hat{f}_{21}(0.4) = 0.030$$

$$\hat{f}_{22}(1.5) = 0.130$$

$$\hat{f}_{23}(1) = 0.616$$

$$\Pr(Y = 1|X = x^*) = 0.944 \text{ and } \Pr(Y = 2|X = x^*) = 0.056$$

## Naïve Bayes and GAMs



$$\begin{aligned}\log \left( \frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)} \right) &= \log \left( \frac{\pi_k f_k(x)}{\pi_K f_K(x)} \right) \\ &= \log \left( \frac{\pi_k \prod_{j=1}^p f_{kj}(x_j)}{\pi_K \prod_{j=1}^p f_{Kj}(x_j)} \right) \\ &= \log \left( \frac{\pi_k}{\pi_K} \right) + \sum_{j=1}^p \log \left( \frac{f_{kj}(x_j)}{f_{Kj}(x_j)} \right) \\ &= a_k + \sum_{j=1}^p g_{kj}(x_j),\end{aligned}$$

where  $a_k = \log \left( \frac{\pi_k}{\pi_K} \right)$  and  $g_{kj}(x_j) = \log \left( \frac{f_{kj}(x_j)}{f_{Kj}(x_j)} \right)$ .

Hence, the Naïve Bayes model takes the form of a *generalized additive model* from Chapter 7.

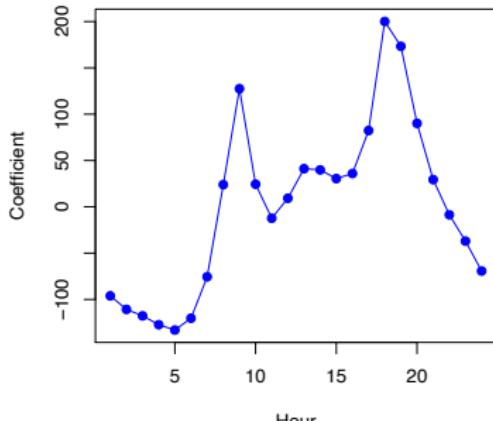
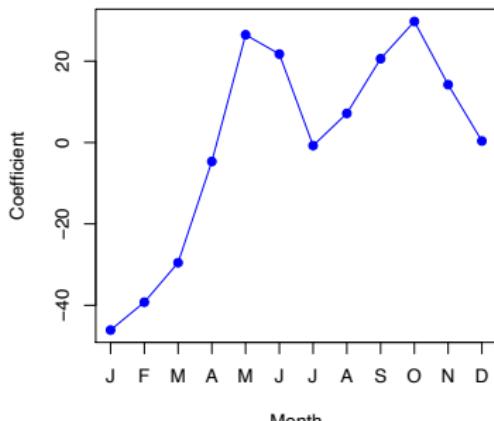
# Generalized Linear Models

- Linear regression is used for quantitative responses.
- Linear logistic regression is the counterpart for a binary response, and models the logit of the probability as a linear model.
- Other response types exist, such as non-negative responses, skewed distributions, and more.
- *Generalized linear models* provide a unified framework for dealing with many different response types.

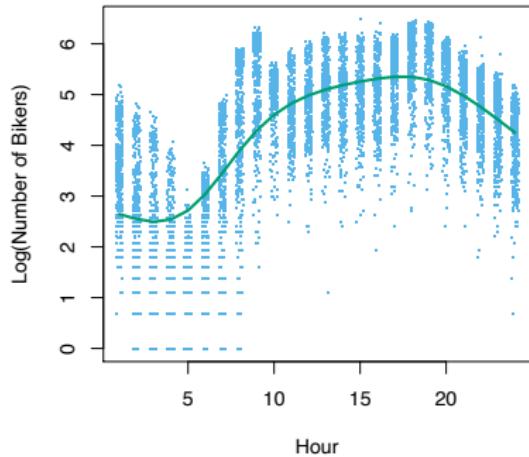
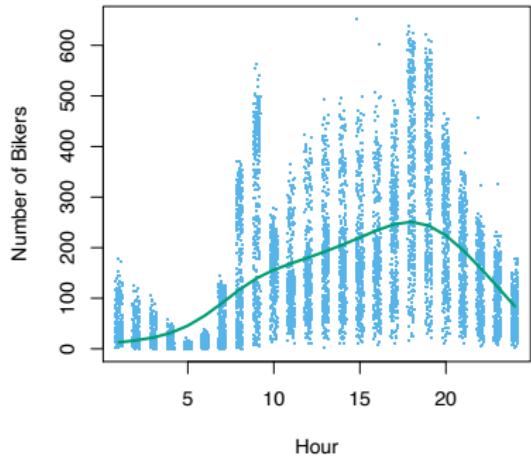
## Example: Bikeshare Data

Linear regression with response **bikers**: number of hourly users in bikeshare program in Washington, DC.

	Coefficient	Std. error	z-statistic	p-value
<b>Intercept</b>	73.60	5.13	14.34	0.00
<b>workingday</b>	1.27	1.78	0.71	0.48
<b>temp</b>	157.21	10.26	15.32	0.00
<b>weathersit [cloudy/misty]</b>	-12.89	1.96	-6.56	0.00
<b>weathersit [light rain/snow]</b>	-66.49	2.97	-22.43	0.00
<b>weathersit [heavy rain/snow]</b>	-109.75	76.67	-1.43	0.15

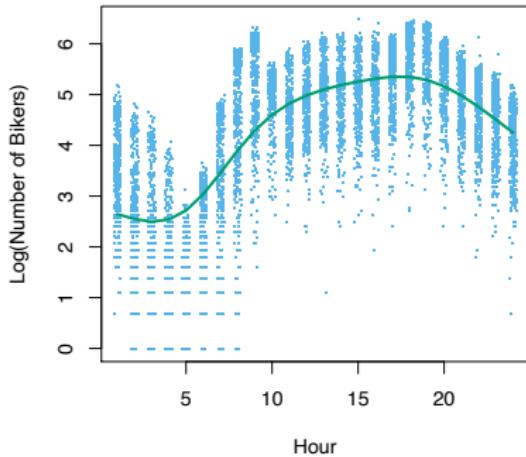
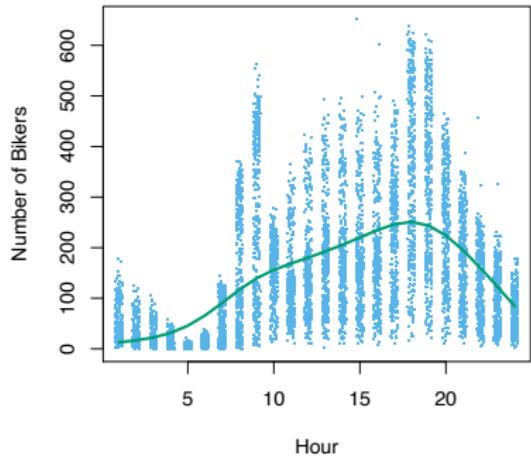


# Mean/Variance Relationship



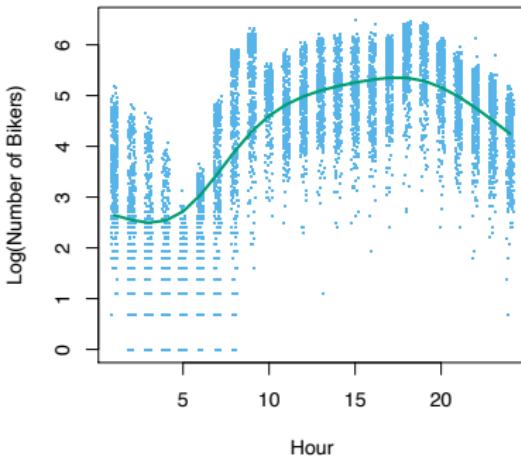
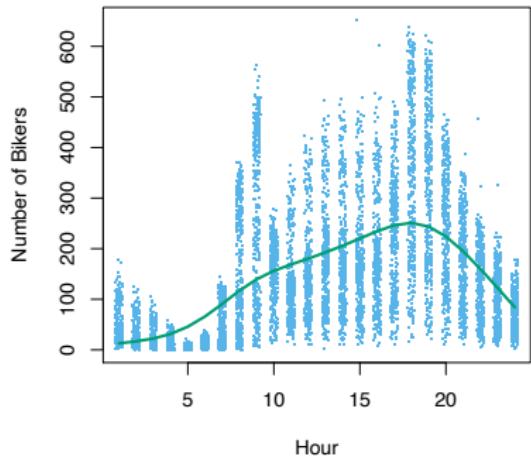
- In left plot we see that the variance mostly increases with the mean.

# Mean/Variance Relationship



- In left plot we see that the variance mostly increases with the mean.
- 10% of linear model predictions are negative! (not shown here.)

# Mean/Variance Relationship



- In left plot we see that the variance mostly increases with the mean.
- 10% of linear model predictions are negative! (not shown here.)
- Taking  $\log(\text{bikers})$  alleviates this, but has its own problems: e.g. predictions are on the wrong scale, and some counts are zero!

## Poisson Regression Model

- Poisson distribution is useful for modeling counts:

$$\Pr(Y = k) = \frac{e^{-\lambda} \lambda^k}{k!} \quad \text{for } k = 0, 1, 2, \dots$$

- $\lambda = \text{E}(Y) = \text{Var}(Y)$  — i.e. there is a mean/variance dependence.

## Poisson Regression Model

- Poisson distribution is useful for modeling counts:

$$\Pr(Y = k) = \frac{e^{-\lambda} \lambda^k}{k!} \quad \text{for } k = 0, 1, 2, \dots$$

- $\lambda = \text{E}(Y) = \text{Var}(Y)$  — i.e. there is a mean/variance dependence.
- With covariates, we model

$$\log(\lambda(X_1, \dots, X_p)) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$$

or equivalently

$$\lambda(X_1, \dots, X_p) = e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}.$$

## Poisson Regression Model

- Poisson distribution is useful for modeling counts:

$$\Pr(Y = k) = \frac{e^{-\lambda} \lambda^k}{k!} \quad \text{for } k = 0, 1, 2, \dots$$

- $\lambda = \text{E}(Y) = \text{Var}(Y)$  — i.e. there is a mean/variance dependence.
- With covariates, we model

$$\log(\lambda(X_1, \dots, X_p)) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$$

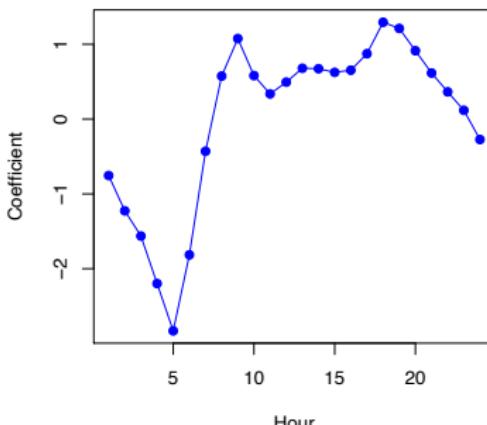
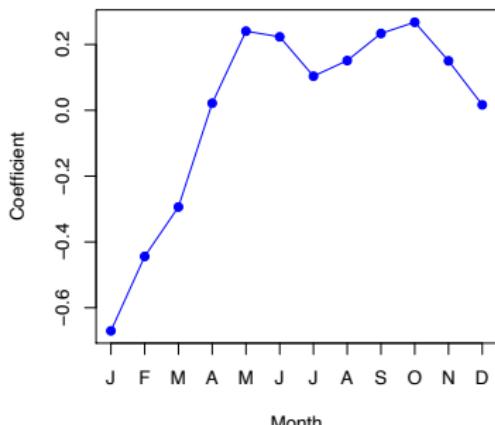
or equivalently

$$\lambda(X_1, \dots, X_p) = e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}.$$

- Model automatically guarantees that the predictions are non-negative.

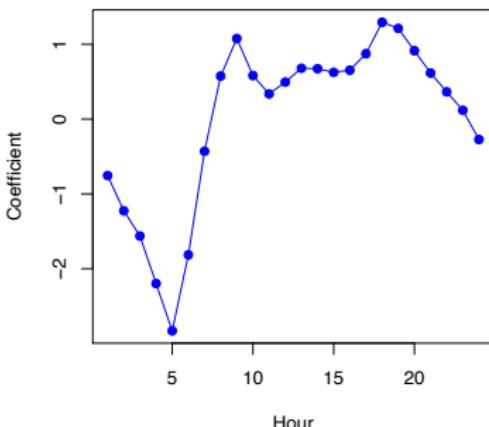
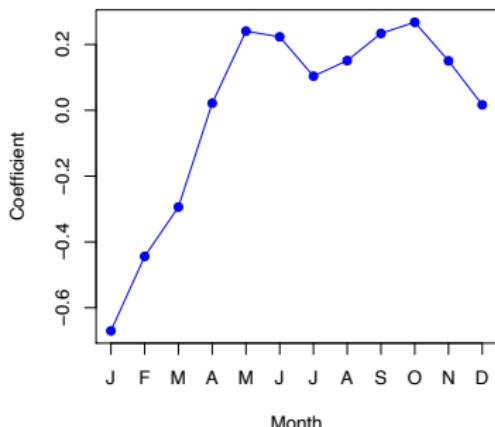
## Poisson Regression on Bikeshare Data

	Coefficient	Std. error	z-statistic	p-value
Intercept	4.12	0.01	683.96	0.00
workingday	0.01	0.00	7.5	0.00
temp	0.79	0.01	68.43	0.00
weathersit [cloudy/misty]	-0.08	0.00	-34.53	0.00
weathersit [light rain/snow]	-0.58	0.00	-141.91	0.00
weathersit [heavy rain/snow]	-0.93	0.17	-5.55	0.00



## Poisson Regression on Bikeshare Data

	Coefficient	Std. error	z-statistic	p-value
Intercept	4.12	0.01	683.96	0.00
workingday	0.01	0.00	7.5	0.00
temp	0.79	0.01	68.43	0.00
weathersit [cloudy/misty]	-0.08	0.00	-34.53	0.00
weathersit [light rain/snow]	-0.58	0.00	-141.91	0.00
weathersit [heavy rain/snow]	-0.93	0.17	-5.55	0.00



\*In this case the variance is somewhat larger than the mean — a situation known as **overdispersion** — so the p-values are misleadingly small.

## Generalized Linear Models

- We have covered three GLMs in this course: Gaussian, binomial and Poisson.
- They each have a characteristic *link* function. This is the transformation of the mean that is represented by a linear model:

$$\eta(\text{E}(Y|X_1, X_2, \dots, X_p)) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p.$$

The link functions for linear, logistic and Poisson regression are  $\eta(\mu) = \mu$ ,  $\eta(\mu) = \log(\mu/(1 - \mu))$ , and  $\eta(\mu) = \log(\mu)$ , respectively.

- They also each have characteristic *variance* functions.
- The models are fit by maximum-likelihood, and model summaries are produced by `glm()` in R.
- Other GLMS include *Gamma*, *Negative-binomial*, *Inverse Gaussian* and more.

# Cross-validation and the Bootstrap

- In the section we discuss two *resampling* methods: cross-validation and the bootstrap.

# Cross-validation and the Bootstrap

- In the section we discuss two *resampling* methods: cross-validation and the bootstrap.
- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.

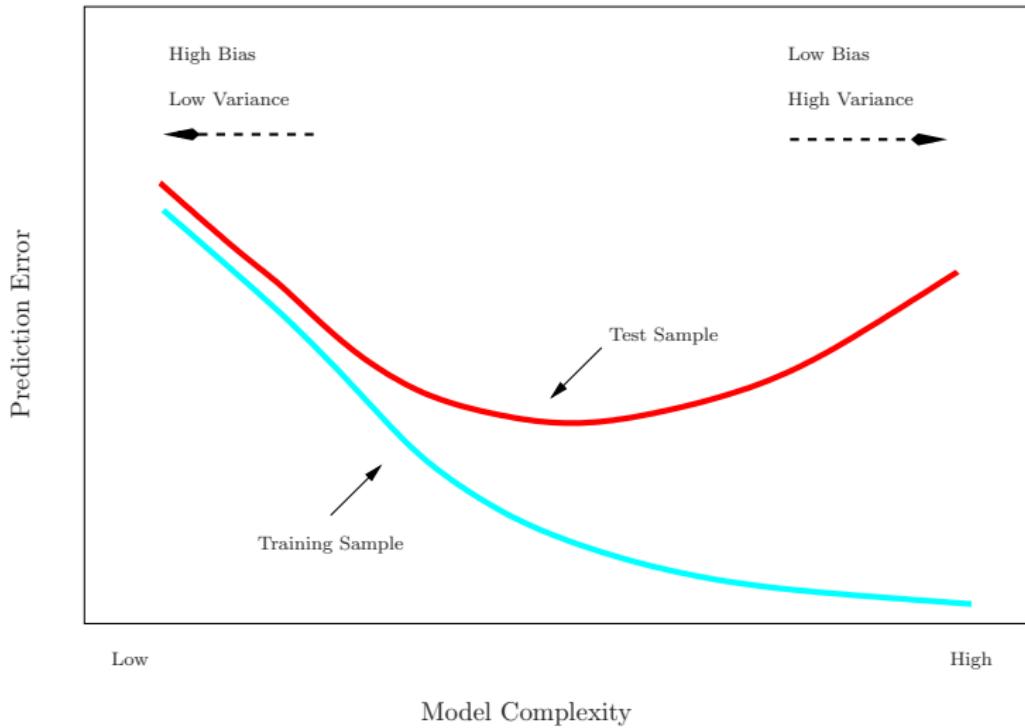
# Cross-validation and the Bootstrap

- In the section we discuss two *resampling* methods: cross-validation and the bootstrap.
- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.
- For example, they provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates

## Training Error versus Test error

- Recall the distinction between the *test error* and the *training error*:
- The *test error* is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
- In contrast, the *training error* can be easily calculated by applying the statistical learning method to the observations used in its training.
- But the training error rate often is quite different from the test error rate, and in particular the former can *dramatically underestimate* the latter.

# Training- versus Test-Set Performance



## More on prediction-error estimates

- Best solution: a large designated test set. Often not available
- Some methods make a *mathematical adjustment* to the training error rate in order to estimate the test error rate. These include the *Cp statistic*, *AIC* and *BIC*. They are discussed elsewhere in this course
- Here we instead consider a class of methods that estimate the test error by *holding out* a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations

## Validation-set approach

- Here we randomly divide the available set of samples into two parts: a *training set* and a *validation* or *hold-out set*.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error. This is typically assessed using MSE in the case of a quantitative response and misclassification rate in the case of a qualitative (discrete) response.

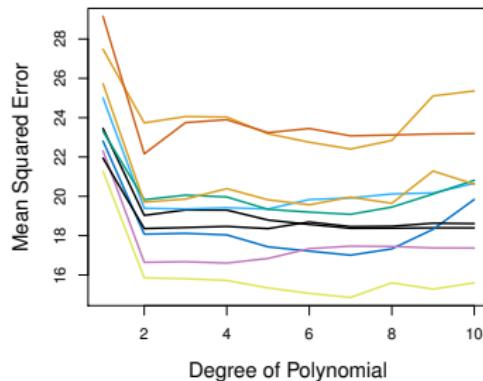
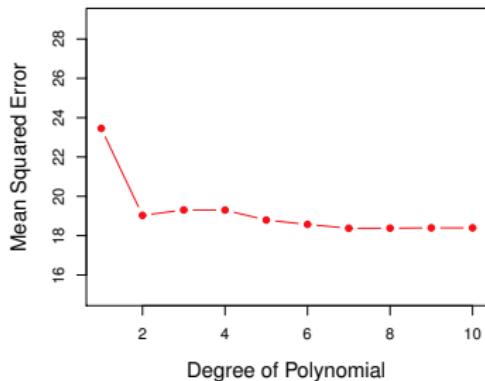
# The Validation process



A random splitting into two halves: left part is training set,  
right part is validation set

## Example: automobile data

- Want to compare linear vs higher-order polynomial terms in a linear regression
- We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.



*Left panel shows single split; right panel shows multiple splits*

## Drawbacks of validation set approach

- the validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to *overestimate* the test error for the model fit on the entire data set.

## Drawbacks of validation set approach

- the validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to *overestimate* the test error for the model fit on the entire data set. *Why?*

## $K$ -fold Cross-validation

- *Widely used approach* for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into  $K$  equal-sized parts. We leave out part  $k$ , fit the model to the other  $K - 1$  parts (combined), and then obtain predictions for the left-out  $k$ th part.
- This is done in turn for each part  $k = 1, 2, \dots, K$ , and then the results are combined.

## $K$ -fold Cross-validation in detail

Divide data into  $K$  roughly equal-sized parts ( $K = 5$  here)

1	2	3	4	5
Validation	Train	Train	Train	Train

## The details

- Let the  $K$  parts be  $C_1, C_2, \dots, C_K$ , where  $C_k$  denotes the indices of the observations in part  $k$ . There are  $n_k$  observations in part  $k$ : if  $N$  is a multiple of  $K$ , then  $n_k = n/K$ .
- Compute

$$\text{CV}_{(K)} = \sum_{k=1}^K \frac{n_k}{n} \text{MSE}_k$$

where  $\text{MSE}_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$ , and  $\hat{y}_i$  is the fit for observation  $i$ , obtained from the data with part  $k$  removed.

## The details

- Let the  $K$  parts be  $C_1, C_2, \dots, C_K$ , where  $C_k$  denotes the indices of the observations in part  $k$ . There are  $n_k$  observations in part  $k$ : if  $N$  is a multiple of  $K$ , then  $n_k = n/K$ .
- Compute

$$\text{CV}_{(K)} = \sum_{k=1}^K \frac{n_k}{n} \text{MSE}_k$$

where  $\text{MSE}_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$ , and  $\hat{y}_i$  is the fit for observation  $i$ , obtained from the data with part  $k$  removed.

- Setting  $K = n$  yields  $n$ -fold or *leave-one out cross-validation* (LOOCV).

## A nice special case!

- With least-squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \hat{y}_i}{1 - h_i} \right)^2,$$

where  $\hat{y}_i$  is the  $i$ th fitted value from the original least squares fit, and  $h_i$  is the leverage (diagonal of the “hat” matrix; see book for details.) This is like the ordinary MSE, except the  $i$ th residual is divided by  $1 - h_i$ .

## A nice special case!

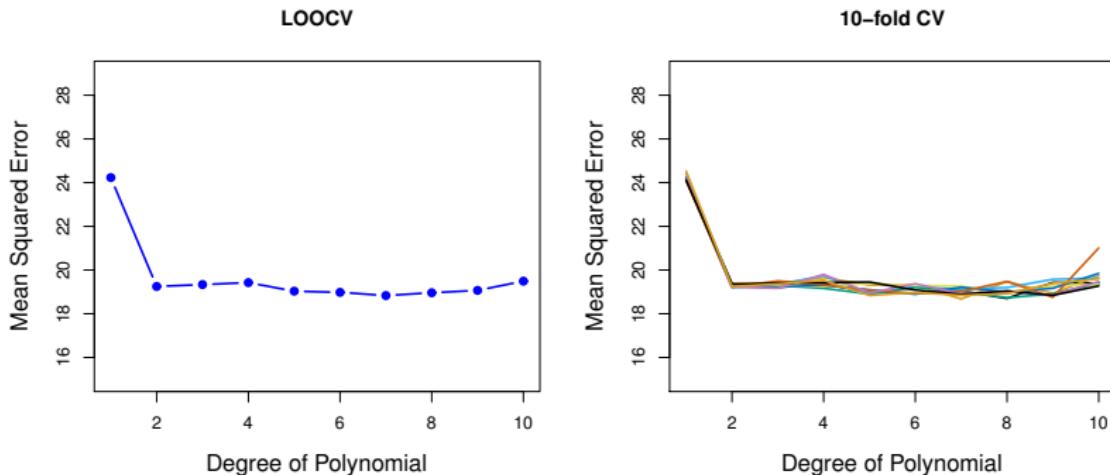
- With least-squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \hat{y}_i}{1 - h_i} \right)^2,$$

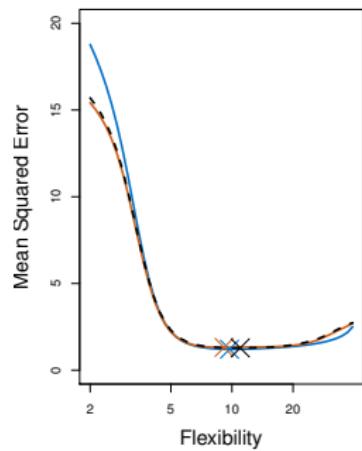
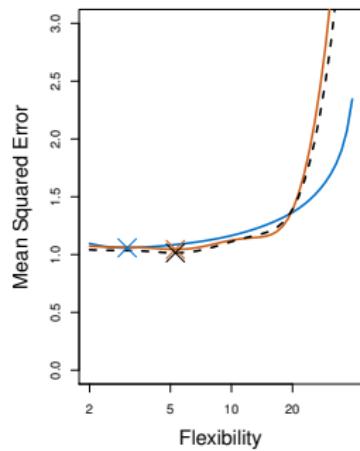
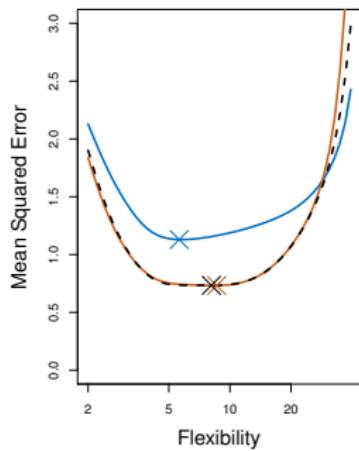
where  $\hat{y}_i$  is the  $i$ th fitted value from the original least squares fit, and  $h_i$  is the leverage (diagonal of the “hat” matrix; see book for details.) This is like the ordinary MSE, except the  $i$ th residual is divided by  $1 - h_i$ .

- LOOCV sometimes useful, but typically doesn’t *shake up* the data enough. The estimates from each fold are highly correlated and hence their average can have high variance.
- a better choice is  $K = 5$  or  $10$ .

# Auto data revisited



# True and estimated test MSE for the simulated data



## Other issues with Cross-validation

- Since each training set is only  $(K - 1)/K$  as big as the original training set, the estimates of prediction error will typically be biased upward.

## Other issues with Cross-validation

- Since each training set is only  $(K - 1)/K$  as big as the original training set, the estimates of prediction error will typically be biased upward. *Why?*

## Other issues with Cross-validation

- Since each training set is only  $(K - 1)/K$  as big as the original training set, the estimates of prediction error will typically be biased upward. *Why?*
- This bias is minimized when  $K = n$  (LOOCV), but this estimate has high variance, as noted earlier.
- $K = 5$  or  $10$  provides a good compromise for this bias-variance tradeoff.

## Cross-Validation for Classification Problems

- We divide the data into  $K$  roughly equal-sized parts  $C_1, C_2, \dots, C_K$ .  $C_k$  denotes the indices of the observations in part  $k$ . There are  $n_k$  observations in part  $k$ : if  $n$  is a multiple of  $K$ , then  $n_k = n/K$ .
- Compute

$$\text{CV}_K = \sum_{k=1}^K \frac{n_k}{n} \text{Err}_k$$

where  $\text{Err}_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i) / n_k$ .

- The estimated standard deviation of  $\text{CV}_K$  is

$$\widehat{\text{SE}}(\text{CV}_K) = \sqrt{\frac{1}{K} \sum_{k=1}^K \frac{(\text{Err}_k - \overline{\text{Err}}_k)^2}{K-1}}$$

- This is a useful estimate, but strictly speaking, not quite valid.

## Cross-Validation for Classification Problems

- We divide the data into  $K$  roughly equal-sized parts  $C_1, C_2, \dots, C_K$ .  $C_k$  denotes the indices of the observations in part  $k$ . There are  $n_k$  observations in part  $k$ : if  $n$  is a multiple of  $K$ , then  $n_k = n/K$ .
- Compute

$$\text{CV}_K = \sum_{k=1}^K \frac{n_k}{n} \text{Err}_k$$

where  $\text{Err}_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i) / n_k$ .

- The estimated standard deviation of  $\text{CV}_K$  is

$$\widehat{\text{SE}}(\text{CV}_K) = \sqrt{\frac{1}{K} \sum_{k=1}^K \frac{(\text{Err}_k - \overline{\text{Err}}_k)^2}{K-1}}$$

- This is a useful estimate, but strictly speaking, not quite valid. *Why not?*

## Cross-validation: right and wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these 100 predictors.

How do we estimate the test set performance of this classifier?

## Cross-validation: right and wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these 100 predictors.

How do we estimate the test set performance of this classifier?

Can we apply cross-validation in step 2, forgetting about step 1?

# NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50%, but the CV error estimate that ignores Step 1 is zero!

# NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50%, but the CV error estimate that ignores Step 1 is zero!

*Try to do this yourself*

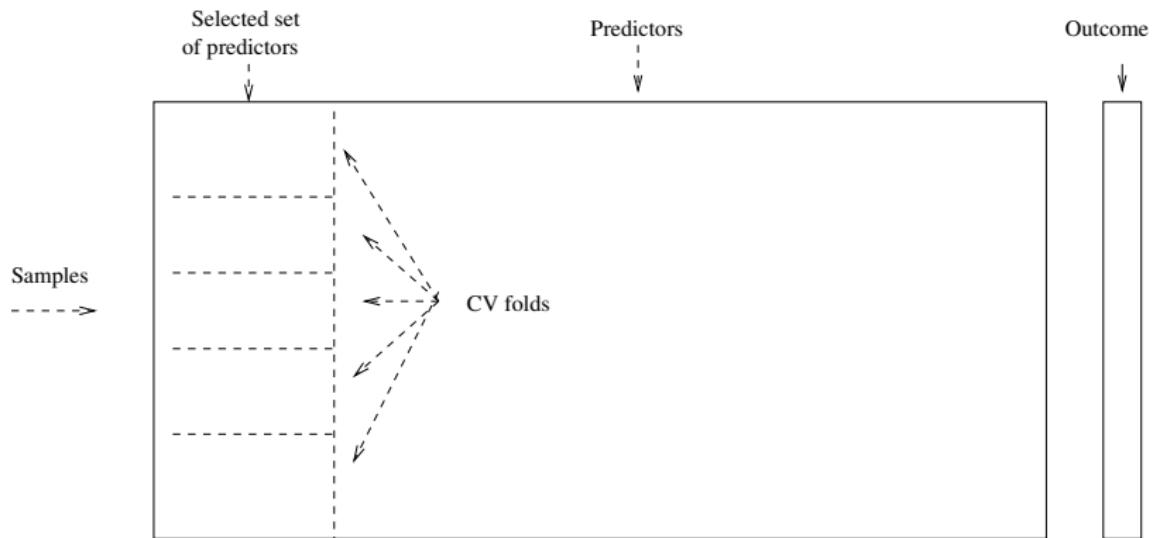
# NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50%, but the CV error estimate that ignores Step 1 is zero!  
*Try to do this yourself*
- We have seen this error made in many high profile genomics papers.

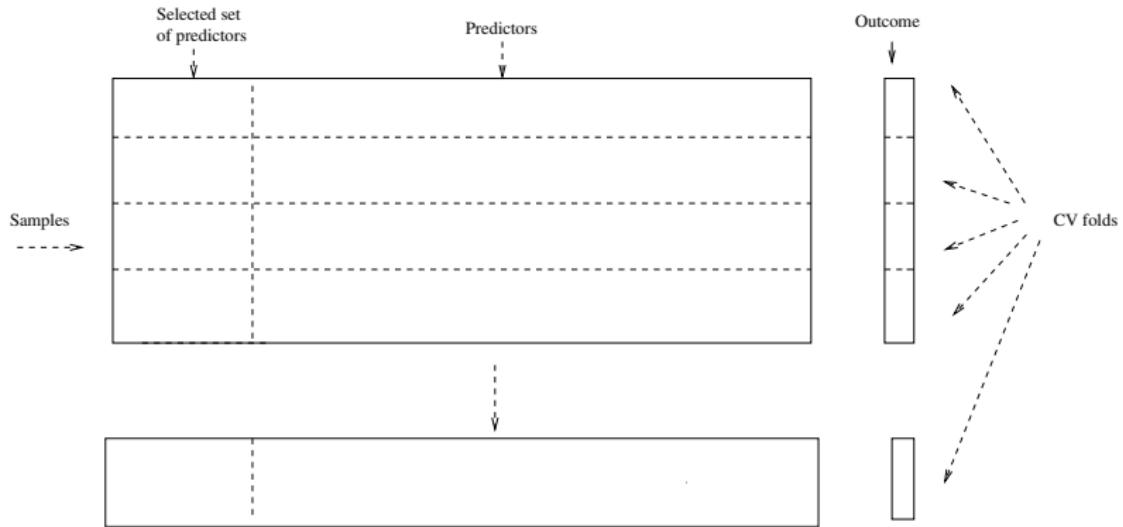
## The Wrong and Right Way

- *Wrong:* Apply cross-validation in step 2.
- *Right:* Apply cross-validation to steps 1 and 2.

# Wrong Way



# Right Way



# The Bootstrap

- The *bootstrap* is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

## Where does the name came from?

- The use of the term bootstrap derives from the phrase *to pull oneself up by one's bootstraps*, widely thought to be based on one of the eighteenth century “The Surprising Adventures of Baron Munchausen” by Rudolph Erich Raspe:

*The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.*

- It is not the same as the term “bootstrap” used in computer science meaning to “boot” a computer from a set of core instructions, though the derivation is similar.

## A simple example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of  $X$  and  $Y$ , respectively, where  $X$  and  $Y$  are random quantities.
- We will invest a fraction  $\alpha$  of our money in  $X$ , and will invest the remaining  $1 - \alpha$  in  $Y$ .
- We wish to choose  $\alpha$  to minimize the total risk, or variance, of our investment. In other words, we want to minimize  $\text{Var}(\alpha X + (1 - \alpha)Y)$ .

## A simple example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of  $X$  and  $Y$ , respectively, where  $X$  and  $Y$  are random quantities.
- We will invest a fraction  $\alpha$  of our money in  $X$ , and will invest the remaining  $1 - \alpha$  in  $Y$ .
- We wish to choose  $\alpha$  to minimize the total risk, or variance, of our investment. In other words, we want to minimize  $\text{Var}(\alpha X + (1 - \alpha)Y)$ .
- One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

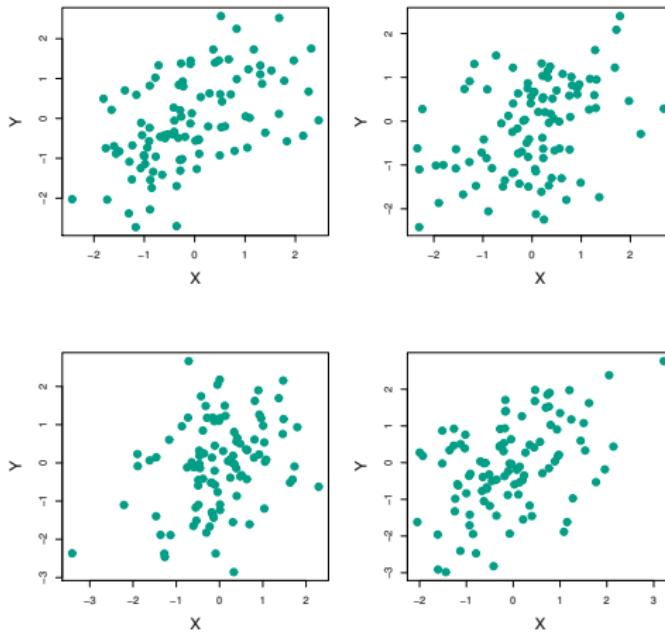
where  $\sigma_X^2 = \text{Var}(X)$ ,  $\sigma_Y^2 = \text{Var}(Y)$ , and  $\sigma_{XY} = \text{Cov}(X, Y)$ .

## Example continued

- But the values of  $\sigma_X^2$ ,  $\sigma_Y^2$ , and  $\sigma_{XY}$  are unknown.
- We can compute estimates for these quantities,  $\hat{\sigma}_X^2$ ,  $\hat{\sigma}_Y^2$ , and  $\hat{\sigma}_{XY}$ , using a data set that contains measurements for  $X$  and  $Y$ .
- We can then estimate the value of  $\alpha$  that minimizes the variance of our investment using

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

## Example continued



*Each panel displays 100 simulated returns for investments X and Y. From left to right and top to bottom, the resulting estimates for  $\alpha$  are 0.576, 0.532, 0.657, and 0.651.*

## Example continued

- To estimate the standard deviation of  $\hat{\alpha}$ , we repeated the process of simulating 100 paired observations of  $X$  and  $Y$ , and estimating  $\alpha$  1,000 times.
- We thereby obtained 1,000 estimates for  $\alpha$ , which we can call  $\hat{\alpha}_1, \hat{\alpha}_2, \dots, \hat{\alpha}_{1000}$ .
- The left-hand panel of the Figure on slide 29 displays a histogram of the resulting estimates.
- For these simulations the parameters were set to  $\sigma_X^2 = 1$ ,  $\sigma_Y^2 = 1.25$ , and  $\sigma_{XY} = 0.5$ , and so we know that the true value of  $\alpha$  is 0.6 (indicated by the red line).

## Example continued

- The mean over all 1,000 estimates for  $\alpha$  is

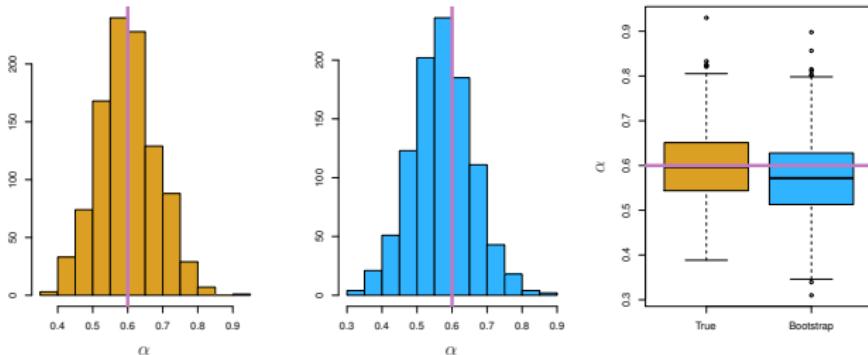
$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996,$$

very close to  $\alpha = 0.6$ , and the standard deviation of the estimates is

$$\sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083.$$

- This gives us a very good idea of the accuracy of  $\hat{\alpha}$ :  $SE(\hat{\alpha}) \approx 0.083$ .
- So roughly speaking, for a random sample from the population, we would expect  $\hat{\alpha}$  to differ from  $\alpha$  by approximately 0.08, on average.

# Results

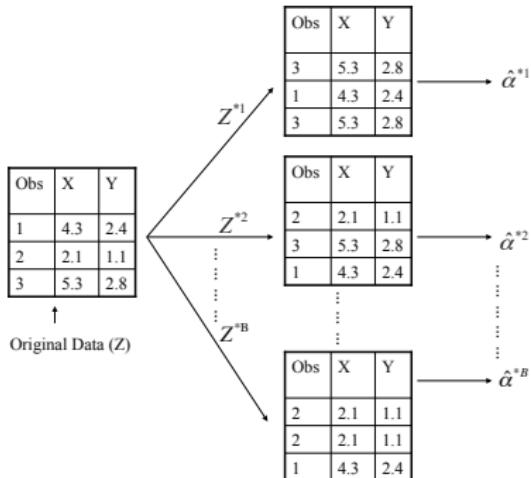


*Left:* A histogram of the estimates of  $\alpha$  obtained by generating 1,000 simulated data sets from the true population. *Center:* A histogram of the estimates of  $\alpha$  obtained from 1,000 bootstrap samples from a single data set. *Right:* The estimates of  $\alpha$  displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of  $\alpha$ .

## Now back to the real world

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set *with replacement*.
- Each of these “bootstrap data sets” is created by sampling *with replacement*, and is the *same size* as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.

## Example with just 3 observations



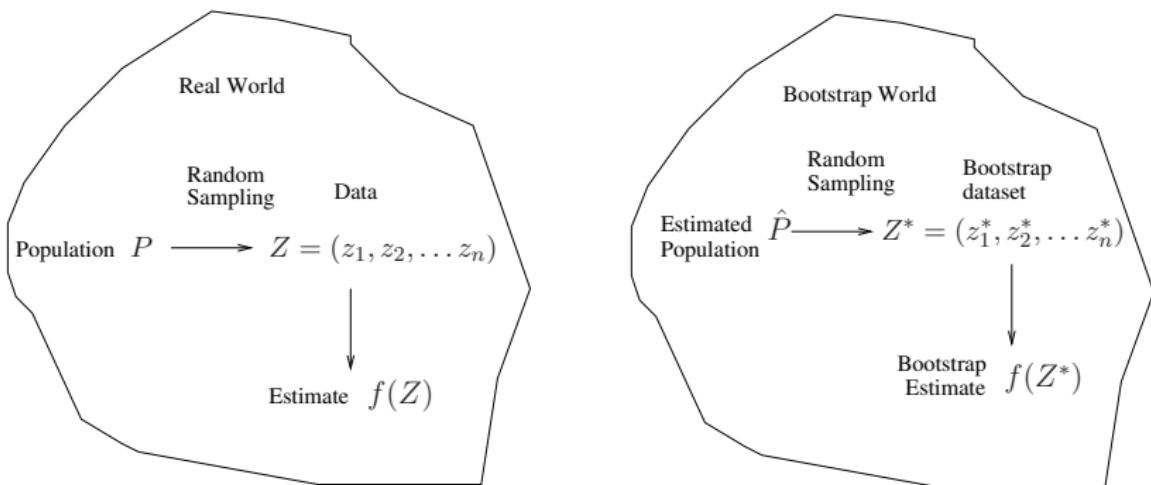
A graphical illustration of the bootstrap approach on a small sample containing  $n = 3$  observations. Each bootstrap data set contains  $n$  observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of  $\alpha$

- Denoting the first bootstrap data set by  $Z^{*1}$ , we use  $Z^{*1}$  to produce a new bootstrap estimate for  $\alpha$ , which we call  $\hat{\alpha}^{*1}$
- This procedure is repeated  $B$  times for some large value of  $B$  (say 100 or 1000), in order to produce  $B$  different bootstrap data sets,  $Z^{*1}, Z^{*2}, \dots, Z^{*B}$ , and  $B$  corresponding  $\alpha$  estimates,  $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \dots, \hat{\alpha}^{*B}$ .
- We estimate the standard error of these bootstrap estimates using the formula

$$\text{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1} \sum_{r=1}^B (\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^*)^2}.$$

- This serves as an estimate of the standard error of  $\hat{\alpha}$  estimated from the original data set. See center and right panels of Figure on slide 29. Bootstrap results are in blue. For this example  $\text{SE}_B(\hat{\alpha}) = 0.087$ .

# A general picture for the bootstrap



## The bootstrap in general

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).

## The bootstrap in general

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).
- We can instead create blocks of consecutive observations, and sample those with replacements. Then we paste together sampled blocks to obtain a bootstrap dataset.

## Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is (.43, .72).
- This represents an approximate 90% confidence interval for the true  $\alpha$ .

## Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is (.43, .72).
- This represents an approximate 90% confidence interval for the true  $\alpha$ . *How do we interpret this confidence interval?*

## Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is (.43, .72).
- This represents an approximate 90% confidence interval for the true  $\alpha$ . *How do we interpret this confidence interval?*
- The above interval is called a *Bootstrap Percentile* confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success.

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success. *Why?*

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample.

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error.

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*

## Can the bootstrap estimate prediction error?

- In cross-validation, each of the  $K$  validation folds is distinct from the other  $K - 1$  folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around—with original sample = training sample, bootstrap dataset = validation sample—is worse!

## Removing the overlap

- Can partly fix this problem by only using predictions for those observations that did not (by chance) occur in the current bootstrap sample.
- But the method gets complicated, and in the end, cross-validation provides a simpler, more attractive approach for estimating prediction error.

## Pre-validation

- In microarray and other genomic studies, an important problem is to compare a predictor of disease outcome derived from a large number of “biomarkers” to standard clinical predictors.
- Comparing them on the same dataset that was used to derive the biomarker predictor can lead to results strongly biased in favor of the biomarker predictor.

## Pre-validation

- In microarray and other genomic studies, an important problem is to compare a predictor of disease outcome derived from a large number of “biomarkers” to standard clinical predictors.
- Comparing them on the same dataset that was used to derive the biomarker predictor can lead to results strongly biased in favor of the biomarker predictor.
- *Pre-validation* can be used to make a fairer comparison between the two sets of predictors.

## Motivating example

An example of this problem arose in the paper of van't Veer *et al.* *Nature* (2002). Their microarray data has 4918 genes measured over 78 cases, taken from a study of breast cancer. There are 44 cases in the good prognosis group and 34 in the poor prognosis group. A “microarray” predictor was constructed as follows:

1. 70 genes were selected, having largest absolute correlation with the 78 class labels.
2. Using these 70 genes, a nearest-centroid classifier  $C(x)$  was constructed.
3. Applying the classifier to the 78 microarrays gave a dichotomous predictor  $z_i = C(x_i)$  for each case  $i$ .

## Results

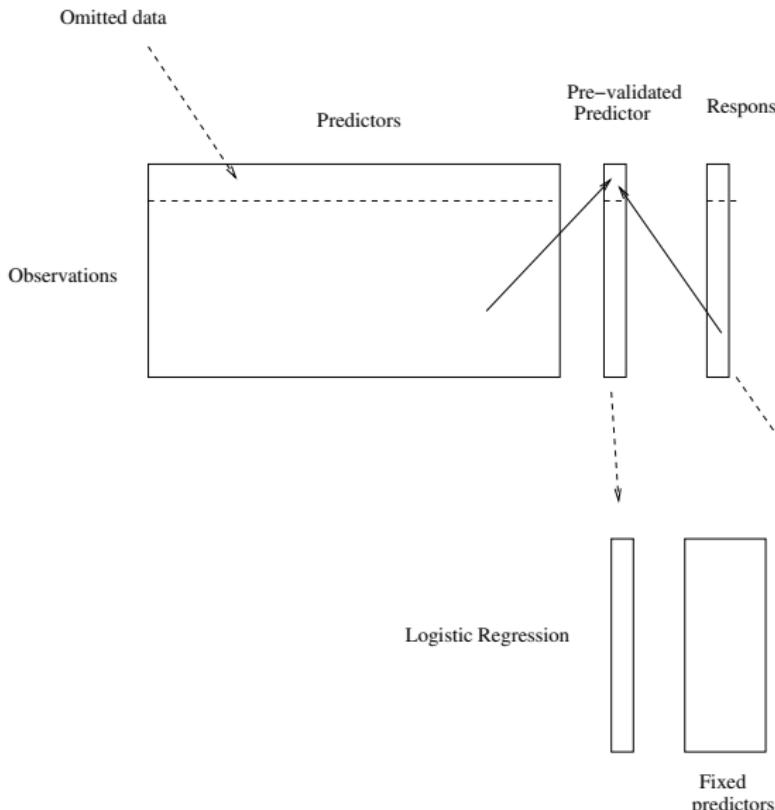
Comparison of the microarray predictor with some clinical predictors, using logistic regression with outcome **prognosis**:

Model	Coef	Stand. Err.	Z score	p-value
Re-use				
microarray	4.096	1.092	3.753	0.000
angio	1.208	0.816	1.482	0.069
er	-0.554	1.044	-0.530	0.298
grade	-0.697	1.003	-0.695	0.243
pr	1.214	1.057	1.149	0.125
age	-1.593	0.911	-1.748	0.040
size	1.483	0.732	2.026	0.021
Pre-validated				
microarray	1.549	0.675	2.296	0.011
angio	1.589	0.682	2.329	0.010
er	-0.617	0.894	-0.690	0.245
grade	0.719	0.720	0.999	0.159
pr	0.537	0.863	0.622	0.267
age	-1.471	0.701	-2.099	0.018
size	0.998	0.594	1.681	0.046

## Idea behind Pre-validation

- Designed for comparison of adaptively derived predictors to fixed, pre-defined predictors.
- The idea is to form a “pre-validated” version of the adaptive predictor: specifically, a “fairer” version that hasn’t “seen” the response  $y$ .

# Pre-validation process



## Pre-validation in detail for this example

1. Divide the cases up into  $K = 13$  equal-sized parts of 6 cases each.
2. Set aside one of parts. Using only the data from the other 12 parts, select the features having absolute correlation at least .3 with the class labels, and form a nearest centroid classification rule.
3. Use the rule to predict the class labels for the 13th part
4. Do steps 2 and 3 for each of the 13 parts, yielding a “pre-validated” microarray predictor  $\tilde{z}_i$  for each of the 78 cases.
5. Fit a logistic regression model to the pre-validated microarray predictor and the 6 clinical predictors.

## The Bootstrap versus Permutation tests

- The bootstrap samples from the estimated population, and uses the results to estimate standard errors and confidence intervals.
- Permutation methods sample from an estimated *null* distribution for the data, and use this to estimate p-values and False Discovery Rates for hypothesis tests.
- The bootstrap can be used to test a null hypothesis in simple situations. Eg if  $\theta = 0$  is the null hypothesis, we check whether the confidence interval for  $\theta$  contains zero.
- Can also adapt the bootstrap to sample from a null distribution (See Efron and Tibshirani book “An Introduction to the Bootstrap” (1993), chapter 16) but there’s no real advantage over permutations.

# Linear Model Selection and Regularization

- Recall the linear model

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon.$$

- In the lectures that follow, we consider some approaches for extending the linear model framework. In the lectures covering Chapter 7 of the text, we generalize the linear model in order to accommodate *non-linear*, but still *additive*, relationships.
- In the lectures covering Chapter 8 we consider even more general *non-linear* models.

# In praise of linear models!

- Despite its simplicity, the linear model has distinct advantages in terms of its *interpretability* and often shows good *predictive performance*.
- Hence we discuss in this lecture some ways in which the simple linear model can be improved, by replacing ordinary least squares fitting with some alternative fitting procedures.

# Why consider alternatives to least squares?

- *Prediction Accuracy*: especially when  $p > n$ , to control the variance.
- *Model Interpretability*: By removing irrelevant features — that is, by setting the corresponding coefficient estimates to zero — we can obtain a model that is more easily interpreted. We will present some approaches for automatically performing *feature selection*.

## Three classes of methods

- *Subset Selection.* We identify a subset of the  $p$  predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.
- *Shrinkage.* We fit a model involving all  $p$  predictors, but the estimated coefficients are shrunk towards zero relative to the least squares estimates. This shrinkage (also known as *regularization*) has the effect of reducing variance and can also perform variable selection.
- *Dimension Reduction.* We project the  $p$  predictors into a  $M$ -dimensional subspace, where  $M < p$ . This is achieved by computing  $M$  different *linear combinations*, or *projections*, of the variables. Then these  $M$  projections are used as predictors to fit a linear regression model by least squares.

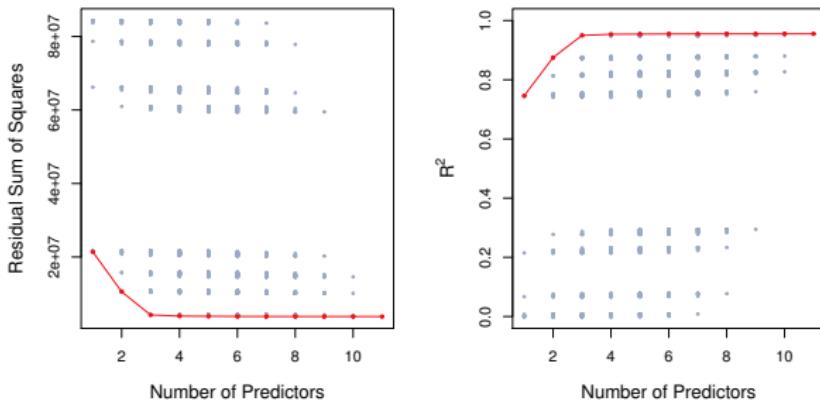
# Subset Selection

*Best subset and stepwise model selection procedures*

## *Best Subset Selection*

1. Let  $\mathcal{M}_0$  denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For  $k = 1, 2, \dots, p$ :
  - (a) Fit all  $\binom{p}{k}$  models that contain exactly  $k$  predictors.
  - (b) Pick the best among these  $\binom{p}{k}$  models, and call it  $\mathcal{M}_k$ . Here *best* is defined as having the smallest RSS, or equivalently largest  $R^2$ .
3. Select a single best model from among  $\mathcal{M}_0, \dots, \mathcal{M}_p$  using cross-validated prediction error,  $C_p$  (AIC), BIC, or adjusted  $R^2$ .

## Example- Credit data set



For each possible model containing a subset of the ten predictors in the **Credit** data set, the  $RSS$  and  $R^2$  are displayed. The red frontier tracks the **best** model for a given number of predictors, according to  $RSS$  and  $R^2$ . Though the data set contains only ten predictors, the  $x$ -axis ranges from 1 to 11, since one of the variables is categorical and takes on three values, leading to the creation of two dummy variables

## Extensions to other models

- Although we have presented best subset selection here for least squares regression, the same ideas apply to other types of models, such as logistic regression.
- The *deviance*— negative two times the maximized log-likelihood— plays the role of RSS for a broader class of models.

## Stepwise Selection

- For computational reasons, best subset selection cannot be applied with very large  $p$ . *Why not?*
- Best subset selection may also suffer from statistical problems when  $p$  is large: larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data.
- Thus an enormous search space can lead to *overfitting* and high variance of the coefficient estimates.
- For both of these reasons, *stepwise* methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

## Forward Stepwise Selection

- Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model.
- In particular, at each step the variable that gives the greatest *additional* improvement to the fit is added to the model.

# In Detail

## *Forward Stepwise Selection*

1. Let  $\mathcal{M}_0$  denote the *null* model, which contains no predictors.
2. For  $k = 0, \dots, p - 1$ :
  - 2.1 Consider all  $p - k$  models that augment the predictors in  $\mathcal{M}_k$  with one additional predictor.
  - 2.2 Choose the *best* among these  $p - k$  models, and call it  $\mathcal{M}_{k+1}$ . Here *best* is defined as having smallest RSS or highest  $R^2$ .
3. Select a single best model from among  $\mathcal{M}_0, \dots, \mathcal{M}_p$  using cross-validated prediction error,  $C_p$  (AIC), BIC, or adjusted  $R^2$ .

## More on Forward Stepwise Selection

- Computational advantage over best subset selection is clear.
- It is not guaranteed to find the best possible model out of all  $2^p$  models containing subsets of the  $p$  predictors. *Why not? Give an example.*

## Credit data example

# Variables	Best subset	Forward stepwise
One	rating	rating
Two	rating, income	rating, income
Three	rating, income, student	rating, income, student
Four	cards, income student, limit	rating, income, student, limit

The first four selected models for best subset selection and forward stepwise selection on the Credit data set. The first three models are identical but the fourth models differ.

## Backward Stepwise Selection

- Like forward stepwise selection, *backward stepwise selection* provides an efficient alternative to best subset selection.
- However, unlike forward stepwise selection, it begins with the full least squares model containing all  $p$  predictors, and then iteratively removes the least useful predictor, one-at-a-time.

# Backward Stepwise Selection: details

## *Backward Stepwise Selection*

1. Let  $\mathcal{M}_p$  denote the *full* model, which contains all  $p$  predictors.
2. For  $k = p, p-1, \dots, 1$ :
  - 2.1 Consider all  $k$  models that contain all but one of the predictors in  $\mathcal{M}_k$ , for a total of  $k-1$  predictors.
  - 2.2 Choose the *best* among these  $k$  models, and call it  $\mathcal{M}_{k-1}$ . Here *best* is defined as having smallest RSS or highest  $R^2$ .
3. Select a single best model from among  $\mathcal{M}_0, \dots, \mathcal{M}_p$  using cross-validated prediction error,  $C_p$  (AIC), BIC, or adjusted  $R^2$ .

## More on Backward Stepwise Selection

- Like forward stepwise selection, the backward selection approach searches through only  $1 + p(p + 1)/2$  models, and so can be applied in settings where  $p$  is too large to apply best subset selection
- Like forward stepwise selection, backward stepwise selection is not guaranteed to yield the *best* model containing a subset of the  $p$  predictors.
- Backward selection requires that the *number of samples  $n$  is larger than the number of variables  $p$*  (so that the full model can be fit). In contrast, forward stepwise can be used even when  $n < p$ , and so is the only viable subset method when  $p$  is very large.

## Choosing the Optimal Model

- The model containing all of the predictors will always have the smallest RSS and the largest  $R^2$ , since these quantities are related to the training error.
- We wish to choose a model with low test error, not a model with low training error. Recall that training error is usually a poor estimate of test error.
- Therefore, RSS and  $R^2$  are not suitable for selecting the best model among a collection of models with different numbers of predictors.

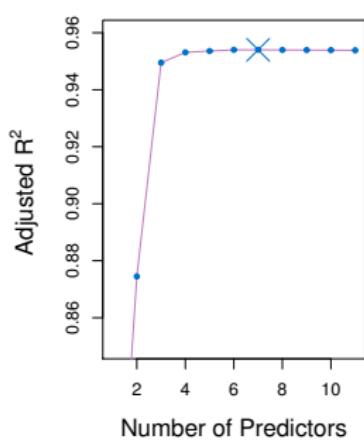
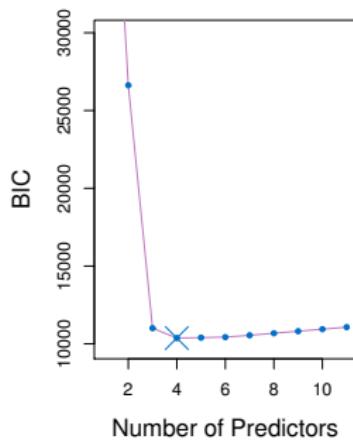
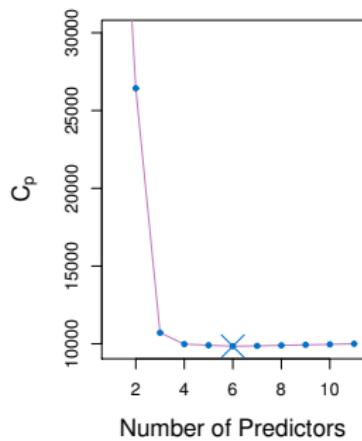
## Estimating test error: two approaches

- We can indirectly estimate test error by making an *adjustment* to the training error to account for the bias due to overfitting.
- We can *directly* estimate the test error, using either a validation set approach or a cross-validation approach, as discussed in previous lectures.
- We illustrate both approaches next.

## $C_p$ , AIC, BIC, and Adjusted $R^2$

- These techniques adjust the training error for the model size, and can be used to select among a set of models with different numbers of variables.
- The next figure displays  $C_p$ , BIC, and adjusted  $R^2$  for the best model of each size produced by best subset selection on the **Credit** data set.

# Credit data example



## Now for some details

- *Mallow's  $C_p$ :*

$$C_p = \frac{1}{n} (\text{RSS} + 2d\hat{\sigma}^2),$$

where  $d$  is the total # of parameters used and  $\hat{\sigma}^2$  is an estimate of the variance of the error  $\epsilon$  associated with each response measurement.

- The *AIC* criterion is defined for a large class of models fit by maximum likelihood:

$$\text{AIC} = -2 \log L + 2 \cdot d$$

where  $L$  is the maximized value of the likelihood function for the estimated model.

- In the case of the linear model with Gaussian errors, maximum likelihood and least squares are the same thing, and  $C_p$  and AIC are equivalent. *Prove this.*

## Details on BIC

$$\text{BIC} = \frac{1}{n} (\text{RSS} + \log(n)d\hat{\sigma}^2).$$

- Like  $C_p$ , the BIC will tend to take on a small value for a model with a low test error, and so generally we select the model that has the lowest BIC value.
- Notice that BIC replaces the  $2d\hat{\sigma}^2$  used by  $C_p$  with a  $\log(n)d\hat{\sigma}^2$  term, where  $n$  is the number of observations.
- Since  $\log n > 2$  for any  $n > 7$ , the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than  $C_p$ . See Figure on slide 19.

## Adjusted $R^2$

- For a least squares model with  $d$  variables, the adjusted  $R^2$  statistic is calculated as

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n - d - 1)}{\text{TSS}/(n - 1)}.$$

where TSS is the total sum of squares.

- Unlike  $C_p$ , AIC, and BIC, for which a *small* value indicates a model with a low test error, a *large* value of adjusted  $R^2$  indicates a model with a small test error.
- Maximizing the adjusted  $R^2$  is equivalent to minimizing  $\frac{\text{RSS}}{n-d-1}$ . While RSS always decreases as the number of variables in the model increases,  $\frac{\text{RSS}}{n-d-1}$  may increase or decrease, due to the presence of  $d$  in the denominator.
- Unlike the  $R^2$  statistic, the adjusted  $R^2$  statistic *pays a price* for the inclusion of unnecessary variables in the model. See Figure on slide 19.

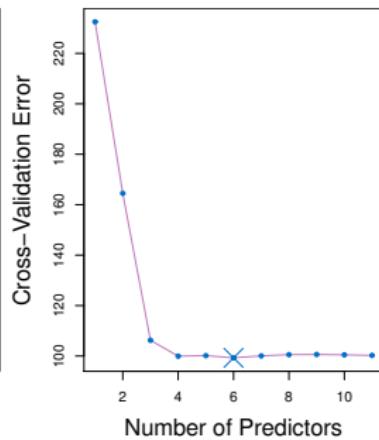
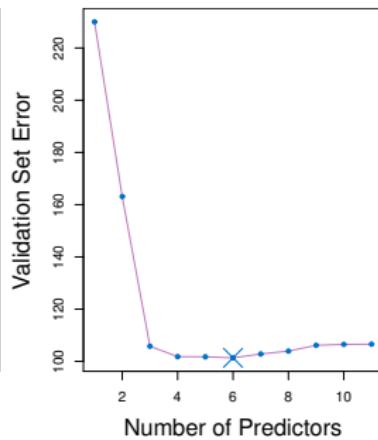
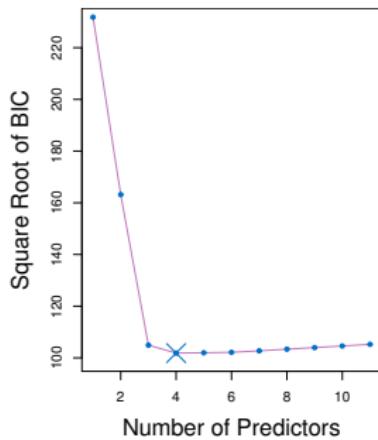
## Validation and Cross-Validation

- Each of the procedures returns a sequence of models  $\mathcal{M}_k$  indexed by model size  $k = 0, 1, 2, \dots$ . Our job here is to select  $\hat{k}$ . Once selected, we will return model  $\mathcal{M}_{\hat{k}}$

## Validation and Cross-Validation

- Each of the procedures returns a sequence of models  $\mathcal{M}_k$  indexed by model size  $k = 0, 1, 2, \dots$ . Our job here is to select  $\hat{k}$ . Once selected, we will return model  $\mathcal{M}_{\hat{k}}$ .
- We compute the validation set error or the cross-validation error for each model  $\mathcal{M}_k$  under consideration, and then select the  $k$  for which the resulting estimated test error is smallest.
- This procedure has an advantage relative to AIC, BIC,  $C_p$ , and adjusted  $R^2$ , in that it provides a direct estimate of the test error, and *doesn't require an estimate of the error variance  $\sigma^2$* .
- It can also be used in a wider range of model selection tasks, even in cases where it is hard to pinpoint the model degrees of freedom (e.g. the number of predictors in the model) or hard to estimate the error variance  $\sigma^2$ .

# Credit data example



## Details of Previous Figure

- The validation errors were calculated by randomly selecting three-quarters of the observations as the training set, and the remainder as the validation set.
- The cross-validation errors were computed using  $k = 10$  folds. In this case, the validation and cross-validation methods both result in a six-variable model.
- However, all three approaches suggest that the four-, five-, and six-variable models are roughly equivalent in terms of their test errors.
- In this setting, we can select a model using the *one-standard-error rule*. We first calculate the standard error of the estimated test MSE for each model size, and then select the smallest model for which the estimated test error is within one standard error of the lowest point on the curve. *What is the rationale for this?*

# Shrinkage Methods

## *Ridge regression* and *Lasso*

- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all  $p$  predictors using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero.
- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance.

## Ridge regression

- Recall that the least squares fitting procedure estimates  $\beta_0, \beta_1, \dots, \beta_p$  using the values that minimize

$$\text{RSS} = \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2.$$

- In contrast, the ridge regression coefficient estimates  $\hat{\beta}^R$  are the values that minimize

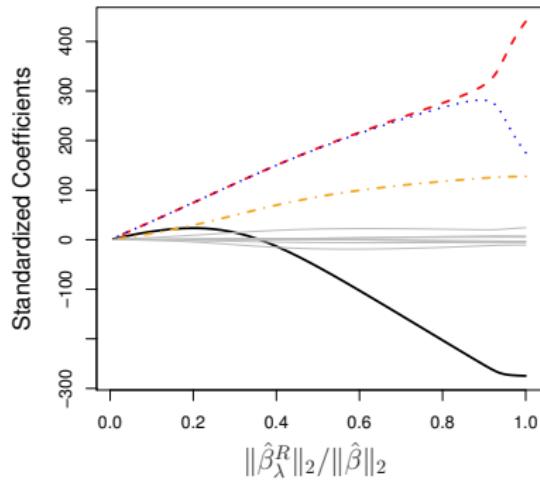
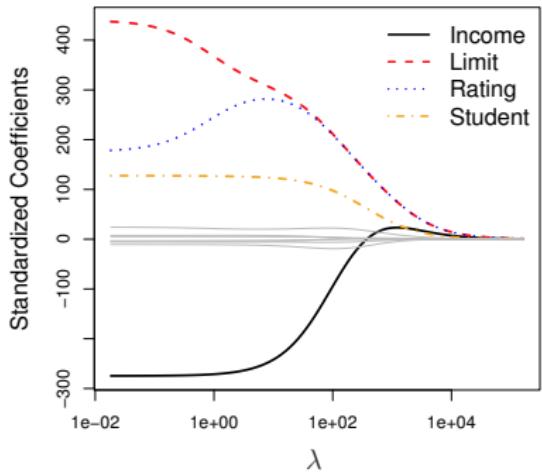
$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2,$$

where  $\lambda \geq 0$  is a *tuning parameter*, to be determined separately.

## Ridge regression: continued

- As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small.
- However, the second term,  $\lambda \sum_j \beta_j^2$ , called a *shrinkage penalty*, is small when  $\beta_1, \dots, \beta_p$  are close to zero, and so it has the effect of *shrinking* the estimates of  $\beta_j$  towards zero.
- The tuning parameter  $\lambda$  serves to control the relative impact of these two terms on the regression coefficient estimates.
- Selecting a good value for  $\lambda$  is critical; cross-validation is used for this.

# Credit data example



## Details of Previous Figure

- In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of  $\lambda$ .
- The right-hand panel displays the same ridge coefficient estimates as the left-hand panel, but instead of displaying  $\lambda$  on the  $x$ -axis, we now display  $\|\hat{\beta}_\lambda^R\|_2/\|\hat{\beta}\|_2$ , where  $\hat{\beta}$  denotes the vector of least squares coefficient estimates.
- The notation  $\|\beta\|_2$  denotes the  $\ell_2$  norm (pronounced “ell 2”) of a vector, and is defined as  $\|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}$ .

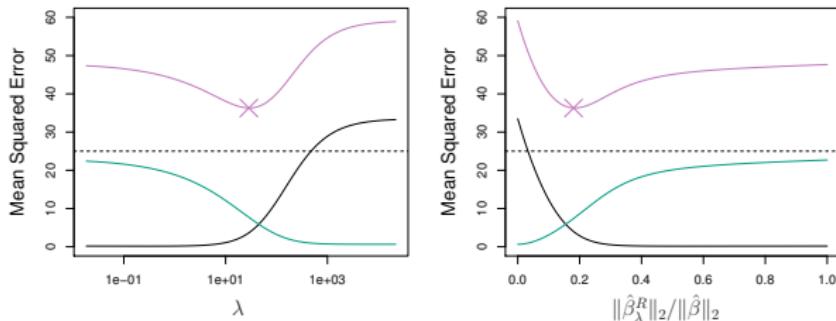
## Ridge regression: scaling of predictors

- The standard least squares coefficient estimates are *scale equivariant*: multiplying  $X_j$  by a constant  $c$  simply leads to a scaling of the least squares coefficient estimates by a factor of  $1/c$ . In other words, regardless of how the  $j$ th predictor is scaled,  $X_j \hat{\beta}_j$  will remain the same.
- In contrast, the ridge regression coefficient estimates can change *substantially* when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- Therefore, it is best to apply ridge regression after *standardizing the predictors*, using the formula

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2}}$$

# Why Does Ridge Regression Improve Over Least Squares?

*The Bias-Variance tradeoff*



*Simulated data with  $n = 50$  observations,  $p = 45$  predictors, all having nonzero coefficients. Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of  $\lambda$  and  $\|\hat{\beta}_\lambda^R\|_2/\|\hat{\beta}\|_2$ . The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.*

## The Lasso

- Ridge regression does have one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all  $p$  predictors in the final model
- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients,  $\hat{\beta}_\lambda^L$ , minimize the quantity

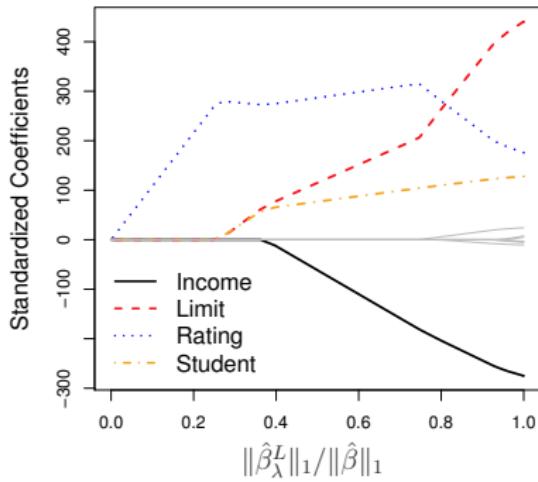
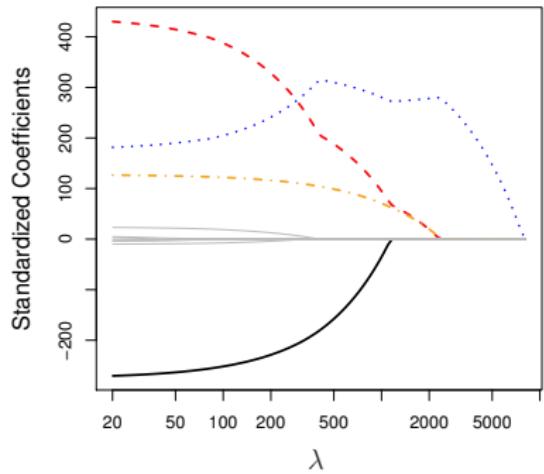
$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|.$$

- In statistical parlance, the lasso uses an  $\ell_1$  (pronounced “ell 1”) penalty instead of an  $\ell_2$  penalty. The  $\ell_1$  norm of a coefficient vector  $\beta$  is given by  $\|\beta\|_1 = \sum |\beta_j|$ .

## The Lasso: continued

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the  $\ell_1$  penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter  $\lambda$  is sufficiently large.
- Hence, much like best subset selection, the lasso performs *variable selection*.
- We say that the lasso yields *sparse* models — that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of  $\lambda$  for the lasso is critical; cross-validation is again the method of choice.

## Example: Credit dataset



## The Variable Selection Property of the Lasso

Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero?

## The Variable Selection Property of the Lasso

Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero?

One can show that the lasso and ridge regression coefficient estimates solve the problems

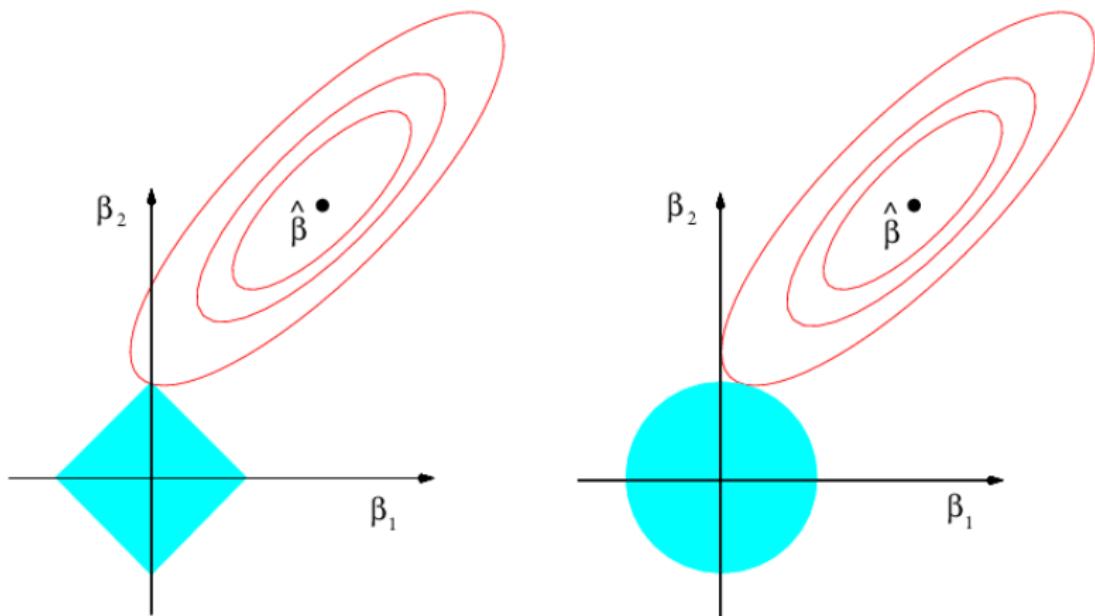
$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq s$$

and

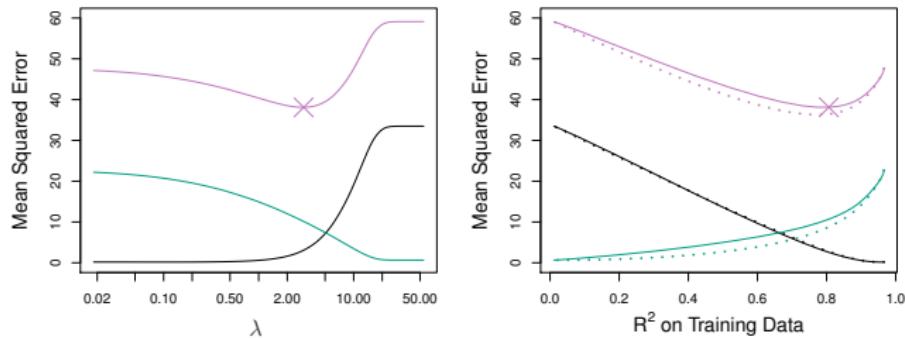
$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 \leq s,$$

respectively.

## The Lasso Picture



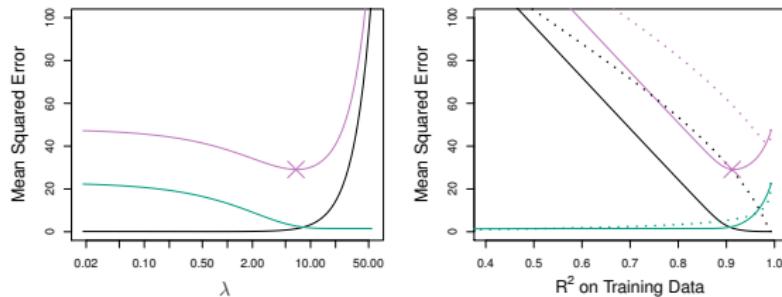
# Comparing the Lasso and Ridge Regression



*Left:* Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on simulated data set of Slide 32.

*Right:* Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their  $R^2$  on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

# Comparing the Lasso and Ridge Regression: continued



*Left:* Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to that in Slide 38, except that now only two predictors are related to the response. *Right:* Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their  $R^2$  on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

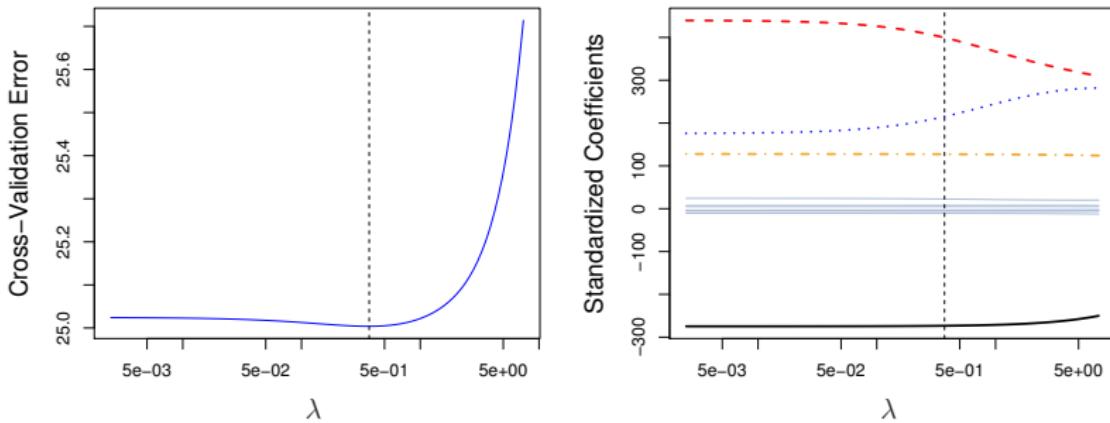
## Conclusions

- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better when the response is a function of only a relatively small number of predictors.
- However, the number of predictors that is related to the response is never known *a priori* for real data sets.
- A technique such as cross-validation can be used in order to determine which approach is better on a particular data set.

# Selecting the Tuning Parameter for Ridge Regression and Lasso

- As for subset selection, for ridge regression and lasso we require a method to determine which of the models under consideration is best.
- That is, we require a method selecting a value for the tuning parameter  $\lambda$  or equivalently, the value of the constraint  $s$ .
- *Cross-validation* provides a simple way to tackle this problem. We choose a grid of  $\lambda$  values, and compute the cross-validation error rate for each value of  $\lambda$ .
- We then select the tuning parameter value for which the cross-validation error is smallest.
- Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter.

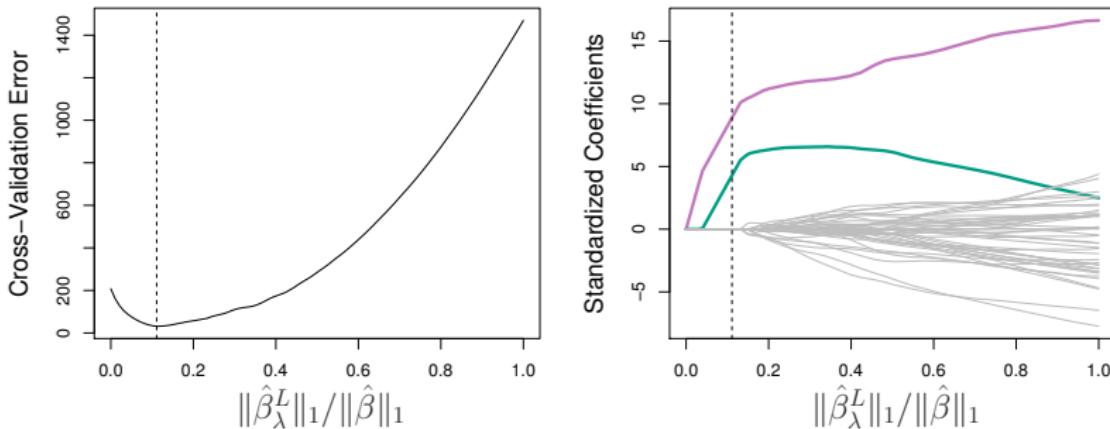
## Credit data example



**Left:** Cross-validation errors that result from applying ridge regression to the **Credit** data set with various values of  $\lambda$ .

**Right:** The coefficient estimates as a function of  $\lambda$ . The vertical dashed lines indicates the value of  $\lambda$  selected by cross-validation.

## Simulated data example



*Left:* Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set from Slide 39. *Right:* The corresponding lasso coefficient estimates are displayed. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

## Dimension Reduction Methods

- The methods that we have discussed so far in this chapter have involved fitting linear regression models, via least squares or a shrunken approach, using the original predictors,  $X_1, X_2, \dots, X_p$ .
- We now explore a class of approaches that *transform* the predictors and then fit a least squares model using the transformed variables. We will refer to these techniques as *dimension reduction* methods.

## Dimension Reduction Methods: details

- Let  $Z_1, Z_2, \dots, Z_M$  represent  $M < p$  *linear combinations* of our original  $p$  predictors. That is,

$$Z_m = \sum_{j=1}^p \phi_{mj} X_j \quad (1)$$

for some constants  $\phi_{m1}, \dots, \phi_{mp}$ .

- We can then fit the linear regression model,

$$y_i = \theta_0 + \sum_{m=1}^M \theta_m z_{im} + \epsilon_i, \quad i = 1, \dots, n, \quad (2)$$

using ordinary least squares.

- Note that in model (2), the regression coefficients are given by  $\theta_0, \theta_1, \dots, \theta_M$ . If the constants  $\phi_{m1}, \dots, \phi_{mp}$  are chosen wisely, then such dimension reduction approaches can often outperform OLS regression.

- Notice that from definition (1),

$$\sum_{m=1}^M \theta_m z_{im} = \sum_{m=1}^M \theta_m \sum_{j=1}^p \phi_{mj} x_{ij} = \sum_{j=1}^p \sum_{m=1}^M \theta_m \phi_{mj} x_{ij} = \sum_{j=1}^p \beta_j x_{ij},$$

where

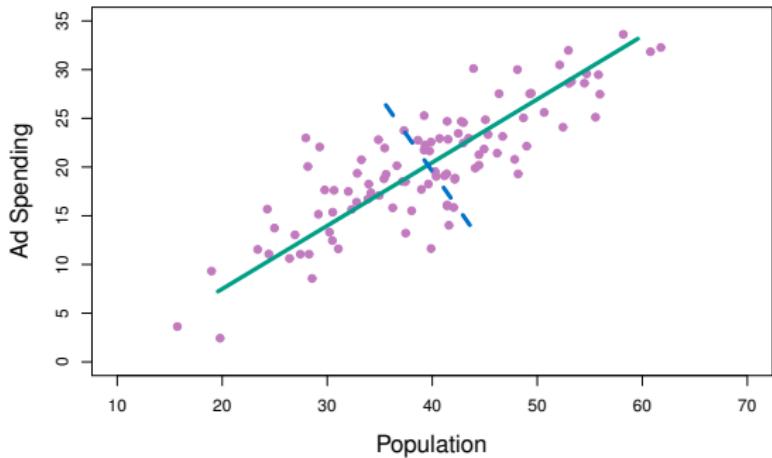
$$\beta_j = \sum_{m=1}^M \theta_m \phi_{mj}. \quad (3)$$

- Hence model (2) can be thought of as a special case of the original linear regression model.
- Dimension reduction serves to constrain the estimated  $\beta_j$  coefficients, since now they must take the form (3).
- Can win in the bias-variance tradeoff.

## Principal Components Regression

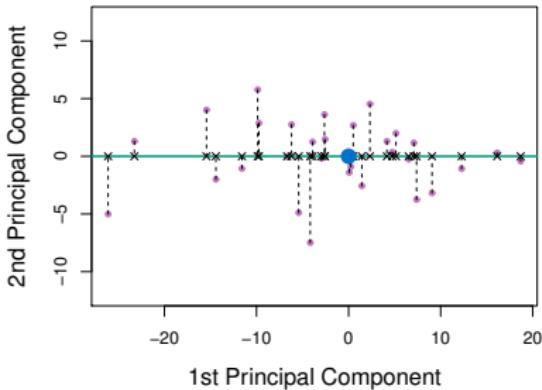
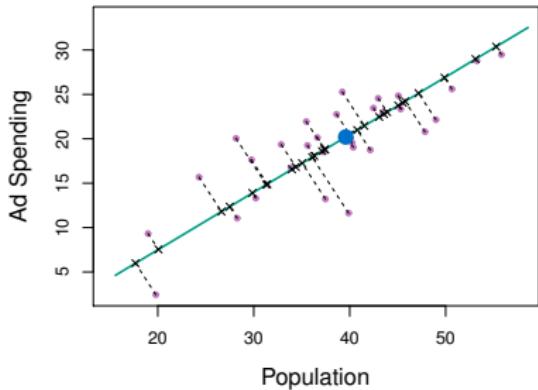
- Here we apply principal components analysis (PCA) (discussed in Chapter 10 of the text) to define the linear combinations of the predictors, for use in our regression.
- The first principal component is that (normalized) linear combination of the variables with the largest variance.
- The second principal component has largest variance, subject to being uncorrelated with the first.
- And so on.
- Hence with many correlated original variables, we replace them with a small set of principal components that capture their joint variation.

# Pictures of PCA



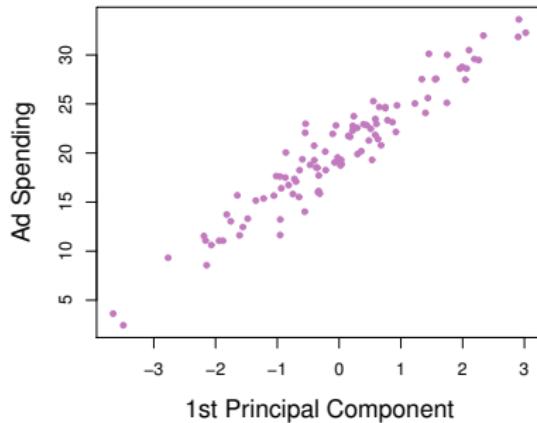
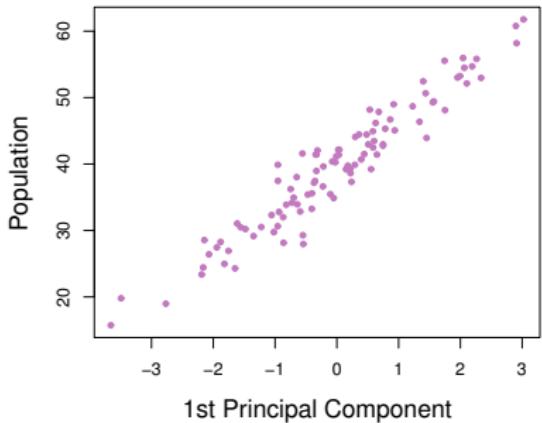
The population size (**pop**) and ad spending (**ad**) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

## Pictures of PCA: continued



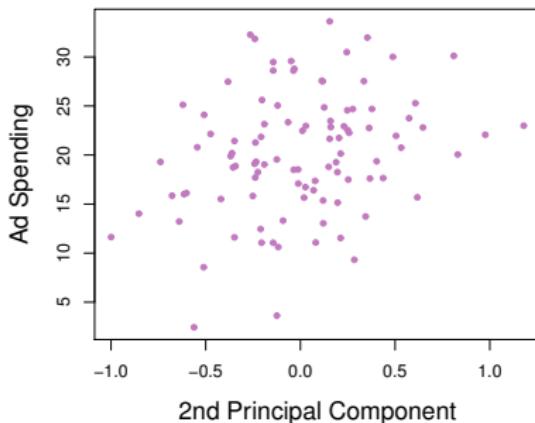
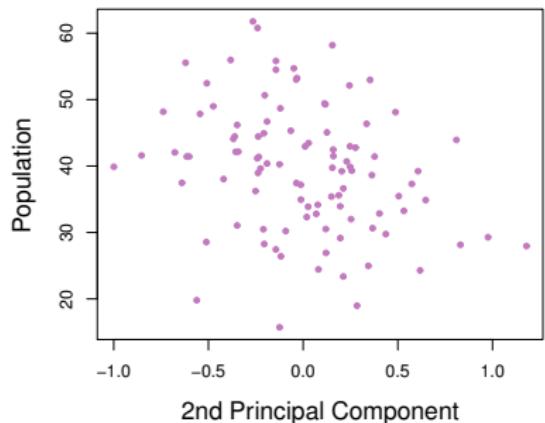
*A subset of the advertising data. Left: The first principal component, chosen to minimize the sum of the squared perpendicular distances to each point, is shown in green. These distances are represented using the black dashed line segments. Right: The left-hand panel has been rotated so that the first principal component lies on the x-axis.*

## Pictures of PCA: continued



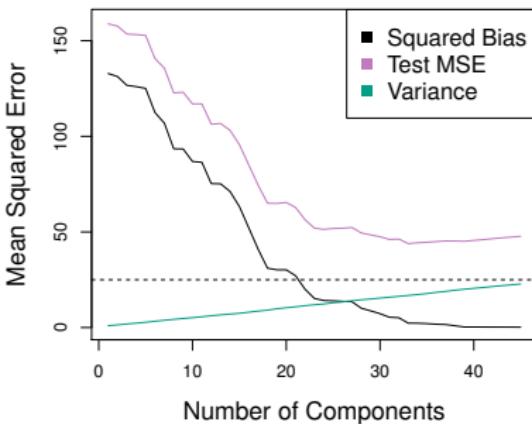
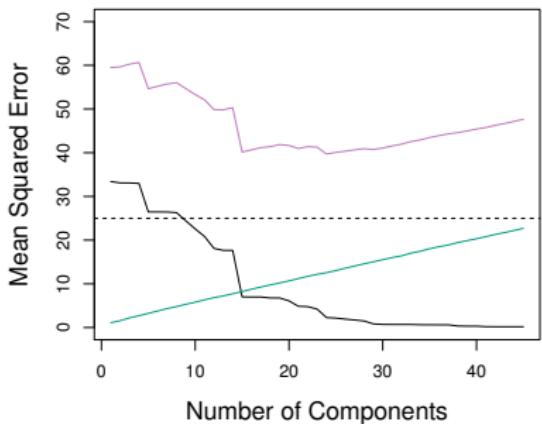
*Plots of the first principal component scores  $z_{i1}$  versus **pop** and **ad**. The relationships are strong.*

## Pictures of PCA: continued



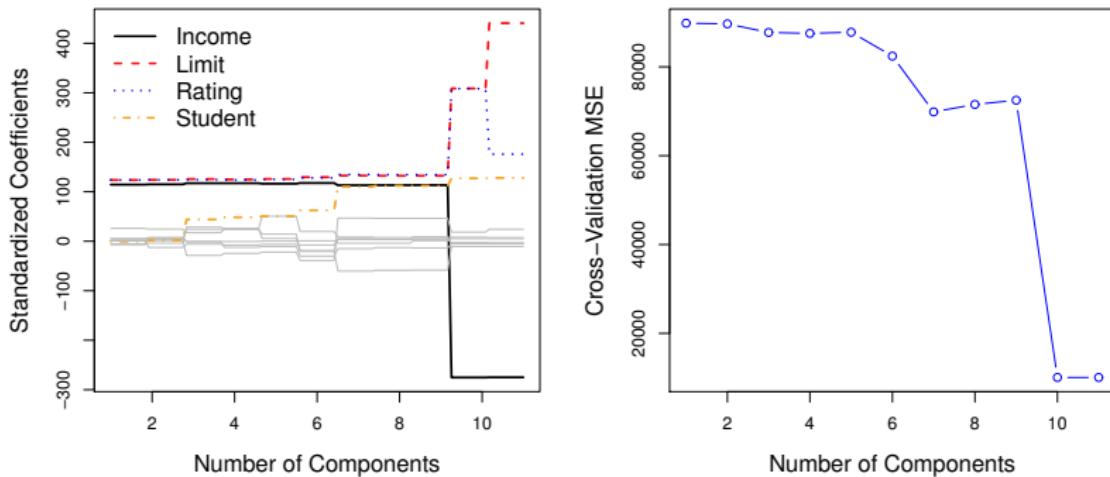
*Plots of the second principal component scores  $z_{i2}$  versus **pop** and **ad**. The relationships are weak.*

# Application to Principal Components Regression



PCR was applied to two simulated data sets. The black, green, and purple lines correspond to squared bias, variance, and test mean squared error, respectively. **Left:** Simulated data from slide 32. **Right:** Simulated data from slide 39.

# Choosing the number of directions $M$



**Left:** PCR standardized coefficient estimates on the Credit data set for different values of  $M$ . **Right:** The 10-fold cross validation MSE obtained using PCR, as a function of  $M$ .

## Partial Least Squares

- PCR identifies linear combinations, or *directions*, that best represent the predictors  $X_1, \dots, X_p$ .
- These directions are identified in an *unsupervised* way, since the response  $Y$  is not used to help determine the principal component directions.
- That is, the response does not *supervise* the identification of the principal components.
- Consequently, PCR suffers from a potentially serious drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response.

## Partial Least Squares: continued

- Like PCR, PLS is a dimension reduction method, which first identifies a new set of features  $Z_1, \dots, Z_M$  that are linear combinations of the original features, and then fits a linear model via OLS using these  $M$  new features.
- But unlike PCR, PLS identifies these new features in a supervised way – that is, it makes use of the response  $Y$  in order to identify new features that not only approximate the old features well, but also that *are related to the response*.
- Roughly speaking, the PLS approach attempts to find directions that help explain both the response and the predictors.

## Details of Partial Least Squares

- After standardizing the  $p$  predictors, PLS computes the first direction  $Z_1$  by setting each  $\phi_{1j}$  in (1) equal to the coefficient from the simple linear regression of  $Y$  onto  $X_j$ .
- One can show that this coefficient is proportional to the correlation between  $Y$  and  $X_j$ .
- Hence, in computing  $Z_1 = \sum_{j=1}^p \phi_{1j} X_j$ , PLS places the highest weight on the variables that are most strongly related to the response.
- Subsequent directions are found by taking residuals and then repeating the above prescription.

# Summary

- Model selection methods are an essential tool for data analysis, especially for big datasets involving many predictors.
- Research into methods that give *sparsity*, such as the *lasso* is an especially hot area.
- Later, we will return to sparsity in more detail, and will describe related approaches such as the *elastic net*.

# Moving Beyond Linearity

The truth is never linear!

# Moving Beyond Linearity

The truth is never linear!  
Or almost never!

# Moving Beyond Linearity

The truth is never linear!

Or almost never!

But often the linearity assumption is good enough.

# Moving Beyond Linearity

The truth is never linear!

Or almost never!

But often the linearity assumption is good enough.

When its not ...

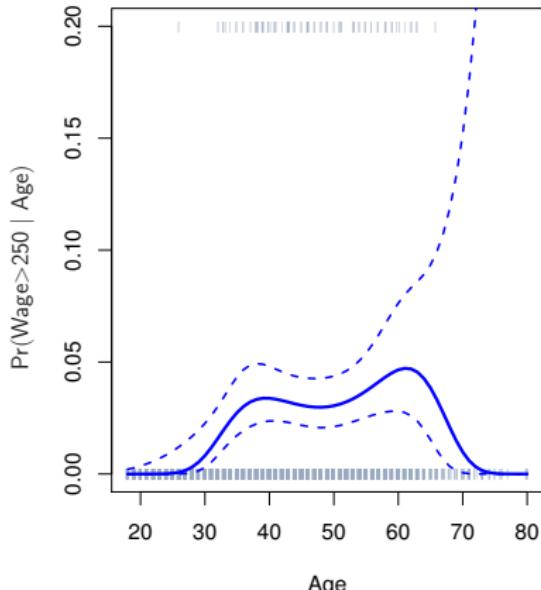
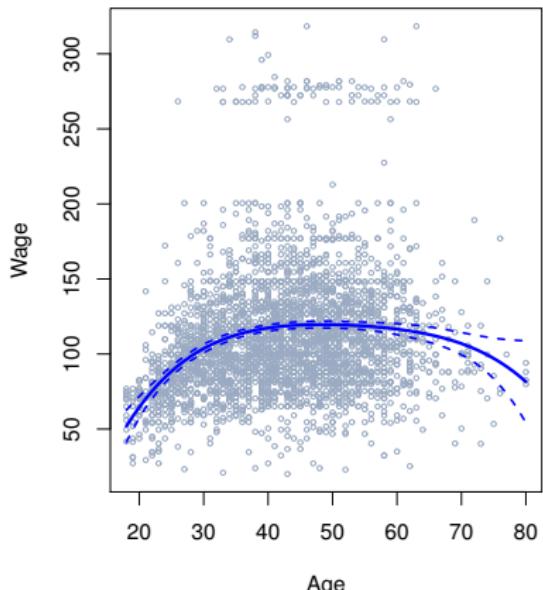
- polynomials,
- step functions,
- splines,
- local regression, and
- generalized additive models

offer a lot of flexibility, without losing the ease and interpretability of linear models.

# Polynomial Regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \dots + \beta_d x_i^d + \epsilon_i$$

Degree-4 Polynomial



## Details

- Create new variables  $X_1 = X$ ,  $X_2 = X^2$ , etc and then treat as multiple linear regression.

## Details

- Create new variables  $X_1 = X$ ,  $X_2 = X^2$ , etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value  $x_0$ :

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

## Details

- Create new variables  $X_1 = X$ ,  $X_2 = X^2$ , etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value  $x_0$ :

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

- Since  $\hat{f}(x_0)$  is a linear function of the  $\hat{\beta}_\ell$ , can get a simple expression for *pointwise-variances*  $\text{Var}[\hat{f}(x_0)]$  at any value  $x_0$ . In the figure we have computed the fit and pointwise standard errors on a grid of values for  $x_0$ . We show  $\hat{f}(x_0) \pm 2 \cdot \text{se}[\hat{f}(x_0)]$ .

## Details

- Create new variables  $X_1 = X$ ,  $X_2 = X^2$ , etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value  $x_0$ :

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

- Since  $\hat{f}(x_0)$  is a linear function of the  $\hat{\beta}_\ell$ , can get a simple expression for *pointwise-variances*  $\text{Var}[\hat{f}(x_0)]$  at any value  $x_0$ . In the figure we have computed the fit and pointwise standard errors on a grid of values for  $x_0$ . We show  $\hat{f}(x_0) \pm 2 \cdot \text{se}[\hat{f}(x_0)]$ .
- We either fix the degree  $d$  at some reasonably low value, else use cross-validation to choose  $d$ .

## Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250 | x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.

## Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250 | x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).

## Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250 | x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d)}.$$

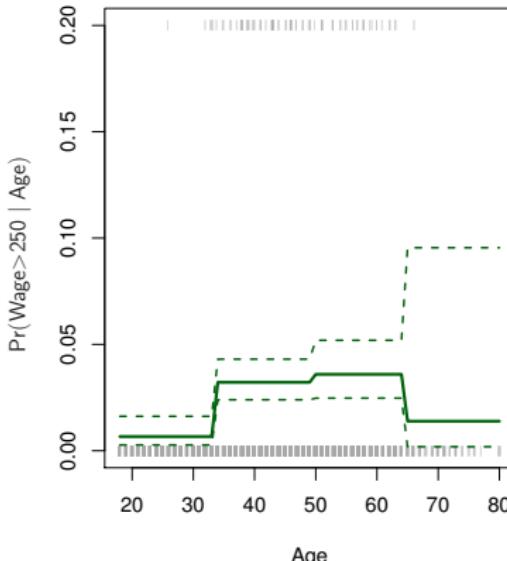
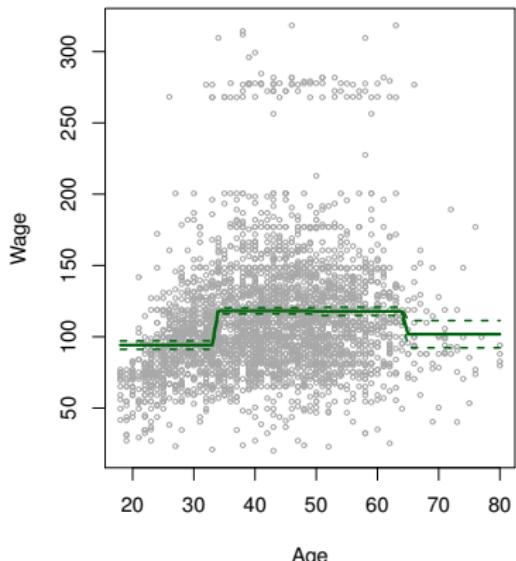
- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).
- Caveat: polynomials have notorious tail behavior — very bad for extrapolation.
- Can fit using `y ~ poly(x, degree = 3)` in formula.

## Step Functions

Another way of creating transformations of a variable — cut the variable into distinct regions.

$$C_1(X) = I(X < 35), \quad C_2(X) = I(35 \leq X < 50), \dots, C_3(X) = I(X \geq 65)$$

Piecewise Constant



## Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.

## Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\text{Year} < 2005) \cdot \text{Age}, \quad I(\text{Year} \geq 2005) \cdot \text{Age}$$

would allow for different linear functions in each age category.

## Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\text{Year} < 2005) \cdot \text{Age}, \quad I(\text{Year} \geq 2005) \cdot \text{Age}$$

would allow for different linear functions in each age category.

- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.

## Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\text{Year} < 2005) \cdot \text{Age}, \quad I(\text{Year} \geq 2005) \cdot \text{Age}$$

would allow for different linear functions in each age category.

- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.
- Choice of cutpoints or *knots* can be problematic. For creating nonlinearities, smoother alternatives such as *splines* are available.

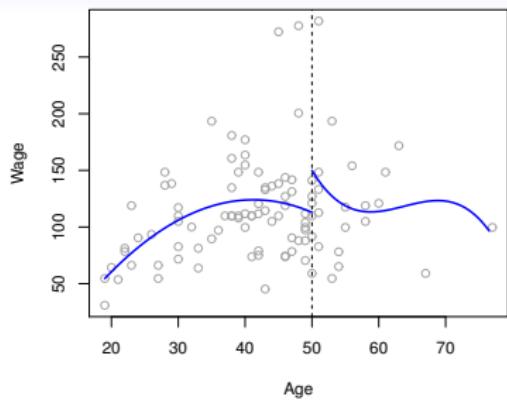
# Piecewise Polynomials

- Instead of a single polynomial in  $X$  over its whole domain, we can rather use different polynomials in regions defined by knots. E.g. (see figure)

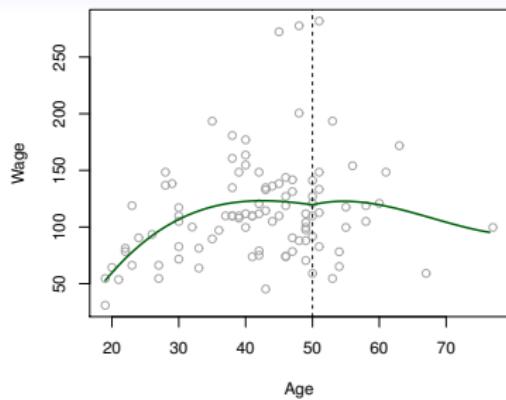
$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

- Better to add constraints to the polynomials, e.g. continuity.
- Splines* have the “maximum” amount of continuity.

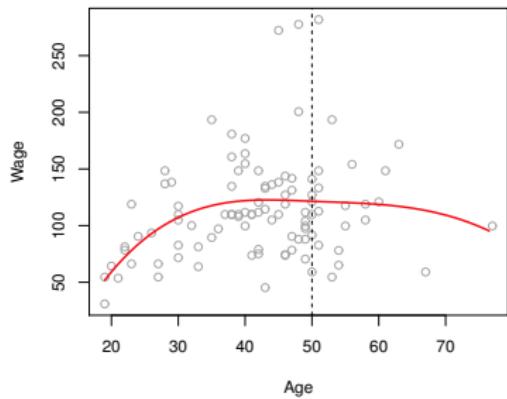
**Piecewise Cubic**



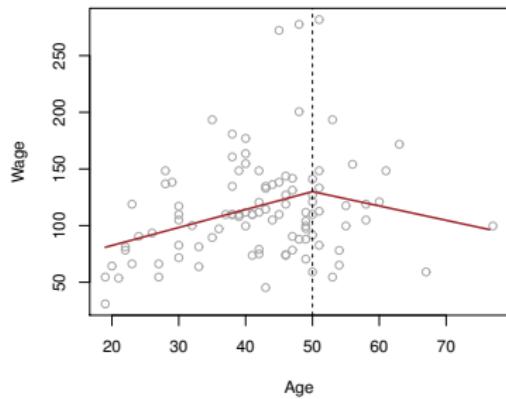
**Continuous Piecewise Cubic**



**Cubic Spline**



**Linear Spline**



## Linear Splines

A linear spline with knots at  $\xi_k$ ,  $k = 1, \dots, K$  is a piecewise linear polynomial continuous at each knot.

We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+1} b_{K+1}(x_i) + \epsilon_i,$$

where the  $b_k$  are *basis functions*.

## Linear Splines

A linear spline with knots at  $\xi_k$ ,  $k = 1, \dots, K$  is a piecewise linear polynomial continuous at each knot.

We can represent this model as

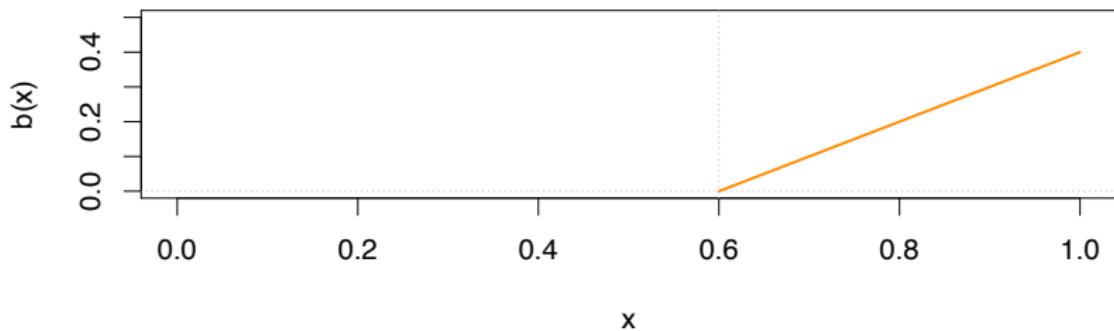
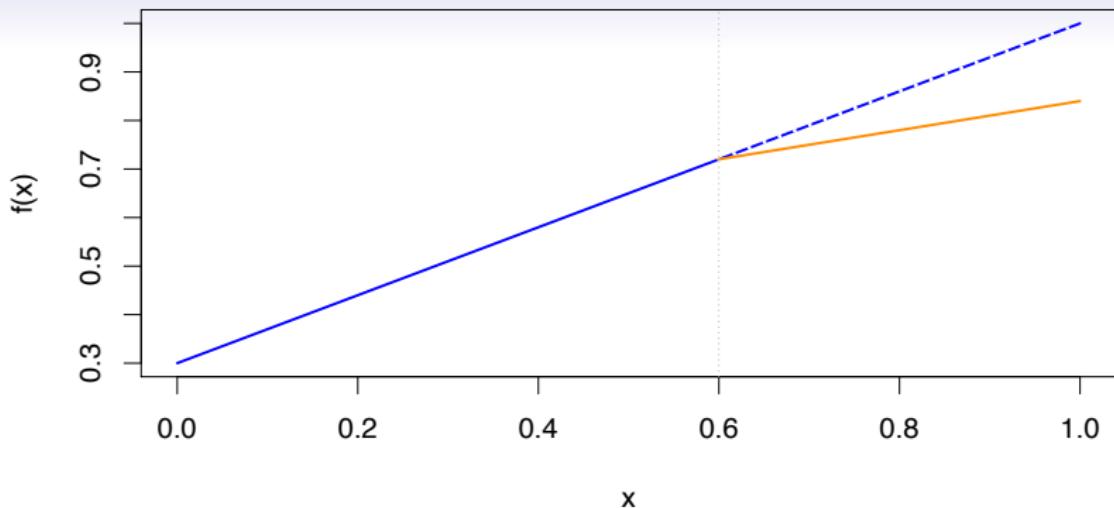
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+1} b_{K+1}(x_i) + \epsilon_i,$$

where the  $b_k$  are *basis functions*.

$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \dots, K \end{aligned}$$

Here the  $(\cdot)_+$  means *positive part*; i.e.

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$



## Cubic Splines

A cubic spline with knots at  $\xi_k$ ,  $k = 1, \dots, K$  is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.

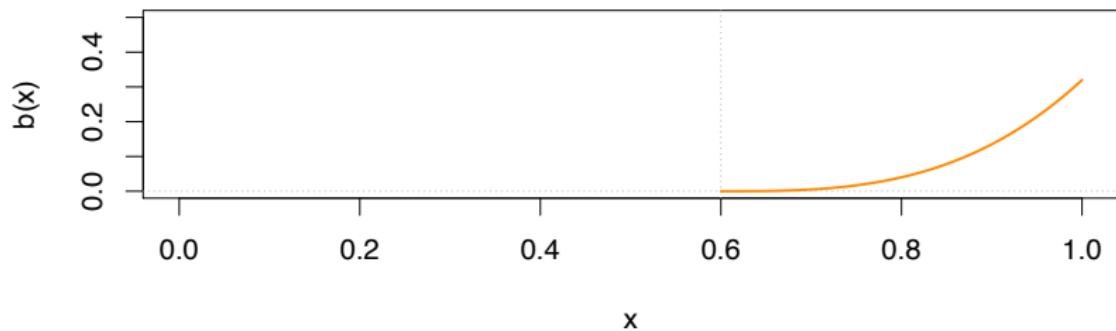
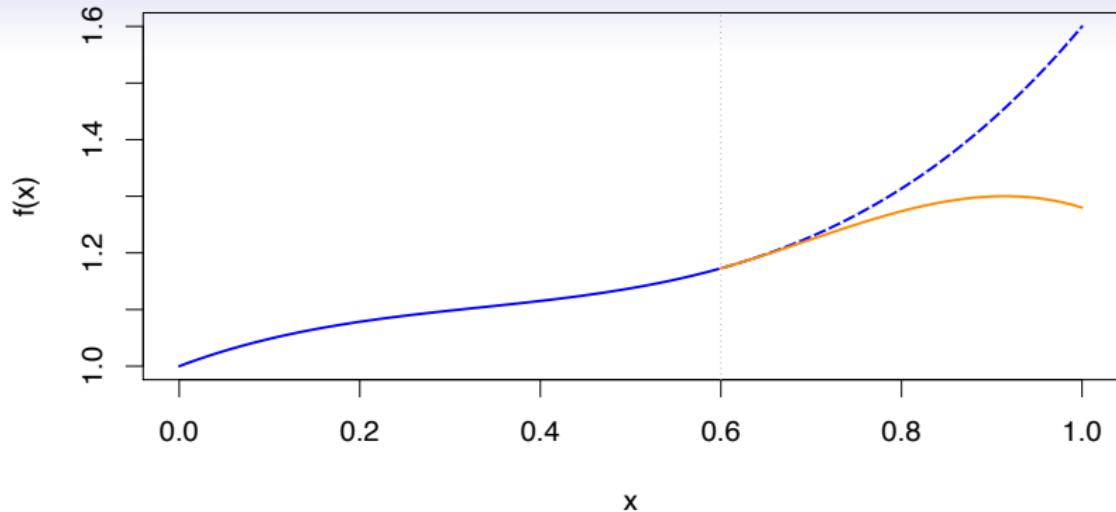
Again we can represent this model with truncated power basis functions

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \dots, K \end{aligned}$$

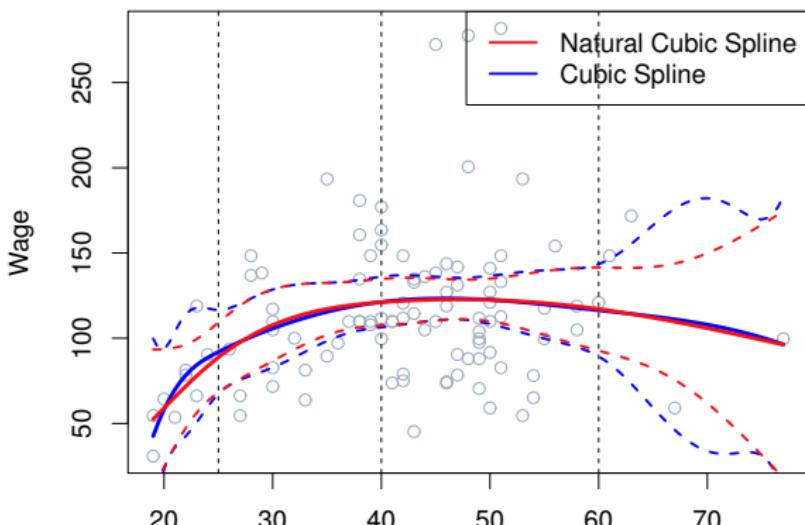
where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$



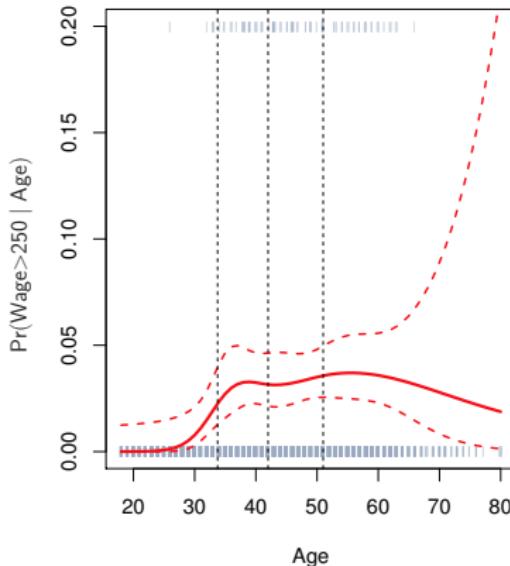
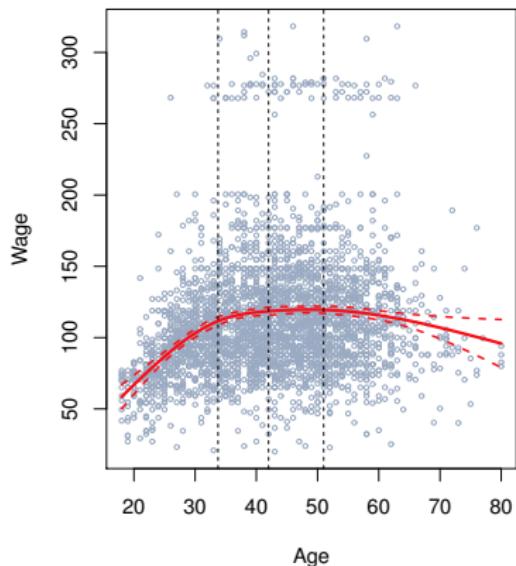
## Natural Cubic Splines

A natural cubic spline extrapolates linearly beyond the boundary knots. This adds  $4 = 2 \times 2$  extra constraints, and allows us to put more internal knots for the same degrees of freedom as a regular cubic spline.



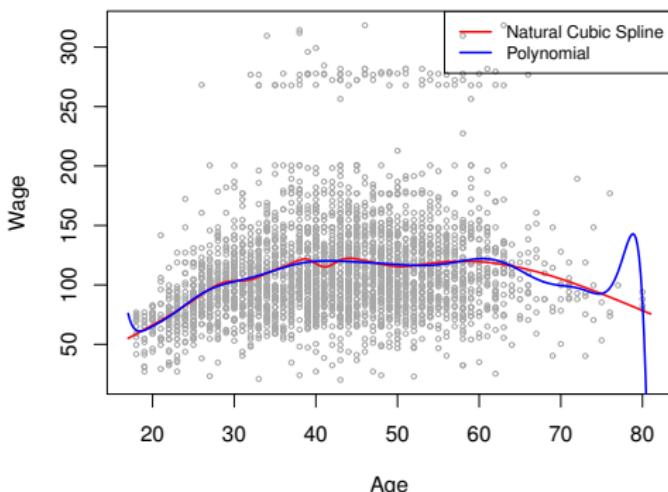
Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package `splines`.

Natural Cubic Spline



## Knot placement

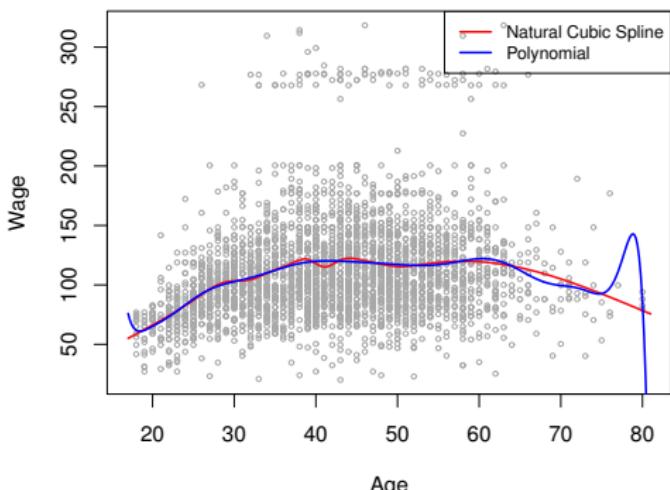
- One strategy is to decide  $K$ , the number of knots, and then place them at appropriate quantiles of the observed  $X$ .
- A cubic spline with  $K$  knots has  $K + 4$  parameters or degrees of freedom.
- A natural spline with  $K$  knots has  $K$  degrees of freedom.



Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

## Knot placement

- One strategy is to decide  $K$ , the number of knots, and then place them at appropriate quantiles of the observed  $X$ .
- A cubic spline with  $K$  knots has  $K + 4$  parameters or degrees of freedom.
- A natural spline with  $K$  knots has  $K$  degrees of freedom.



Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

`ns(age, df=14)`  
`poly(age, deg=14)`

## Smoothing Splines

This section is a little bit mathematical



Consider this criterion for fitting a smooth function  $g(x)$  to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

## Smoothing Splines

This section is a little bit mathematical



Consider this criterion for fitting a smooth function  $g(x)$  to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make  $g(x)$  match the data at each  $x_i$ .

# Smoothing Splines

This section is a little bit mathematical



Consider this criterion for fitting a smooth function  $g(x)$  to some data:

$$\underset{g \in S}{\text{minimize}} \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make  $g(x)$  match the data at each  $x_i$ .
- The second term is a *roughness penalty* and controls how wiggly  $g(x)$  is. It is modulated by the *tuning parameter*  $\lambda \geq 0$ .

# Smoothing Splines

This section is a little bit mathematical



Consider this criterion for fitting a smooth function  $g(x)$  to some data:

$$\underset{g \in S}{\text{minimize}} \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make  $g(x)$  match the data at each  $x_i$ .
- The second term is a *roughness penalty* and controls how wiggly  $g(x)$  is. It is modulated by the *tuning parameter*  $\lambda \geq 0$ .
  - The smaller  $\lambda$ , the more wiggly the function, eventually interpolating  $y_i$  when  $\lambda = 0$ .

# Smoothing Splines

This section is a little bit mathematical



Consider this criterion for fitting a smooth function  $g(x)$  to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make  $g(x)$  match the data at each  $x_i$ .
- The second term is a *roughness penalty* and controls how wiggly  $g(x)$  is. It is modulated by the *tuning parameter*  $\lambda \geq 0$ .
  - The smaller  $\lambda$ , the more wiggly the function, eventually interpolating  $y_i$  when  $\lambda = 0$ .
  - As  $\lambda \rightarrow \infty$ , the function  $g(x)$  becomes linear.

## Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of  $x_i$ . The roughness penalty still controls the roughness via  $\lambda$ .

## Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of  $x_i$ . The roughness penalty still controls the roughness via  $\lambda$ .

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single  $\lambda$  to be chosen.

## Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of  $x_i$ . The roughness penalty still controls the roughness via  $\lambda$ .

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single  $\lambda$  to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.

## Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of  $x_i$ . The roughness penalty still controls the roughness via  $\lambda$ .

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single  $\lambda$  to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.
- The vector of  $n$  fitted values can be written as  $\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y}$ , where  $\mathbf{S}_\lambda$  is a  $n \times n$  matrix (determined by the  $x_i$  and  $\lambda$ ).
- The *effective degrees of freedom* are given by

$$df_\lambda = \sum_{i=1}^n \{\mathbf{S}_\lambda\}_{ii}.$$

## Smoothing Splines continued — choosing $\lambda$

- We can specify  $df$  rather than  $\lambda$ !

In R: `smooth.spline(age, wage, df = 10)`

## Smoothing Splines continued — choosing $\lambda$

- We can specify  $df$  rather than  $\lambda$ !

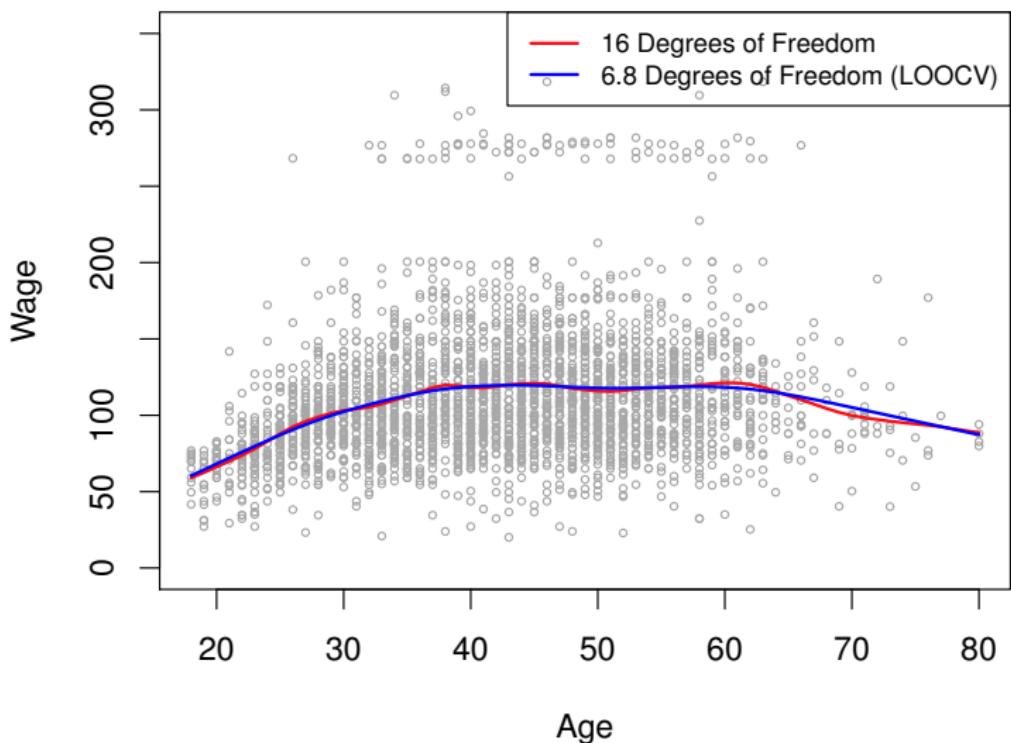
In R: `smooth.spline(age, wage, df = 10)`

- The leave-one-out (LOO) cross-validated error is given by

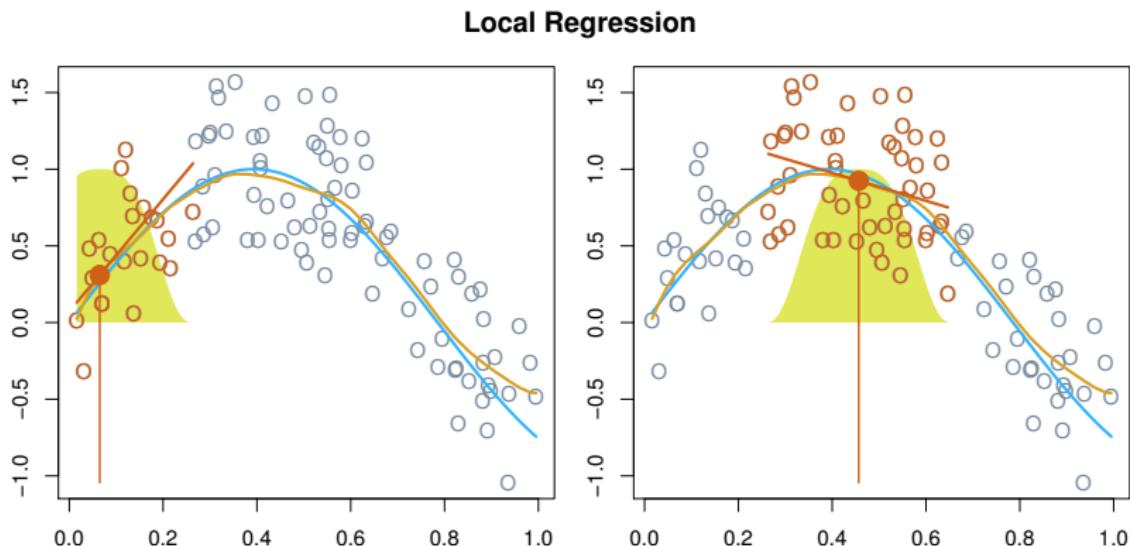
$$\text{RSS}_{cv}(\lambda) = \sum_{i=1}^n (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^n \left[ \frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}} \right]^2.$$

In R: `smooth.spline(age, wage)`

## Smoothing Spline



# Local Regression



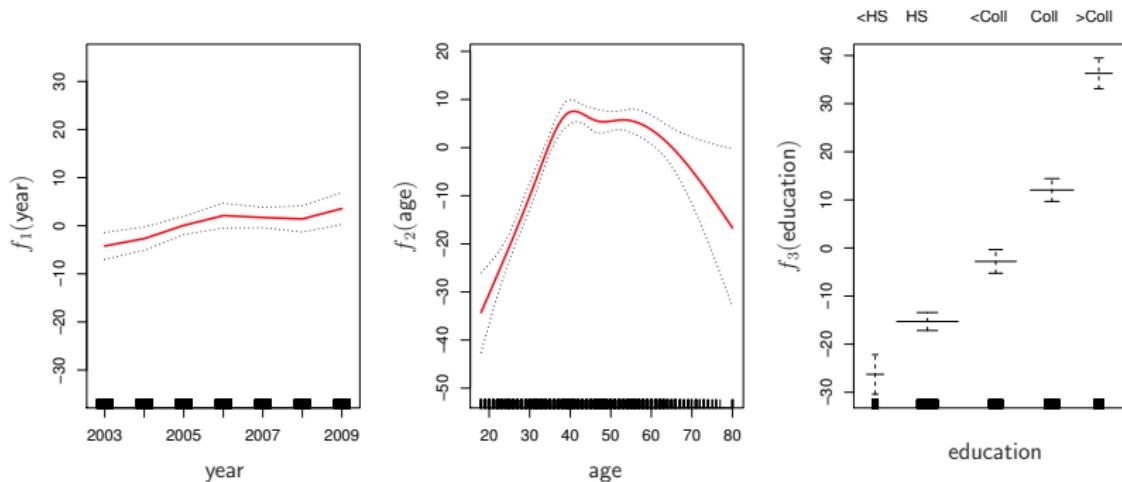
With a sliding weight function, we fit separate linear fits over the range of  $X$  by weighted least squares.

See text for more details, and `loess()` function in R.

# Generalized Additive Models

Allows for flexible nonlinearities in several variables, but retains the additive structure of linear models.

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i.$$



## GAM details

- Can fit a GAM simply using, e.g. natural splines:

```
lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + education)
```

## GAM details

- Can fit a GAM simply using, e.g. natural splines:

```
lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + education)
```

- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.

## GAM details

- Can fit a GAM simply using, e.g. natural splines:

```
lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + education)
```

- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.

## GAM details

- Can fit a GAM simply using, e.g. natural splines:

```
lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + education)
```

- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.
- Can use smoothing splines or local regression as well:

```
gam(wage ~ s(year, df = 5) + lo(age, span = .5) + education)
```

## GAM details

- Can fit a GAM simply using, e.g. natural splines:

```
lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + education)
```

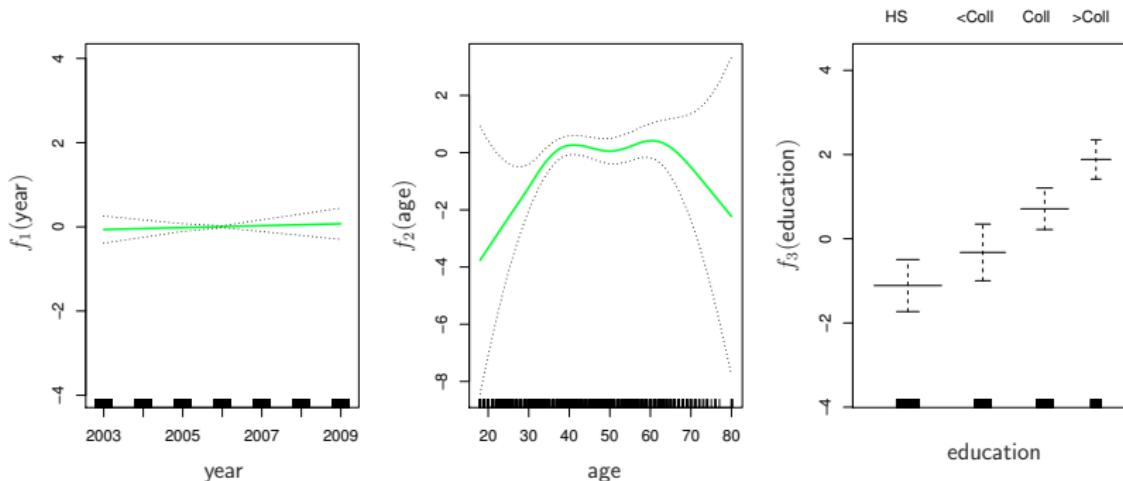
- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.
- Can use smoothing splines or local regression as well:

```
gam(wage ~ s(year, df = 5) + lo(age, span = .5) + education)
```

- GAMs are additive, although low-order interactions can be included in a natural way using, e.g. bivariate smoothers or interactions of the form `ns(age, df=5):ns(year, df=5)`.

# GAMs for classification

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p).$$



```
gam(I(wage > 250) ~ year + s(age, df = 5) + education, family = binomial)
```

## Tree-based Methods

- Here we describe *tree-based* methods for regression and classification.
- These involve *stratifying* or *segmenting* the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *decision-tree* methods.

## Pros and Cons

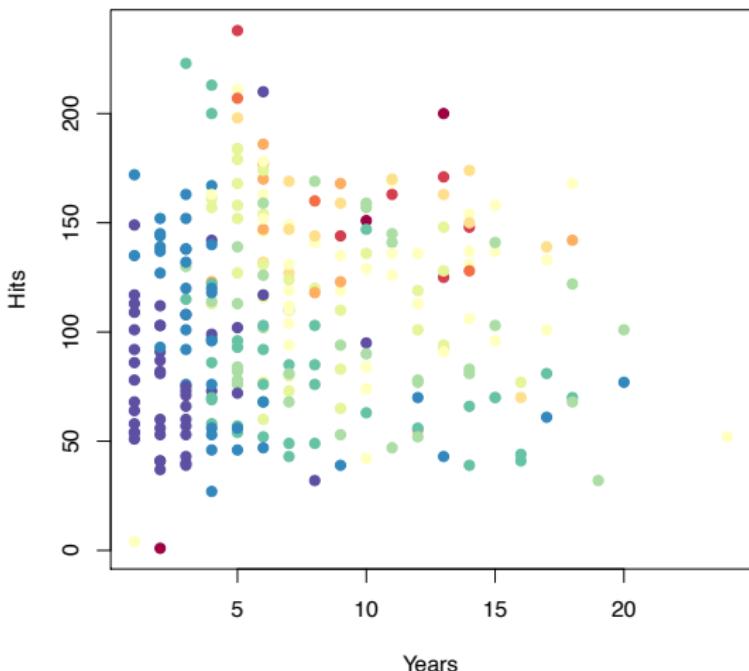
- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Hence we also discuss *bagging*, *random forests*, and *boosting*. These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

# The Basics of Decision Trees

- Decision trees can be applied to both regression and classification problems.
- We first consider regression problems, and then move on to classification.

# Baseball salary data: how would you stratify it?

Salary is color-coded from low (blue, green) to high (yellow,red)



## Decision tree for these data

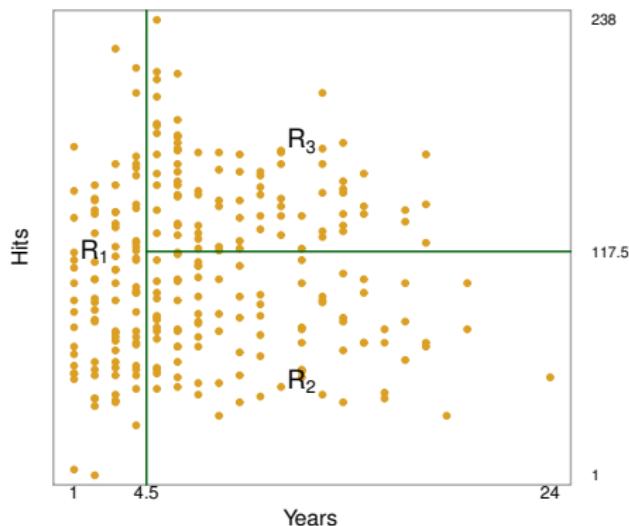


## Details of previous figure

- For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.
- At a given internal node, the label (of the form  $X_j < t_k$ ) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to  $X_j \geq t_k$ . For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to **Years**<4.5, and the right-hand branch corresponds to **Years**>=4.5.
- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

## Results

- Overall, the tree stratifies or segments the players into three regions of predictor space:  $R_1 = \{X \mid \text{Years} < 4.5\}$ ,  $R_2 = \{X \mid \text{Years} \geq 4.5, \text{Hits} < 117.5\}$ , and  $R_3 = \{X \mid \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$ .



## Terminology for Trees

- In keeping with the *tree* analogy, the regions  $R_1$ ,  $R_2$ , and  $R_3$  are known as *terminal nodes*
- Decision trees are typically drawn *upside down*, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as *internal nodes*
- In the hitters tree, the two internal nodes are indicated by the text `Years<4.5` and `Hits<117.5`.

## Interpretation of Results

- **Years** is the most important factor in determining **Salary**, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of **Hits** that he made in the previous year seems to play little role in his **Salary**.
- But among players who have been in the major leagues for five or more years, the number of **Hits** made in the previous year does affect **Salary**, and players who made more **Hits** last year tend to have higher salaries.
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain

## Details of the tree-building process

1. We divide the predictor space — that is, the set of possible values for  $X_1, X_2, \dots, X_p$  — into  $J$  distinct and non-overlapping regions,  $R_1, R_2, \dots, R_J$ .
2. For every observation that falls into the region  $R_j$ , we make the same prediction, which is simply the mean of the response values for the training observations in  $R_j$ .

## More details of the tree-building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes  $R_1, \dots, R_J$  that minimize the RSS, given by

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where  $\hat{y}_{R_j}$  is the mean response for the training observations within the  $j$ th box.

## More details of the tree-building process

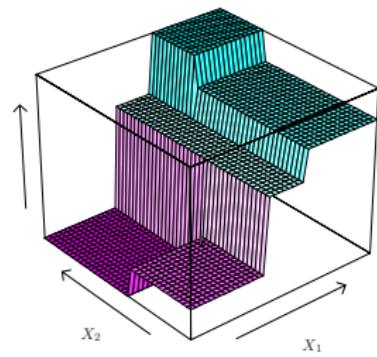
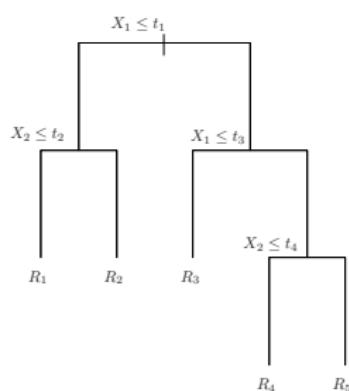
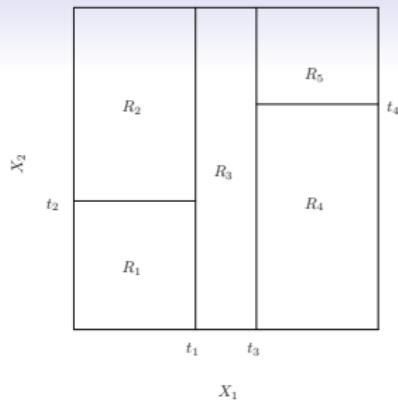
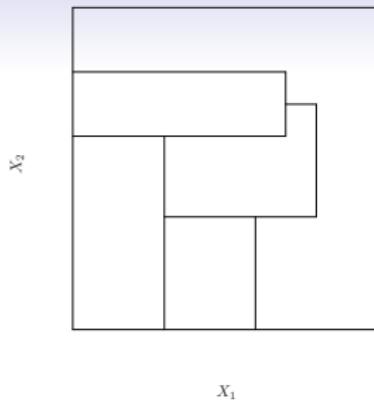
- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into  $J$  boxes.
- For this reason, we take a *top-down, greedy* approach that is known as recursive binary splitting.
- The approach is *top-down* because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is *greedy* because at each step of the tree-building process, the *best* split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

## Details— Continued

- We first select the predictor  $X_j$  and the cutpoint  $s$  such that splitting the predictor space into the regions  $\{X|X_j < s\}$  and  $\{X|X_j \geq s\}$  leads to the greatest possible reduction in RSS.
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

## Predictions

- We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- A five-region example of this approach is shown in the next slide.



## Details of previous figure

*Top Left:* A partition of two-dimensional feature space that could not result from recursive binary splitting.

*Top Right:* The output of recursive binary splitting on a two-dimensional example.

*Bottom Left:* A tree corresponding to the partition in the top right panel.

*Bottom Right:* A perspective plot of the prediction surface corresponding to that tree.

## Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance. *Why?*

## Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance. *Why?*
- A smaller tree with fewer splits (that is, fewer regions  $R_1, \dots, R_J$ ) might lead to lower variance and better interpretation at the cost of a little bias.
- One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, but is too *short-sighted*: a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on.

## Pruning a tree— continued

- A better strategy is to grow a very large tree  $T_0$ , and then *prune* it back in order to obtain a *subtree*
- *Cost complexity pruning* — also known as *weakest link pruning* — is used to do this
- we consider a sequence of trees indexed by a nonnegative tuning parameter  $\alpha$ . For each value of  $\alpha$  there corresponds a subtree  $T \subset T_0$  such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. Here  $|T|$  indicates the number of terminal nodes of the tree  $T$ ,  $R_m$  is the rectangle (i.e. the subset of predictor space) corresponding to the  $m$ th terminal node, and  $\hat{y}_{R_m}$  is the mean of the training observations in  $R_m$ .

## Choosing the best subtree

- The tuning parameter  $\alpha$  controls a trade-off between the subtree's complexity and its fit to the training data.
- We select an optimal value  $\hat{\alpha}$  using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to  $\hat{\alpha}$ .

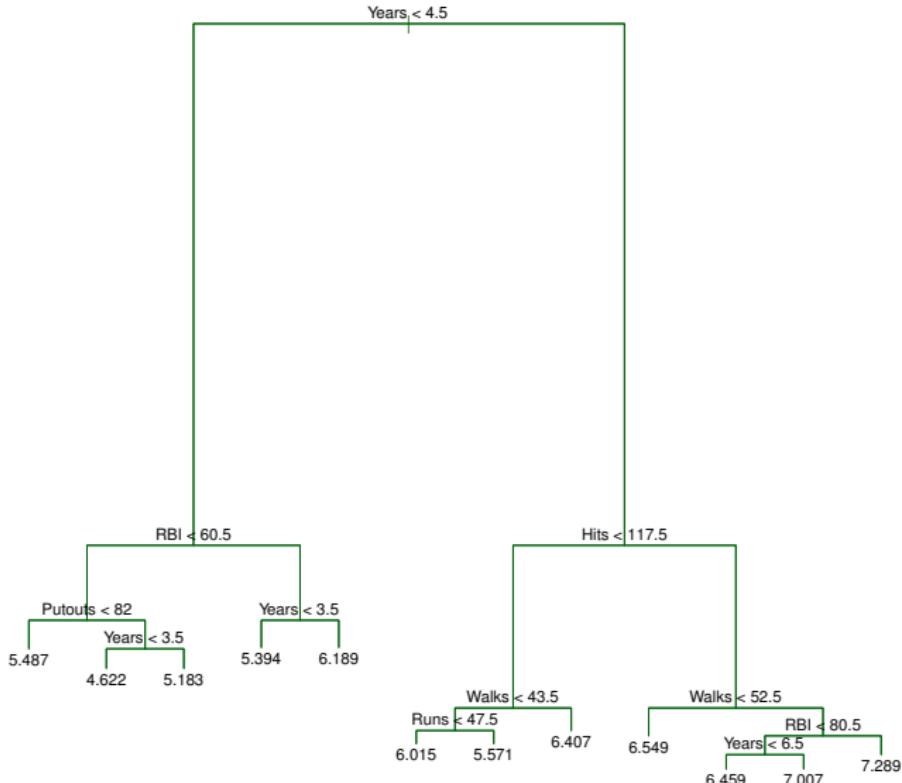
## Summary: tree algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of  $\alpha$ .
3. Use K-fold cross-validation to choose  $\alpha$ . For each  $k = 1, \dots, K$ :
  - 3.1 Repeat Steps 1 and 2 on the  $\frac{K-1}{K}$ th fraction of the training data, excluding the  $k$ th fold.
  - 3.2 Evaluate the mean squared prediction error on the data in the left-out  $k$ th fold, as a function of  $\alpha$ .Average the results, and pick  $\alpha$  to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of  $\alpha$ .

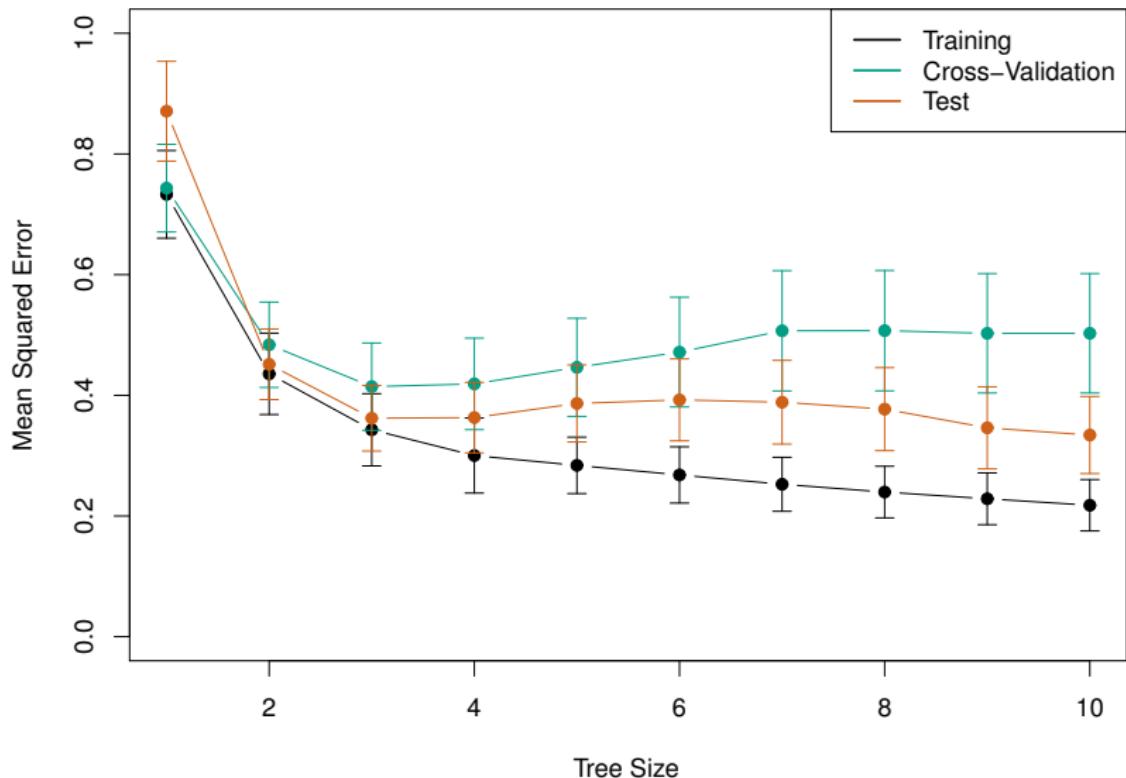
## Baseball example continued

- First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set.
- We then built a large regression tree on the training data and varied  $\alpha$  in order to create subtrees with different numbers of terminal nodes.
- Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of  $\alpha$ .

## Baseball example continued



## Baseball example continued



# Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs.

## Details of classification trees

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
- In the classification setting, RSS cannot be used as a criterion for making the binary splits
- A natural alternative to RSS is the *classification error rate*. this is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk}).$$

Here  $\hat{p}_{mk}$  represents the proportion of training observations in the  $m$ th region that are from the  $k$ th class.

- However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

## Gini index and Deviance

- The *Gini index* is defined by

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

a measure of total variance across the  $K$  classes. The Gini index takes on a small value if all of the  $\hat{p}_{mk}$ 's are close to zero or one.

- For this reason the Gini index is referred to as a measure of node *purity* — a small value indicates that a node contains predominantly observations from a single class.

## Gini index and Deviance

- The *Gini index* is defined by

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

a measure of total variance across the  $K$  classes. The Gini index takes on a small value if all of the  $\hat{p}_{mk}$ 's are close to zero or one.

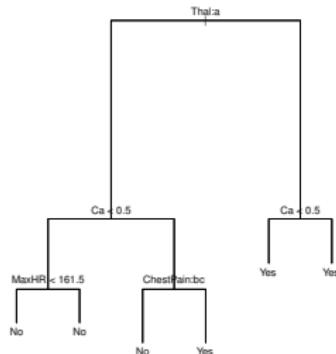
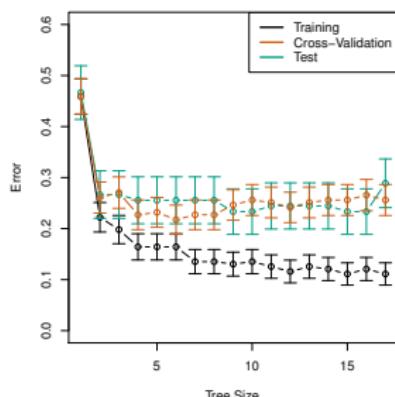
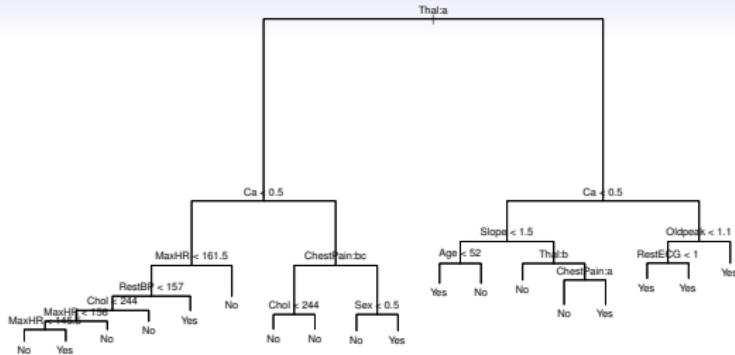
- For this reason the Gini index is referred to as a measure of node *purity* — a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is *cross-entropy*, given by

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}.$$

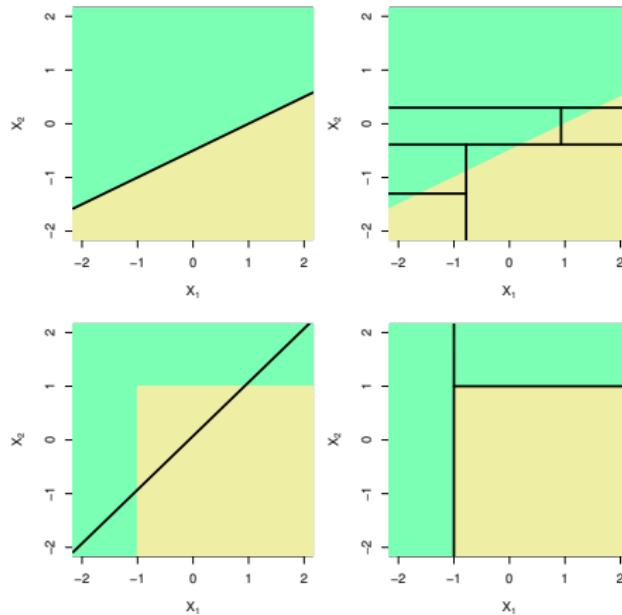
- It turns out that the Gini index and the cross-entropy are very similar numerically.

## Example: heart data

- These data contain a binary outcome **HD** for 303 patients who presented with chest pain.
- An outcome value of **Yes** indicates the presence of heart disease based on an angiographic test, while **No** means no heart disease.
- There are 13 predictors including **Age**, **Sex**, **Chol** (a cholesterol measurement), and other heart and lung function measurements.
- Cross-validation yields a tree with six terminal nodes. See next figure.



# Trees Versus Linear Models



Top Row: True linear boundary; Bottom row: true non-linear boundary.

Left column: linear model; Right column: tree-based model

## Advantages and Disadvantages of Trees

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

# Bagging

- *Bootstrap aggregation*, or *bagging*, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- Recall that given a set of  $n$  independent observations  $Z_1, \dots, Z_n$ , each with variance  $\sigma^2$ , the variance of the mean  $\bar{Z}$  of the observations is given by  $\sigma^2/n$ .
- In other words, *averaging a set of observations reduces variance*. Of course, this is not practical because we generally do not have access to multiple training sets.

## Bagging— continued

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate  $B$  different bootstrapped training data sets. We then train our method on the  $b$ th bootstrapped training set in order to get  $\hat{f}^{*b}(x)$ , the prediction at a point  $x$ . We then average all the predictions to obtain

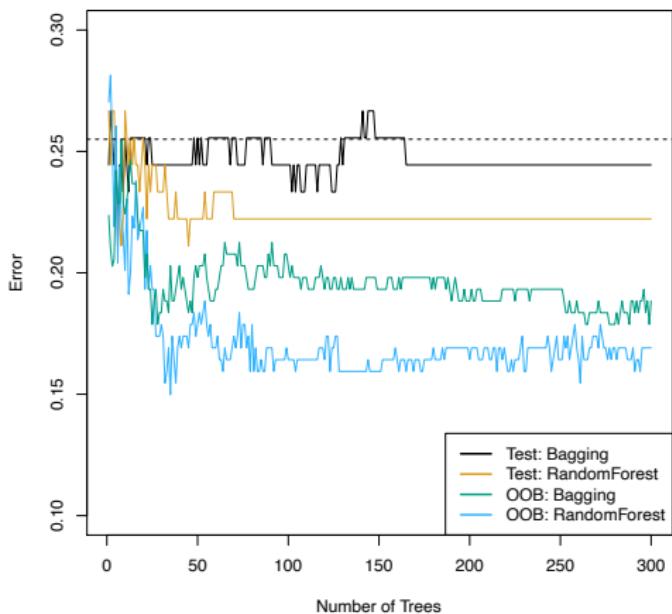
$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x).$$

This is called *bagging*.

## Bagging classification trees

- The above prescription applied to regression trees
- For classification trees: for each test observation, we record the class predicted by each of the  $B$  trees, and take a *majority vote*: the overall prediction is the most commonly occurring class among the  $B$  predictions.

## Bagging the heart data



## Details of previous figure

Bagging and random forest results for the Heart data.

- The test error (black and orange) is shown as a function of  $B$ , the number of bootstrapped training sets used.
- Random forests were applied with  $m = \sqrt{p}$ .
- The dashed line indicates the test error resulting from a single classification tree.
- The green and blue traces show the OOB error, which in this case is considerably lower

## Out-of-Bag Error Estimation

- It turns out that there is a very straightforward way to estimate the test error of a bagged model.
- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. One can show that on average, each bagged tree makes use of around two-thirds of the observations.
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the *out-of-bag* (OOB) observations.
- We can predict the response for the  $i$ th observation using each of the trees in which that observation was OOB. This will yield around  $B/3$  predictions for the  $i$ th observation, which we average.
- This estimate is essentially the LOO cross-validation error for bagging, if  $B$  is large.

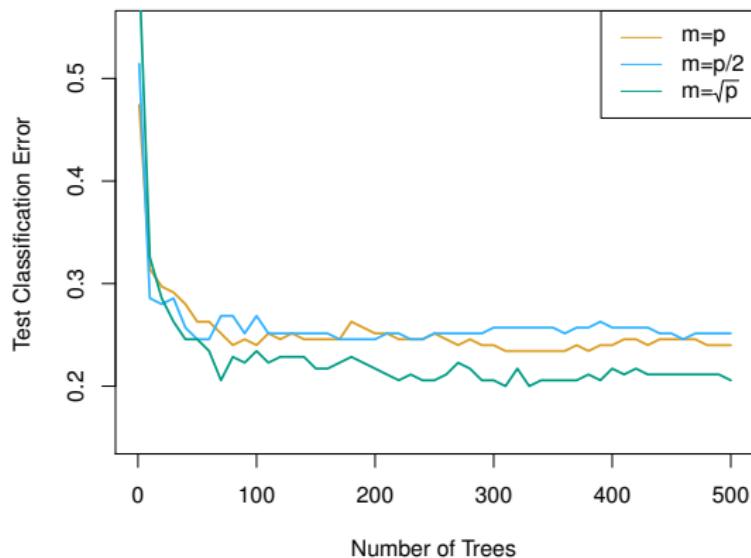
## Random Forests

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, *a random selection of  $m$  predictors* is chosen as split candidates from the full set of  $p$  predictors. The split is allowed to use only one of those  $m$  predictors.
- A fresh selection of  $m$  predictors is taken at each split, and typically we choose  $m \approx \sqrt{p}$  — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

## Example: gene expression data

- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables  $m$ .

## Results: gene expression data



## Details of previous figure

- Results from random forests for the fifteen-class gene expression data set with  $p = 500$  predictors.
- The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of  $m$ , the number of predictors available for splitting at each interior tree node.
- Random forests ( $m < p$ ) lead to a slight improvement over bagging ( $m = p$ ). A single classification tree has an error rate of 45.7%.

## Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We only discuss boosting for decision trees.
- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown *sequentially*: each tree is grown using information from previously grown trees.

## Boosting algorithm for regression trees

1. Set  $\hat{f}(x) = 0$  and  $r_i = y_i$  for all  $i$  in the training set.
2. For  $b = 1, 2, \dots, B$ , repeat:
  - 2.1 Fit a tree  $\hat{f}^b$  with  $d$  splits ( $d + 1$  terminal nodes) to the training data  $(X, r)$ .
  - 2.2 Update  $\hat{f}$  by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

- 2.3 Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x).$$

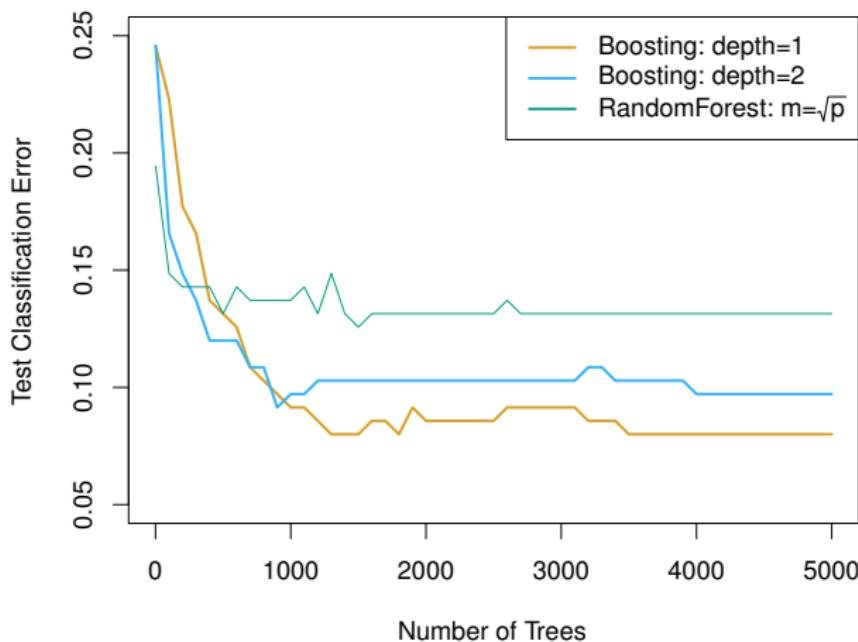
## What is the idea behind this procedure?

- Unlike fitting a single large decision tree to the data, which amounts to *fitting the data hard* and potentially overfitting, the boosting approach instead *learns slowly*.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter  $d$  in the algorithm.
- By fitting small trees to the residuals, we slowly improve  $\hat{f}$  in areas where it does not perform well. The shrinkage parameter  $\lambda$  slows the process down even further, allowing more and different shaped trees to attack the residuals.

## Boosting for classification

- Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex. We will not go into detail here, nor do we in the text book.
- Students can learn about the details in *Elements of Statistical Learning, chapter 10.*
- The R package `gbm` (gradient boosted models) handles a variety of regression and classification problems.

## Gene expression data continued



## Details of previous figure

- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict *cancer* versus *normal*.
- The test error is displayed as a function of the number of trees. For the two boosted models,  $\lambda = 0.01$ . Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.
- The test error rate for a single tree is 24%.

## Tuning parameters for boosting

1. The *number of trees*  $B$ . Unlike bagging and random forests, boosting can overfit if  $B$  is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select  $B$ .

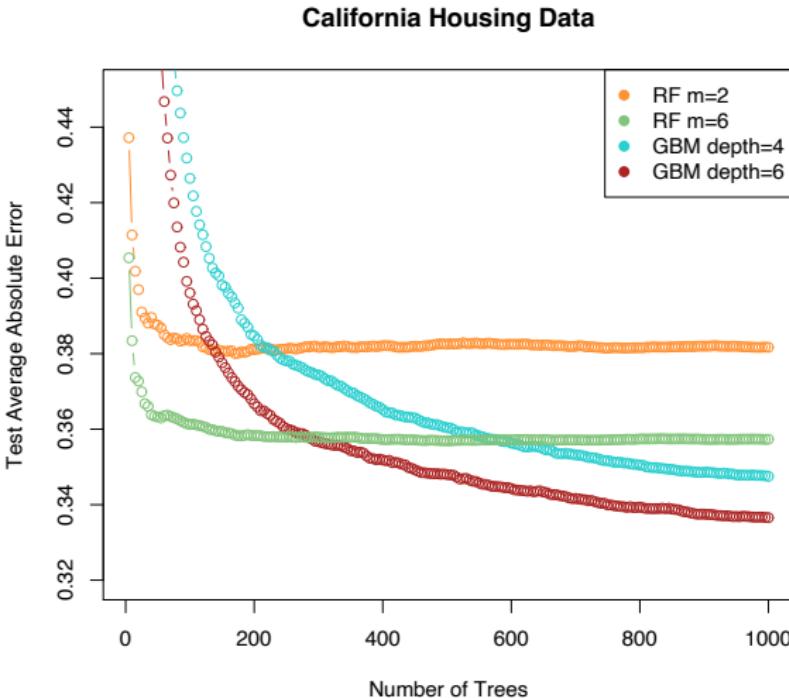
## Tuning parameters for boosting

1. The *number of trees*  $B$ . Unlike bagging and random forests, boosting can overfit if  $B$  is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select  $B$ .
2. The *shrinkage parameter*  $\lambda$ , a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small  $\lambda$  can require using a very large value of  $B$  in order to achieve good performance.

## Tuning parameters for boosting

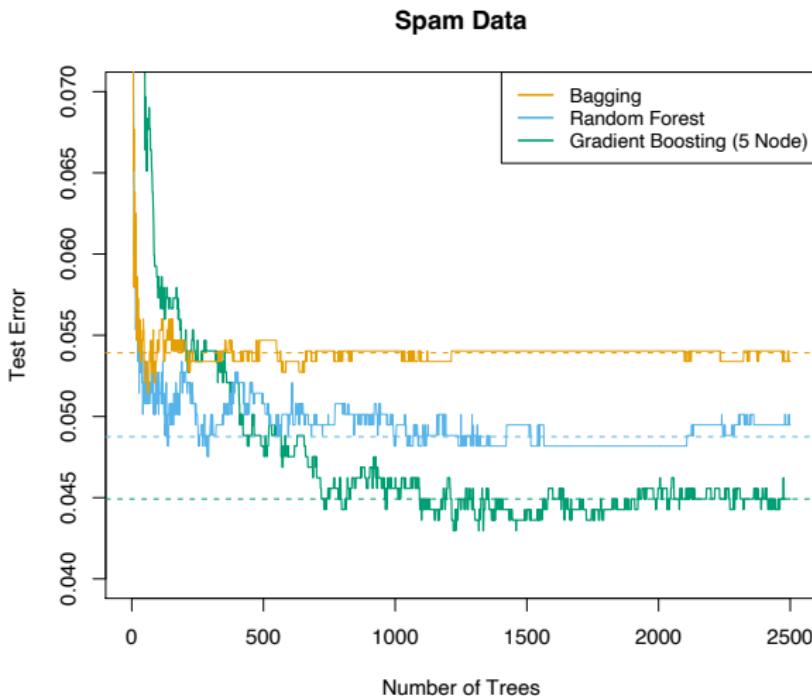
1. The *number of trees*  $B$ . Unlike bagging and random forests, boosting can overfit if  $B$  is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select  $B$ .
2. The *shrinkage parameter*  $\lambda$ , a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small  $\lambda$  can require using a very large value of  $B$  in order to achieve good performance.
3. The *number of splits*  $d$  in each tree, which controls the complexity of the boosted ensemble. Often  $d = 1$  works well, in which case each tree is a *stump*, consisting of a single split and resulting in an additive model. More generally  $d$  is the *interaction depth*, and controls the interaction order of the boosted model, since  $d$  splits can involve at most  $d$  variables.

# Another regression example



from *Elements of Statistical Learning, chapter 15.*

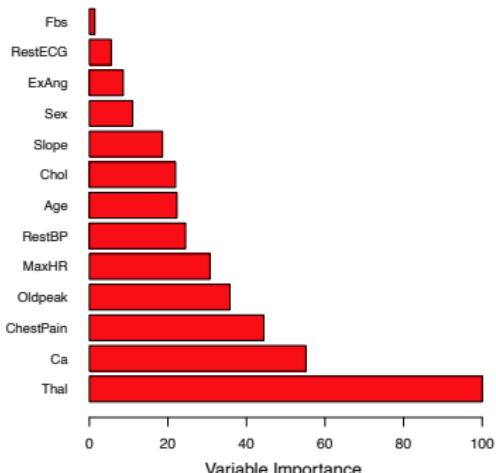
## Another classification example



from *Elements of Statistical Learning, chapter 15.*

## Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all  $B$  trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all  $B$  trees.



Variable importance plot  
for the **Heart** data

# Summary

- Decision trees are simple and interpretable models for regression and classification
- However they are often not competitive with other methods in terms of prediction accuracy
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods— random forests and boosting— are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

# Bayesian Additive Regression Trees

# Bayesian Additive Regression Trees

- We discuss *Bayesian additive regression trees* (BART), an ensemble method that uses decision trees as its building blocks.

# Bayesian Additive Regression Trees

- We discuss *Bayesian additive regression trees* (BART), an ensemble method that uses decision trees as its building blocks.
- Recall that *bagging* and *random forests* make predictions from an average of regression trees, each of which is built using a random sample of data and/or predictors. Each tree is built separately from the others.

# Bayesian Additive Regression Trees

- We discuss *Bayesian additive regression trees* (BART), an ensemble method that uses decision trees as its building blocks.
- Recall that *bagging* and *random forests* make predictions from an average of regression trees, each of which is built using a random sample of data and/or predictors. Each tree is built separately from the others.
- By contrast, *boosting* uses a weighted sum of trees, each of which is constructed by fitting a tree to the residual of the current fit. Thus, each new tree attempts to capture signal that is not yet accounted for by the current set of trees.

# Bayesian Additive Regression Trees — Details

## Bayesian Additive Regression Trees — Details

- BART is related to both random forests and boosting: each tree is constructed in a random manner as in bagging and random forests, and each tree tries to capture signal not yet accounted for by the current model, as in boosting.

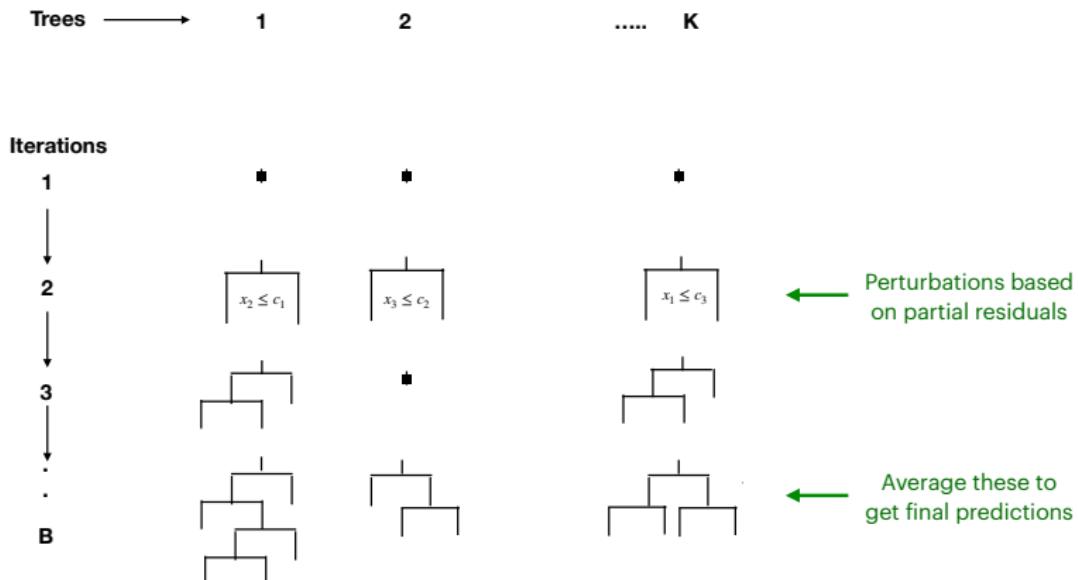
## Bayesian Additive Regression Trees — Details

- BART is related to both random forests and boosting: each tree is constructed in a random manner as in bagging and random forests, and each tree tries to capture signal not yet accounted for by the current model, as in boosting.
- The main novelty in BART is the way in which new trees are generated.

# Bayesian Additive Regression Trees — Details

- BART is related to both random forests and boosting: each tree is constructed in a random manner as in bagging and random forests, and each tree tries to capture signal not yet accounted for by the current model, as in boosting.
- The main novelty in BART is the way in which new trees are generated.
- BART can be applied to *regression*, *classification* and other problems; we will focus here just on regression.

# BART algorithm — the idea



# Bayesian Additive Regression Trees — Some Notation

## Bayesian Additive Regression Trees — Some Notation

- We let  $K$  denote the number of regression trees, and  $B$  the number of iterations for which the BART algorithm will be run.

## Bayesian Additive Regression Trees — Some Notation

- We let  $K$  denote the number of regression trees, and  $B$  the number of iterations for which the BART algorithm will be run.
- The notation  $\hat{f}_k^b(x)$  represents the prediction at  $x$  for the  $k$ th regression tree used in the  $b$ th iteration. At the end of each iteration, the  $K$  trees from that iteration will be summed, i.e.  $\hat{f}^b(x) = \sum_{k=1}^K \hat{f}_k^b(x)$  for  $b = 1, \dots, B$ .

# BART iterations

## BART iterations

- In the *first iteration* of the BART algorithm, all trees are initialized to have a single root node, with  $\hat{f}_k^1(x) = \frac{1}{nK} \sum_{i=1}^n y_i$ , the mean of the response values divided by the total number of trees. Thus,

$$\hat{f}^1(x) = \sum_{k=1}^K \hat{f}_k^1(x) = \frac{1}{n} \sum_{i=1}^n y_i$$

## BART iterations

- In the *first iteration* of the BART algorithm, all trees are initialized to have a single root node, with  $\hat{f}_k^1(x) = \frac{1}{nK} \sum_{i=1}^n y_i$ , the mean of the response values divided by the total number of trees. Thus,

$$\hat{f}^1(x) = \sum_{k=1}^K \hat{f}_k^1(x) = \frac{1}{n} \sum_{i=1}^n y_i$$

- In subsequent iterations*, BART updates each of the  $K$  trees, one at a time. In the  $b$ th iteration, to update the  $k$ th tree, we subtract from each response value the predictions from all but the  $k$ th tree, in order to obtain a *partial residual*

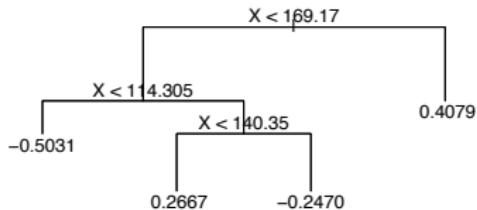
$$r_i = y_i - \sum_{k' < k} \hat{f}_{k'}^b(x_i) - \sum_{k' > k} \hat{f}_{k'}^{b-1}(x_i), \quad i = 1, \dots, n$$

## New trees are chosen by perturbations

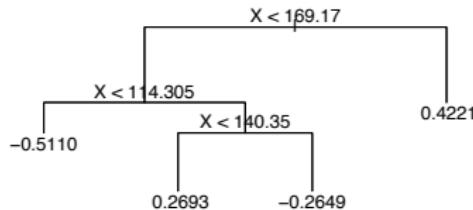
- Rather than fitting a fresh tree to this partial residual, BART randomly chooses a perturbation to the tree from the previous iteration ( $\hat{f}_k^{b-1}$ ) from a set of possible perturbations, favoring ones that improve the fit to the partial residual.
- There are two components to this perturbation:
  1. We may change the structure of the tree by adding or pruning branches.
  2. We may change the prediction in each terminal node of the tree.

## Examples of possible perturbations to a tree

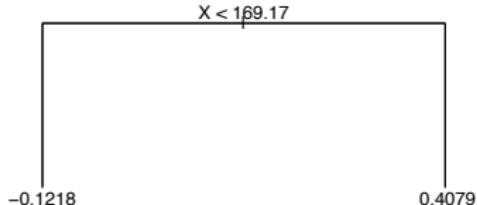
(a):  $\hat{f}_k^{b-1}(X)$



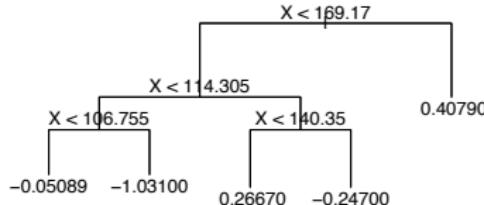
(b): Possibility #1 for  $\hat{f}_k^b(X)$



(c): Possibility #2 for  $\hat{f}_k^b(X)$



(d): Possibility #3 for  $\hat{f}_k^b(X)$



## What does BART Deliver?

## What does BART Deliver?

- The output of BART is a collection of prediction models,

$$\hat{f}^b(x) = \sum_{k=1}^K \hat{f}_k^b(x), \text{ for } b = 1, 2, \dots, B.$$

## What does BART Deliver?

- The output of BART is a collection of prediction models,

$$\hat{f}^b(x) = \sum_{k=1}^K \hat{f}_k^b(x), \text{ for } b = 1, 2, \dots, B.$$

- To obtain a single prediction, we simply take the average after some  $L$  *burn-in iterations*,  $\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^B \hat{f}^b(x)$ .

## What does BART Deliver?

- The output of BART is a collection of prediction models,

$$\hat{f}^b(x) = \sum_{k=1}^K \hat{f}_k^b(x), \text{ for } b = 1, 2, \dots, B.$$

- To obtain a single prediction, we simply take the average after some  $L$  *burn-in iterations*,  $\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^B \hat{f}^b(x)$ .
- The perturbation-style moves guard against overfitting since they limit how *hard* we fit the data in each iteration.

## What does BART Deliver?

- The output of BART is a collection of prediction models,

$$\hat{f}^b(x) = \sum_{k=1}^K \hat{f}_k^b(x), \text{ for } b = 1, 2, \dots, B.$$

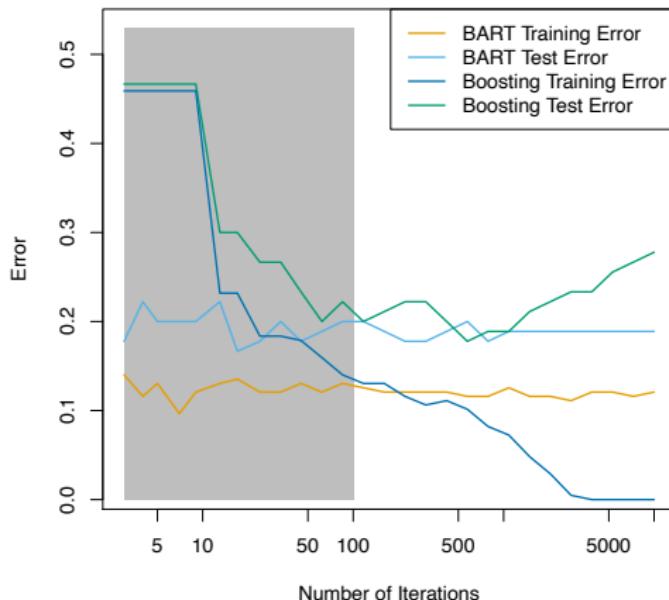
- To obtain a single prediction, we simply take the average after some  $L$  *burn-in iterations*,  $\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^B \hat{f}^b(x)$ .
- The perturbation-style moves guard against overfitting since they limit how *hard* we fit the data in each iteration.
- We can also compute quantities other than the average: for instance, the *percentiles* of  $f^{L+1}(x), \dots, f^B(x)$  provide a measure of uncertainty of the final prediction.

## BART applied to the Heart data

$K = 200$  trees; the number of iterations is increased to 10,000.

During the initial iterations (in gray), the test and training errors jump around a bit. After this initial burn-in period, the error rates settle down.

The tree perturbation process largely avoids overfitting.



BART is a Bayesian Method

## BART is a Bayesian Method

- It turns out that the BART method can be viewed as a *Bayesian* approach to fitting an ensemble of trees: each time we randomly perturb a tree in order to fit the residuals, we are in fact drawing a new tree from a *posterior* distribution.
- Furthermore, the BART algorithm can be viewed as a *Markov chain Monte Carlo* procedure for fitting the BART model.

## BART is a Bayesian Method

- It turns out that the BART method can be viewed as a *Bayesian* approach to fitting an ensemble of trees: each time we randomly perturb a tree in order to fit the residuals, we are in fact drawing a new tree from a *posterior* distribution.
- Furthermore, the BART algorithm can be viewed as a *Markov chain Monte Carlo* procedure for fitting the BART model.
- We typically choose large values for  $B$  and  $K$ , and a moderate value for  $L$ : for instance,  $K = 200$ ,  $B = 1,000$ , and  $L = 100$  are reasonable choices. BART has been shown to have impressive out-of-box performance — that is, it performs well with minimal tuning.

# Support Vector Machines

Here we approach the two-class classification problem in a direct way:

*We try and find a plane that separates the classes in feature space.*

If we cannot, we get creative in two ways:

- We soften what we mean by “separates”, and
- We enrich and enlarge the feature space so that separation is possible.

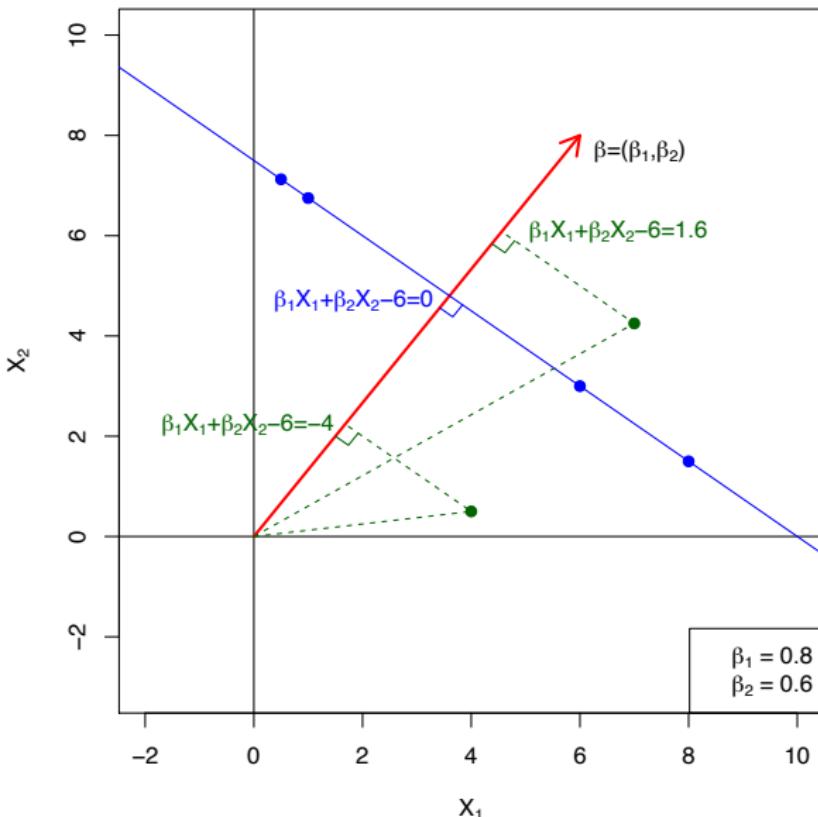
## What is a Hyperplane?

- A hyperplane in  $p$  dimensions is a flat affine subspace of dimension  $p - 1$ .
- In general the equation for a hyperplane has the form

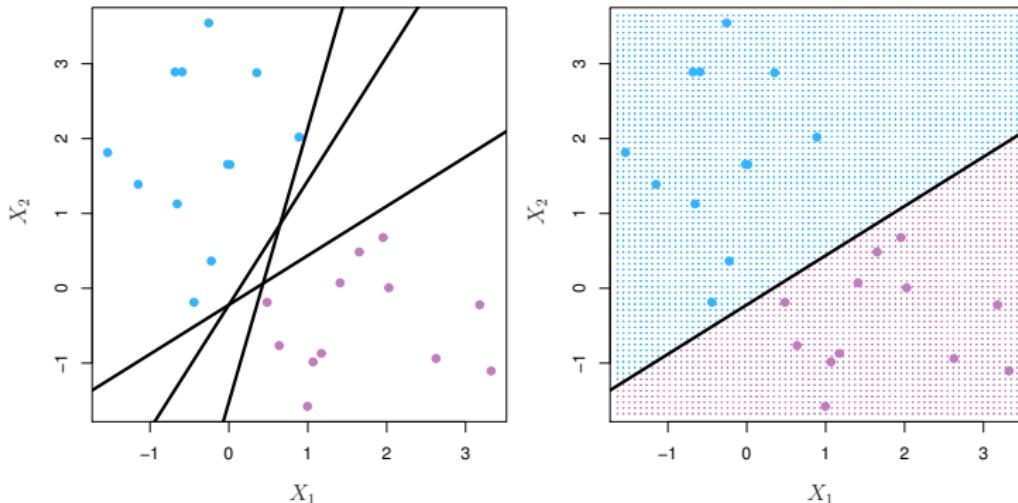
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$$

- In  $p = 2$  dimensions a hyperplane is a line.
- If  $\beta_0 = 0$ , the hyperplane goes through the origin, otherwise not.
- The vector  $\beta = (\beta_1, \beta_2, \dots, \beta_p)$  is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.

# Hyperplane in 2 Dimensions



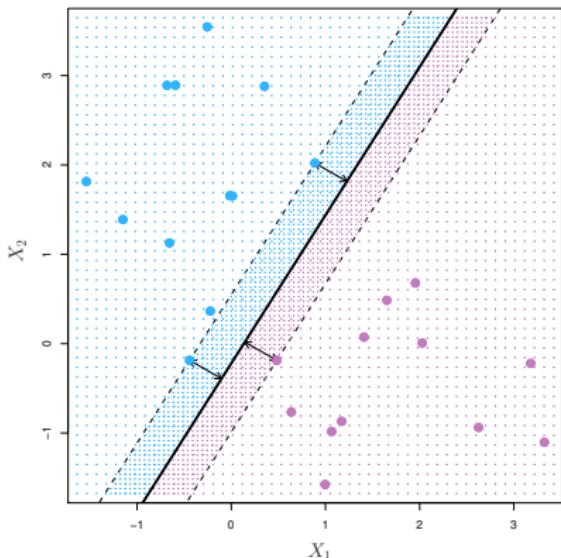
# Separating Hyperplanes



- If  $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ , then  $f(X) > 0$  for points on one side of the hyperplane, and  $f(X) < 0$  for points on the other.
- If we code the colored points as  $Y_i = +1$  for blue, say, and  $Y_i = -1$  for mauve, then if  $Y_i \cdot f(X_i) > 0$  for all  $i$ ,  $f(X) = 0$  defines a *separating hyperplane*.

# Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

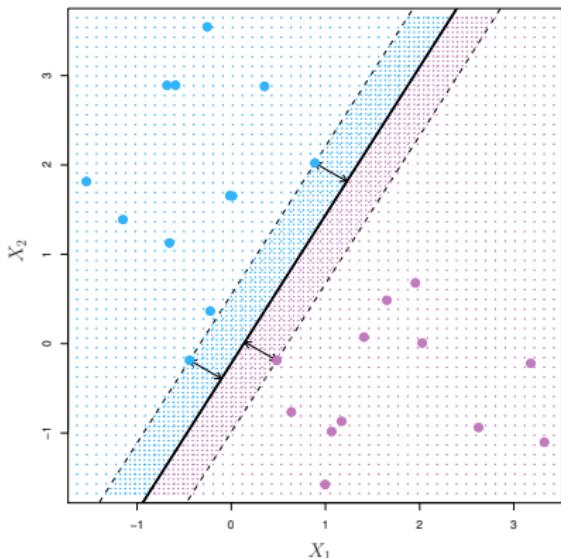
$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximize}} M$$

$$\text{subject to } \sum_{j=1}^p \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \text{for all } i = 1, \dots, N.$$

# Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximize}} M$$

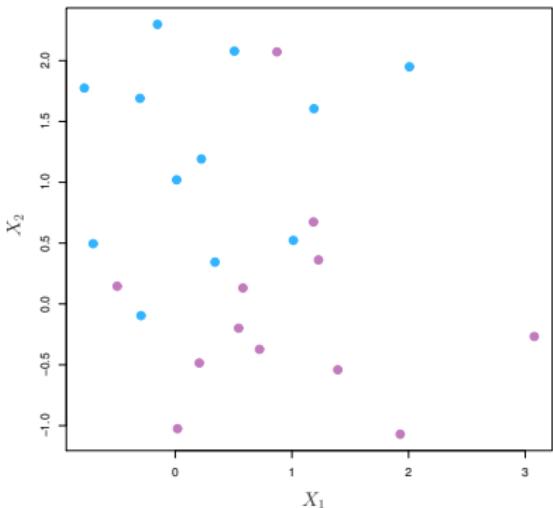
$$\text{subject to } \sum_{j=1}^p \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \text{for all } i = 1, \dots, N.$$



This can be rephrased as a convex quadratic program, and solved efficiently. The function `svm()` in package `e1071` solves this problem efficiently

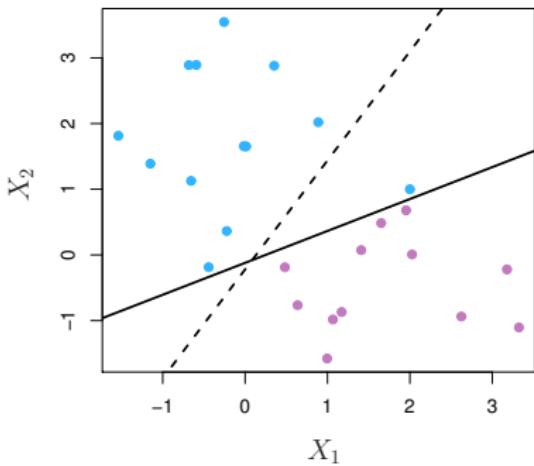
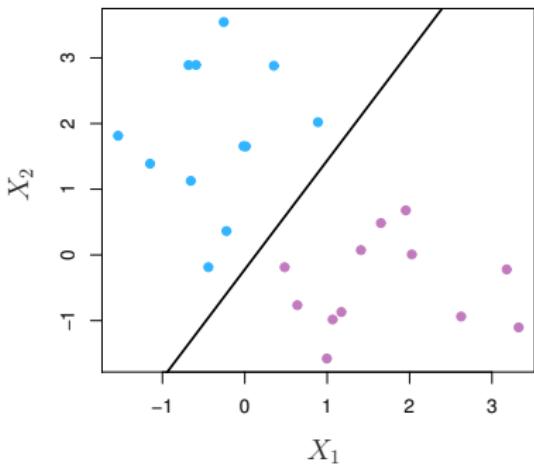
# Non-separable Data



The data on the left are not separable by a linear boundary.

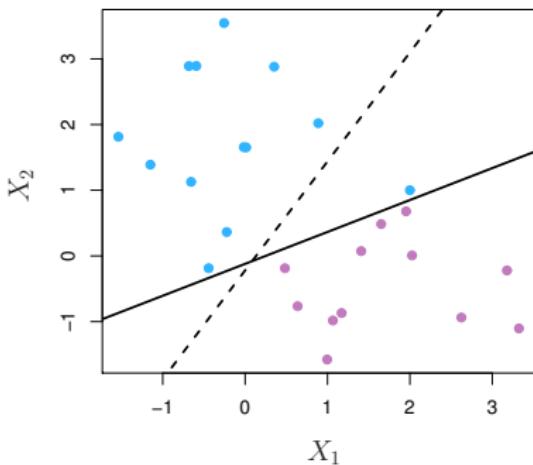
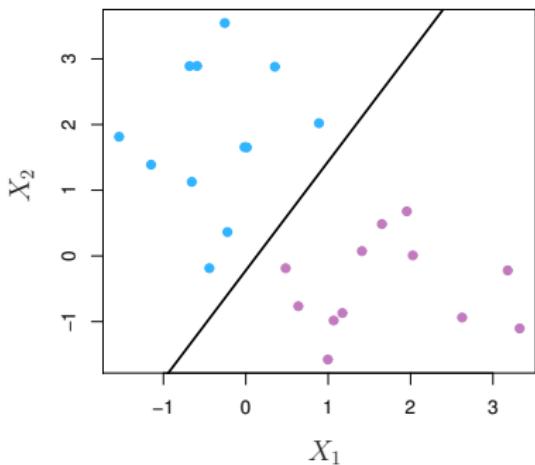
This is often the case, unless  $N < p$ .

# Noisy Data



Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

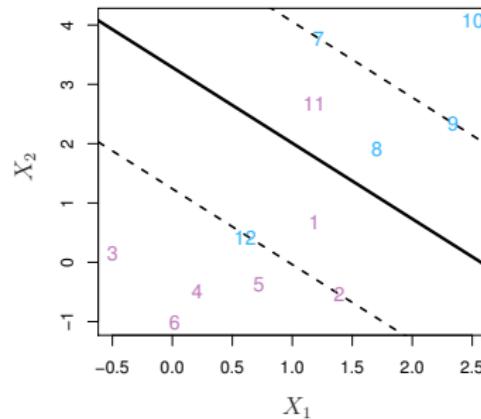
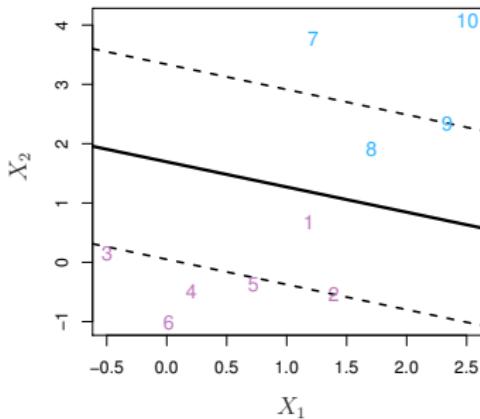
## Noisy Data



Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

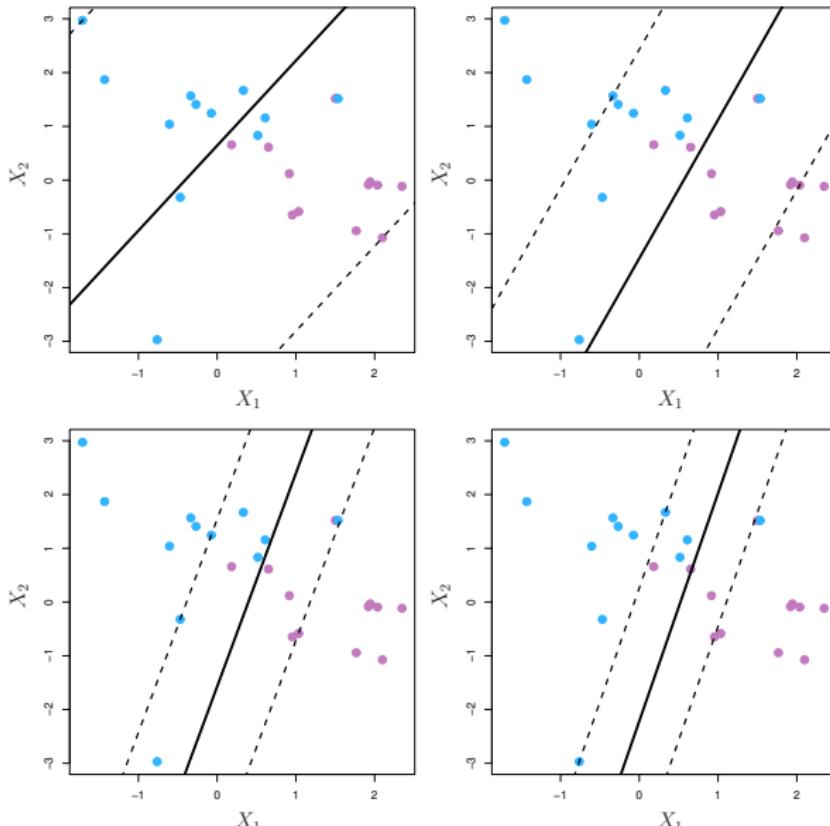
The *support vector classifier* maximizes a *soft* margin.

# Support Vector Classifier

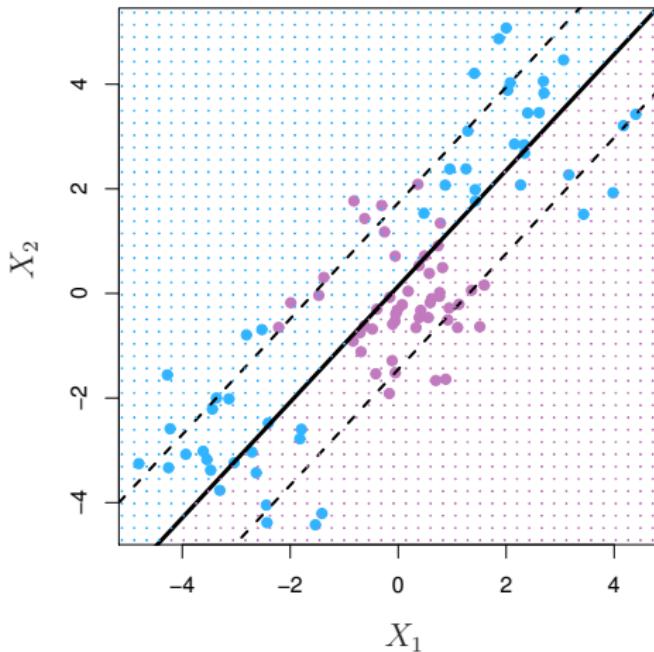


$$\begin{aligned} & \underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1, \\ & y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \\ & \epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C, \end{aligned}$$

$C$  is a regularization parameter



## Linear boundary can fail



Sometime a linear boundary simply won't work, no matter what value of  $C$ .

The example on the left is such a case.

What to do?

## Feature Expansion

- Enlarge the space of features by including transformations; e.g.  $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \dots$ . Hence go from a  $p$ -dimensional space to a  $M > p$  dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

## Feature Expansion

- Enlarge the space of features by including transformations; e.g.  $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \dots$ . Hence go from a  $p$ -dimensional space to a  $M > p$  dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

Example: Suppose we use  $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$  instead of just  $(X_1, X_2)$ . Then the decision boundary would be of the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

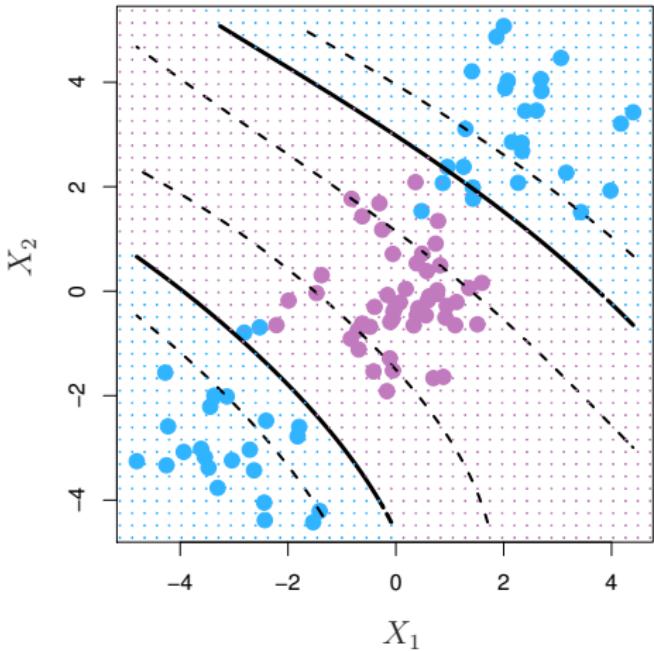
This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

## Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space

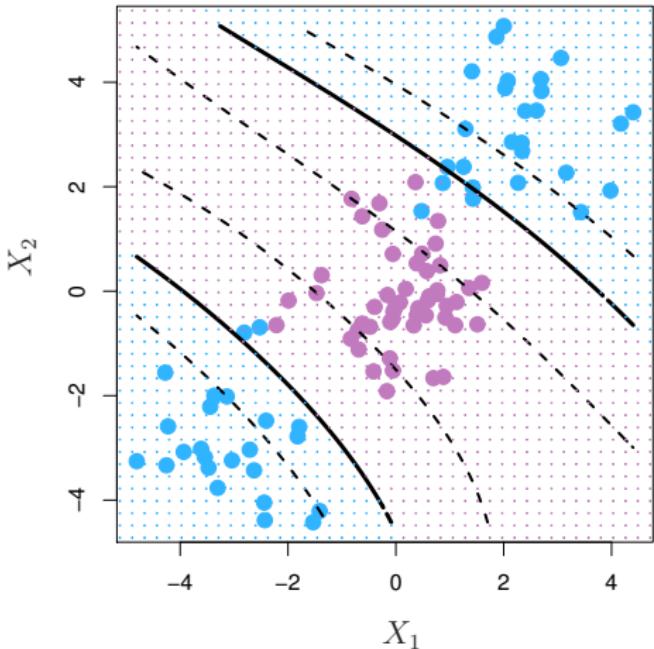


## Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space



$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 + \beta_7 X_2^3 + \beta_8 X_1 X_2^2 + \beta_9 X_1^2 X_2 = 0$$

# Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of *kernels*.
- Before we discuss these, we must understand the role of *inner products* in support-vector classifiers.

## Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j} \quad — \text{inner product between vectors}$$

## Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j} \quad - \text{inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle \quad - n \text{ parameters}$$

## Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j} \quad - \text{inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle \quad - n \text{ parameters}$$

- To estimate the parameters  $\alpha_1, \dots, \alpha_n$  and  $\beta_0$ , all we need are the  $\binom{n}{2}$  inner products  $\langle x_i, x_{i'} \rangle$  between all pairs of training observations.

## Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j} \quad - \text{inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle \quad - n \text{ parameters}$$

- To estimate the parameters  $\alpha_1, \dots, \alpha_n$  and  $\beta_0$ , all we need are the  $\binom{n}{2}$  inner products  $\langle x_i, x_{i'} \rangle$  between all pairs of training observations.

It turns out that most of the  $\hat{\alpha}_i$  can be zero:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i \langle x, x_i \rangle$$

$\mathcal{S}$  is the *support set* of indices  $i$  such that  $\hat{\alpha}_i > 0$ . [see slide 8]

## Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

$$K(x_i, x_{i'}) = \left( 1 + \sum_{j=1}^p x_{ij} x_{i'j} \right)^d$$

computes the inner-products needed for  $d$  dimensional polynomials —  $\binom{p+d}{d}$  basis functions!

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij}x_{i'j}\right)^d$$

computes the inner-products needed for  $d$  dimensional polynomials —  $\binom{p+d}{d}$  basis functions!

*Try it for  $p = 2$  and  $d = 2$ .*

# Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij}x_{i'j}\right)^d$$

computes the inner-products needed for  $d$  dimensional polynomials —  $\binom{p+d}{d}$  basis functions!

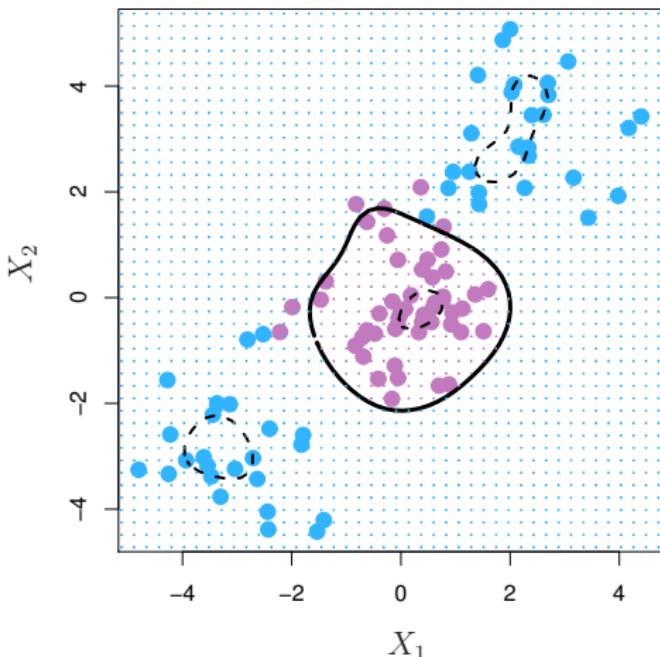
*Try it for  $p = 2$  and  $d = 2$ .*

- The solution has the form

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i).$$

## Radial Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2).$$

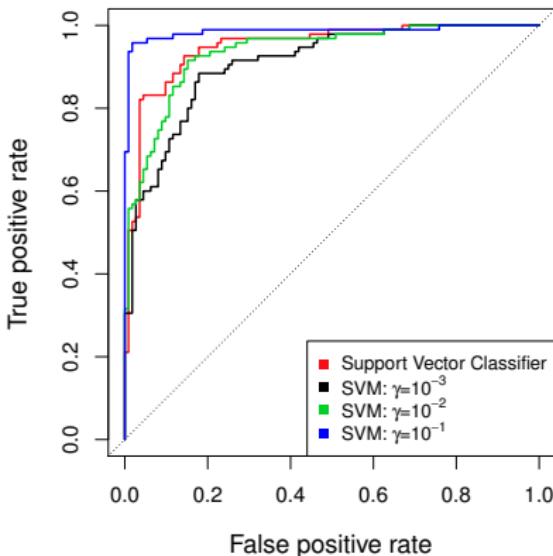
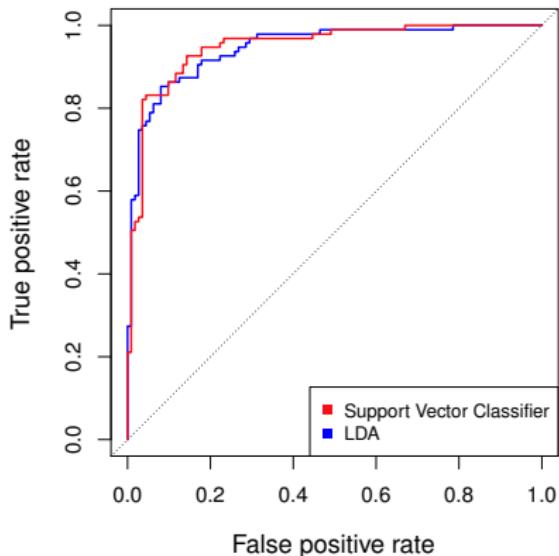


$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space;  
very high dimensional.

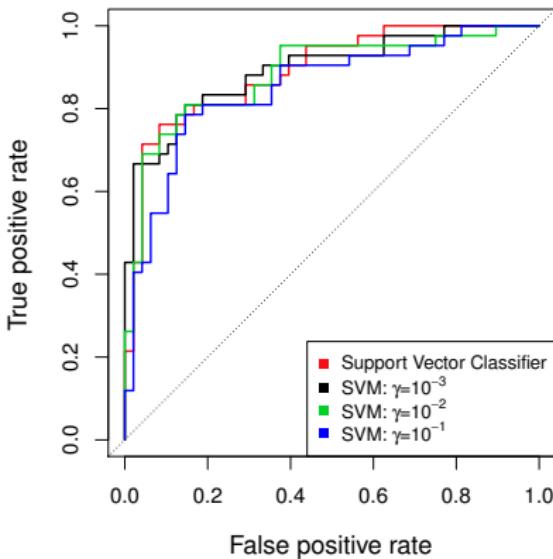
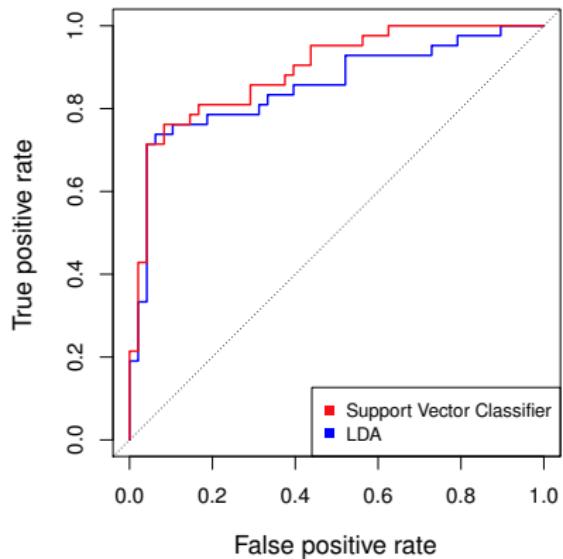
Controls variance by  
squashing down most  
dimensions severely

## Example: Heart Data



ROC curve is obtained by changing the threshold 0 to threshold  $t$  in  $\hat{f}(X) > t$ , and recording *false positive* and *true positive* rates as  $t$  varies. Here we see ROC curves on training data.

## Example continued: Heart Test Data



## SVMs: more than 2 classes?

The SVM as defined works for  $K = 2$  classes. What do we do if we have  $K > 2$  classes?

## SVMs: more than 2 classes?

The SVM as defined works for  $K = 2$  classes. What do we do if we have  $K > 2$  classes?

**OVA** One versus All. Fit  $K$  different 2-class SVM classifiers  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$ ; each class versus the rest. Classify  $x^*$  to the class for which  $\hat{f}_k(x^*)$  is largest.

## SVMs: more than 2 classes?

The SVM as defined works for  $K = 2$  classes. What do we do if we have  $K > 2$  classes?

**OVA** One versus All. Fit  $K$  different 2-class SVM classifiers  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$ ; each class versus the rest. Classify  $x^*$  to the class for which  $\hat{f}_k(x^*)$  is largest.

**OVO** One versus One. Fit all  $\binom{K}{2}$  pairwise classifiers  $\hat{f}_{k\ell}(x)$ . Classify  $x^*$  to the class that wins the most pairwise competitions.

## SVMs: more than 2 classes?

The SVM as defined works for  $K = 2$  classes. What do we do if we have  $K > 2$  classes?

**OVA** One versus All. Fit  $K$  different 2-class SVM classifiers  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$ ; each class versus the rest. Classify  $x^*$  to the class for which  $\hat{f}_k(x^*)$  is largest.

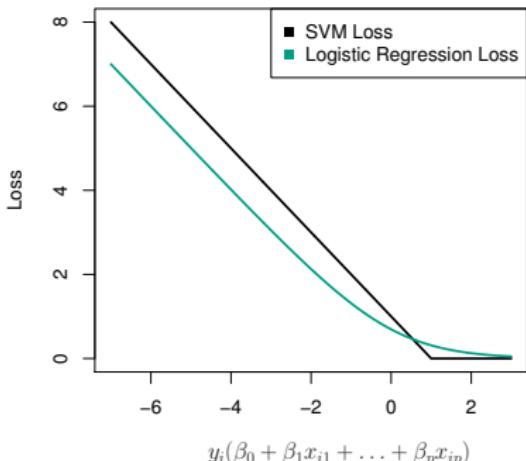
**OVO** One versus One. Fit all  $\binom{K}{2}$  pairwise classifiers  $\hat{f}_{k\ell}(x)$ . Classify  $x^*$  to the class that wins the most pairwise competitions.

Which to choose? If  $K$  is not too large, use OVO.

## Support Vector versus Logistic Regression?

With  $f(X) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$  can rephrase support-vector classifier optimization as

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^n \max [0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$



This has the form  
*loss plus penalty.*

The loss is known as the  
*hinge loss.*

Very similar to “loss” in logistic regression (negative log-likelihood).

## Which to use: SVM or Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not, LR (with ridge penalty) and SVM very similar.
- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.

# Deep Learning

# Deep Learning

Neural networks became popular in the 1980s.

Lots of successes, hype, and great conferences: NeurIPS,  
Snowbird.

# Deep Learning

Neural networks became popular in the 1980s.

Lots of successes, hype, and great conferences: NeurIPS, Snowbird.

Then along came SVMs, Random Forests and Boosting in the 1990s, and Neural Networks took a back seat.

# Deep Learning

Neural networks became popular in the 1980s.

Lots of successes, hype, and great conferences: NeurIPS, Snowbird.

Then along came SVMs, Random Forests and Boosting in the 1990s, and Neural Networks took a back seat.

Re-emerged around 2010 as *Deep Learning*.

By 2020s very dominant and successful.

Part of success due to vast improvements in computing power, larger training sets, and software: Tensorflow and PyTorch.

# Deep Learning

Neural networks became popular in the 1980s.

Lots of successes, hype, and great conferences: NeurIPS, Snowbird.

Then along came SVMs, Random Forests and Boosting in the 1990s, and Neural Networks took a back seat.

Re-emerged around 2010 as *Deep Learning*.

By 2020s very dominant and successful.

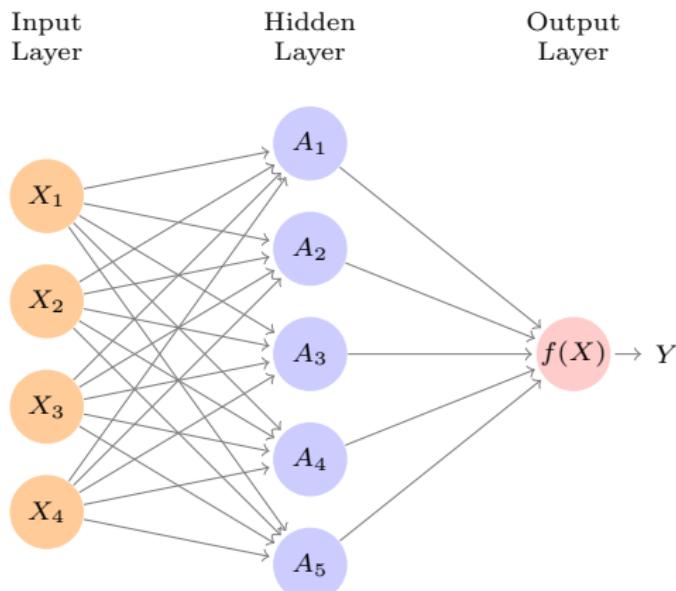
Part of success due to vast improvements in computing power, larger training sets, and software: Tensorflow and PyTorch.

Much of the credit goes to three pioneers and their students: Yann LeCun, Geoffrey Hinton and Yoshua Bengio, who received the 2019 ACM Turing Award for their work in Neural Networks.

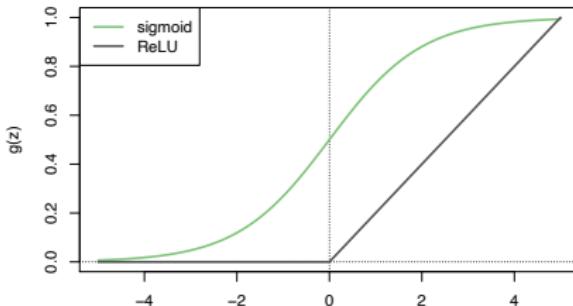


# Single Layer Neural Network

$$\begin{aligned}f(X) &= \beta_0 + \sum_{k=1}^K \beta_k h_k(X) \\&= \beta_0 + \sum_{k=1}^K \beta_k g(w_{k0} + \sum_{j=1}^p w_{kj} X_j).\end{aligned}$$

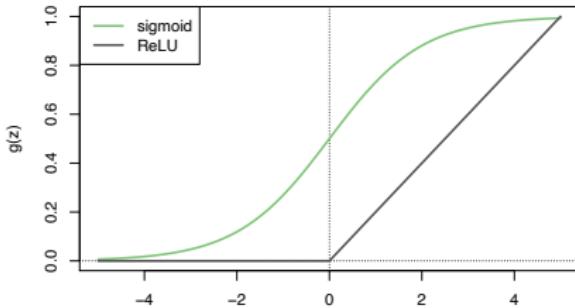


## Details



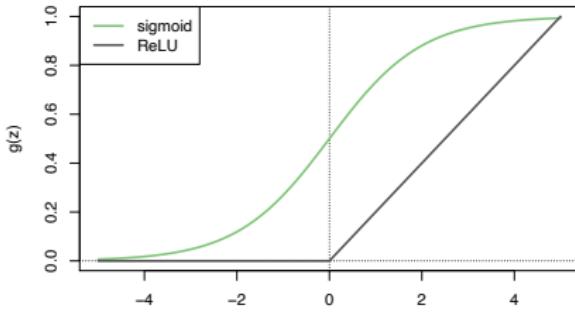
- $A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^p w_{kj} X_j)$  are called the *activations* in the *hidden layer*.

## Details



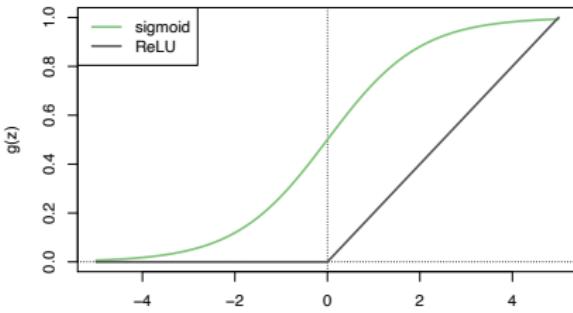
- $A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^p w_{kj} X_j)$  are called the *activations* in the *hidden layer*.
- $g(z)$  is called the *activation function*. Popular are the *sigmoid* and *rectified linear*, shown in figure.

## Details



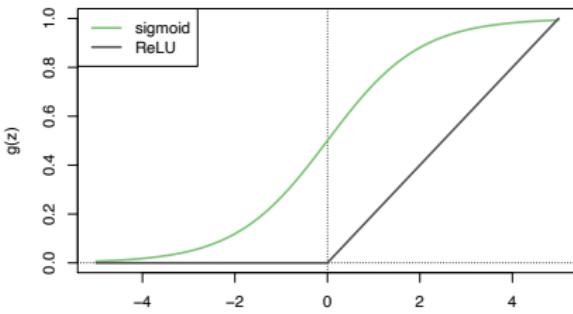
- $A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^p w_{kj} X_j)$  are called the *activations* in the *hidden layer*.
- $g(z)$  is called the *activation function*. Popular are the *sigmoid* and *rectified linear*, shown in figure.
- Activation functions in hidden layers are typically nonlinear, otherwise the model collapses to a linear model.

## Details



- $A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^p w_{kj} X_j)$  are called the *activations* in the *hidden layer*.
- $g(z)$  is called the *activation function*. Popular are the *sigmoid* and *rectified linear*, shown in figure.
- Activation functions in hidden layers are typically nonlinear, otherwise the model collapses to a linear model.
- So the activations are like derived features — nonlinear transformations of linear combinations of the features.

## Details



- $A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^p w_{kj} X_j)$  are called the *activations* in the *hidden layer*.
- $g(z)$  is called the *activation function*. Popular are the *sigmoid* and *rectified linear*, shown in figure.
- Activation functions in hidden layers are typically nonlinear, otherwise the model collapses to a linear model.
- So the activations are like derived features — nonlinear transformations of linear combinations of the features.
- The model is fit by minimizing  $\sum_{i=1}^n (y_i - f(x_i))^2$  (e.g. for regression).

## Example: MNIST Digits

0 1 2 3 4 5 6 7 8 9

Handwritten digits

0 1 2 3 4 5 6 7 8 9

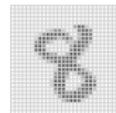
28 × 28 grayscale images

0 1 2 3 4 5 6 7 8 9

60K train, 10K test images

0 1 2 3 4 5 6 7 8 9

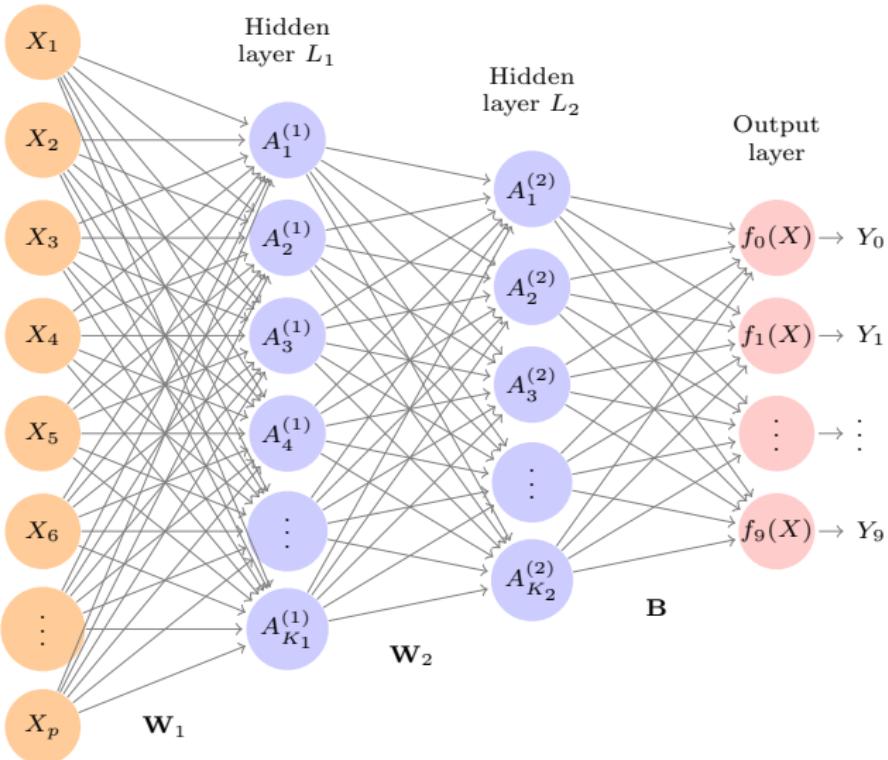
Features are the 784 pixel  
grayscale values  $\in (0, 255)$



Labels are the digit class 0–9

- Goal: build a classifier to predict the image class.
- We build a two-layer network with 256 units at first layer, 128 units at second layer, and 10 units at output layer.
- Along with intercepts (called *biases*) there are 235,146 parameters (referred to as *weights*)

Input  
layer



## Details of Output Layer

- Let  $Z_m = \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} A_\ell^{(2)}$ ,  $m = 0, 1, \dots, 9$  be 10 linear combinations of activations at second layer.
- Output activation function encodes the *softmax* function

$$f_m(X) = \Pr(Y = m|X) = \frac{e^{Z_m}}{\sum_{\ell=0}^9 e^{Z_\ell}}.$$

## Details of Output Layer

- Let  $Z_m = \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} A_\ell^{(2)}$ ,  $m = 0, 1, \dots, 9$  be 10 linear combinations of activations at second layer.
- Output activation function encodes the *softmax* function

$$f_m(X) = \Pr(Y = m|X) = \frac{e^{Z_m}}{\sum_{\ell=0}^9 e^{Z_\ell}}.$$

- We fit the model by minimizing the negative multinomial log-likelihood (or cross-entropy):

$$-\sum_{i=1}^n \sum_{m=0}^9 y_{im} \log(f_m(x_i)).$$

- $y_{im}$  is 1 if true class for observation  $i$  is  $m$ , else 0 — i.e. *one-hot encoded*.

## Results

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

- Early success for neural networks in the 1990s.
- With so many parameters, regularization is essential.
- Some details of regularization and fitting will come later.

## Results

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

- Early success for neural networks in the 1990s.
- With so many parameters, regularization is essential.
- Some details of regularization and fitting will come later.
- Very overworked problem — best reported rates are < 0.5%!
- Human error rate is reported to be around 0.2%, or 20 of the 10K test images.

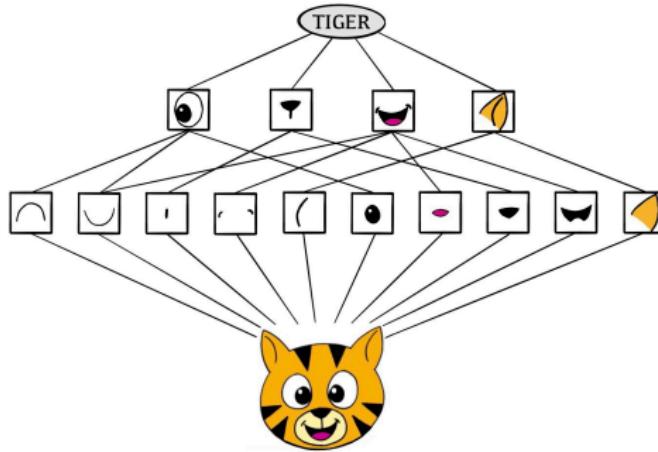
# Convolutional Neural Network — CNN



- Major success story for classifying images.
- Shown are samples from **CIFAR100** database.  $32 \times 32$  color natural images, with 100 classes.
- $50K$  training images,  $10K$  test images.

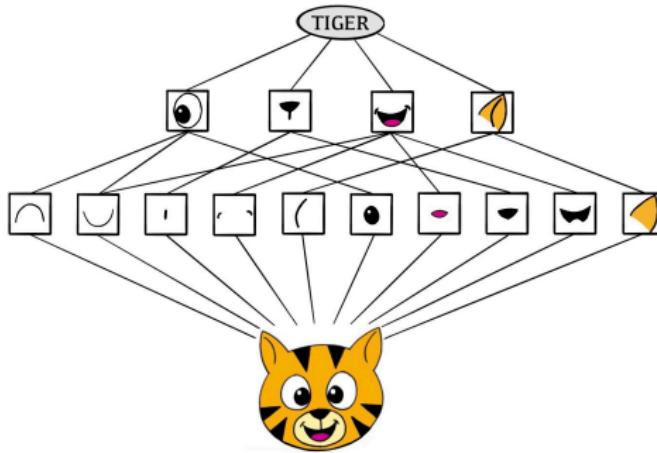
Each image is a three-dimensional array or *feature map*:  
 $32 \times 32 \times 3$  array of 8-bit numbers. The last dimension  
represents the three color channels for red, green and blue.

# How CNNs Work



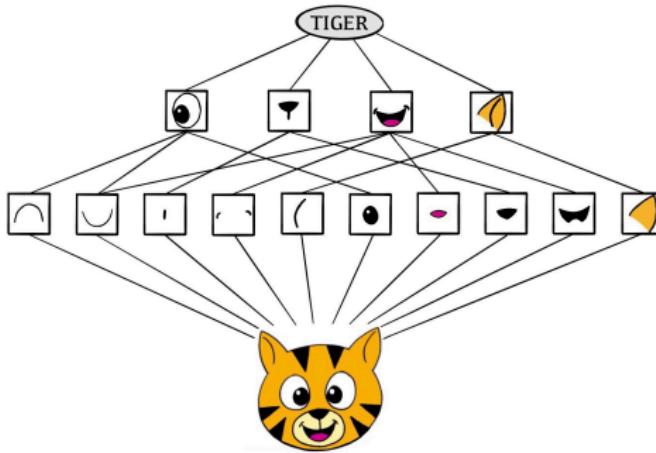
- The CNN builds up an image in a hierarchical fashion.

# How CNNs Work



- The CNN builds up an image in a hierarchical fashion.
- Edges and shapes are recognized and pieced together to form more complex shapes, eventually assembling the target image.

# How CNNs Work



- The CNN builds up an image in a hierarchical fashion.
- Edges and shapes are recognized and pieced together to form more complex shapes, eventually assembling the target image.
- This hierarchical construction is achieved using *convolution* and *pooling* layers.

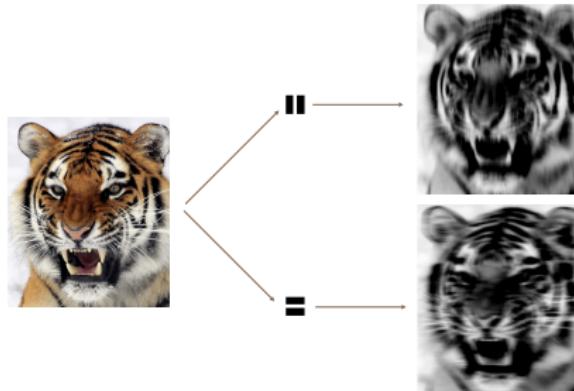
## Convolution Filter

$$\text{Input Image} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix} \quad \text{Convolution Filter} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}.$$

$$\text{Convolved Image} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

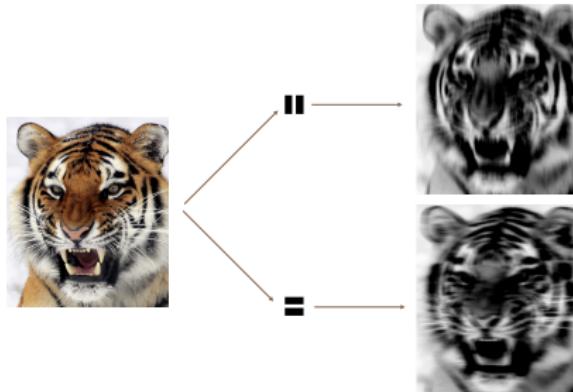
- The filter is itself an image, and represents a small shape, edge etc.
- We slide it around the input image, scoring for matches.
- The scoring is done via *dot-products*, illustrated above.
- If the subimage of the input image is similar to the filter, the score is high, otherwise low.
- The filters are *learned* during training.

## Convolution Example



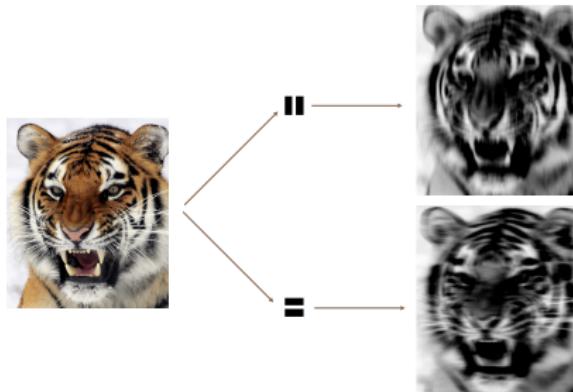
- The idea of convolution with a filter is to find common patterns that occur in different parts of the image.

## Convolution Example



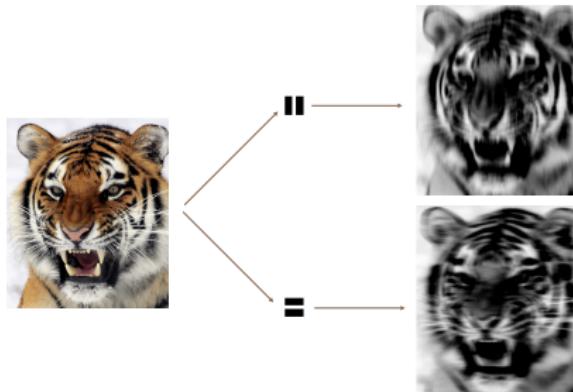
- The idea of convolution with a filter is to find common patterns that occur in different parts of the image.
- The two filters shown here highlight vertical and horizontal stripes.

## Convolution Example



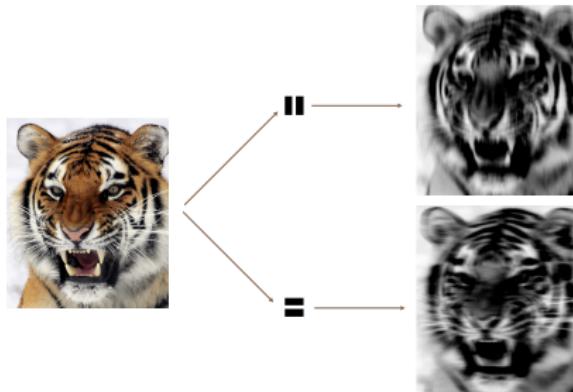
- The idea of convolution with a filter is to find common patterns that occur in different parts of the image.
- The two filters shown here highlight vertical and horizontal stripes.
- The result of the convolution is a new feature map.

## Convolution Example



- The idea of convolution with a filter is to find common patterns that occur in different parts of the image.
- The two filters shown here highlight vertical and horizontal stripes.
- The result of the convolution is a new feature map.
- Since images have three colors channels, the filter does as well: one filter per channel, and dot-products are summed.

## Convolution Example



- The idea of convolution with a filter is to find common patterns that occur in different parts of the image.
- The two filters shown here highlight vertical and horizontal stripes.
- The result of the convolution is a new feature map.
- Since images have three colors channels, the filter does as well: one filter per channel, and dot-products are summed.
- The weights in the filters are *learned* by the network.

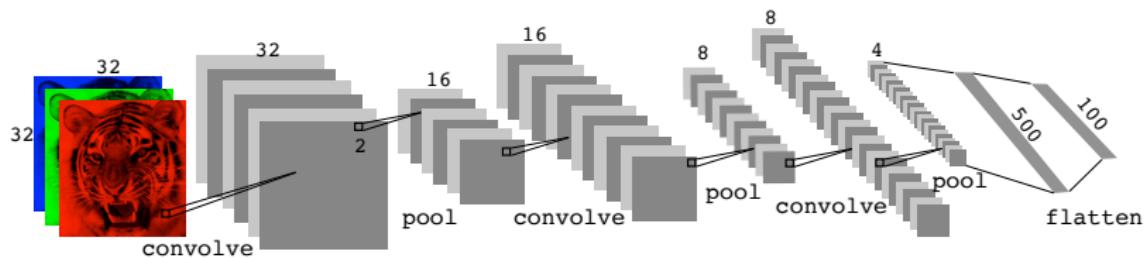
# Pooling

Max pool

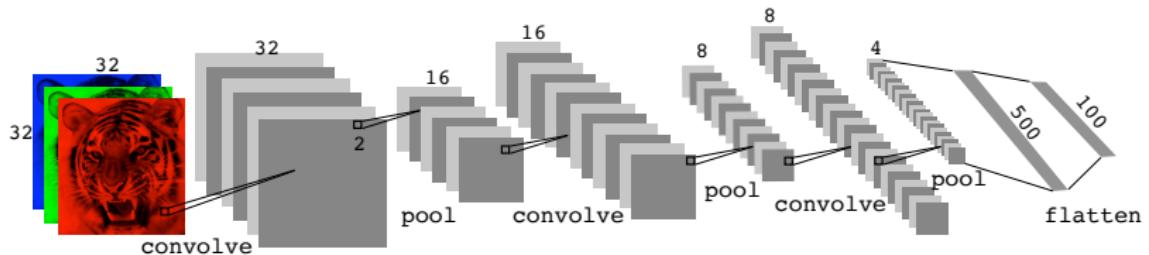
$$\begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

- Each non-overlapping  $2 \times 2$  block is replaced by its maximum.
- This sharpens the feature identification.
- Allows for locational invariance.
- Reduces the dimension by a factor of 4 — i.e. factor of 2 in each dimension.

# Architecture of a CNN

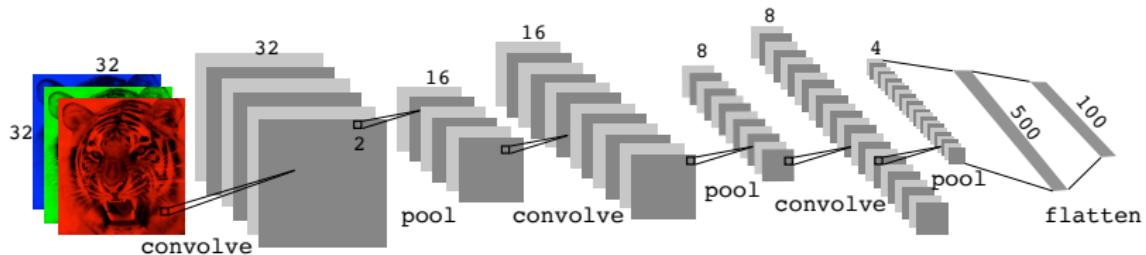


# Architecture of a CNN



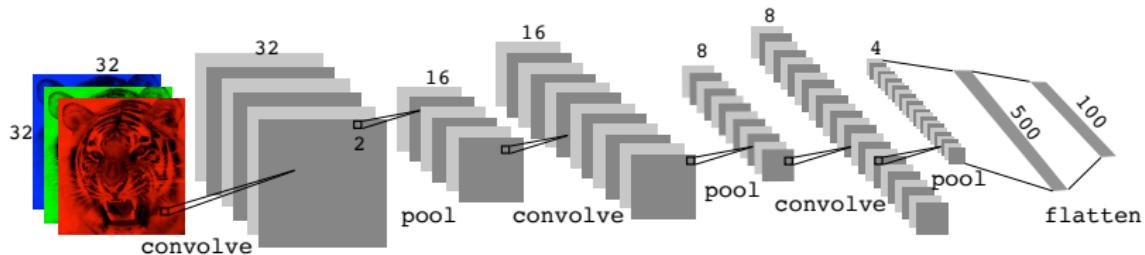
- Many convolve + pool layers.

# Architecture of a CNN



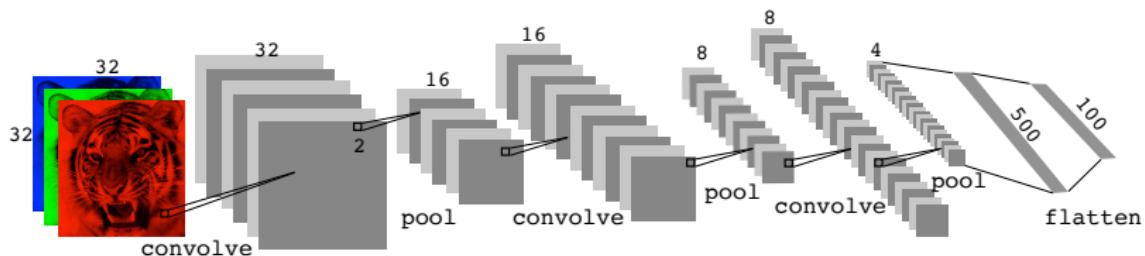
- Many convolve + pool layers.
- Filters are typically small, e.g. each channel  $3 \times 3$ .

# Architecture of a CNN



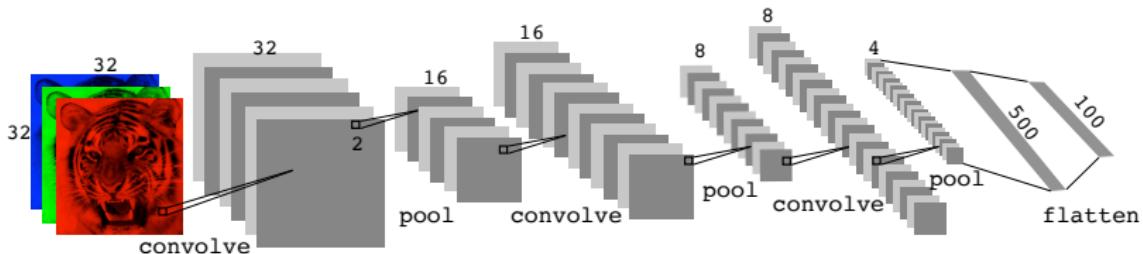
- Many convolve + pool layers.
- Filters are typically small, e.g. each channel  $3 \times 3$ .
- Each filter creates a new channel in convolution layer.

# Architecture of a CNN



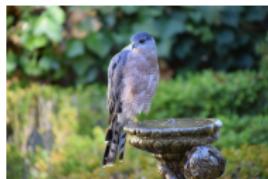
- Many convolve + pool layers.
- Filters are typically small, e.g. each channel  $3 \times 3$ .
- Each filter creates a new channel in convolution layer.
- As pooling reduces size, the number of filters/channels is typically increased.

# Architecture of a CNN



- Many convolve + pool layers.
- Filters are typically small, e.g. each channel  $3 \times 3$ .
- Each filter creates a new channel in convolution layer.
- As pooling reduces size, the number of filters/channels is typically increased.
- Number of layers can be very large. E.g. **resnet50** trained on **imagenet** 1000-class image data base has 50 layers!

# Using Pretrained Networks to Classify Images



# Using Pretrained Networks to Classify Images



flamingo

Cooper's hawk

Cooper's hawk

flamingo	0.83	kite (raptor)	0.60	fountain	0.35
spoonbill	0.17	great grey owl	0.09	nail	0.12
white stork	0.00	robin	0.06	hook	0.07

Lhasa Apso

cat

Cape weaver

Tibetan terrier	0.56	Old English sheepdog	0.82	jacamar	0.28
Lhasa	0.32	Shih-Tzu	0.04	macaw	0.12
cocker spaniel	0.03	Persian cat	0.04	robin	0.12

Here we use the 50-layer **resnet50** network trained on the 1000-class **imagenet** corpus to classify some photographs.

## Document Classification: IMDB Movie Reviews

The **IMDB** corpus consists of user-supplied movie ratings for a large collection of movies. Each has been labeled for **sentiment** as **positive** or **negative**. Here is the beginning of a negative review:

This has to be one of the worst films of the 1990s. When my friends & I were watching this film (being the target audience it was aimed at) we just sat & watched the first half an hour with our jaws touching the floor at how bad it really was. The rest of the time, everyone else in the theater just started talking to each other, leaving or generally crying into their popcorn ...

We have labeled training and test sets, each consisting of 25,000 reviews, and each balanced with regard to sentiment.

## Document Classification: IMDB Movie Reviews

The **IMDB** corpus consists of user-supplied movie ratings for a large collection of movies. Each has been labeled for **sentiment** as **positive** or **negative**. Here is the beginning of a negative review:

This has to be one of the worst films of the 1990s. When my friends & I were watching this film (being the target audience it was aimed at) we just sat & watched the first half an hour with our jaws touching the floor at how bad it really was. The rest of the time, everyone else in the theater just started talking to each other, leaving or generally crying into their popcorn ...

We have labeled training and test sets, each consisting of 25,000 reviews, and each balanced with regard to sentiment.

We wish to build a classifier to predict the sentiment of a review.

## Featurization: Bag-of-Words

Documents have different lengths, and consist of sequences of words. How do we create features  $X$  to characterize a document?

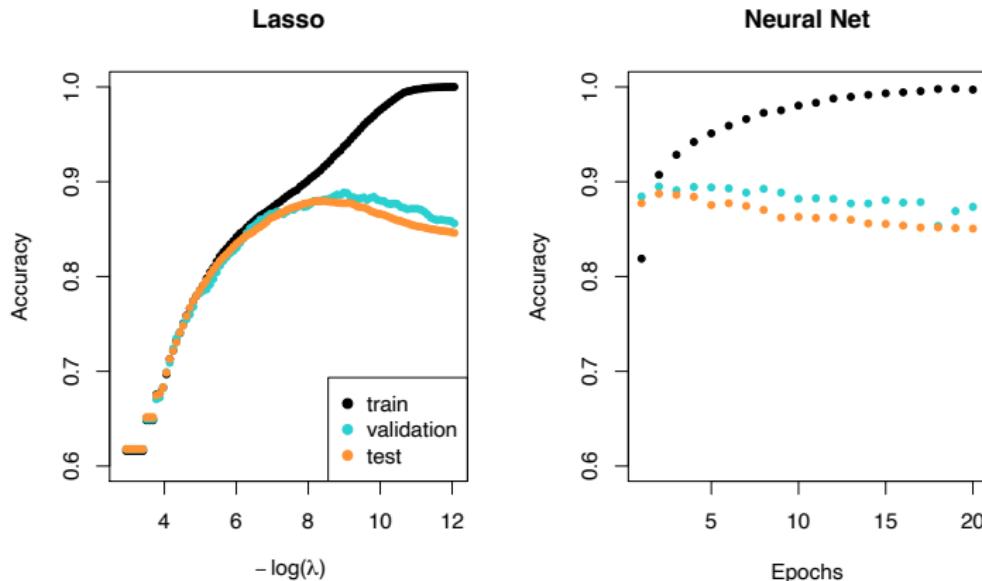
- From a dictionary, identify the  $10K$  most frequently occurring words.
- Create a binary vector of length  $p = 10K$  for each document, and score a 1 in every position that the corresponding word occurred.
- With  $n$  documents, we now have a  $n \times p$  *sparse* feature matrix  $\mathbf{X}$ .
- We compare a lasso logistic regression model to a two-hidden-layer neural network on the next slide. (No convolutions here!)

## Featurization: Bag-of-Words

Documents have different lengths, and consist of sequences of words. How do we create features  $X$  to characterize a document?

- From a dictionary, identify the  $10K$  most frequently occurring words.
- Create a binary vector of length  $p = 10K$  for each document, and score a 1 in every position that the corresponding word occurred.
- With  $n$  documents, we now have a  $n \times p$  *sparse* feature matrix  $\mathbf{X}$ .
- We compare a lasso logistic regression model to a two-hidden-layer neural network on the next slide. (No convolutions here!)
- Bag-of-words are *unigrams*. We can instead use *bigrams* (occurrences of adjacent word pairs), and in general *m-grams*.

# Lasso versus Neural Network — IMDB Reviews



- Simpler lasso logistic regression model works as well as neural network in this case.
- **glmnet** was used to fit the lasso model, and is very effective because it can exploit sparsity in the  $\mathbf{X}$  matrix.

# Recurrent Neural Networks

Often data arise as sequences:

- Documents are sequences of words, and their relative positions have meaning.
- Time-series such as weather data or financial indices.
- Recorded speech or music.
- Handwriting, such as doctor's notes.

RNNs build models that take into account this sequential nature of the data, and build a memory of the past.

# Recurrent Neural Networks

Often data arise as sequences:

- Documents are sequences of words, and their relative positions have meaning.
- Time-series such as weather data or financial indices.
- Recorded speech or music.
- Handwriting, such as doctor's notes.

RNNs build models that take into account this sequential nature of the data, and build a memory of the past.

# Recurrent Neural Networks

Often data arise as sequences:

- Documents are sequences of words, and their relative positions have meaning.
- Time-series such as weather data or financial indices.
- Recorded speech or music.
- Handwriting, such as doctor's notes.

RNNs build models that take into account this sequential nature of the data, and build a memory of the past.

- The feature for each observation is a *sequence* of vectors  $X = \{X_1, X_2, \dots, X_L\}$ .

# Recurrent Neural Networks

Often data arise as sequences:

- Documents are sequences of words, and their relative positions have meaning.
- Time-series such as weather data or financial indices.
- Recorded speech or music.
- Handwriting, such as doctor's notes.

RNNs build models that take into account this sequential nature of the data, and build a memory of the past.

- The feature for each observation is a *sequence* of vectors  $X = \{X_1, X_2, \dots, X_L\}$ .
- The target  $Y$  is often of the usual kind — e.g. a single variable such as **Sentiment**, or a one-hot vector for multiclass.

# Recurrent Neural Networks

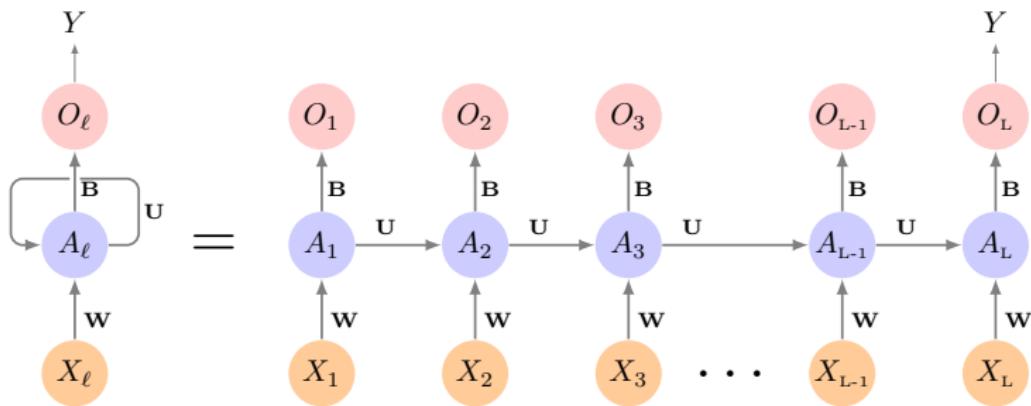
Often data arise as sequences:

- Documents are sequences of words, and their relative positions have meaning.
- Time-series such as weather data or financial indices.
- Recorded speech or music.
- Handwriting, such as doctor's notes.

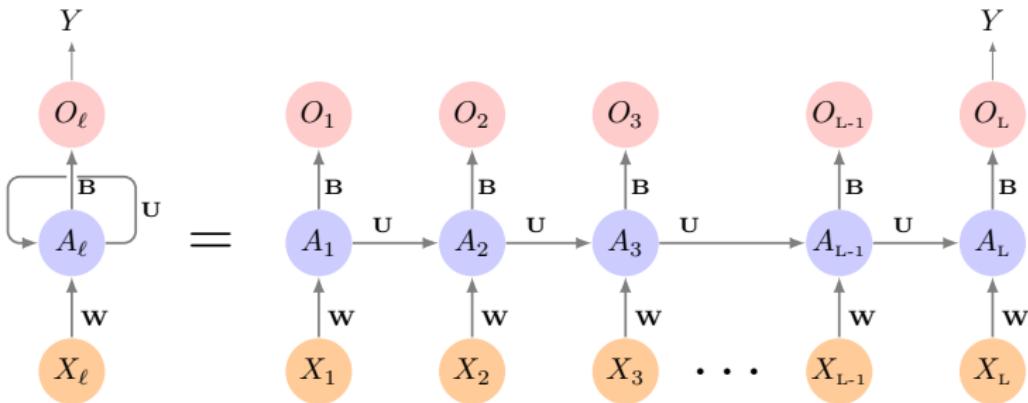
RNNs build models that take into account this sequential nature of the data, and build a memory of the past.

- The feature for each observation is a *sequence* of vectors  $X = \{X_1, X_2, \dots, X_L\}$ .
- The target  $Y$  is often of the usual kind — e.g. a single variable such as **Sentiment**, or a one-hot vector for multiclass.
- However,  $Y$  can also be a sequence, such as the same document in a different language.

# Simple Recurrent Neural Network Architecture

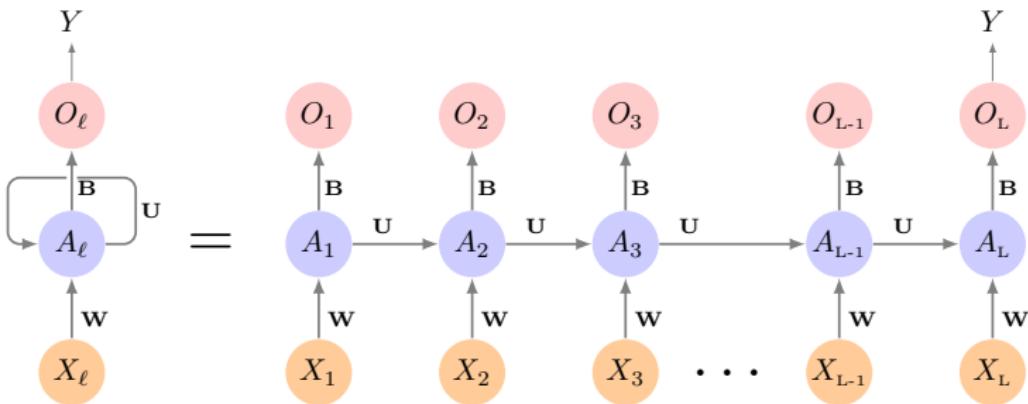


# Simple Recurrent Neural Network Architecture



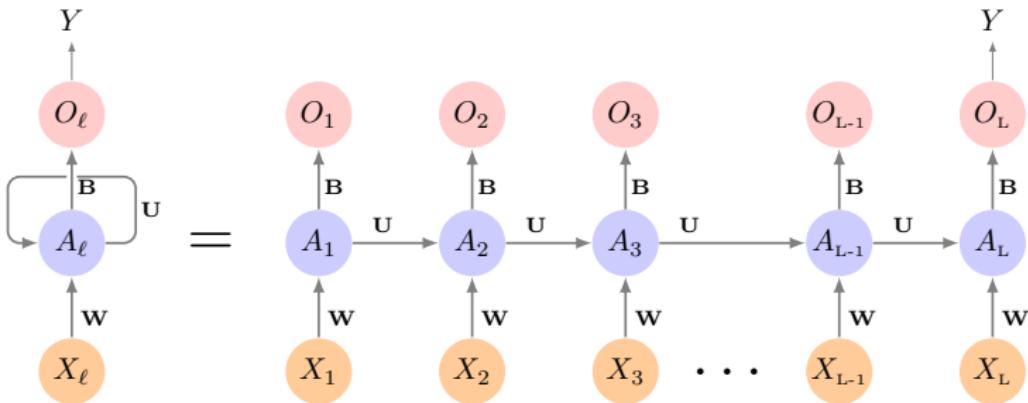
- The hidden layer is a sequence of vectors  $A_\ell$ , receiving as input  $X_\ell$  as well as  $A_{\ell-1}$ .  $A_\ell$  produces an output  $O_\ell$ .

# Simple Recurrent Neural Network Architecture



- The hidden layer is a sequence of vectors  $A_\ell$ , receiving as input  $X_\ell$  as well as  $A_{\ell-1}$ .  $A_\ell$  produces an output  $O_\ell$ .
- The *same* weights  $\mathbf{W}$ ,  $\mathbf{U}$  and  $\mathbf{B}$  are used at each step in the sequence — hence the term *recurrent*.

# Simple Recurrent Neural Network Architecture



- The hidden layer is a sequence of vectors  $A_\ell$ , receiving as input  $X_\ell$  as well as  $A_{\ell-1}$ .  $A_\ell$  produces an output  $O_\ell$ .
- The *same* weights  $\mathbf{W}$ ,  $\mathbf{U}$  and  $\mathbf{B}$  are used at each step in the sequence — hence the term *recurrent*.
- The  $A_\ell$  sequence represents an evolving model for the response that is updated as each element  $X_\ell$  is processed.

## RNN in Detail

Suppose  $X_\ell = (X_{\ell 1}, X_{\ell 2}, \dots, X_{\ell p})$  has  $p$  components, and  $A_\ell = (A_{\ell 1}, A_{\ell 2}, \dots, A_{\ell K})$  has  $K$  components. Then the computation at the  $k$ th components of hidden unit  $A_\ell$  is

$$\begin{aligned} A_{\ell k} &= g\left(w_{k0} + \sum_{j=1}^p w_{kj} X_{\ell j} + \sum_{s=1}^K u_{ks} A_{\ell-1,s}\right) \\ O_\ell &= \beta_0 + \sum_{k=1}^K \beta_k A_{\ell k} \end{aligned}$$

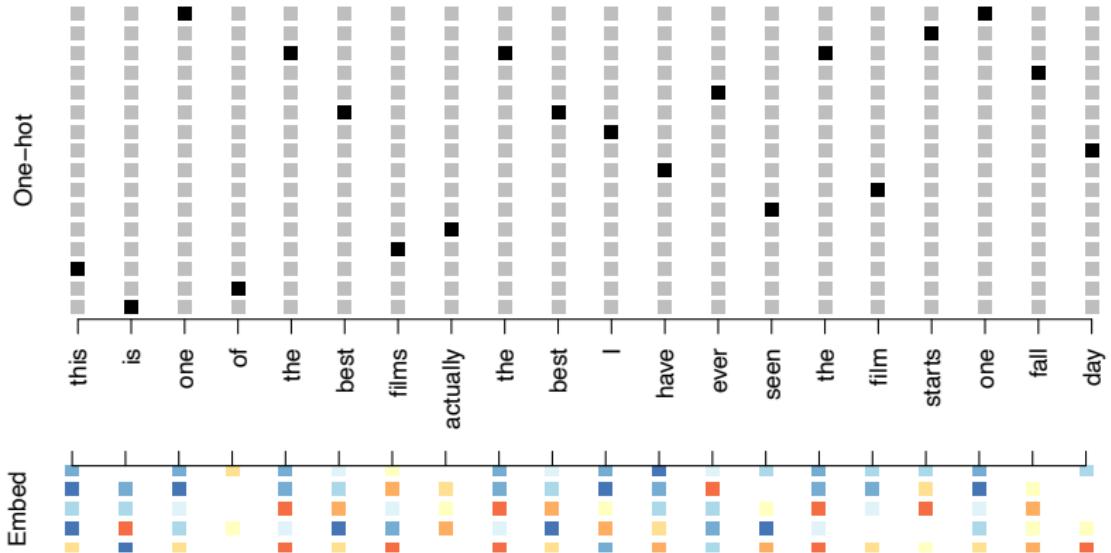
Often we are concerned only with the prediction  $O_L$  at the last unit. For squared error loss, and  $n$  sequence/response pairs, we would minimize

$$\sum_{i=1}^n (y_i - o_{iL})^2 = \sum_{i=1}^n \left( y_i - \left( \beta_0 + \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{iLj} + \sum_{s=1}^K u_{ks} a_{i,L-1,s}\right) \right) \right)^2.$$

## RNN and IMDB Reviews

- The document feature is a sequence of words  $\{\mathcal{W}_\ell\}_1^L$ . We typically truncate/pad the documents to the same number  $L$  of words (we use  $L = 500$ ).
- Each word  $\mathcal{W}_\ell$  is represented as a *one-hot encoded* binary vector  $X_\ell$  (dummy variable) of length  $10K$ , with all zeros and a single one in the position for that word in the dictionary.
- This results in an extremely sparse feature representation, and would not work well.
- Instead we use a lower-dimensional pretrained *word embedding* matrix  $\mathbf{E}$  ( $m \times 10K$ , next slide).
- This reduces the binary feature vector of length  $10K$  to a real feature vector of dimension  $m \ll 10K$  (e.g.  $m$  in the low hundreds.)

# Word Embedding



this is one of the best films actually the best I have ever seen the film starts one fall day ... .

Embeddings are pretrained on very large corpora of documents, using methods similar to principal components. **word2vec** and **GloVe** are popular.

## RNN on IMDB Reviews

- After a lot of work, the results are a disappointing 76% accuracy.

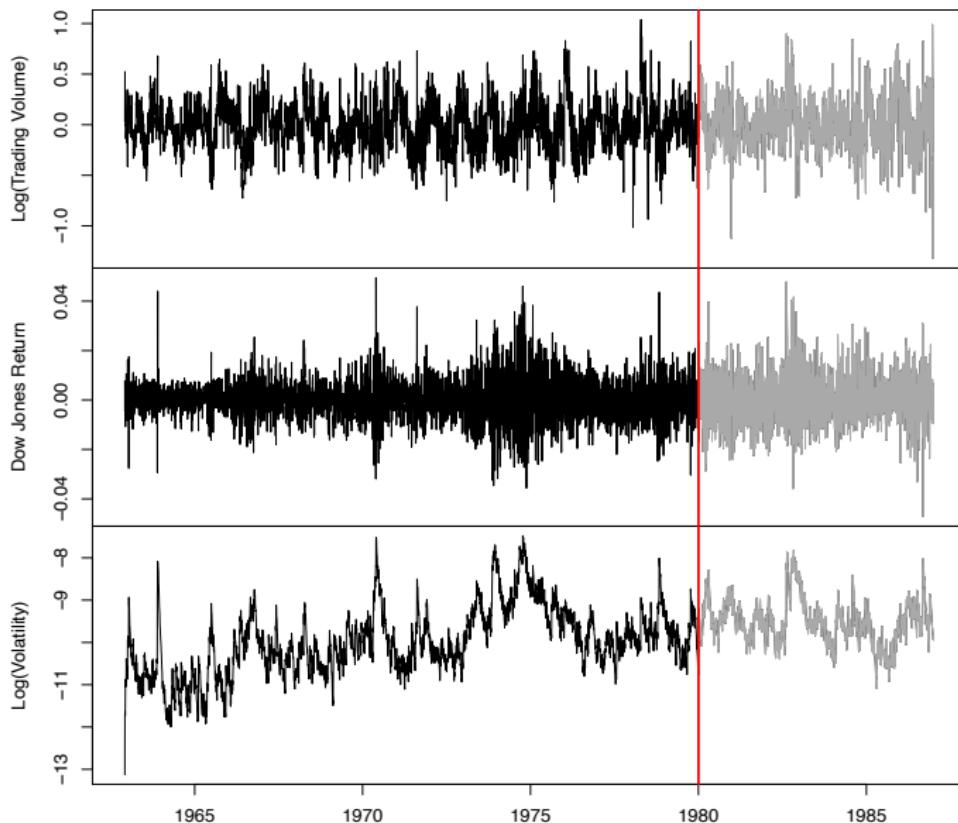
## RNN on IMDB Reviews

- After a lot of work, the results are a disappointing 76% accuracy.
- We then fit a more exotic RNN than the one displayed — a *LSTM* with *long and short term memory*. Here  $A_\ell$  receives input from  $A_{\ell-1}$  (short term memory) as well as from a version that reaches further back in time (long term memory). Now we get 87% accuracy, slightly less than the 88% achieved by `glmnet`.

## RNN on IMDB Reviews

- After a lot of work, the results are a disappointing 76% accuracy.
- We then fit a more exotic RNN than the one displayed — a *LSTM* with *long and short term memory*. Here  $A_\ell$  receives input from  $A_{\ell-1}$  (short term memory) as well as from a version that reaches further back in time (long term memory). Now we get 87% accuracy, slightly less than the 88% achieved by *glmnet*.
- These data have been used as a benchmark for new RNN architectures. The best reported result found at the time of writing (2020) was around 95%. We point to a *leaderboard* in Section 10.5.1.

# Time Series Forecasting



## New-York Stock Exchange Data

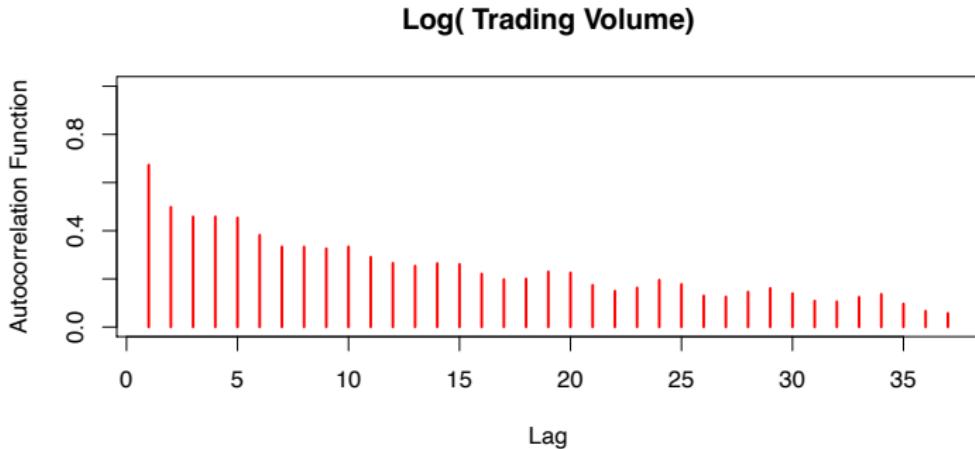
Shown in previous slide are three daily time series for the period December 3, 1962 to December 31, 1986 (6,051 trading days):

- **Log trading volume.** This is the fraction of all outstanding shares that are traded on that day, relative to a 100-day moving average of past turnover, on the log scale.
- **Dow Jones return.** This is the difference between the log of the Dow Jones Industrial Index on consecutive trading days.
- **Log volatility.** This is based on the absolute values of daily price movements.

Goal: predict **Log trading volume** tomorrow, given its observed values up to today, as well as those of **Dow Jones return** and **Log volatility**.

These data were assembled by LeBaron and Weigend (1998) *IEEE Transactions on Neural Networks*, 9(1): 213–220.

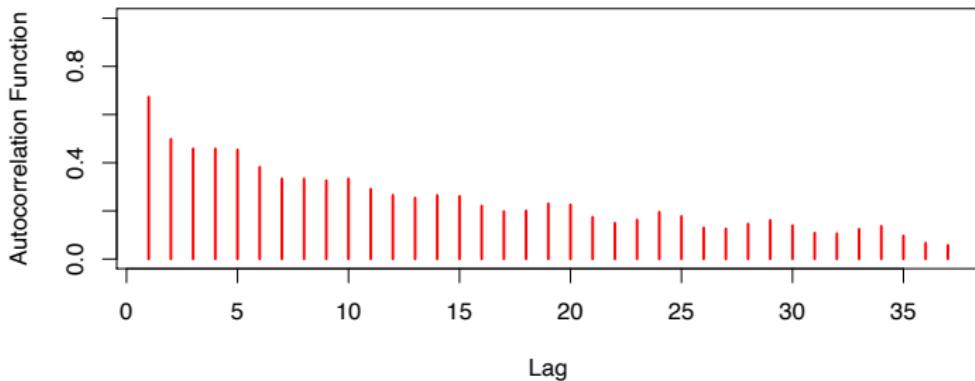
# Autocorrelation



- The *autocorrelation* at lag  $\ell$  is the correlation of all pairs  $(v_t, v_{t-\ell})$  that are  $\ell$  trading days apart.

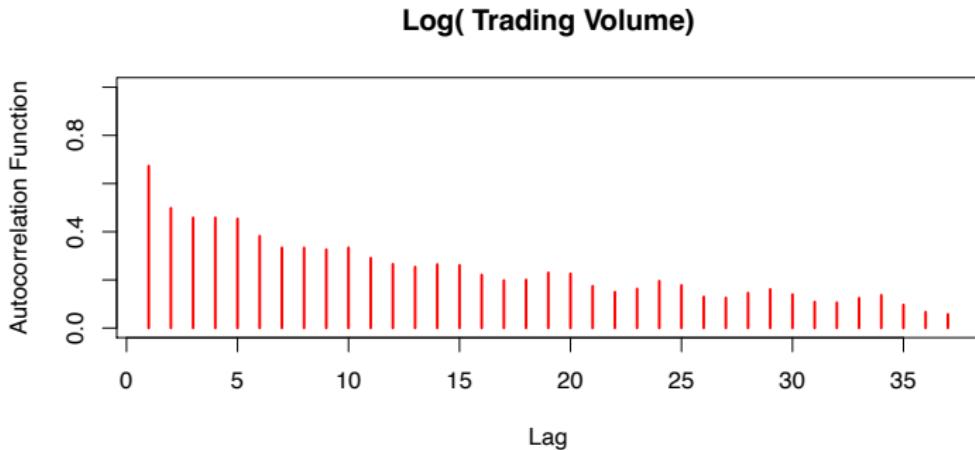
# Autocorrelation

Log( Trading Volume)



- The *autocorrelation* at lag  $\ell$  is the correlation of all pairs  $(v_t, v_{t-\ell})$  that are  $\ell$  trading days apart.
- These sizable correlations give us confidence that past values will be helpful in predicting the future.

# Autocorrelation



- The *autocorrelation* at lag  $\ell$  is the correlation of all pairs  $(v_t, v_{t-\ell})$  that are  $\ell$  trading days apart.
- These sizable correlations give us confidence that past values will be helpful in predicting the future.
- This is a curious prediction problem: the response  $v_t$  is also a feature  $v_{t-\ell}$ !

## RNN Forecaster

We only have one series of data! How do we set up for an RNN?

We extract many short mini-series of input sequences

$X = \{X_1, X_2, \dots, X_L\}$  with a predefined length  $L$  known as the *lag*:

$$X_1 = \begin{pmatrix} v_{t-L} \\ r_{t-L} \\ z_{t-L} \end{pmatrix}, \quad X_2 = \begin{pmatrix} v_{t-L+1} \\ r_{t-L+1} \\ z_{t-L+1} \end{pmatrix}, \dots, \quad X_L = \begin{pmatrix} v_{t-1} \\ r_{t-1} \\ z_{t-1} \end{pmatrix}, \quad \text{and } Y = v_t.$$

Since  $T = 6,051$ , with  $L = 5$  we can create 6,046 such  $(X, Y)$  pairs.

We use the first 4,281 as training data, and the following 1,770 as test data. We fit an RNN with 12 hidden units per lag step (i.e. per  $A_\ell$ .)

# RNN Results for NYSE Data

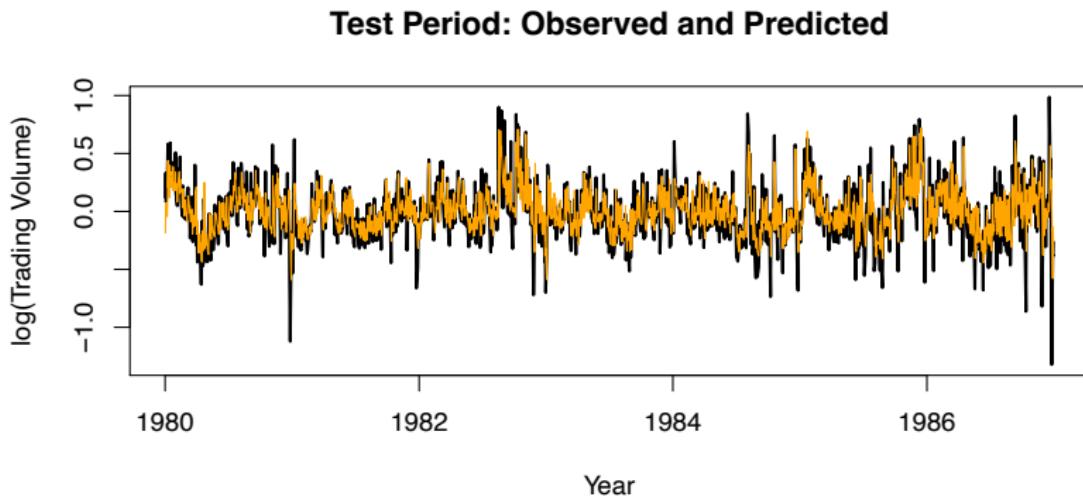


Figure shows predictions and truth for test period.

$R^2 = 0.42$  for RNN

$R^2 = 0.18$  for *straw man* — use yesterday's value of **Log trading volume** to predict that of today.

## Autoregression Forecaster

The RNN forecaster is similar in structure to a traditional *autoregression* procedure.

$$\mathbf{y} = \begin{bmatrix} v_{L+1} \\ v_{L+2} \\ v_{L+3} \\ \vdots \\ v_T \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & v_L & v_{L-1} & \cdots & v_1 \\ 1 & v_{L+1} & v_L & \cdots & v_2 \\ 1 & v_{L+2} & v_{L+1} & \cdots & v_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_{T-1} & v_{T-2} & \cdots & v_{T-L} \end{bmatrix}.$$

Fit an OLS regression of  $\mathbf{y}$  on  $\mathbf{M}$ , giving

$$\hat{v}_t = \hat{\beta}_0 + \hat{\beta}_1 v_{t-1} + \hat{\beta}_2 v_{t-2} + \cdots + \hat{\beta}_L v_{t-L}.$$

Known as an *order- $L$  autoregression* model or AR( $L$ ).

For the **NYSE** data we can include lagged versions of **DJ\_return** and **log\_volatility** in matrix  $\mathbf{M}$ , resulting in  $3L + 1$  columns.

## Autoregression Results for NYSE Data

$R^2 = 0.41$  for AR(5) model (16 parameters)

$R^2 = 0.42$  for RNN model (205 parameters)

$R^2 = 0.42$  for AR(5) model fit by neural network.

$R^2 = 0.46$  for all models if we include **day\_of\_week** of day being predicted.

## Summary of RNNs

- We have presented the simplest of RNNs. Many more complex variations exist.
- One variation treats the sequence as a one-dimensional image, and uses CNNs for fitting. For example, a sequence of words using an embedding representation can be viewed as an image, and the CNN convolves by sliding a convolutional filter along the sequence.
- Can have additional hidden layers, where each hidden layer is a sequence, and treats the previous hidden layer as an input sequence.
- Can have output also be a sequence, and input and output share the hidden units. So called **seq2seq** learning are used for language translation.

## When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.

## When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.
- RNNs have had big wins in speech modeling, language translation, and forecasting.

## When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.
- RNNs have had big wins in speech modeling, language translation, and forecasting.

Should we always use deep learning models?

## When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.
- RNNs have had big wins in speech modeling, language translation, and forecasting.

Should we always use deep learning models?

- Often the big successes occur when the *signal to noise ratio* is high — e.g. image recognition and language translation. Datasets are large, and overfitting is not a big problem.

## When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.
- RNNs have had big wins in speech modeling, language translation, and forecasting.

Should we always use deep learning models?

- Often the big successes occur when the *signal to noise ratio* is high — e.g. image recognition and language translation. Datasets are large, and overfitting is not a big problem.
- For noisier data, simpler models can often work better.
  - On the **NYSE** data, the AR(5) model is much simpler than a RNN, and performed as well.
  - On the **IMDB** review data, the linear model fit by **glmnet** did as well as the neural network, and better than the RNN.

## When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.
- RNNs have had big wins in speech modeling, language translation, and forecasting.

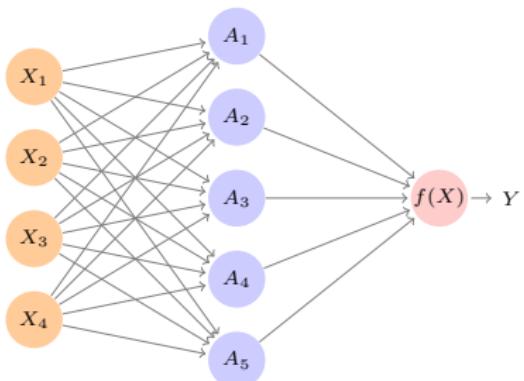
Should we always use deep learning models?

- Often the big successes occur when the *signal to noise ratio* is high — e.g. image recognition and language translation. Datasets are large, and overfitting is not a big problem.
- For noisier data, simpler models can often work better.
  - On the **NYSE** data, the AR(5) model is much simpler than a RNN, and performed as well.
  - On the **IMDB** review data, the linear model fit by **glmnet** did as well as the neural network, and better than the RNN.
- We endorse the *Occam's razor* principle — we prefer simpler models if they work as well. More interpretable!

# Fitting Neural Networks



Input Layer      Hidden Layer      Output Layer



$$\underset{\{w_k\}_1^K, \beta}{\text{minimize}} \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2,$$

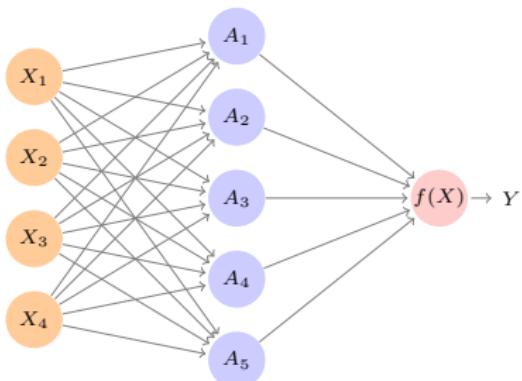
where

$$f(x_i) = \beta_0 + \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}\right).$$

# Fitting Neural Networks



Input Layer      Hidden Layer      Output Layer



$$\underset{\{w_k\}_1^K, \beta}{\text{minimize}} \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2,$$

where

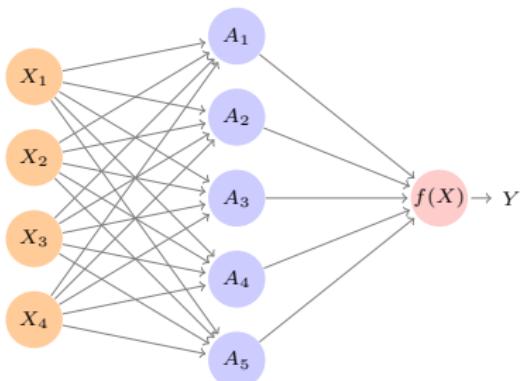
$$f(x_i) = \beta_0 + \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}\right).$$

This problem is difficult because the objective is *non-convex*.

# Fitting Neural Networks



Input Layer      Hidden Layer      Output Layer



$$\underset{\{w_k\}_1^K, \beta}{\text{minimize}} \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2,$$

where

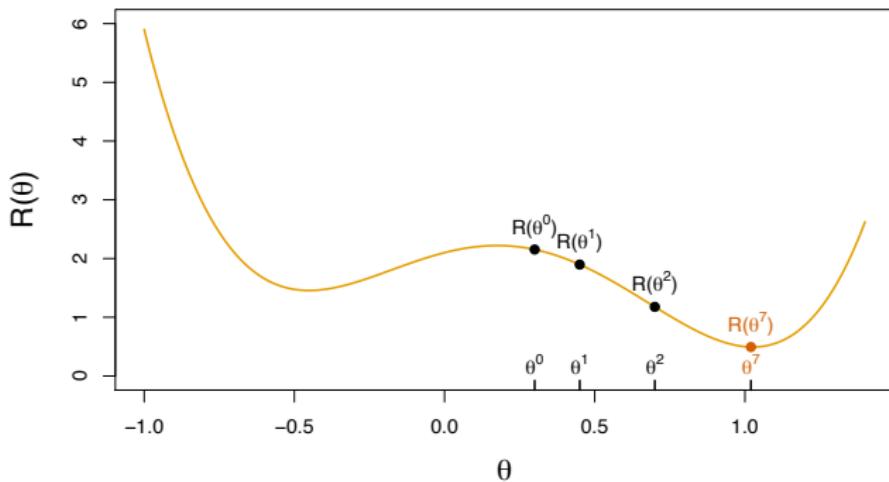
$$f(x_i) = \beta_0 + \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}\right).$$

This problem is difficult because the objective is *non-convex*.

Despite this, effective algorithms have evolved that can optimize complex neural network problems efficiently.

# Non Convex Functions and Gradient Descent

Let  $R(\theta) = \frac{1}{2} \sum_{i=1}^n (y_i - f_\theta(x_i))^2$  with  $\theta = (\{w_k\}_1^K, \beta)$ .



1. Start with a guess  $\theta^0$  for all the parameters in  $\theta$ , and set  $t = 0$ .
2. Iterate until the objective  $R(\theta)$  fails to decrease:
  - (a) Find a vector  $\delta$  that reflects a small change in  $\theta$ , such that  $\theta^{t+1} = \theta^t + \delta$  **reduces** the objective; i.e.  $R(\theta^{t+1}) < R(\theta^t)$ .
  - (b) Set  $t \leftarrow t + 1$ .

## Gradient Descent Continued

- In this simple example we reached the *global minimum*.
- If we had started a little to the left of  $\theta^0$  we would have gone in the other direction, and ended up in a *local minimum*.
- Although  $\theta$  is multi-dimensional, we have depicted the process as one-dimensional. It is much harder to identify whether one is in a local minimum in high dimensions.

How to find a direction  $\delta$  that points downhill? We compute the *gradient vector*

$$\nabla R(\theta^t) = \frac{\partial R(\theta)}{\partial \theta} \Big|_{\theta=\theta^t}$$

i.e. the vector of *partial derivatives* at the current guess  $\theta^t$ .

The gradient points uphill, so our update is  $\delta = -\rho \nabla R(\theta^t)$  or

$$\theta^{t+1} \leftarrow \theta^t - \rho \nabla R(\theta^t),$$

where  $\rho$  is the *learning rate* (typically small, e.g.  $\rho = 0.001$ .

## Gradients and Backpropagation

$R(\theta) = \sum_{i=1}^n R_i(\theta)$  is a sum, so gradient is sum of gradients.

$$R_i(\theta) = \frac{1}{2}(y_i - f_\theta(x_i))^2 = \frac{1}{2} \left( y_i - \beta_0 - \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}\right) \right)^2$$

For ease of notation, let  $z_{ik} = w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}$ .

Backpropagation uses the *chain rule for differentiation*:

$$\begin{aligned} \frac{\partial R_i(\theta)}{\partial \beta_k} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial \beta_k} \\ &= -(y_i - f_\theta(x_i)) \cdot g(z_{ik}). \end{aligned}$$

$$\begin{aligned} \frac{\partial R_i(\theta)}{\partial w_{kj}} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}} \\ &= -(y_i - f_\theta(x_i)) \cdot \beta_k \cdot g'(z_{ik}) \cdot x_{ij}. \end{aligned}$$

## Tricks of the Trade

- *Slow learning.* Gradient descent is slow, and a small learning rate  $\rho$  slows it even further. With *early stopping*, this is a form of regularization.

## Tricks of the Trade

- *Slow learning.* Gradient descent is slow, and a small learning rate  $\rho$  slows it even further. With *early stopping*, this is a form of regularization.
- *Stochastic gradient descent.* Rather than compute the gradient using *all* the data, use a small *minibatch* drawn at random at each step. E.g. for **MNIST** data, with  $n = 60K$ , we use minibatches of 128 observations.

## Tricks of the Trade

- *Slow learning.* Gradient descent is slow, and a small learning rate  $\rho$  slows it even further. With *early stopping*, this is a form of regularization.
- *Stochastic gradient descent.* Rather than compute the gradient using *all* the data, use a small *minibatch* drawn at random at each step. E.g. for **MNIST** data, with  $n = 60K$ , we use minibatches of 128 observations.
- An *epoch* is a count of iterations and amounts to the number of minibatch updates such that  $n$  samples in total have been processed; i.e.  $60K/128 \approx 469$  for **MNIST**.

## Tricks of the Trade

- *Slow learning.* Gradient descent is slow, and a small learning rate  $\rho$  slows it even further. With *early stopping*, this is a form of regularization.
- *Stochastic gradient descent.* Rather than compute the gradient using *all* the data, use a small *minibatch* drawn at random at each step. E.g. for **MNIST** data, with  $n = 60K$ , we use minibatches of 128 observations.
- An *epoch* is a count of iterations and amounts to the number of minibatch updates such that  $n$  samples in total have been processed; i.e.  $60K/128 \approx 469$  for **MNIST**.
- *Regularization.* Ridge and lasso regularization can be used to shrink the weights at each layer. Two other popular forms of regularization are *dropout* and *augmentation*, discussed next.

## Dropout Learning



- At each SGD update, randomly remove units with probability  $\phi$ , and scale up the weights of those retained by  $1/(1 - \phi)$  to compensate.

## Dropout Learning



- At each SGD update, randomly remove units with probability  $\phi$ , and scale up the weights of those retained by  $1/(1 - \phi)$  to compensate.
- In simple scenarios like linear regression, a version of this process can be shown to be equivalent to ridge regularization.

## Dropout Learning



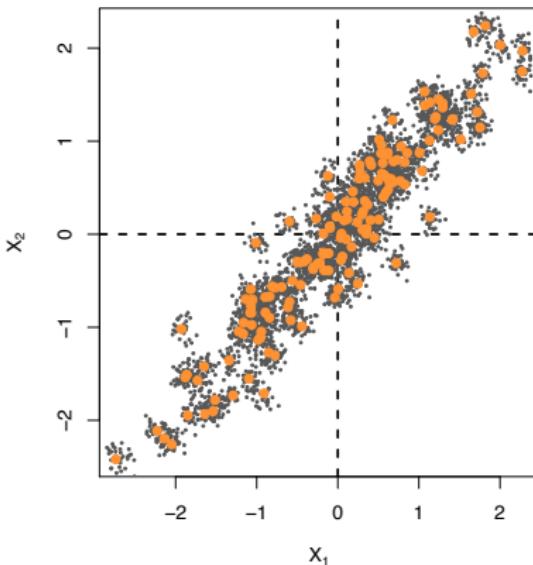
- At each SGD update, randomly remove units with probability  $\phi$ , and scale up the weights of those retained by  $1/(1 - \phi)$  to compensate.
- In simple scenarios like linear regression, a version of this process can be shown to be equivalent to ridge regularization.
- As in ridge, the other units *stand in* for those temporarily removed, and their weights are drawn closer together.

## Dropout Learning



- At each SGD update, randomly remove units with probability  $\phi$ , and scale up the weights of those retained by  $1/(1 - \phi)$  to compensate.
- In simple scenarios like linear regression, a version of this process can be shown to be equivalent to ridge regularization.
- As in ridge, the other units *stand in* for those temporarily removed, and their weights are drawn closer together.
- Similar to randomly omitting variables when growing trees in random forests (Chapter 8).

## Ridge and Data Augmentation



- Make many copies of each  $(x_i, y_i)$  and add a small amount of Gaussian noise to the  $x_i$  — a little cloud around each observation — but *leave the copies of  $y_i$  alone!*
- This makes the fit robust to small perturbations in  $x_i$ , and is equivalent to ridge regularization in an OLS setting.

## Data Augmentation on the Fly



- Data augmentation is especially effective with SGD, here demonstrated for a CNN and image classification.

## Data Augmentation on the Fly



- Data augmentation is especially effective with SGD, here demonstrated for a CNN and image classification.
- Natural transformations are made of each training image when it is sampled by SGD, thus ultimately making a cloud of images around each original training image.

## Data Augmentation on the Fly



- Data augmentation is especially effective with SGD, here demonstrated for a CNN and image classification.
- Natural transformations are made of each training image when it is sampled by SGD, thus ultimately making a cloud of images around each original training image.
- The label is left unchanged — in each case still **tiger**.

## Data Augmentation on the Fly



- Data augmentation is especially effective with SGD, here demonstrated for a CNN and image classification.
- Natural transformations are made of each training image when it is sampled by SGD, thus ultimately making a cloud of images around each original training image.
- The label is left unchanged — in each case still **tiger**.
- Improves performance of CNN and is similar to ridge.

## Double Descent

- With neural networks, it seems better to have too many hidden units than too few.

## Double Descent

- With neural networks, it seems better to have too many hidden units than too few.
- Likewise more hidden layers better than few.

## Double Descent

- With neural networks, it seems better to have too many hidden units than too few.
- Likewise more hidden layers better than few.
- Running stochastic gradient descent till zero training error often gives good out-of-sample error.

## Double Descent

- With neural networks, it seems better to have too many hidden units than too few.
- Likewise more hidden layers better than few.
- Running stochastic gradient descent till zero training error often gives good out-of-sample error.
- Increasing the number of units or layers and again training till zero error sometimes gives *even better* out-of-sample error.

## Double Descent

- With neural networks, it seems better to have too many hidden units than too few.
- Likewise more hidden layers better than few.
- Running stochastic gradient descent till zero training error often gives good out-of-sample error.
- Increasing the number of units or layers and again training till zero error sometimes gives *even better* out-of-sample error.

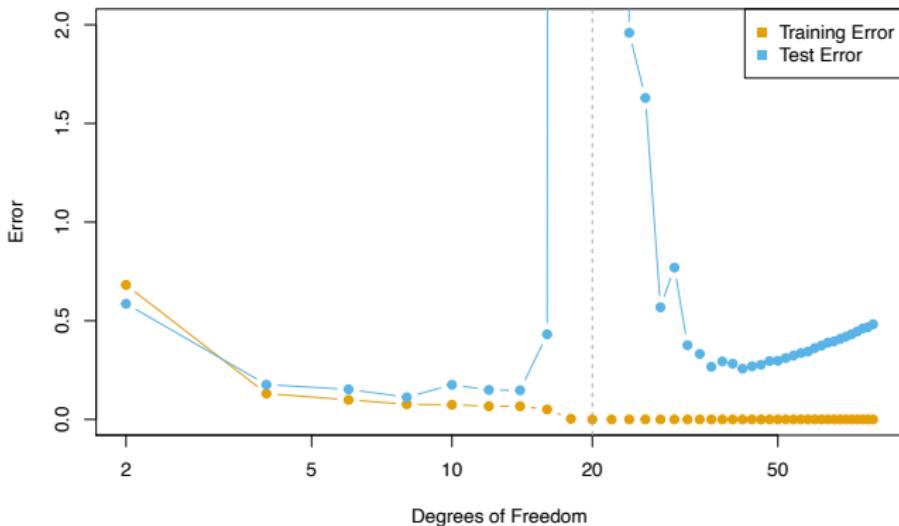
What happened to overfitting and the usual bias-variance trade-off?

Belkin, Hsu, Ma and Mandal (arXiv 2018) *Reconciling Modern Machine Learning and the Bias-Variance Trade-off*.

## Simulation

- $y = \sin(x) + \varepsilon$  with  $x \sim U[-5, 5]$  and  $\varepsilon$  Gaussian with S.D. = 0.3.
- Training set  $n = 20$ , test set very large (10K).
- We fit a natural spline to the data (Section 7.4) with  $d$  degrees of freedom — i.e. a linear regression onto  $d$  basis functions:  $\hat{y}_i = \hat{\beta}_1 N_1(x_i) + \hat{\beta}_2 N_2(x_i) + \cdots + \hat{\beta}_d N_d(x_i)$ .
- When  $d = 20$  we fit the training data exactly, and get all residuals equal to zero.
- When  $d > 20$ , we still fit the data exactly, but the solution is not unique. Among the zero-residual solutions, we pick the one with *minimum norm* — i.e. the zero-residual solution with smallest  $\sum_{j=1}^d \hat{\beta}_j^2$ .

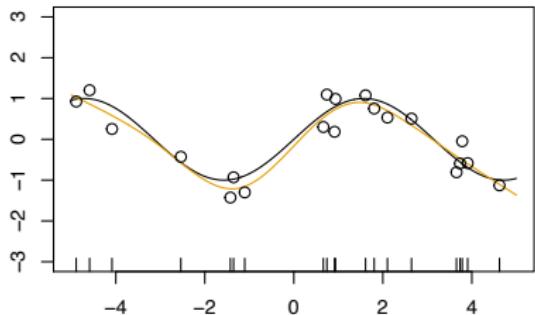
# The Double-Descent Error Curve



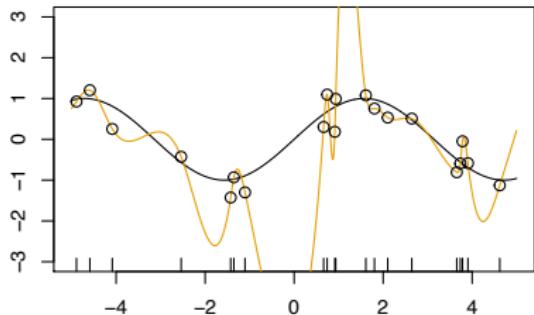
- When  $d \leq 20$ , model is OLS, and we see usual bias-variance trade-off
- When  $d > 20$ , we revert to minimum-norm. As  $d$  increases above 20,  $\sum_{j=1}^d \hat{\beta}_j^2$  *decreases* since it is easier to achieve zero error, and hence less wiggly solutions.

# Less Wiggly Solutions

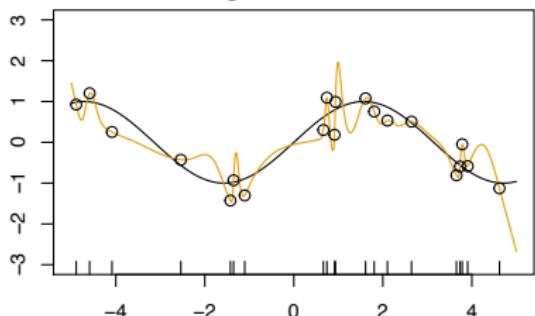
8 Degrees of Freedom



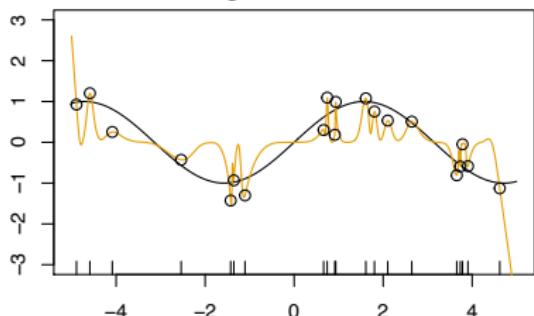
20 Degrees of Freedom



42 Degrees of Freedom



80 Degrees of Freedom



To achieve a zero-residual solution with  $d = 20$  is a real stretch!  
Easier for larger  $d$ .

## Some Facts

- In a wide linear model ( $p \gg n$ ) fit by least squares, SGD with a small step size leads to a *minimum norm* zero-residual solution.

## Some Facts

- In a wide linear model ( $p \gg n$ ) fit by least squares, SGD with a small step size leads to a *minimum norm* zero-residual solution.
- Stochastic gradient *flow* — i.e. the entire path of SGD solutions — is somewhat similar to ridge path.

## Some Facts

- In a wide linear model ( $p \gg n$ ) fit by least squares, SGD with a small step size leads to a *minimum norm* zero-residual solution.
- Stochastic gradient *flow* — i.e. the entire path of SGD solutions — is somewhat similar to ridge path.
- By analogy, deep and wide neural networks fit by SGD down to zero training error often give good solutions that generalize well.

## Some Facts

- In a wide linear model ( $p \gg n$ ) fit by least squares, SGD with a small step size leads to a *minimum norm* zero-residual solution.
- Stochastic gradient *flow* — i.e. the entire path of SGD solutions — is somewhat similar to ridge path.
- By analogy, deep and wide neural networks fit by SGD down to zero training error often give good solutions that generalize well.
- In particular cases with *high signal-to-noise ratio* — e.g. image recognition — are less prone to overfitting; the zero-error solution is mostly signal!

## Software

- Wonderful software available for neural networks and deep learning. **Tensorflow** from Google and **PyTorch** from Facebook. Both are **Python** packages.
- In the Chapter 10 lab we demonstrate **tensorflow** and **keras** packages in **R**, which interface to Python. See textbook and online resources for **Rmarkdown** and **Jupyter** notebooks for these and all labs for the second edition of ISLR book.
- The **torch** package in R is available as well, and implements the **PyTorch** dialect. The Chapter 10 lab will be available in this dialect as well; watch the resources page at [www.statlearning.com](http://www.statlearning.com).

# Survival Analysis

# Survival Analysis

- Survival analysis concerns a special kind of outcome variable: the *time until an event occurs*.

## Survival Analysis

- Survival analysis concerns a special kind of outcome variable: the *time until an event occurs*.
- For example, suppose that we have conducted a five-year medical study, in which patients have been treated for cancer.

## Survival Analysis

- Survival analysis concerns a special kind of outcome variable: the *time until an event occurs*.
- For example, suppose that we have conducted a five-year medical study, in which patients have been treated for cancer.
- We would like to fit a model to predict patient survival time, using features such as baseline health measurements or type of treatment.

## Survival Analysis

- Survival analysis concerns a special kind of outcome variable: the *time until an event occurs*.
- For example, suppose that we have conducted a five-year medical study, in which patients have been treated for cancer.
- We would like to fit a model to predict patient survival time, using features such as baseline health measurements or type of treatment.
- Sounds like a *regression problem*. But there is an important complication: some of the patients have survived until the end of the study. Such a patient's survival time is said to be *censored*.

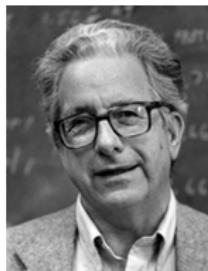
## Survival Analysis

- Survival analysis concerns a special kind of outcome variable: the *time until an event occurs*.
- For example, suppose that we have conducted a five-year medical study, in which patients have been treated for cancer.
- We would like to fit a model to predict patient survival time, using features such as baseline health measurements or type of treatment.
- Sounds like a *regression problem*. But there is an important complication: some of the patients have survived until the end of the study. Such a patient's survival time is said to be *censored*.
- We do not want to discard this subset of surviving patients, since the fact that they survived at least five years amounts to valuable information.

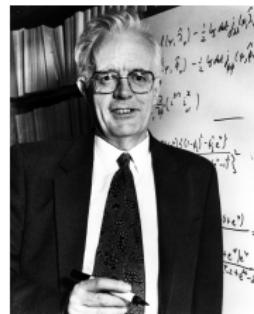
## Some of the big names in this field



**Edward Kaplan**



**Paul Meier**



**David Cox**



**Nathan Mantel**

(log rank test)



**William Haenszel**



**Terry Therneau**

(author of *Survival package in R*)

## Non-medical Examples

## Non-medical Examples

- The applications of survival analysis extend far beyond medicine. For example, consider a company that wishes to model *churn*, the event when customers cancel subscription to a service.

## Non-medical Examples

- The applications of survival analysis extend far beyond medicine. For example, consider a company that wishes to model *churn*, the event when customers cancel subscription to a service.
- The company might collect data on customers over some time period, in order to predict each customer's time to cancellation.

## Non-medical Examples

- The applications of survival analysis extend far beyond medicine. For example, consider a company that wishes to model *churn*, the event when customers cancel subscription to a service.
- The company might collect data on customers over some time period, in order to predict each customer's time to cancellation.
- However, presumably not all customers will have cancelled their subscription by the end of this time period; for such customers, the time to cancellation is censored.

## Non-medical Examples

- The applications of survival analysis extend far beyond medicine. For example, consider a company that wishes to model *churn*, the event when customers cancel subscription to a service.
- The company might collect data on customers over some time period, in order to predict each customer's time to cancellation.
- However, presumably not all customers will have cancelled their subscription by the end of this time period; for such customers, the time to cancellation is censored.
- Survival analysis is a very well-studied topic within statistics. However, it has received relatively little attention in the machine learning community.

# Survival and Censoring Times

## Survival and Censoring Times

- For each individual, we suppose that there is a true *failure* or *event* time  $T$ , as well as a true censoring time  $C$ .

## Survival and Censoring Times

- For each individual, we suppose that there is a true *failure* or *event* time  $T$ , as well as a true censoring time  $C$ .
- The survival time represents the time at which the event of interest occurs (such as death).

## Survival and Censoring Times

- For each individual, we suppose that there is a true *failure* or *event* time  $T$ , as well as a true censoring time  $C$ .
- The survival time represents the time at which the event of interest occurs (such as death).
- By contrast, the *censoring* is the time at which censoring occurs: for example, the time at which the patient drops out of the study or the study ends.

## Survival and Censoring Times — Continued

## Survival and Censoring Times — Continued

- We observe either the survival time  $T$  or else the censoring time  $C$ . Specifically, we observe the random variable

$$Y = \min(T, C).$$

## Survival and Censoring Times — Continued

- We observe either the survival time  $T$  or else the censoring time  $C$ . Specifically, we observe the random variable

$$Y = \min(T, C).$$

- If the event occurs before censoring (i.e.  $T < C$ ) then we observe the true survival time  $T$ ; if censoring occurs before the event ( $T > C$ ) then we observe the censoring time. We also observe a status indicator,

$$\delta = \begin{cases} 1 & \text{if } T \leq C \\ 0 & \text{if } T > C. \end{cases}$$

## Survival and Censoring Times — Continued

- We observe either the survival time  $T$  or else the censoring time  $C$ . Specifically, we observe the random variable

$$Y = \min(T, C).$$

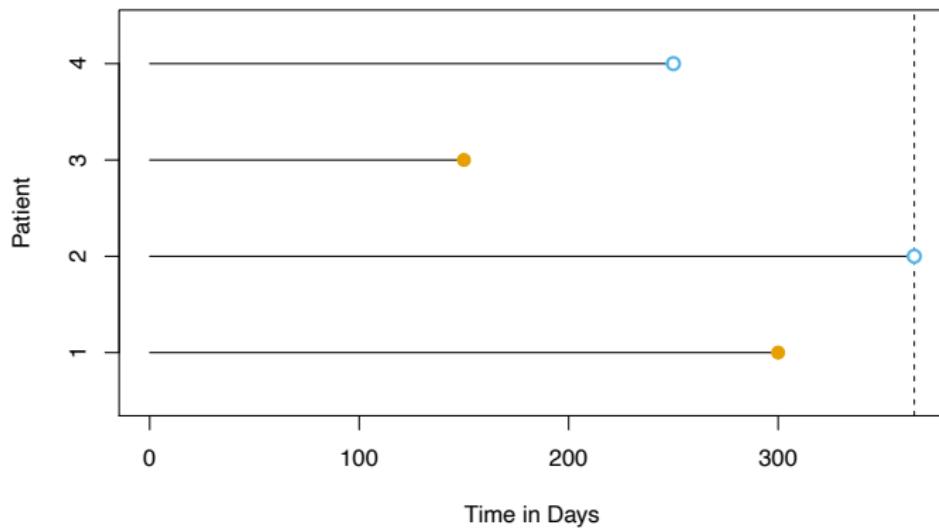
- If the event occurs before censoring (i.e.  $T < C$ ) then we observe the true survival time  $T$ ; if censoring occurs before the event ( $T > C$ ) then we observe the censoring time. We also observe a status indicator,

$$\delta = \begin{cases} 1 & \text{if } T \leq C \\ 0 & \text{if } T > C. \end{cases}$$

- Finally, in our dataset we observe  $n$  pairs  $(Y, \delta)$ , which we denote as  $(y_1, \delta_1), \dots, (y_n, \delta_n)$ .

## Illustration

Here is an illustration of censored survival data. For patients 1 and 3, the event was observed. Patient 2 was alive when the study ended. Patient 4 dropped out of the study.



## A Closer Look at Censoring

## A Closer Look at Censoring

- Suppose that a number of patients drop out of a cancer study early because they are very sick.

## A Closer Look at Censoring

- Suppose that a number of patients drop out of a cancer study early because they are very sick.
- An analysis that does not take into consideration the reason why the patients dropped out will likely overestimate the true average survival time.

## A Closer Look at Censoring

- Suppose that a number of patients drop out of a cancer study early because they are very sick.
- An analysis that does not take into consideration the reason why the patients dropped out will likely overestimate the true average survival time.
- Similarly, suppose that males who are very sick are more likely to drop out of the study than females who are very sick. Then a comparison of male and female survival times may wrongly suggest that males survive longer than females.

## A Closer Look at Censoring

- Suppose that a number of patients drop out of a cancer study early because they are very sick.
- An analysis that does not take into consideration the reason why the patients dropped out will likely overestimate the true average survival time.
- Similarly, suppose that males who are very sick are more likely to drop out of the study than females who are very sick. Then a comparison of male and female survival times may wrongly suggest that males survive longer than females.
- In general, we need to assume that, conditional on the features, the event time  $T$  is *independent* of the censoring time  $C$ . The two examples above violate the assumption of independent censoring.

# The Survival Curve

## The Survival Curve

- The survival function (or curve) is defined as

$$S(t) = \Pr(T > t).$$

## The Survival Curve

- The survival function (or curve) is defined as

$$S(t) = \Pr(T > t).$$

- This decreasing function quantifies the probability of surviving past time  $t$ .

## The Survival Curve

- The survival function (or curve) is defined as

$$S(t) = \Pr(T > t).$$

- This decreasing function quantifies the probability of surviving past time  $t$ .
- For example, suppose that a company is interested in modeling customer churn. Let  $T$  represent the time that a customer cancels a subscription to the company's service.

## The Survival Curve

- The survival function (or curve) is defined as

$$S(t) = \Pr(T > t).$$

- This decreasing function quantifies the probability of surviving past time  $t$ .
- For example, suppose that a company is interested in modeling customer churn. Let  $T$  represent the time that a customer cancels a subscription to the company's service.
- Then  $S(t)$  represents the probability that a customer cancels later than time  $t$ . The larger the value of  $S(t)$ , the less likely that the customer will cancel before time  $t$ .

## Estimating the Survival Curve

## Estimating the Survival Curve

- Consider the **BrainCancer** dataset, which contains the survival times for patients with primary brain tumors undergoing treatment with stereotactic radiation methods.

## Estimating the Survival Curve

- Consider the **BrainCancer** dataset, which contains the survival times for patients with primary brain tumors undergoing treatment with stereotactic radiation methods.
- The predictors are **gtv** (gross tumor volume, in cubic centimeters); **sex** (male or female); **diagnosis** (meningioma, LG glioma, HG glioma, or other); **loc** (the tumor location: either infratentorial or supratentorial); **ki** (Karnofsky index); and **stereo** (stereotactic method).

## Estimating the Survival Curve

- Consider the **BrainCancer** dataset, which contains the survival times for patients with primary brain tumors undergoing treatment with stereotactic radiation methods.
- The predictors are **gtv** (gross tumor volume, in cubic centimeters); **sex** (male or female); **diagnosis** (meningioma, LG glioma, HG glioma, or other); **loc** (the tumor location: either infratentorial or supratentorial); **ki** (Karnofsky index); and **stereo** (stereotactic method).
- Only 53 of the 88 patients were still alive at the end of the study.

## Estimating the Survival Curve — Continued

## Estimating the Survival Curve — Continued

- Suppose we'd like to estimate  $S(20) = \Pr(T > 20)$ , the probability that a patient survives for at least 20 months,

## Estimating the Survival Curve — Continued

- Suppose we'd like to estimate  $S(20) = \Pr(T > 20)$ , the probability that a patient survives for at least 20 months,
- It is tempting to simply compute the proportion of patients who are known to have survived past 20 months, that is, the proportion of patients for whom  $Y > 20$ .

## Estimating the Survival Curve — Continued

- Suppose we'd like to estimate  $S(20) = \Pr(T > 20)$ , the probability that a patient survives for at least 20 months,
- It is tempting to simply compute the proportion of patients who are known to have survived past 20 months, that is, the proportion of patients for whom  $Y > 20$ .
- This turns out to be 48/88, or approximately 55%.

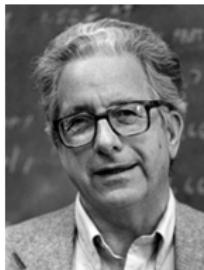
## Estimating the Survival Curve — Continued

- Suppose we'd like to estimate  $S(20) = \Pr(T > 20)$ , the probability that a patient survives for at least 20 months,
- It is tempting to simply compute the proportion of patients who are known to have survived past 20 months, that is, the proportion of patients for whom  $Y > 20$ .
- This turns out to be 48/88, or approximately 55%.
- However, this does not seem quite right: 17 of the 40 patients who did not survive to 20 months were actually censored, and this analysis implicitly assumes they died before 20 months. Hence it is probably an underestimate.

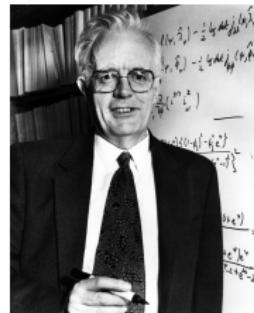
# Those big names again



Edward Kaplan



Paul Meier



David Cox



Nathan Mantel



William Haenszel

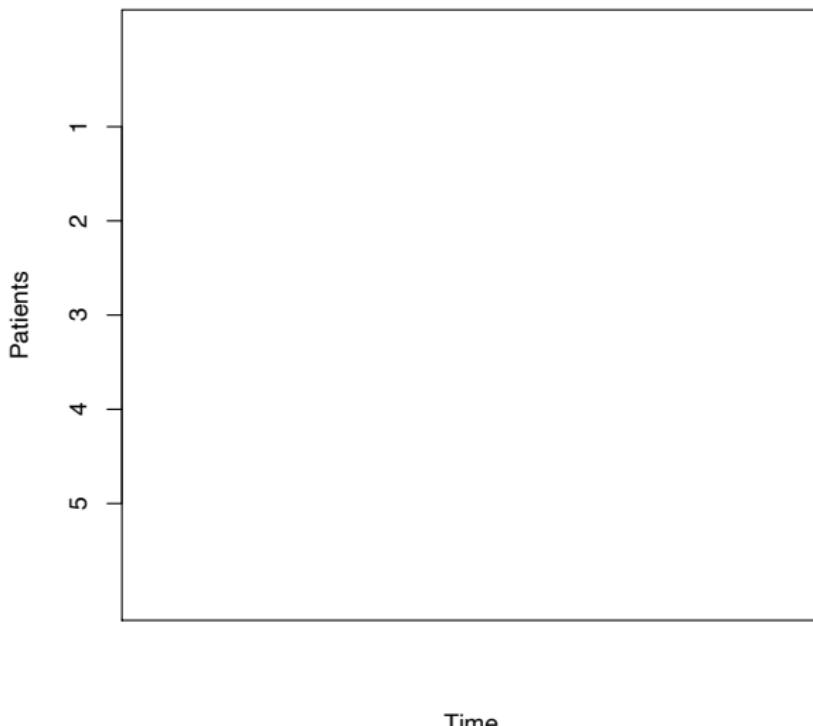


Terry Therneau

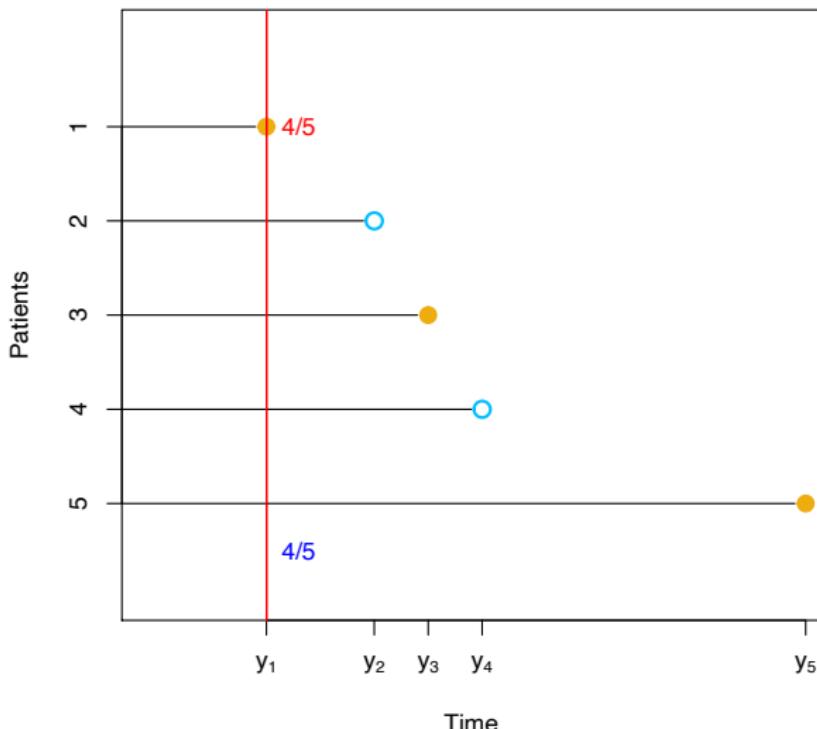
(log rank test)

(author of *Survival package in R*)

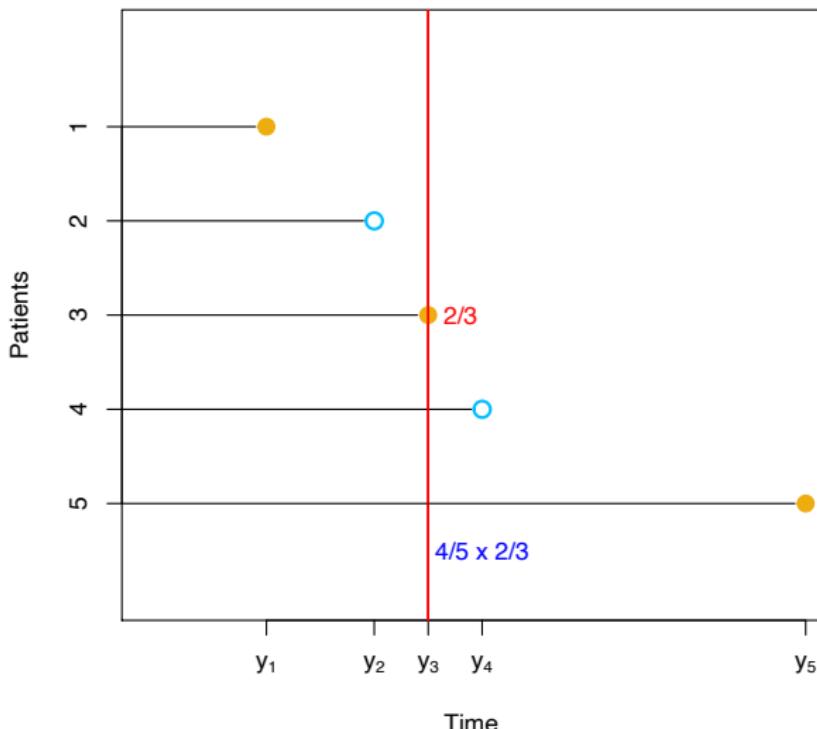
## The Kaplan-Meier Estimate: Example



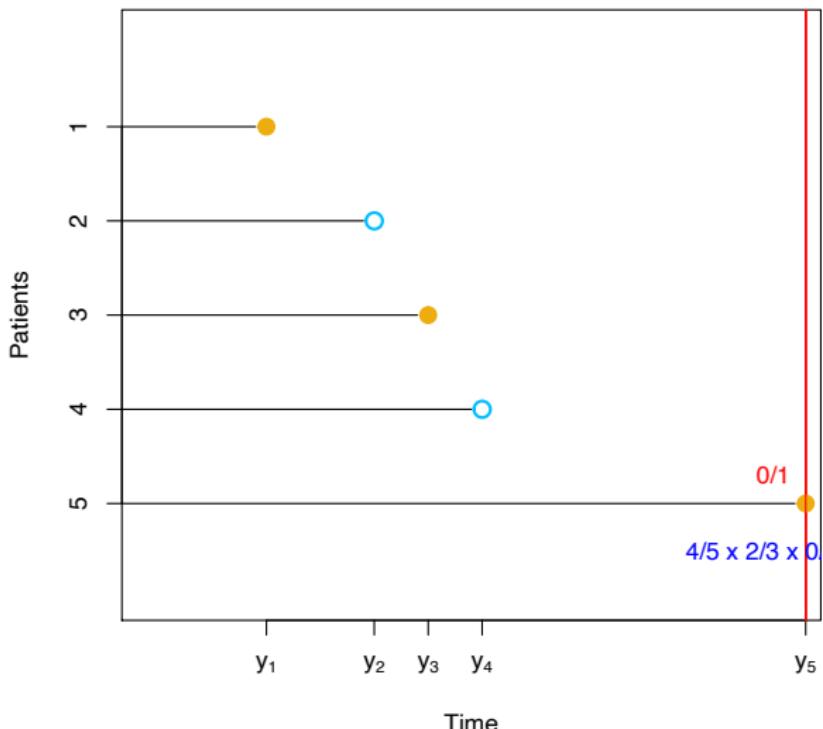
# First Failure



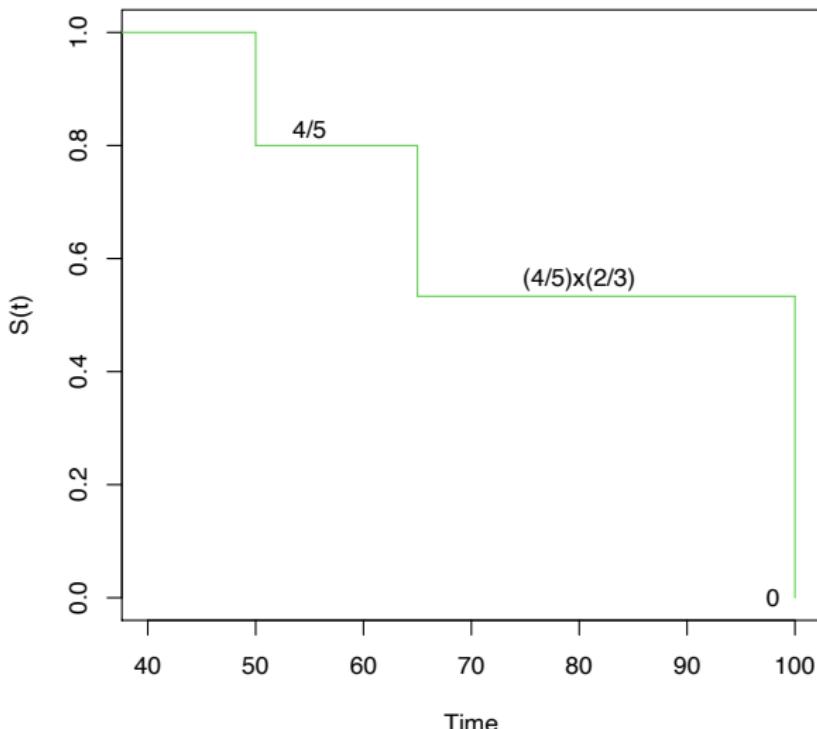
## Second Failure



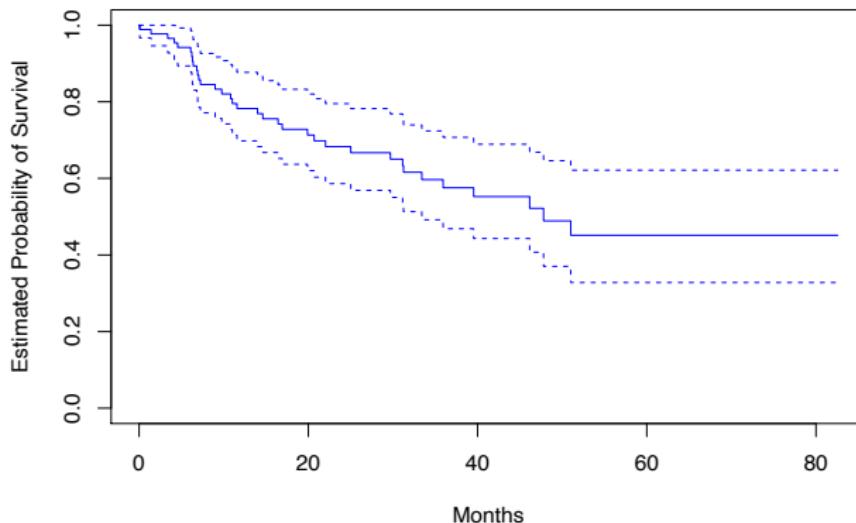
## Third Failure



## Resulting KM Survival Curve



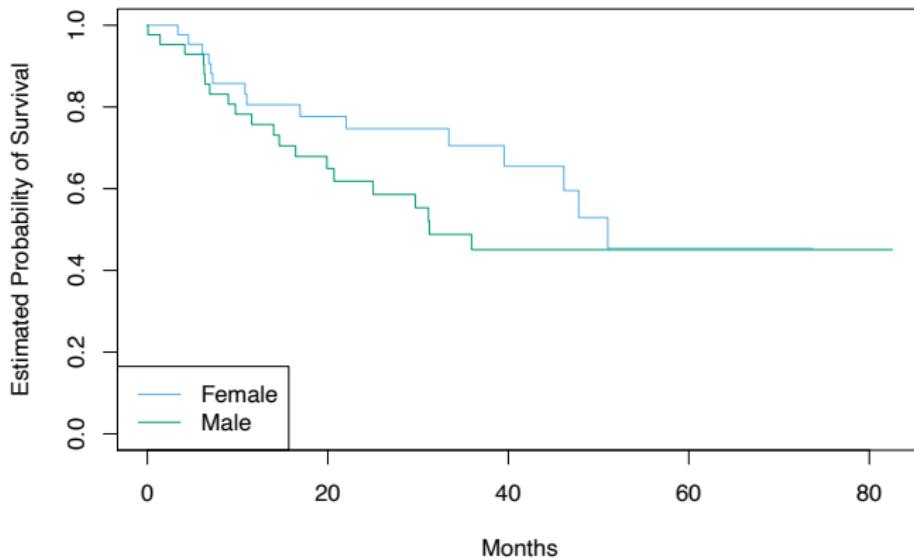
## Kaplan-Meier Survival Curve for the BrainCancer Data



Each point in the solid step-like curve shows the estimated probability of surviving past the time indicated on the horizontal axis.

The estimated probability of survival past 20 months is 71%, which is quite a bit higher than the naive estimate of 55% presented earlier.

# The Log-Rank Test

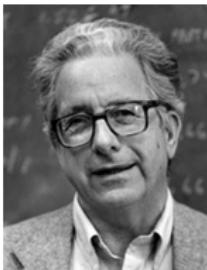


We wish to compare the survival of males to that of females.  
Shown are the Kaplan-Meier survival curves for the two groups.

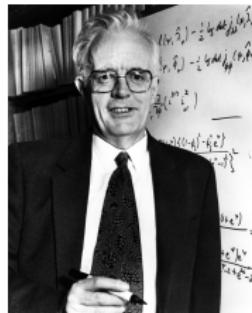
# Those big names again



Edward Kaplan



Paul Meier



David Cox



Nathan Mantel



William Haenszel

(log rank test)



Terry Therneau

(author of *Survival package in R*)

## The Log-Rank Test — Continued

## The Log-Rank Test — Continued

- Females seem to fare a little better up to about 50 months, but then the two curves both level off to about 50%. How can we carry out a formal test of equality of the two survival curves?

## The Log-Rank Test — Continued

- Females seem to fare a little better up to about 50 months, but then the two curves both level off to about 50%. How can we carry out a formal test of equality of the two survival curves?
- At first glance, a two-sample  $t$ -test seems like an obvious choice: but the presence of censoring again creates a complication.

## The Log-Rank Test — Continued

- Females seem to fare a little better up to about 50 months, but then the two curves both level off to about 50%. How can we carry out a formal test of equality of the two survival curves?
- At first glance, a two-sample  $t$ -test seems like an obvious choice: but the presence of censoring again creates a complication.
- To overcome this challenge, we will conduct a log-rank test.

## The Log-Rank Test — Continued

## The Log-Rank Test — Continued

- Recall that  $d_1 < d_2 < \dots < d_K$  are the unique death times among the non-censored patients,  $r_k$  is the number of patients at risk at time  $d_k$ , and  $q_k$  is the number of patients who died at time  $d_k$ .

## The Log-Rank Test — Continued

- Recall that  $d_1 < d_2 < \dots < d_K$  are the unique death times among the non-censored patients,  $r_k$  is the number of patients at risk at time  $d_k$ , and  $q_k$  is the number of patients who died at time  $d_k$ .
- We further define  $r_{1k}$  and  $r_{2k}$  to be the number of patients in groups 1 and 2, respectively, who are at risk at time  $d_k$ .

## The Log-Rank Test — Continued

- Recall that  $d_1 < d_2 < \dots < d_K$  are the unique death times among the non-censored patients,  $r_k$  is the number of patients at risk at time  $d_k$ , and  $q_k$  is the number of patients who died at time  $d_k$ .
- We further define  $r_{1k}$  and  $r_{2k}$  to be the number of patients in groups 1 and 2, respectively, who are at risk at time  $d_k$ .
- Similarly, we define  $q_{1k}$  and  $q_{2k}$  to be the number of patients in groups 1 and 2, respectively, who died at time  $d_k$ . Note that  $r_{1k} + r_{2k} = r_k$  and  $q_{1k} + q_{2k} = q_k$ .

## Details of the Test Statistic

	Group 1	Group 2	Total
Died	$q_{1k}$	$q_{2k}$	$q_k$
Survived	$r_{1k} - q_{1k}$	$r_{2k} - q_{2k}$	$r_k - q_k$
Total	$r_{1k}$	$r_{2k}$	$r_k$

At each death time  $d_k$ , we construct a  $2 \times 2$  table of counts of the form shown above.

Note that if the death times are unique (i.e. no two individuals die at the same time), then one of  $q_{1k}$  and  $q_{2k}$  equals one, and the other equals zero.

## Log Rank Test: the Main Idea

## Log Rank Test: the Main Idea

- To test  $H_0 : E(X) = 0$  for some random variable  $X$ , one approach is to construct a test statistic of the form

$$W = \frac{X - E(X)}{\sqrt{\text{Var}(X)}},$$

where  $E(X)$  and  $\text{Var}(X)$  are the expectation and variance, respectively, of  $X$  under  $H_0$ .

## Log Rank Test: the Main Idea

- To test  $H_0 : E(X) = 0$  for some random variable  $X$ , one approach is to construct a test statistic of the form

$$W = \frac{X - E(X)}{\sqrt{\text{Var}(X)}},$$

where  $E(X)$  and  $\text{Var}(X)$  are the expectation and variance, respectively, of  $X$  under  $H_0$ .

- In order to construct the log-rank test statistic, we compute a quantity that takes exactly the form above, with  $X = \sum_{k=1}^K q_{1k}$ , where  $q_{1k}$  is given in the top left of the table above.

## The Final Result

The resulting formula for the log-rank test statistic is

$$W = \frac{\sum_{k=1}^K (q_{1k} - \text{E}(q_{1k}))}{\sqrt{\sum_{k=1}^K \text{Var}(q_{1k})}} = \frac{\sum_{k=1}^K \left( q_{1k} - \frac{q_k}{r_k} r_{1k} \right)}{\sqrt{\sum_{k=1}^K \frac{q_k(r_{1k}/r_k)(1-r_{1k}/r_k)(r_k-q_k)}{r_k-1}}}.$$

When the sample size is large, the log-rank test statistic  $W$  has approximately a standard normal distribution.

This can be used to compute a  $p$ -value for the null hypothesis that there is no difference between the survival curves in the two groups.

## Application to the Brain Cancer Dataset

## Application to the Brain Cancer Dataset

- Comparing the survival times of females and males on the **BrainCancer** data gives a log-rank test statistic of  $W = 1.2$ , which corresponds to a two-sided  $p$ -value of 0.2.

## Application to the Brain Cancer Dataset

- Comparing the survival times of females and males on the **BrainCancer** data gives a log-rank test statistic of  $W = 1.2$ , which corresponds to a two-sided  $p$ -value of 0.2.
- Thus, we cannot reject the null hypothesis of no difference in survival curves between females and males.

## Application to the Brain Cancer Dataset

- Comparing the survival times of females and males on the **BrainCancer** data gives a log-rank test statistic of  $W = 1.2$ , which corresponds to a two-sided  $p$ -value of 0.2.
- Thus, we cannot reject the null hypothesis of no difference in survival curves between females and males.
- The log-rank test is closely related to Cox's proportional hazards model, which we discuss next.

# Regression Models with a Survival Response

## Regression Models with a Survival Response

- We now consider the task of fitting a regression model to survival data.

## Regression Models with a Survival Response

- We now consider the task of fitting a regression model to survival data.
- We wish to predict the true survival time  $T$ . Since the observed quantity  $Y = \min(T, C)$  is positive and may have a long right tail, we might be tempted to fit a linear regression of  $\log(Y)$  on  $X$ . But *censoring again creates a problem*.

## Regression Models with a Survival Response

- We now consider the task of fitting a regression model to survival data.
- We wish to predict the true survival time  $T$ . Since the observed quantity  $Y = \min(T, C)$  is positive and may have a long right tail, we might be tempted to fit a linear regression of  $\log(Y)$  on  $X$ . But *censoring again creates a problem*.
- To overcome this difficulty, we instead make use of a sequential construction, similar to the idea used for the Kaplan-Meier survival curve.

## The Hazard Function

The *hazard function* or *hazard rate* — also known as the *force of mortality* — is formally defined as

$$h(t) = \lim_{\Delta t \rightarrow 0} \frac{\Pr(t < T \leq t + \Delta t | T > t)}{\Delta t},$$

where  $T$  is the (true) survival time.

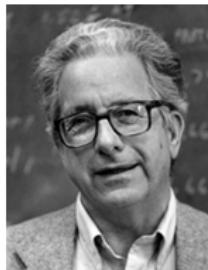
It is the death rate in the instant after time  $t$ , given survival up to that time.

The hazard function is the basis for the *Proportional Hazards Model*, discussed next.

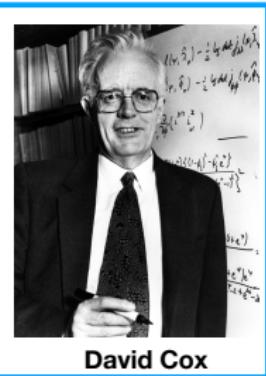
# Bringing in the covariates: those big names again



Edward Kaplan



Paul Meier



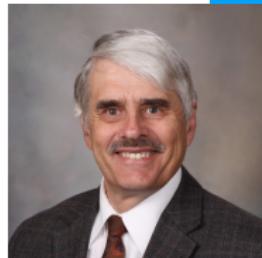
David Cox



Nathan Mantel



William Haenszel



Terry Therneau

(log rank test)

(author of *Survival package in R*)

# The Proportional Hazards Model

## The Proportional Hazards Model

- The proportional hazards assumption states that

$$h(t|x_i) = h_0(t) \exp\left(\sum_{j=1}^p x_{ij}\beta_j\right),$$

where  $h_0(t) \geq 0$  is an unspecified function, known as the *baseline hazard*. It is the hazard function for an individual with features  $x_{i1} = \dots = x_{ip} = 0$ .

## The Proportional Hazards Model

- The proportional hazards assumption states that

$$h(t|x_i) = h_0(t) \exp\left(\sum_{j=1}^p x_{ij}\beta_j\right),$$

where  $h_0(t) \geq 0$  is an unspecified function, known as the *baseline hazard*. It is the hazard function for an individual with features  $x_{i1} = \dots = x_{ip} = 0$ .

- The name *proportional hazards* arises from the fact that the hazard function for an individual with feature vector  $x_i$  is some unknown function  $h_0(t)$  times the factor  $\exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)$ . The quantity  $\exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)$  is called the *relative risk* for the feature vector  $x_i = (x_{i1}, \dots, x_{ip})$ , relative to that for the feature vector  $x_i = (0, \dots, 0)$ .

## Proportional Hazards Model— Continued

## Proportional Hazards Model—Continued

- What does it mean that the baseline hazard function  $h_0(t)$  is unspecified?

## Proportional Hazards Model—Continued

- What does it mean that the baseline hazard function  $h_0(t)$  is unspecified?
- Basically, we make *no assumptions about its functional form*. We allow the instantaneous probability of death at time  $t$ , given that one has survived at least until time  $t$ , to take any form.

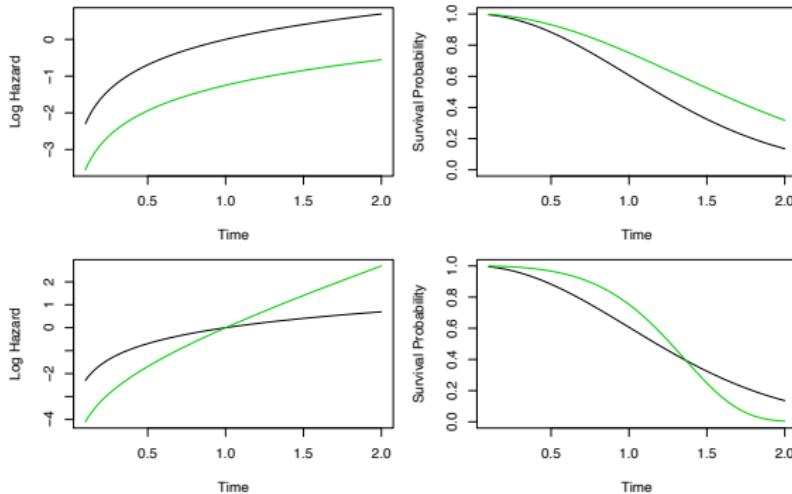
## Proportional Hazards Model— Continued

- What does it mean that the baseline hazard function  $h_0(t)$  is unspecified?
- Basically, we make *no assumptions about its functional form*. We allow the instantaneous probability of death at time  $t$ , given that one has survived at least until time  $t$ , to take any form.
- This means that the hazard function is very flexible and can model a wide range of relationships between the covariates and survival time.

## Proportional Hazards Model—Continued

- What does it mean that the baseline hazard function  $h_0(t)$  is unspecified?
- Basically, we make *no assumptions about its functional form*. We allow the instantaneous probability of death at time  $t$ , given that one has survived at least until time  $t$ , to take any form.
- This means that the hazard function is very flexible and can model a wide range of relationships between the covariates and survival time.
- Our only assumption is that a one-unit increase in  $x_{ij}$  corresponds to an increase in  $h(t|x_i)$  by a factor of  $\exp(\beta_j)$ .

# An Example



Here is an example with  $p = 1$  and a binary covariate  $x_i \in \{0, 1\}$ .

**Top row:** the log hazard and the survival function under the model are shown (green for  $x_i = 0$  and black for  $x_i = 1$ ). Because of the proportional hazards assumption, the log hazard functions differ by a constant, and the survival functions do not cross.

**Bottom row:** the proportional hazards assumption does not hold.

# Partial Likelihood

## Partial Likelihood

- Because the form of the baseline hazard is unknown, we cannot simply plug  $h(t|x_i)$  into the likelihood and then estimate  $\beta = (\beta_1, \dots, \beta_p)^T$  by maximum likelihood.

## Partial Likelihood

- Because the form of the baseline hazard is unknown, we cannot simply plug  $h(t|x_i)$  into the likelihood and then estimate  $\beta = (\beta_1, \dots, \beta_p)^T$  by maximum likelihood.
- The magic of Cox's proportional hazards model lies in the fact that it is in fact possible to estimate  $\beta$  *without having to specify the form of  $h_0(t)$* .

## Partial Likelihood

- Because the form of the baseline hazard is unknown, we cannot simply plug  $h(t|x_i)$  into the likelihood and then estimate  $\beta = (\beta_1, \dots, \beta_p)^T$  by maximum likelihood.
- The magic of Cox's proportional hazards model lies in the fact that it is in fact possible to estimate  $\beta$  *without having to specify the form of  $h_0(t)$* .
- To accomplish this, we make use of the same “sequential in time” logic that we used to derive the Kaplan-Meier survival curve and the log-rank test. Then the total hazard at failure time  $y_i$  for the at-risk observations is

$$\sum_{i': y_{i'} \geq y_i} h_0(y_i) \exp \left( \sum_{j=1}^p x_{i'j} \beta_j \right).$$

## Partial Likelihood — Continued

## Partial Likelihood — Continued

- Therefore, the probability that the  $i$ th observation is the one to fail at time  $y_i$  (as opposed to one of the other observations in the risk set) is

$$\frac{h_0(y_i) \exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)}{\sum_{i':y_{i'} \geq y_i} h_0(y_i) \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)} = \frac{\exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)}{\sum_{i':y_{i'} \geq y_i} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)}.$$

## Partial Likelihood — Continued

- Therefore, the probability that the  $i$ th observation is the one to fail at time  $y_i$  (as opposed to one of the other observations in the risk set) is

$$\frac{h_0(y_i) \exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)}{\sum_{i':y_{i'} \geq y_i} h_0(y_i) \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)} = \frac{\exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)}{\sum_{i':y_{i'} \geq y_i} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)}.$$

- Notice that the unspecified baseline hazard function  $h_0(y_i)$  cancels out of the numerator and denominator!

## Partial Likelihood — Continued

## Partial Likelihood — Continued

- The partial likelihood is simply the product of these probabilities over all of the uncensored observations,

$$PL(\beta) = \prod_{i:\delta_i=1} \frac{\exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)}{\sum_{i':y_{i'} \geq y_i} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)}.$$

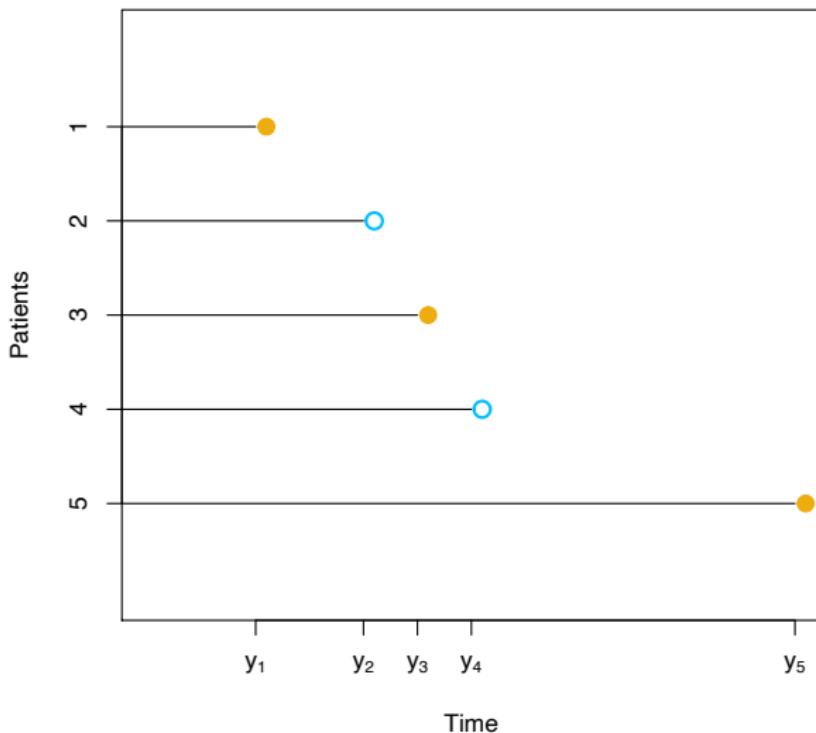
## Partial Likelihood — Continued

- The partial likelihood is simply the product of these probabilities over all of the uncensored observations,

$$PL(\beta) = \prod_{i:\delta_i=1} \frac{\exp\left(\sum_{j=1}^p x_{ij}\beta_j\right)}{\sum_{i':y_{i'} \geq y_i} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)}.$$

- Critically, the partial likelihood is valid regardless of the true value of  $h_0(t)$ , making the model very flexible and robust.

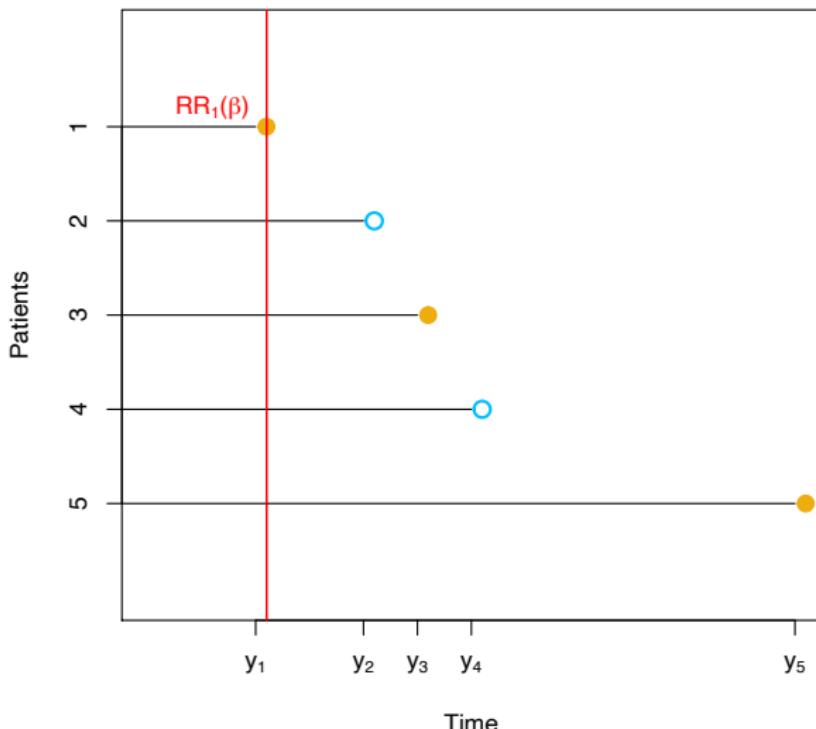
## The Partial Likelihood: Example



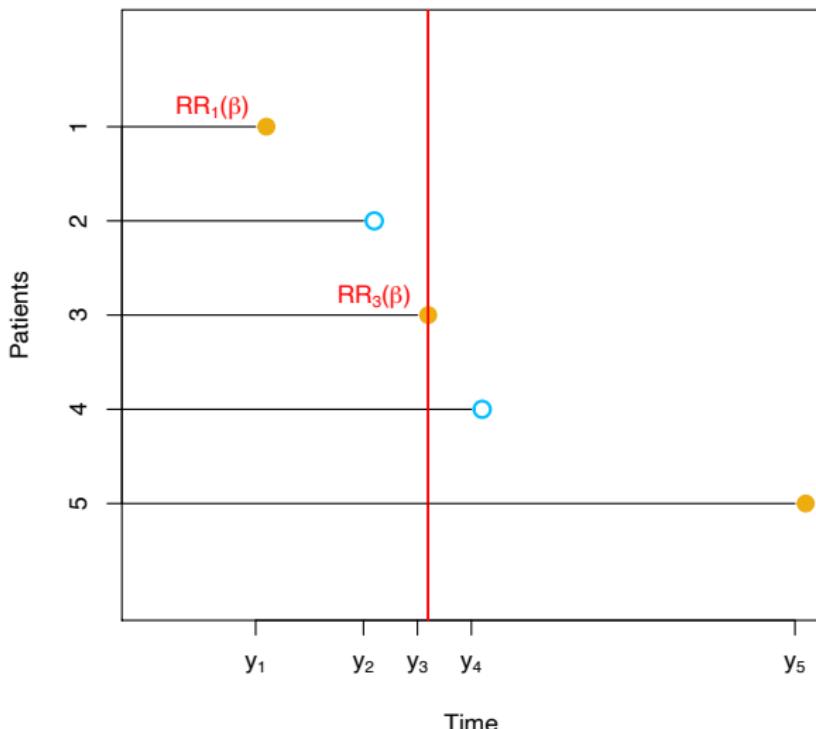
## Relative Risk Functions at each Failure Time

$$\begin{aligned} RR_1(\beta) &= \frac{\exp\left(\sum_{j=1}^p x_{1j}\beta_j\right)}{\sum_{i':y_{i'} \geq y_1} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)} \\ RR_3(\beta) &= \frac{\exp\left(\sum_{j=1}^p x_{3j}\beta_j\right)}{\sum_{i':y_{i'} \geq y_3} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)} \\ RR_5(\beta) &= \frac{\exp\left(\sum_{j=1}^p x_{5j}\beta_j\right)}{\sum_{i':y_{i'} \geq y_5} \exp\left(\sum_{j=1}^p x_{i'j}\beta_j\right)} \end{aligned}$$

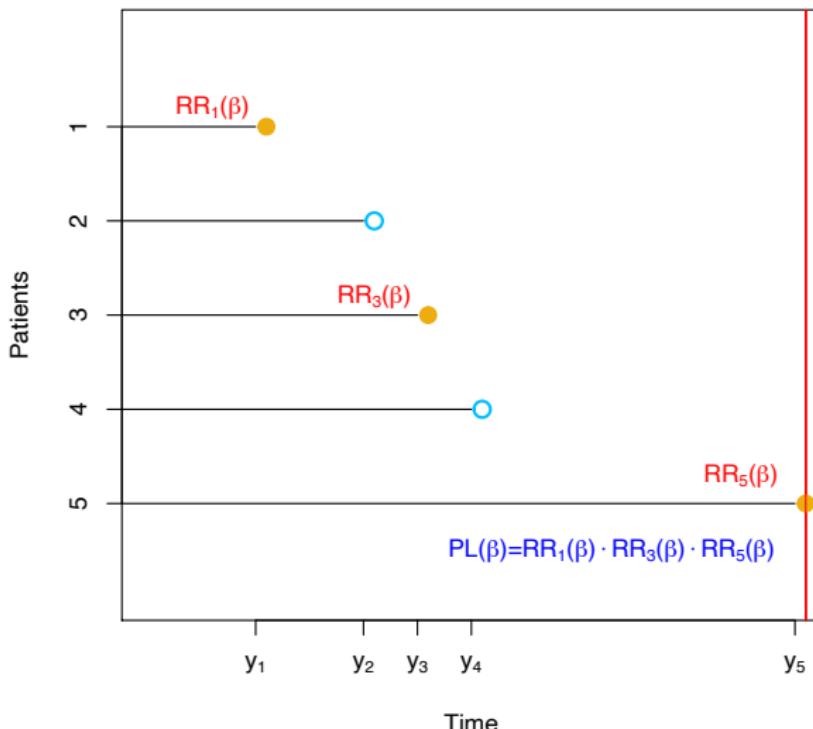
# First Failure



## Second Failure



## Third Failure



## Partial Likelihood — Computation

## Partial Likelihood — Computation

- To estimate  $\beta$ , we simply maximize the partial likelihood with respect to  $\beta$ . As is the case for logistic regression, no closed-form solution is available, and so iterative algorithms are required.

## Partial Likelihood — Computation

- To estimate  $\beta$ , we simply maximize the partial likelihood with respect to  $\beta$ . As is the case for logistic regression, no closed-form solution is available, and so iterative algorithms are required.
- In addition to estimating  $\beta$ , we can also obtain other model outputs, like those in least squares regression and logistic regression.

## Partial Likelihood — Computation

- To estimate  $\beta$ , we simply maximize the partial likelihood with respect to  $\beta$ . As is the case for logistic regression, no closed-form solution is available, and so iterative algorithms are required.
- In addition to estimating  $\beta$ , we can also obtain other model outputs, like those in least squares regression and logistic regression.
- For example, we can obtain *p-values* corresponding to particular null hypotheses (e.g.  $H_0 : \beta_j = 0$ ), as well as *estimated standard errors* and *confidence intervals* associated with the coefficients.

## Connection with the Log-Rank Test

## Connection with the Log-Rank Test

- Suppose that we have just a single predictor ( $p = 1$ ) with  $x_i \in \{0, 1\}$ . To test whether there is a difference between the survival times of the observations in the two groups, we can consider taking two possible approaches:

## Connection with the Log-Rank Test

- Suppose that we have just a single predictor ( $p = 1$ ) with  $x_i \in \{0, 1\}$ . To test whether there is a difference between the survival times of the observations in the two groups, we can consider taking two possible approaches:
  1. Fit a Cox proportional hazards model, and test the null hypothesis  $H_0 : \beta = 0$ . (Since  $p = 1$ ,  $\beta$  is a scalar.)

## Connection with the Log-Rank Test

- Suppose that we have just a single predictor ( $p = 1$ ) with  $x_i \in \{0, 1\}$ . To test whether there is a difference between the survival times of the observations in the two groups, we can consider taking two possible approaches:
  1. Fit a Cox proportional hazards model, and test the null hypothesis  $H_0 : \beta = 0$ . (Since  $p = 1$ ,  $\beta$  is a scalar.)
  2. Perform a log-rank test to compare the two groups.

## Connection with the Log-Rank Test

- Suppose that we have just a single predictor ( $p = 1$ ) with  $x_i \in \{0, 1\}$ . To test whether there is a difference between the survival times of the observations in the two groups, we can consider taking two possible approaches:
  1. Fit a Cox proportional hazards model, and test the null hypothesis  $H_0 : \beta = 0$ . (Since  $p = 1$ ,  $\beta$  is a scalar.)
  2. Perform a log-rank test to compare the two groups.
- Now when taking approach #1, there are a number of possible ways to test  $H_0$ . One way is known as a *score test*.

## Connection with the Log-Rank Test

- Suppose that we have just a single predictor ( $p = 1$ ) with  $x_i \in \{0, 1\}$ . To test whether there is a difference between the survival times of the observations in the two groups, we can consider taking two possible approaches:
  1. Fit a Cox proportional hazards model, and test the null hypothesis  $H_0 : \beta = 0$ . (Since  $p = 1$ ,  $\beta$  is a scalar.)
  2. Perform a log-rank test to compare the two groups.
- Now when taking approach #1, there are a number of possible ways to test  $H_0$ . One way is known as a *score test*.
- It turns out that in the case of a single binary covariate, the score test for  $H_0 : \beta = 0$  in Cox's proportional hazards model is *exactly equal to the log-rank test*.

## The Proportional Hazards Model— Additional Details

The discussion of the proportional hazards model glossed over a few subtleties:

## The Proportional Hazards Model— Additional Details

The discussion of the proportional hazards model glossed over a few subtleties:

- There is *no intercept* in the proportional hazards model because an intercept can be absorbed into the baseline hazard  $h_0(t)$ .

## The Proportional Hazards Model— Additional Details

The discussion of the proportional hazards model glossed over a few subtleties:

- There is *no intercept* in the proportional hazards model because an intercept can be absorbed into the baseline hazard  $h_0(t)$ .
- We have assumed that there are *no tied failure times*. In the case of ties, the exact form of the partial likelihood is more complicated, and a number of computational approximations must be used.

## The Proportional Hazards Model— Additional Details

The discussion of the proportional hazards model glossed over a few subtleties:

- There is *no intercept* in the proportional hazards model because an intercept can be absorbed into the baseline hazard  $h_0(t)$ .
- We have assumed that there are *no tied failure times*. In the case of ties, the exact form of the partial likelihood is more complicated, and a number of computational approximations must be used.
- The *partial* likelihood gets its name because it is not exactly a likelihood. However, it is a very good approximation.

## The Proportional Hazards Model— Additional Details

The discussion of the proportional hazards model glossed over a few subtleties:

- There is *no intercept* in the proportional hazards model because an intercept can be absorbed into the baseline hazard  $h_0(t)$ .
- We have assumed that there are *no tied failure times*. In the case of ties, the exact form of the partial likelihood is more complicated, and a number of computational approximations must be used.
- The *partial* likelihood gets its name because it is not exactly a likelihood. However, it is a very good approximation.
- We have focused only on estimation of the coefficients  $\beta$ . However, we may also wish to estimate the baseline hazard  $h_0(t)$ , for instance so that we can estimate the survival curve  $S(t|x)$ . These are implemented in the **survival** package in **R**.

## Example: Brain Cancer Data

	Coefficient	Std. error	z-statistic	p-value
sex [Male]	0.18	0.36	0.51	0.61
diagnosis [LG Glioma]	0.92	0.64	1.43	0.15
diagnosis [HG Glioma]	2.15	0.45	4.78	0.00
diagnosis [Other]	0.89	0.66	1.35	0.18
loc [Supratentorial]	0.44	0.70	0.63	0.53
ki	-0.05	0.02	-3.00	<0.01
gtv	0.03	0.02	1.54	0.12
stereo [SRT]	0.18	0.60	0.30	0.77

## Example: Brain Cancer Data

	Coefficient	Std. error	z-statistic	p-value
sex [Male]	0.18	0.36	0.51	0.61
diagnosis [LG Glioma]	0.92	0.64	1.43	0.15
diagnosis [HG Glioma]	2.15	0.45	4.78	0.00
diagnosis [Other]	0.89	0.66	1.35	0.18
loc [Supratentorial]	0.44	0.70	0.63	0.53
ki	-0.05	0.02	-3.00	<0.01
gtv	0.03	0.02	1.54	0.12
stereo [SRT]	0.18	0.60	0.30	0.77

- This table shows the result of fitting the proportional hazards model to the **BrainCancer** data.

## Example: Brain Cancer Data

	Coefficient	Std. error	z-statistic	p-value
sex [Male]	0.18	0.36	0.51	0.61
diagnosis [LG Glioma]	0.92	0.64	1.43	0.15
diagnosis [HG Glioma]	2.15	0.45	4.78	0.00
diagnosis [Other]	0.89	0.66	1.35	0.18
loc [Supratentorial]	0.44	0.70	0.63	0.53
ki	-0.05	0.02	-3.00	<0.01
gtv	0.03	0.02	1.54	0.12
stereo [SRT]	0.18	0.60	0.30	0.77

- This table shows the result of fitting the proportional hazards model to the **BrainCancer** data.
- We see for example that each one-unit increase in the Karnofsky index corresponds to a multiplier of  $\exp(-0.05) = 0.95$  in the instantaneous chance of dying.

## Example: Brain Cancer Data

	Coefficient	Std. error	z-statistic	p-value
sex [Male]	0.18	0.36	0.51	0.61
diagnosis [LG Glioma]	0.92	0.64	1.43	0.15
diagnosis [HG Glioma]	2.15	0.45	4.78	0.00
diagnosis [Other]	0.89	0.66	1.35	0.18
loc [Supratentorial]	0.44	0.70	0.63	0.53
ki	-0.05	0.02	-3.00	<0.01
gtv	0.03	0.02	1.54	0.12
stereo [SRT]	0.18	0.60	0.30	0.77

- This table shows the result of fitting the proportional hazards model to the **BrainCancer** data.
- We see for example that each one-unit increase in the Karnofsky index corresponds to a multiplier of  $\exp(-0.05) = 0.95$  in the instantaneous chance of dying.
- In other words, the higher the Karnofsky index, the lower the chance of dying at any given point in time. This effect is highly significant, with a *p*-value of 0.0027.

## Example: Publication Data

Next, we consider the **Publication** dataset involving the time to publication of journal papers reporting the results of clinical trials funded by the National Heart, Lung, and Blood Institute.

## Example: Publication Data

Next, we consider the **Publication** dataset involving the time to publication of journal papers reporting the results of clinical trials funded by the National Heart, Lung, and Blood Institute.

- For 244 trials, the time in months until publication is recorded. Of the 244 trials, only 156 were published during the study period; the remaining studies were censored.

## Example: Publication Data

Next, we consider the **Publication** dataset involving the time to publication of journal papers reporting the results of clinical trials funded by the National Heart, Lung, and Blood Institute.

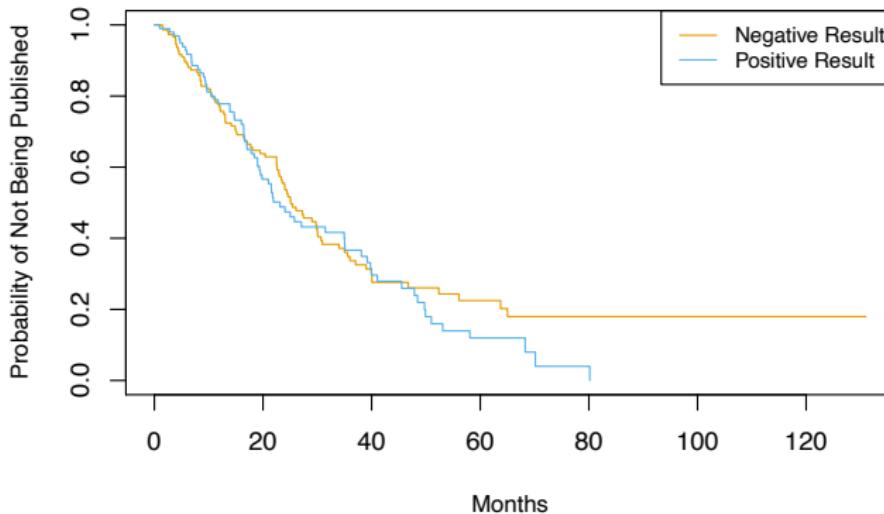
- For 244 trials, the time in months until publication is recorded. Of the 244 trials, only 156 were published during the study period; the remaining studies were censored.
- The covariates include whether the trial focused on a clinical endpoint (**clinend**), whether the trial involved multiple centers (**multi**), the funding mechanism within the National Institutes of Health (**mech**), trial sample size (**sampszie**), budget (**budget**), impact (**impact**, related to the number of citations), and whether the trial produced a positive (significant) result (**posres**).

## Example: Publication Data

Next, we consider the **Publication** dataset involving the time to publication of journal papers reporting the results of clinical trials funded by the National Heart, Lung, and Blood Institute.

- For 244 trials, the time in months until publication is recorded. Of the 244 trials, only 156 were published during the study period; the remaining studies were censored.
- The covariates include whether the trial focused on a clinical endpoint (**clinend**), whether the trial involved multiple centers (**multi**), the funding mechanism within the National Institutes of Health (**mech**), trial sample size (**sampszie**), budget (**budget**), impact (**impact**, related to the number of citations), and whether the trial produced a positive (significant) result (**posres**).
- The last covariate is particularly interesting, as a number of studies have suggested that positive trials have a higher publication rate.

## Publication Data — Continued



- The figure above shows the Kaplan-Meier curves for the time until publication, stratified by whether or not the study produced a positive result.
- We see slight evidence that time until publication is lower for studies with a positive result. However, the log-rank test yields a very unimpressive  $p$ -value of 0.36.

## Publication Data: Multivariate Analysis

	Coefficient	Std. error	z-statistic	p-value
<b>posres</b> [Yes]	0.55	0.18	3.02	0.00
<b>multi</b> [Yes]	0.15	0.31	0.47	0.64
<b>clinend</b> [Yes]	0.51	0.27	1.89	0.06
<b>mech</b> [K01]	1.05	1.06	1.00	0.32
<i>many mech lines omitted</i>				
<b>sampsize</b>	0.00	0.00	0.19	0.85
<b>budget</b>	0.00	0.00	1.67	0.09
<b>impact</b>	0.06	0.01	8.23	0.00

- The results of fitting Cox's proportional hazards model using all of the available features are shown above.
- We find that the chance of publication of a study with a positive result is  $e^{0.55} = 1.74$  times higher than that of a negative result at any point in time, holding all other covariates fixed.
- The very small *p*-value associated with **posres** indicates that this result is highly significant.

## Digging Deeper

In order to gain more insight into this result, on the next slide we display estimates of the survival curves associated with positive and negative results, adjusting for the other predictors.

## Digging Deeper

In order to gain more insight into this result, on the next slide we display estimates of the survival curves associated with positive and negative results, adjusting for the other predictors.

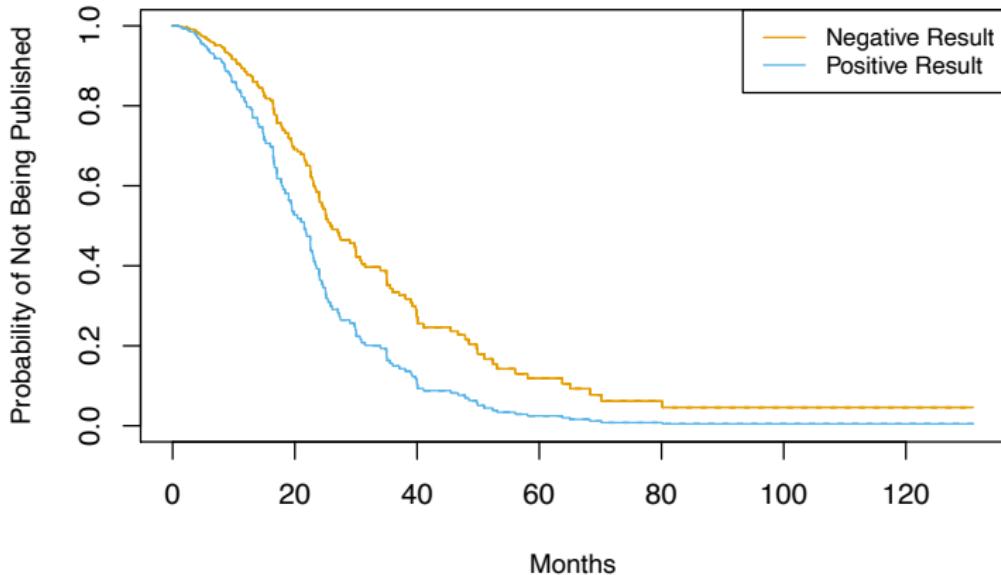
- To produce these survival curves, we estimated the underlying baseline hazard  $h_0(t)$ : this is implemented in the `survival` package in `R`, although the details are beyond the scope of this course.

## Digging Deeper

In order to gain more insight into this result, on the next slide we display estimates of the survival curves associated with positive and negative results, adjusting for the other predictors.

- To produce these survival curves, we estimated the underlying baseline hazard  $h_0(t)$ : this is implemented in the `survival` package in R, although the details are beyond the scope of this course.
- We also needed to select representative values for the other predictors; we used the mean value for each predictor, except for the categorical predictor `mech`, for which we used the most prevalent category (`R01`).

## Adjusted Survival Curves



Adjusting for the other predictors, we now see a clear difference in the survival curves between studies with positive versus negative results. *[What has happened?]*

## AUC for Survival Analysis: the C-index

## AUC for Survival Analysis: the C-index

- This is an appealing method for assessing a fitted Cox model on a test set.

## AUC for Survival Analysis: the C-index

- This is an appealing method for assessing a fitted Cox model on a test set.
- For each observation, we calculate the estimated risk score,  $\hat{\eta}_i = \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_p x_{ip}$ , for  $i = 1, \dots, n$ , using the estimated Cox model coefficients.

## AUC for Survival Analysis: the C-index

- This is an appealing method for assessing a fitted Cox model on a test set.
- For each observation, we calculate the estimated risk score,  $\hat{\eta}_i = \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_p x_{ip}$ , for  $i = 1, \dots, n$ , using the estimated Cox model coefficients.
- Then Harrell's concordance index (or *C-index*) computes the proportion of observation pairs for which  $\hat{\eta}_{i'} > \hat{\eta}_i$  and  $y_i > y_{i'}$ :

$$C = \frac{\sum_{i,i':y_i>y_{i'}} I(\hat{\eta}_{i'} > \hat{\eta}_i) \delta_{i'}}{\sum_{i,i':y_i>y_{i'}} \delta_{i'}}.$$

## AUC for Survival Analysis: the C-index

- This is an appealing method for assessing a fitted Cox model on a test set.
- For each observation, we calculate the estimated risk score,  $\hat{\eta}_i = \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_p x_{ip}$ , for  $i = 1, \dots, n$ , using the estimated Cox model coefficients.
- Then Harrell's concordance index (or *C-index*) computes the proportion of observation pairs for which  $\hat{\eta}_{i'} > \hat{\eta}_i$  and  $y_i > y_{i'}$ :

$$C = \frac{\sum_{i,i':y_i>y_{i'}} I(\hat{\eta}_{i'} > \hat{\eta}_i) \delta_{i'}}{\sum_{i,i':y_i>y_{i'}} \delta_{i'}}.$$

- This is the proportion of pairs for which the model correctly predicts the relative survival time, among all pairs for which this can be determined

## C-index: Example

We fit a Cox proportional hazards model on the training set of the **Publication** data, and computed the  $C$ -index on the test set.

This yielded  $C = 0.733$ . Roughly speaking, given two random papers from the test set, the model can predict with 73.3% accuracy which will be published first.

## Additional Topics

Here are some additional topics that are covered in the text:

## Additional Topics

Here are some additional topics that are covered in the text:

- Other types of censoring: left and interval censoring.

## Additional Topics

Here are some additional topics that are covered in the text:

- Other types of censoring: left and interval censoring.
- The choice of time scale, e.g calendar time or age?

## Additional Topics

Here are some additional topics that are covered in the text:

- Other types of censoring: left and interval censoring.
- The choice of time scale, e.g calendar time or age?
- *Time-dependent covariates* — where we measure a feature (like blood pressure) at different time points

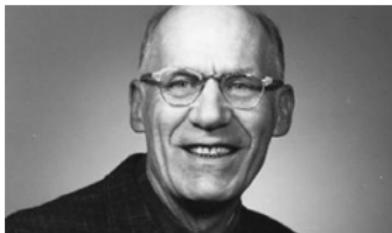
## Additional Topics

Here are some additional topics that are covered in the text:

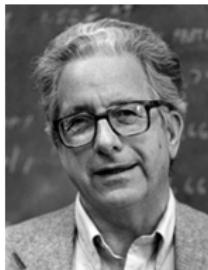
- Other types of censoring: left and interval censoring.
- The choice of time scale, e.g calendar time or age?
- *Time-dependent covariates* — where we measure a feature (like blood pressure) at different time points
- Methods for checking the proportional hazards assumption

There are also approaches for modeling survival data using other machine learning methods such as *random forests*, *boosting* and *neural networks*. Some of these avoid the proportional hazards assumption.

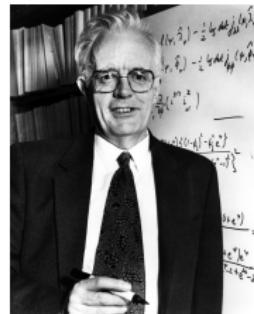
# Those big names one last time



Edward Kaplan



Paul Meier



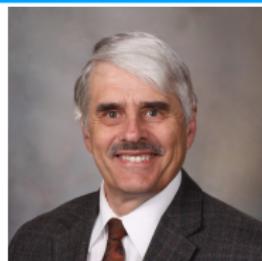
David Cox



Nathan Mantel



William Haenszel



Terry Therneau

(log rank test)

(author of *Survival package in R*)

# Software for Survival Analysis

# Software for Survival Analysis

- The examples in this lecture were created using the `survival` and `glmnet` packages in `R`.
- Both packages can handle time-dependent covariates and general forms of censoring.

# Software for Survival Analysis

- The examples in this lecture were created using the `survival` and `glmnet` packages in `R`.
- Both packages can handle time-dependent covariates and general forms of censoring.
- Software for other machine learning approaches can be found both the `R` repository and the `scikit-survival` Python collection.

# Unsupervised Learning

## *Unsupervised vs Supervised Learning:*

- Most of this course focuses on *supervised learning* methods such as regression and classification.
- In that setting we observe both a set of features  $X_1, X_2, \dots, X_p$  for each object, as well as a response or outcome variable  $Y$ . The goal is then to predict  $Y$  using  $X_1, X_2, \dots, X_p$ .
- Here we instead focus on *unsupervised learning*, where we observe only the features  $X_1, X_2, \dots, X_p$ . We are not interested in prediction, because we do not have an associated response variable  $Y$ .

# The Goals of Unsupervised Learning

- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We discuss two methods:
  - *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied, and
  - *clustering*, a broad class of methods for discovering unknown subgroups in data.

# The Challenge of Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of growing importance in a number of fields:
  - subgroups of breast cancer patients grouped by their gene expression measurements,
  - groups of shoppers characterized by their browsing and purchase histories,
  - movies grouped by the ratings assigned by movie viewers.

## Another advantage

- It is often easier to obtain *unlabeled data* — from a lab instrument or a computer — than *labeled data*, which can require human intervention.
- For example it is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?

# Principal Components Analysis

- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

## Principal Components Analysis: details

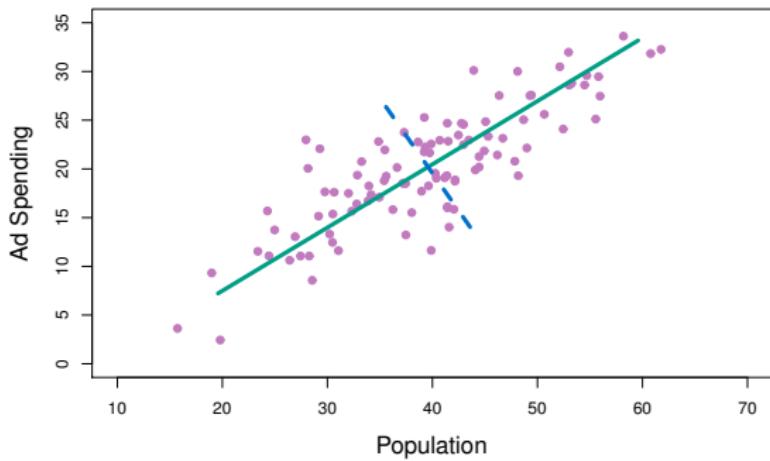
- The *first principal component* of a set of features  $X_1, X_2, \dots, X_p$  is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p$$

that has the largest variance. By *normalized*, we mean that  $\sum_{j=1}^p \phi_{j1}^2 = 1$ .

- We refer to the elements  $\phi_{11}, \dots, \phi_{p1}$  as the loadings of the first principal component; together, the loadings make up the principal component loading vector,  
 $\phi_1 = (\phi_{11} \ \phi_{21} \ \dots \ \phi_{p1})^T$ .
- We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

## PCA: example



The population size (**pop**) and ad spending (**ad**) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component direction, and the blue dashed line indicates the second principal component direction.

## Computation of Principal Components

- Suppose we have a  $n \times p$  data set  $\mathbf{X}$ . Since we are only interested in variance, we assume that each of the variables in  $\mathbf{X}$  has been centered to have mean zero (that is, the column means of  $\mathbf{X}$  are zero).
- We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \dots + \phi_{p1}x_{ip} \quad (1)$$

for  $i = 1, \dots, n$  that has largest sample variance, subject to the constraint that  $\sum_{j=1}^p \phi_{j1}^2 = 1$ .

- Since each of the  $x_{ij}$  has mean zero, then so does  $z_{i1}$  (for any values of  $\phi_{j1}$ ). Hence the sample variance of the  $z_{i1}$  can be written as  $\frac{1}{n} \sum_{i=1}^n z_{i1}^2$ .

## Computation: continued

- Plugging in (1) the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11}, \dots, \phi_{p1}}{\text{maximize}} \frac{1}{n} \sum_{i=1}^n \left( \sum_{j=1}^p \phi_{j1} x_{ij} \right)^2 \quad \text{subject to } \sum_{j=1}^p \phi_{j1}^2 = 1.$$

- This problem can be solved via a singular-value decomposition of the matrix  $\mathbf{X}$ , a standard technique in linear algebra.
- We refer to  $Z_1$  as the first principal component, with realized values  $z_{11}, \dots, z_{n1}$

## Geometry of PCA

- The loading vector  $\phi_1$  with elements  $\phi_{11}, \phi_{21}, \dots, \phi_{p1}$  defines a direction in feature space along which the data vary the most.
- If we project the  $n$  data points  $x_1, \dots, x_n$  onto this direction, the projected values are the principal component scores  $z_{11}, \dots, z_{n1}$  themselves.

## Further principal components

- The second principal component is the linear combination of  $X_1, \dots, X_p$  that has maximal variance among all linear combinations that are *uncorrelated* with  $Z_1$ .
- The second principal component scores  $z_{12}, z_{22}, \dots, z_{n2}$  take the form

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \dots + \phi_{p2}x_{ip},$$

where  $\phi_2$  is the second principal component loading vector, with elements  $\phi_{12}, \phi_{22}, \dots, \phi_{p2}$ .

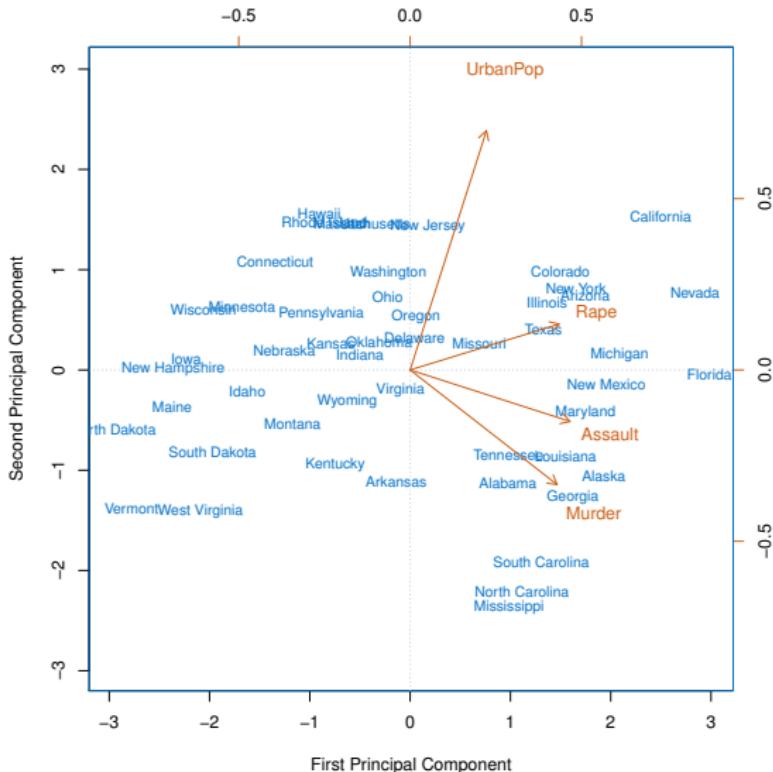
## Further principal components: continued

- It turns out that constraining  $Z_2$  to be uncorrelated with  $Z_1$  is equivalent to constraining the direction  $\phi_2$  to be orthogonal (perpendicular) to the direction  $\phi_1$ . And so on.
- The principal component directions  $\phi_1, \phi_2, \phi_3, \dots$  are the ordered sequence of right singular vectors of the matrix  $\mathbf{X}$ , and the variances of the components are  $\frac{1}{n}$  times the squares of the singular values. There are at most  $\min(n - 1, p)$  principal components.

## Illustration

- **USAarrests** data: For each of the fifty states in the United States, the data set contains the number of arrests per 100,000 residents for each of three crimes: **Assault**, **Murder**, and **Rape**. We also record **UrbanPop** (the percent of the population in each state living in urban areas).
- The principal component score vectors have length  $n = 50$ , and the principal component loading vectors have length  $p = 4$ .
- PCA was performed after standardizing each variable to have mean zero and standard deviation one.

# USAarrests data: PCA plot



## Figure details

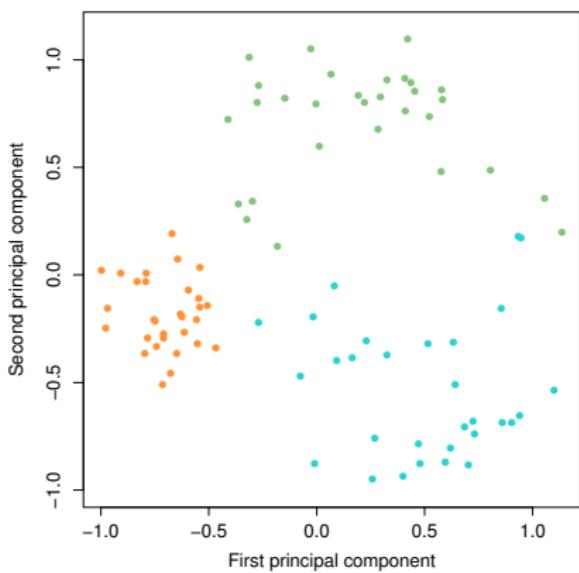
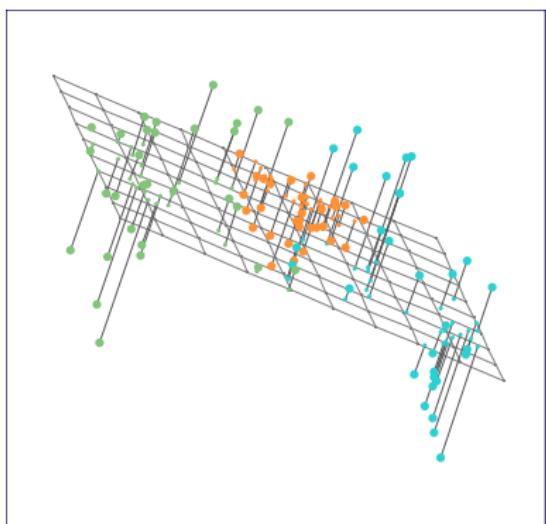
The first two principal components for the USArrests data.

- The blue state names represent the scores for the first two principal components.
- The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for **Rape** on the first component is 0.54, and its loading on the second principal component 0.17 [the word **Rape** is centered at the point (0.54, 0.17)].
- This figure is known as a *biplot*, because it displays both the principal component scores and the principal component loadings.

## PCA loadings

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

# Another Interpretation of Principal Components

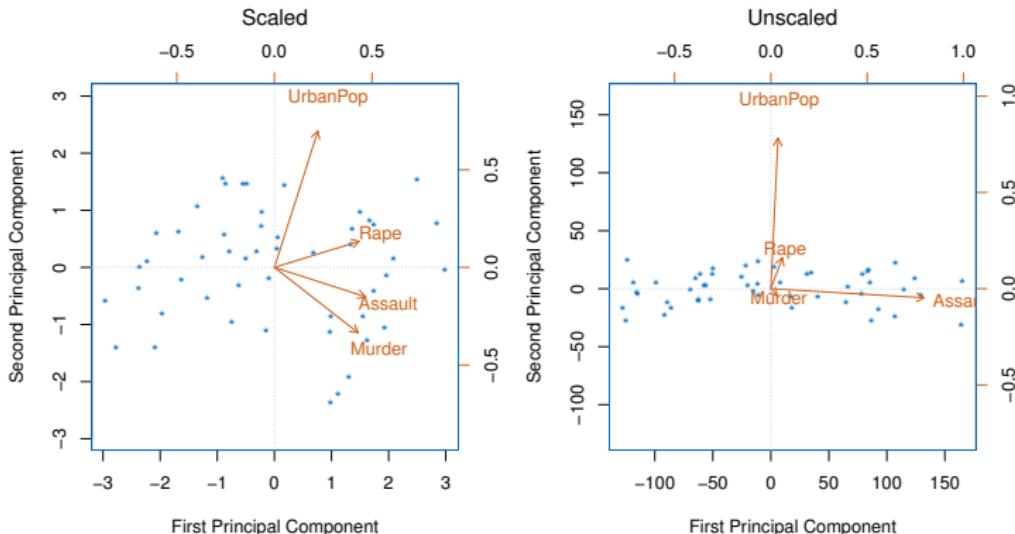


## PCA find the hyperplane closest to the observations

- The first principal component loading vector has a very special property: it defines the line in  $p$ -dimensional space that is *closest* to the  $n$  observations (using average squared Euclidean distance as a measure of closeness)
- The notion of principal components as the dimensions that are closest to the  $n$  observations extends beyond just the first principal component.
- For instance, the first two principal components of a data set span the plane that is closest to the  $n$  observations, in terms of average squared Euclidean distance.

# Scaling of the variables matters

- If the variables are in different units, scaling each to have standard deviation equal to one is recommended.
- If they are in the same units, you might or might not scale the variables.



## Proportion Variance Explained

- To understand the strength of each component, we are interested in knowing the proportion of variance explained (PVE) by each one.
- The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^p \text{Var}(X_j) = \sum_{j=1}^p \frac{1}{n} \sum_{i=1}^n x_{ij}^2,$$

and the variance explained by the  $m$ th principal component is

$$\text{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^n z_{im}^2.$$

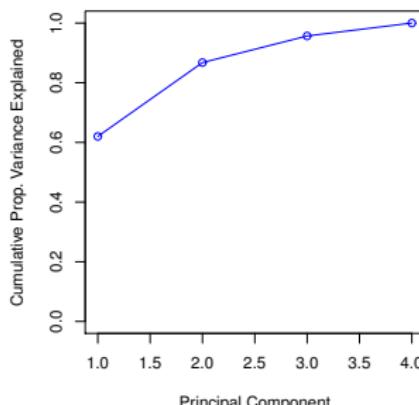
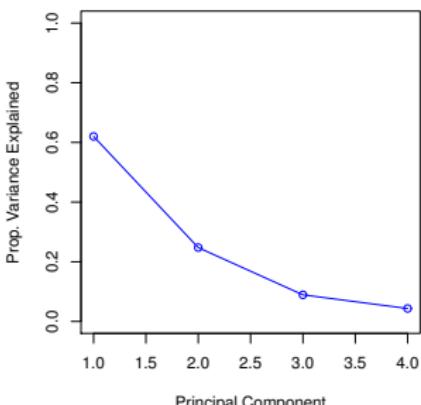
- It can be shown that  $\sum_{j=1}^p \text{Var}(X_j) = \sum_{m=1}^M \text{Var}(Z_m)$ , with  $M = \min(n - 1, p)$ .

## Proportion Variance Explained: continued

- Therefore, the PVE of the  $m$ th principal component is given by the positive quantity between 0 and 1

$$\frac{\sum_{i=1}^n z_{im}^2}{\sum_{j=1}^p \sum_{i=1}^n x_{ij}^2}.$$

- The PVEs sum to one. We sometimes display the cumulative PVEs.



## How many principal components should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
  - *Why not?*

## How many principal components should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
  - *Why not?*
  - When could we use cross-validation to select the number of components?

## How many principal components should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
  - *Why not?*
  - When could we use cross-validation to select the number of components?
- the “scree plot” on the previous slide can be used as a guide: we look for an “elbow”.

# Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- To make this concrete, we must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

## PCA vs Clustering

- PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.
- Clustering looks for homogeneous subgroups among the observations.

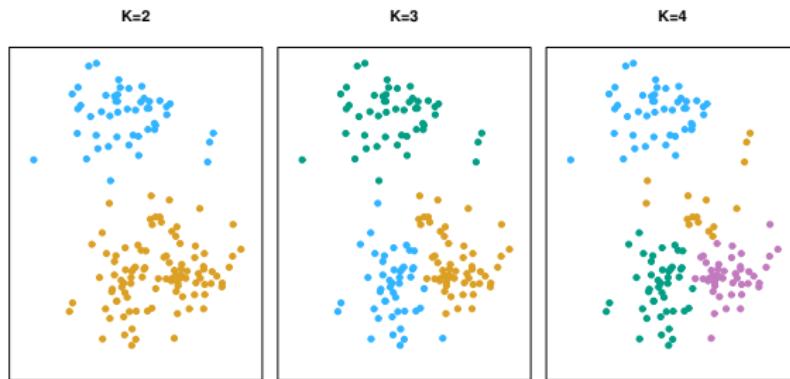
## Clustering for Market Segmentation

- Suppose we have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people.
- Our goal is to perform *market segmentation* by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.
- The task of performing market segmentation amounts to clustering the people in the data set.

## Two clustering methods

- In *K-means clustering*, we seek to partition the observations into a pre-specified number of clusters.
- In *hierarchical clustering*, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to  $n$ .

## $K$ -means clustering



A simulated data set with 150 observations in 2-dimensional space. Panels show the results of applying  $K$ -means clustering with different values of  $K$ , the number of clusters. The color of each observation indicates the cluster to which it was assigned using the  $K$ -means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.

## Details of $K$ -means clustering

Let  $C_1, \dots, C_K$  denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1.  $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$ . In other words, each observation belongs to at least one of the  $K$  clusters.
2.  $C_k \cap C_{k'} = \emptyset$  for all  $k \neq k'$ . In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the  $i$ th observation is in the  $k$ th cluster, then  $i \in C_k$ .

## Details of $K$ -means clustering: continued

- The idea behind  $K$ -means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.
- The within-cluster variation for cluster  $C_k$  is a measure  $\text{WCV}(C_k)$  of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \text{WCV}(C_k) \right\}. \quad (2)$$

- In words, this formula says that we want to partition the observations into  $K$  clusters such that the total within-cluster variation, summed over all  $K$  clusters, is as small as possible.

## How to define within-cluster variation?

- Typically we use Euclidean distance

$$\text{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2, \quad (3)$$

where  $|C_k|$  denotes the number of observations in the  $k$ th cluster.

- Combining (2) and (3) gives the optimization problem that defines  $K$ -means clustering,

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\}. \quad (4)$$

## $K$ -Means Clustering Algorithm

1. Randomly assign a number, from 1 to  $K$ , to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
  - 2.1 For each of the  $K$  clusters, compute the cluster *centroid*.  
The  $k$ th cluster centroid is the vector of the  $p$  feature means for the observations in the  $k$ th cluster.
  - 2.2 Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

## Properties of the Algorithm

- This algorithm is guaranteed to decrease the value of the objective (4) at each step. *Why?*

## Properties of the Algorithm

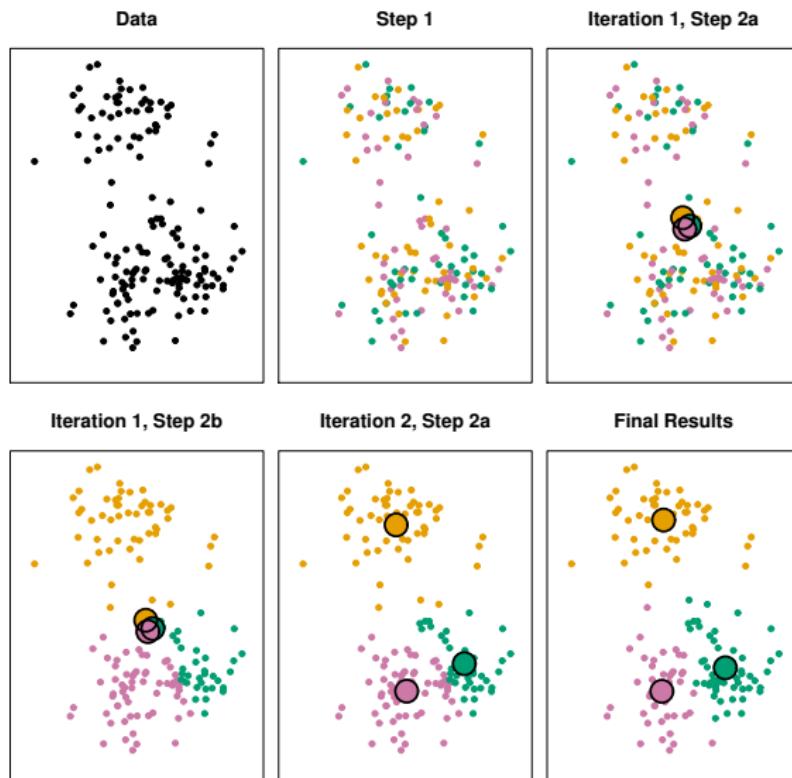
- This algorithm is guaranteed to decrease the value of the objective (4) at each step. *Why?* Note that

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2,$$

where  $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$  is the mean for feature  $j$  in cluster  $C_k$ .

- however it is not guaranteed to give the global minimum.  
*Why not?*

## Example



## Details of Previous Figure

The progress of the K-means algorithm with  $K=3$ .

- *Top left:* The observations are shown.
- *Top center:* In Step 1 of the algorithm, each observation is randomly assigned to a cluster.
- *Top right:* In Step 2(a), the cluster centroids are computed. These are shown as large colored disks. Initially the centroids are almost completely overlapping because the initial cluster assignments were chosen at random.
- *Bottom left:* In Step 2(b), each observation is assigned to the nearest centroid.
- *Bottom center:* Step 2(a) is once again performed, leading to new cluster centroids.
- *Bottom right:* The results obtained after 10 iterations.

## Example: different starting values



## Details of Previous Figure

$K$ -means clustering performed six times on the data from previous figure with  $K = 3$ , each time with a different random assignment of the observations in Step 1 of the  $K$ -means algorithm.

Above each plot is the value of the objective (4).

Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters.

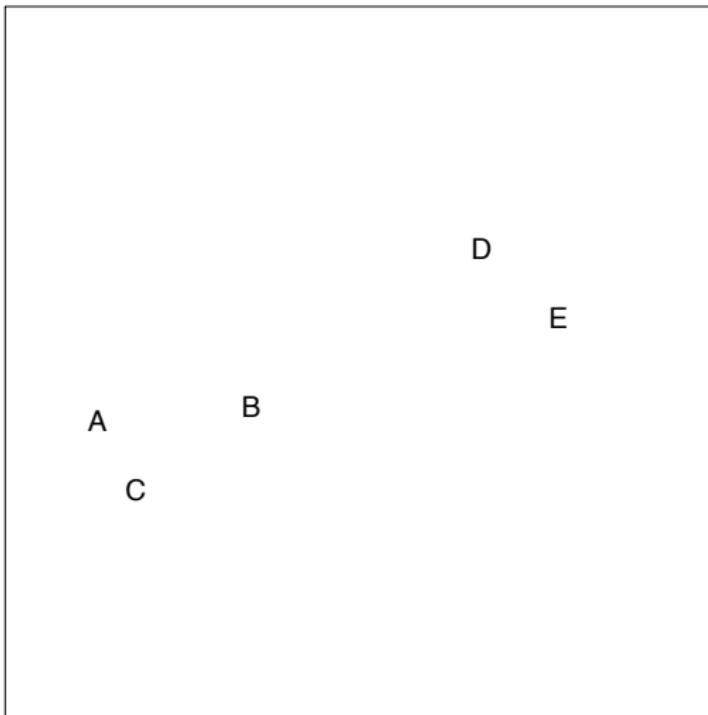
Those labeled in red all achieved the same best solution, with an objective value of 235.8

# Hierarchical Clustering

- $K$ -means clustering requires us to pre-specify the number of clusters  $K$ . This can be a disadvantage (later we discuss strategies for choosing  $K$ )
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of  $K$ .
- In this section, we describe *bottom-up* or *agglomerative* clustering. This is the most common type of hierarchical clustering, and refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

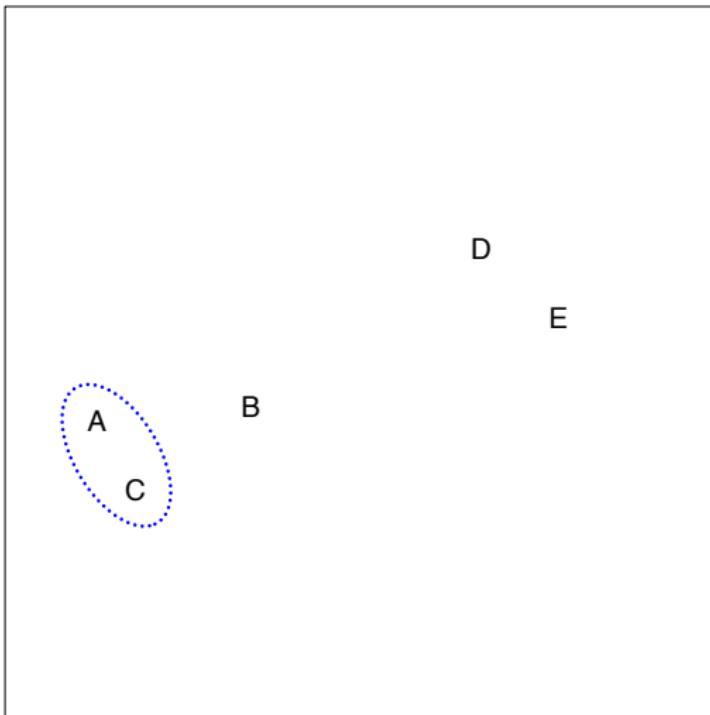
## Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



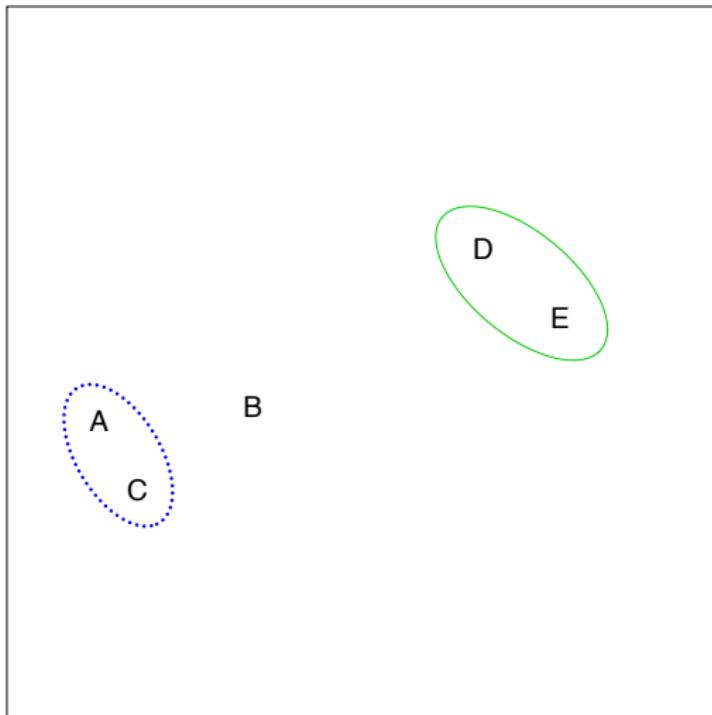
## Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



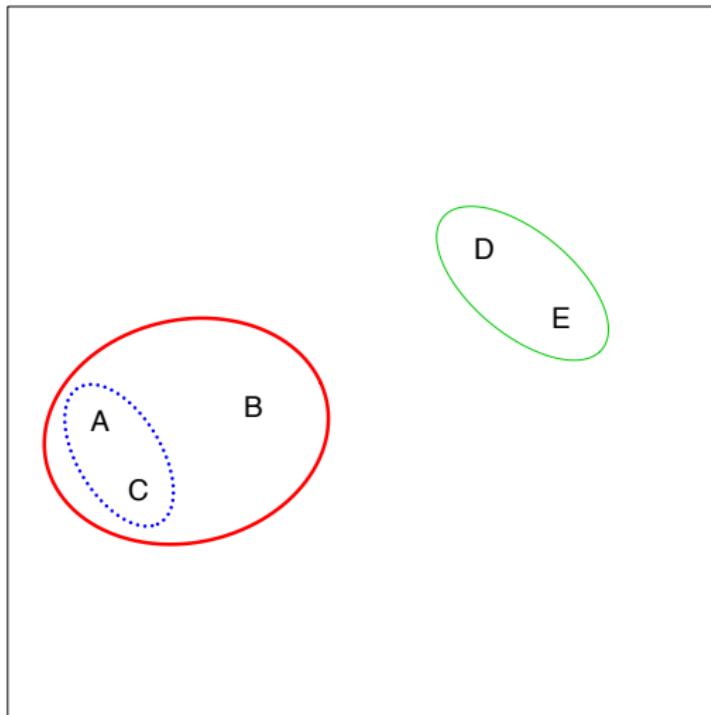
## Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



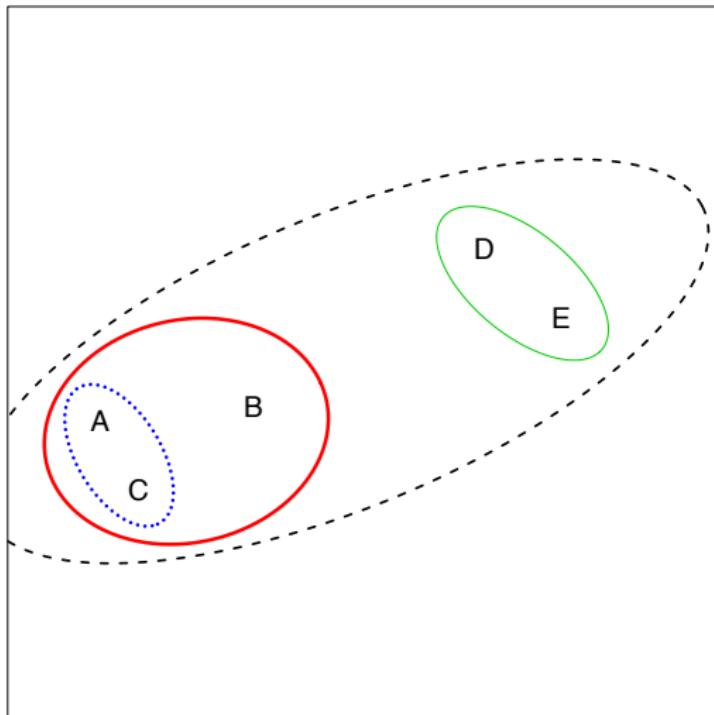
## Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



## Hierarchical Clustering: the idea

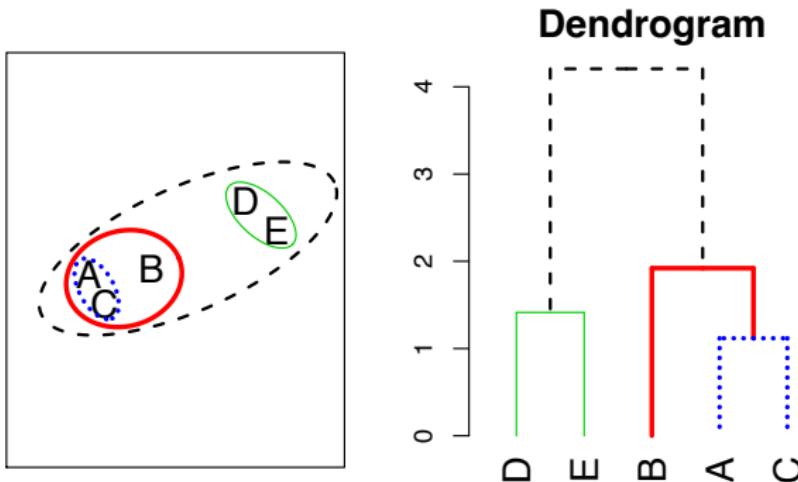
Builds a hierarchy in a “bottom-up” fashion...



# Hierarchical Clustering Algorithm

The approach in words:

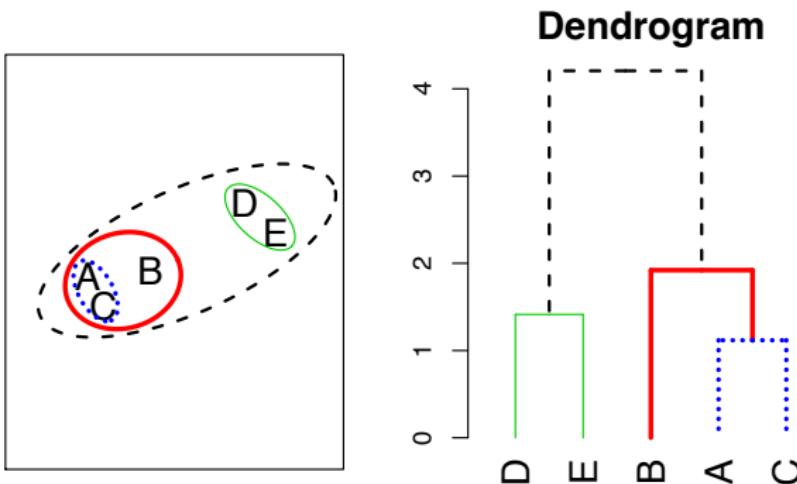
- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.



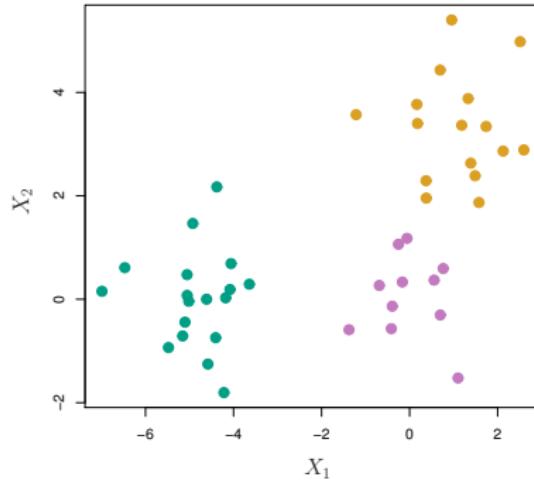
# Hierarchical Clustering Algorithm

The approach in words:

- Start with each point in its own cluster.
- Identify the **closest** two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

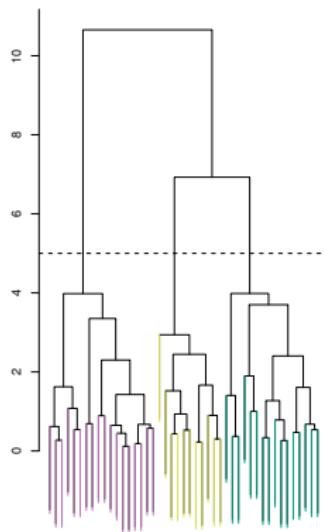
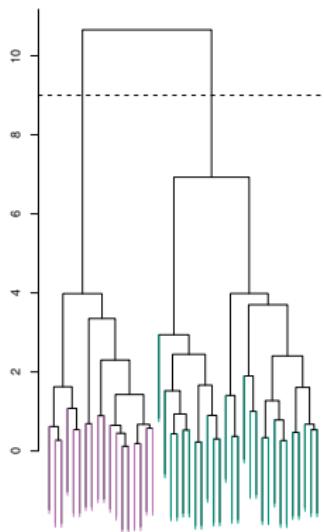
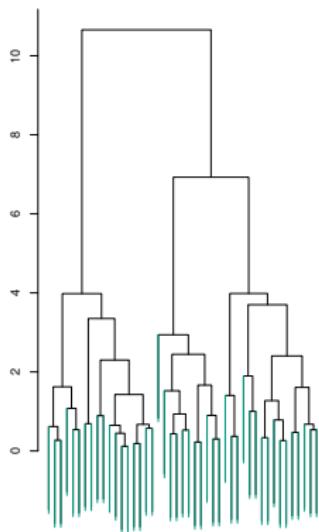


## An Example



45 observations generated in 2-dimensional space. In reality there are three distinct classes, shown in separate colors. However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

# Application of hierarchical clustering



## Details of previous figure

- *Left:* Dendrogram obtained from hierarchically clustering the data from previous slide, with complete linkage and Euclidean distance.
- *Center:* The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.
- *Right:* The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure

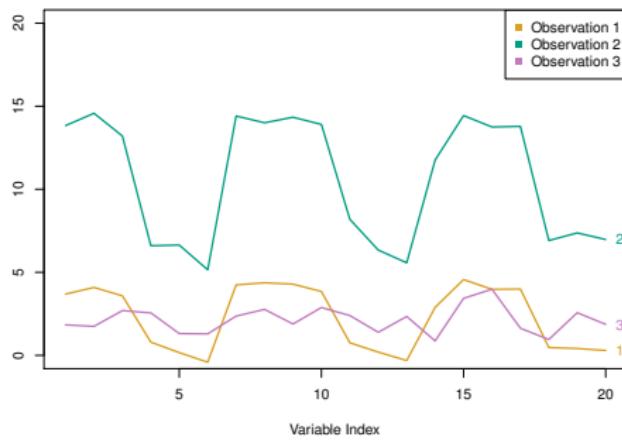
## Types of Linkage

<i>Linkage</i>	<i>Description</i>
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>largest</i> of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>smallest</i> of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>average</i> of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length $p$ ) and the centroid for cluster B. Centroid linkage can result in undesirable <i>inversions</i> .



## Choice of Dissimilarity Measure

- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.

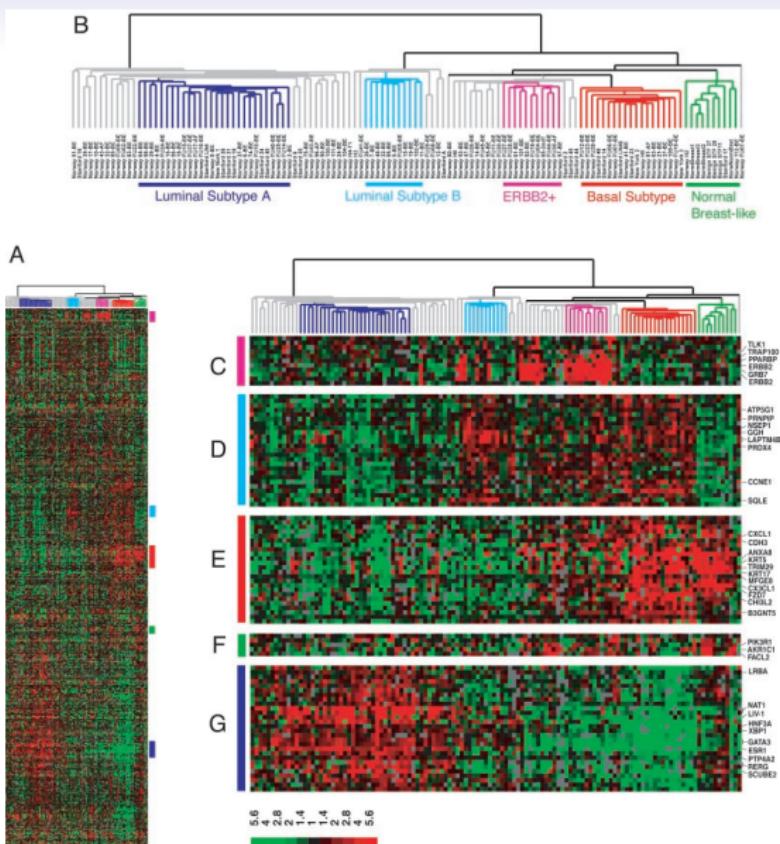


## Practical issues

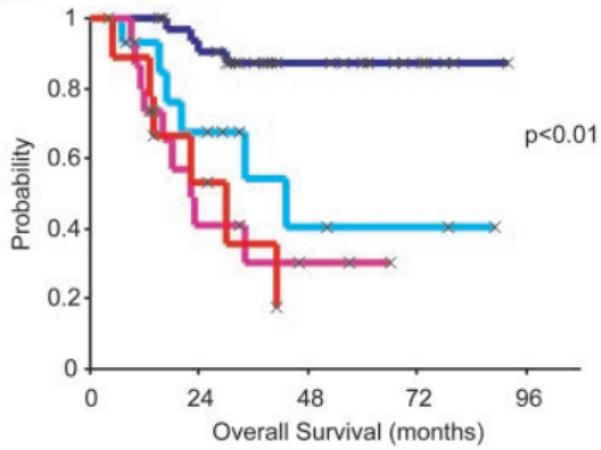
- *Scaling of the variables matters!*. Should the observations or features first be standardized in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering,
  - What dissimilarity measure should be used?
  - What type of linkage should be used?
- How many clusters to choose? (in both  $K$ -means or hierarchical clustering). Difficult problem. No agreed-upon method. See Elements of Statistical Learning, chapter 13 for more details.
- Which features should we use to drive the clustering?

## Example: breast cancer microarray study

- “Repeated observation of breast tumor subtypes in independent gene expression data sets;” Sorlie at el, PNAS 2003
- Gene expression measurements for about  $\sim 8000$  genes, for each of 88 breast cancer patients.
- Average linkage, correlation metric
- Clustered samples using 500 *intrinsic genes*: each woman was measured before and after chemotherapy. Intrinsic genes have smallest within/between variation.



**B** Norway/Stanford data set



# Conclusions

- *Unsupervised learning* is important for understanding the variation and grouping structure of a set of unlabeled data, and can be a useful pre-processor for supervised learning
- It is intrinsically more difficult than *supervised learning* because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy).
- It is an active field of research, with many recently developed tools such as *self-organizing maps*, *independent components analysis* and *spectral clustering*.  
See *The Elements of Statistical Learning*, chapter 14.

# Matrix Completion and Missing Values

## Matrix Completion and Missing Values

- It is often the case that data matrices  $\mathbf{X}$  have missing entries, often represented by **NAs** (not available).

## Matrix Completion and Missing Values

- It is often the case that data matrices  $\mathbf{X}$  have missing entries, often represented by **NAs** (not available).
- This is a nuisance, since many of our modeling procedures, such as linear regression and GLMs require complete data.

## Matrix Completion and Missing Values

- It is often the case that data matrices  $\mathbf{X}$  have missing entries, often represented by **NAs** (not available).
- This is a nuisance, since many of our modeling procedures, such as linear regression and GLMs require complete data.
- Sometimes imputation *is the prediction problem!* — as in recommender systems.

## Matrix Completion and Missing Values

- It is often the case that data matrices  $\mathbf{X}$  have missing entries, often represented by **NAs** (not available).
- This is a nuisance, since many of our modeling procedures, such as linear regression and GLMs require complete data.
- Sometimes imputation *is the prediction problem!* — as in recommender systems.
- One simple approach is *mean imputation* — replace missing values for a variable by the mean of the non-missing entries.
- This ignores the correlations among variables; we should be able to exploit these correlations when imputing missing values.

## Matrix Completion and Missing Values

- It is often the case that data matrices  $\mathbf{X}$  have missing entries, often represented by **NAs** (not available).
- This is a nuisance, since many of our modeling procedures, such as linear regression and GLMs require complete data.
- Sometimes imputation *is the prediction problem!* — as in recommender systems.
- One simple approach is *mean imputation* — replace missing values for a variable by the mean of the non-missing entries.
- This ignores the correlations among variables; we should be able to exploit these correlations when imputing missing values.
- We assume values are missing *at random*; i.e. the missingness should not be informative.
- We present an approach based on principal components.

# Recommender Systems

	Jerry Maguire	Oceans	Road to Perdition	A Fortunate Man	Catch Me If You Can	Driving Miss Daisy	The Two Popes	The Laundromat	Code 8	The Social Network	...
Customer 1	•	•	•	•	4	•	•	•	•	•	...
Customer 2	•	•	3	•	•	•	3	•	•	3	...
Customer 3	•	2	•	4	•	•	•	•	2	•	...
Customer 4	3	•	•	•	•	•	•	•	•	•	...
Customer 5	5	1	•	•	4	•	•	•	•	•	...
Customer 6	•	•	•	•	•	2	4	•	•	•	...
Customer 7	•	•	5	•	•	•	•	3	•	•	...
Customer 8	•	•	•	•	•	•	•	•	•	•	...
Customer 9	3	•	•	•	5	•	•	1	•	•	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Netflix users rate movies they have seen, usually a very small fraction of available movies.
- Predicting missing ratings provides a way to *recommend* movies to users. Matrix completion is one of the primary tools.

## Matrix Approximation via Principal Components

In Section 12.2.2 we gave an interpretation of principal components in terms of *matrix approximation*:

$$\underset{\mathbf{A} \in \mathbb{R}^{n \times M}, \mathbf{B} \in \mathbb{R}^{p \times M}}{\text{minimize}} \left\{ \sum_{j=1}^p \sum_{i=1}^n \left( x_{ij} - \sum_{m=1}^M a_{im} b_{jm} \right)^2 \right\}.$$

**A** is a  $n \times M$  matrix whose  $(i, m)$  element is  $a_{im}$ , and **B** is a  $p \times M$  element whose  $(j, m)$  element is  $b_{jm}$ .

- It can be shown that for any value of  $M$ , the *first  $M$  principal components* provide a solution:  $\hat{a}_{im} = z_{im}$  and  $\hat{b}_{jm} = \phi_{jm}$ .
- But what to do if the matrix has missing elements?

## Matrix Completion via Principal Components

We pose instead a modified version of the approximation criterion:

$$\underset{\mathbf{A} \in \mathbb{R}^{n \times M}, \mathbf{B} \in \mathbb{R}^{p \times M}}{\text{minimize}} \left\{ \sum_{(i,j) \in \mathcal{O}} \left( x_{ij} - \sum_{m=1}^M a_{im} b_{jm} \right)^2 \right\},$$

where  $\mathcal{O}$  is the set of all *observed* pairs of indices  $(i, j)$ , a subset of the possible  $n \times p$  pairs.

## Matrix Completion via Principal Components

We pose instead a modified version of the approximation criterion:

$$\underset{\mathbf{A} \in \mathbb{R}^{n \times M}, \mathbf{B} \in \mathbb{R}^{p \times M}}{\text{minimize}} \left\{ \sum_{(i,j) \in \mathcal{O}} \left( x_{ij} - \sum_{m=1}^M a_{im} b_{jm} \right)^2 \right\},$$

where  $\mathcal{O}$  is the set of all *observed* pairs of indices  $(i, j)$ , a subset of the possible  $n \times p$  pairs.

Once we solve this problem:

- we can estimate a missing observation  $x_{ij}$  using  
 $\hat{x}_{ij} = \sum_{m=1}^M \hat{a}_{im} \hat{b}_{jm}$ , where  $\hat{a}_{im}$  and  $\hat{b}_{jm}$  are the  $(i, m)$  and  $(j, m)$  elements of the solution matrices  $\hat{\mathbf{A}}$  and  $\hat{\mathbf{B}}$ .

## Matrix Completion via Principal Components

We pose instead a modified version of the approximation criterion:

$$\underset{\mathbf{A} \in \mathbb{R}^{n \times M}, \mathbf{B} \in \mathbb{R}^{p \times M}}{\text{minimize}} \left\{ \sum_{(i,j) \in \mathcal{O}} \left( x_{ij} - \sum_{m=1}^M a_{im} b_{jm} \right)^2 \right\},$$

where  $\mathcal{O}$  is the set of all *observed* pairs of indices  $(i, j)$ , a subset of the possible  $n \times p$  pairs.

Once we solve this problem:

- we can estimate a missing observation  $x_{ij}$  using  $\hat{x}_{ij} = \sum_{m=1}^M \hat{a}_{im} \hat{b}_{jm}$ , where  $\hat{a}_{im}$  and  $\hat{b}_{jm}$  are the  $(i, m)$  and  $(j, m)$  elements of the solution matrices  $\hat{\mathbf{A}}$  and  $\hat{\mathbf{B}}$ .
- we can (approximately) recover the  $M$  principal component scores and loadings, as if data were complete.

# Iterative Algorithm for Matrix Completion

1. *Initialize*: create a complete data matrix  $\tilde{\mathbf{X}}$  by filling in the missing values using mean imputation.
2. *Repeat*: steps (a)–(c) until the objective in (c) fails to decrease:

(a)

$$\underset{\mathbf{A} \in \mathbb{R}^{n \times M}, \mathbf{B} \in \mathbb{R}^{p \times M}}{\text{minimize}} \left\{ \sum_{j=1}^p \sum_{i=1}^n \left( \tilde{x}_{ij} - \sum_{m=1}^M a_{im} b_{jm} \right)^2 \right\}$$

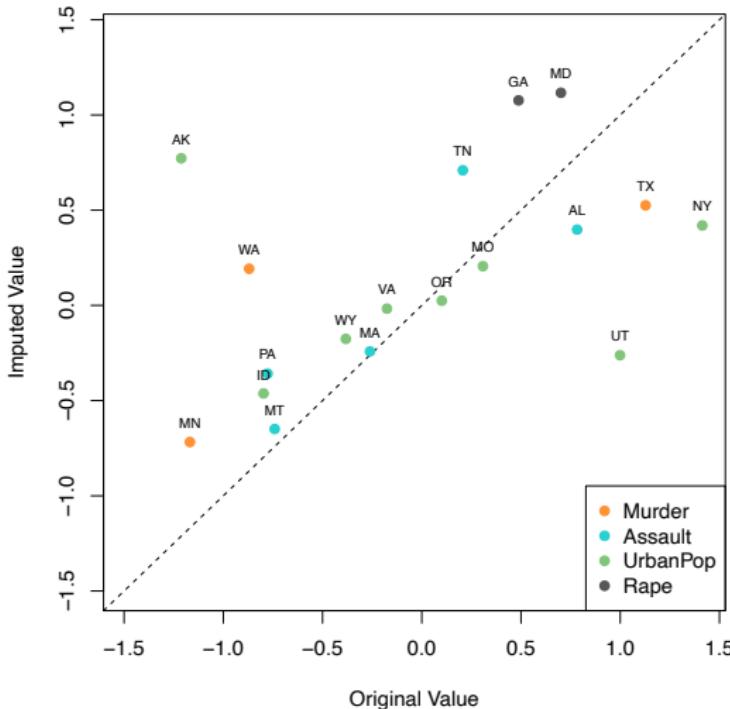
by computing the principal components of  $\tilde{\mathbf{X}}$ .

- (b) For each missing entry  $(i, j) \notin \mathcal{O}$ , set  $\tilde{x}_{ij} \leftarrow \sum_{m=1}^M \hat{a}_{im} \hat{b}_{jm}$ .
- (c) Compute the objective

$$\sum_{(i,j) \in \mathcal{O}} \left( x_{ij} - \sum_{m=1}^M \hat{a}_{im} \hat{b}_{jm} \right)^2.$$

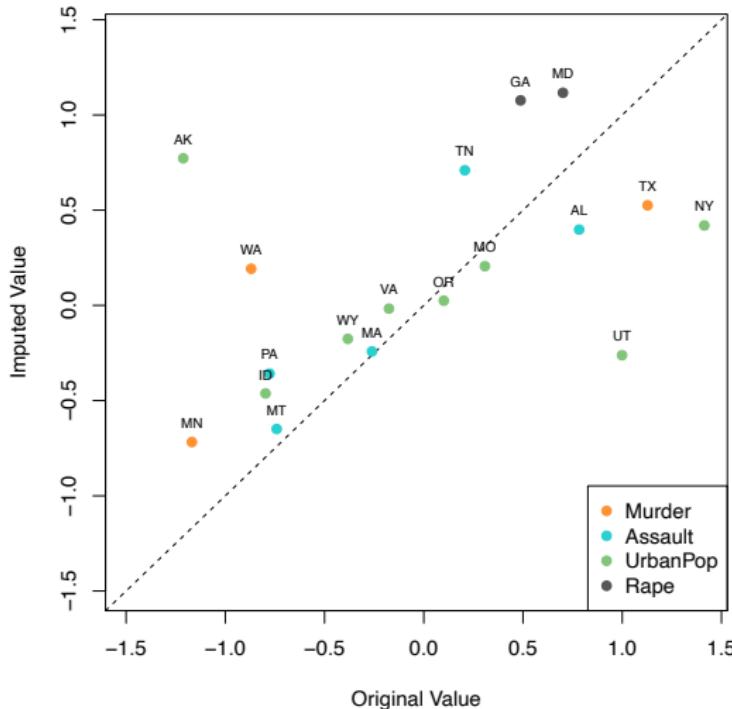
3. Return the estimated missing entries  $\tilde{x}_{ij}$ ,  $(i, j) \notin \mathcal{O}$ .

## Example: USAarrests Data



Here  $\mathbf{X}$  has 50 rows (states) and four columns: **Murder**, **Assault**, **Rape** and **UrbanPop**.

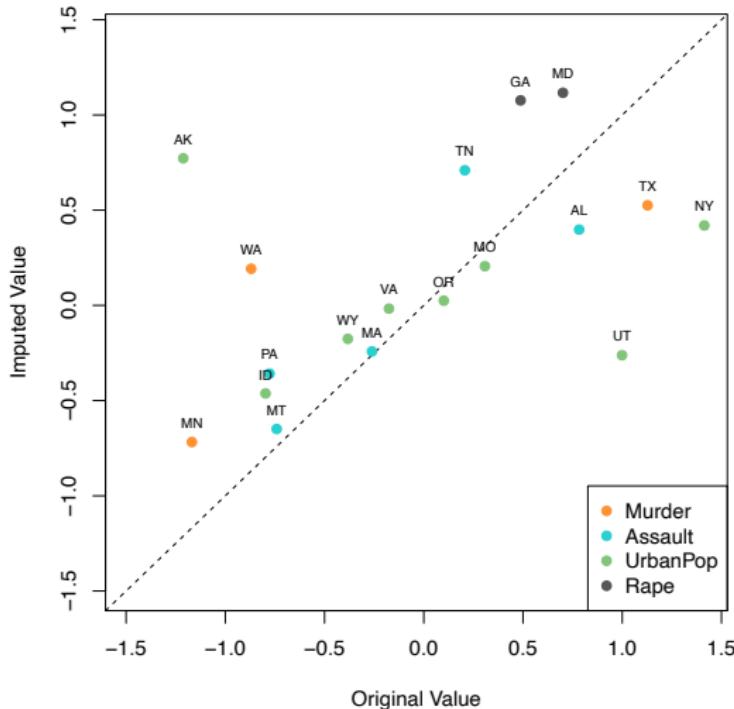
## Example: USAarrests Data



Here  $\mathbf{X}$  has 50 rows (states) and four columns: **Murder**, **Assault**, **Rape** and **UrbanPop**.

We selected 20 states at random, and for each we selected one of the variables at random, and set its value to **NA**.

## Example: USAarrests Data



Here  $\mathbf{X}$  has 50 rows (states) and four columns: **Murder**, **Assault**, **Rape** and **UrbanPop**.

We selected 20 states at random, and for each we selected one of the variables at random, and set its value to **NA**.

Used  $M = 1$  principal component in algorithm.

Correlation 0.63 between original and imputed values.

## Example — Continued

- The **USArrests** data has only four variables, which is on the low end for this method to work well. For this reason, for this demonstration we randomly set at most one variable per state to be missing, and only used  $M = 1$  principal component.

## Example — Continued

- The **USArrests** data has only four variables, which is on the low end for this method to work well. For this reason, for this demonstration we randomly set at most one variable per state to be missing, and only used  $M = 1$  principal component.
- In general, in order to apply this algorithm, we must select  $M$ , the number of principal components to use for the imputation.
- One approach is to randomly set to **NA** some elements that were actually observed, and select  $M$  based on how well those known values are recovered. This is closely related to the validation-set approach seen in Chapter 5.

## Example — Continued

- The **USArrests** data has only four variables, which is on the low end for this method to work well. For this reason, for this demonstration we randomly set at most one variable per state to be missing, and only used  $M = 1$  principal component.
- In general, in order to apply this algorithm, we must select  $M$ , the number of principal components to use for the imputation.
- One approach is to randomly set to **NA** some elements that were actually observed, and select  $M$  based on how well those known values are recovered. This is closely related to the validation-set approach seen in Chapter 5.
- **softImpute** package in R implements matrix completion algorithms, and can manage **Netflix**-scale matrices.

# Multiple Hypothesis Testing

- This session focuses on *multiple hypothesis testing*.

# Multiple Hypothesis Testing

- This session focuses on *multiple hypothesis testing*.
- A single null hypothesis might look like  $H_0$ : *the expected blood pressures of mice in the control and treatment groups are the same.*

# Multiple Hypothesis Testing

- This session focuses on *multiple hypothesis testing*.
- A single null hypothesis might look like  $H_0$ : *the expected blood pressures of mice in the control and treatment groups are the same.*
- We will now consider testing  $m$  null hypotheses,  $H_{01}, \dots, H_{0m}$ , where e.g.  $H_{0j}$ : *the expected values of the  $j^{th}$  biomarker among mice in the control and treatment groups are equal.*

# Multiple Hypothesis Testing

- This session focuses on *multiple hypothesis testing*.
- A single null hypothesis might look like  $H_0$ : *the expected blood pressures of mice in the control and treatment groups are the same.*
- We will now consider testing  $m$  null hypotheses,  $H_{01}, \dots, H_{0m}$ , where e.g.  $H_{0j}$ : *the expected values of the  $j^{\text{th}}$  biomarker among mice in the control and treatment groups are equal.*
- In this setting, we need to be careful to avoid incorrectly rejecting too many null hypotheses, i.e. having too many false positives.

## A Quick Review of Hypothesis Testing

Hypothesis tests allow us to answer simple “yes-or-no” questions, such as:

- Is the true coefficient  $\beta_j$  in a linear regression equal to zero?
- Does the expected blood pressure among mice in the treatment group equal the expected blood pressure among mice in the control group?

## A Quick Review of Hypothesis Testing

Hypothesis tests allow us to answer simple “yes-or-no” questions, such as:

- Is the true coefficient  $\beta_j$  in a linear regression equal to zero?
- Does the expected blood pressure among mice in the treatment group equal the expected blood pressure among mice in the control group?

Hypothesis testing proceeds as follows:

1. Define the null and alternative hypotheses

## A Quick Review of Hypothesis Testing

Hypothesis tests allow us to answer simple “yes-or-no” questions, such as:

- Is the true coefficient  $\beta_j$  in a linear regression equal to zero?
- Does the expected blood pressure among mice in the treatment group equal the expected blood pressure among mice in the control group?

Hypothesis testing proceeds as follows:

1. Define the null and alternative hypotheses
2. Construct the test statistic

## A Quick Review of Hypothesis Testing

Hypothesis tests allow us to answer simple “yes-or-no” questions, such as:

- Is the true coefficient  $\beta_j$  in a linear regression equal to zero?
- Does the expected blood pressure among mice in the treatment group equal the expected blood pressure among mice in the control group?

Hypothesis testing proceeds as follows:

1. Define the null and alternative hypotheses
2. Construct the test statistic
3. Compute the  $p$ -value

# A Quick Review of Hypothesis Testing

Hypothesis tests allow us to answer simple “yes-or-no” questions, such as:

- Is the true coefficient  $\beta_j$  in a linear regression equal to zero?
- Does the expected blood pressure among mice in the treatment group equal the expected blood pressure among mice in the control group?

Hypothesis testing proceeds as follows:

1. Define the null and alternative hypotheses
2. Construct the test statistic
3. Compute the  $p$ -value
4. Decide whether to reject the null hypothesis

## 1. Define the Null and Alternative Hypotheses

- We divide the world into *null* and *alternative* hypotheses.
- The null hypothesis,  $H_0$ , is the default state of belief about the world. For instance:
  1. The true coefficient  $\beta_j$  equals zero.
  2. There is no difference in the expected blood pressures.

## 1. Define the Null and Alternative Hypotheses

- We divide the world into *null* and *alternative* hypotheses.
- The null hypothesis,  $H_0$ , is the default state of belief about the world. For instance:
  1. The true coefficient  $\beta_j$  equals zero.
  2. There is no difference in the expected blood pressures.
- The alternative hypothesis,  $H_a$ , represents something different and unexpected. For instance:
  1. The true coefficient  $\beta_j$  is non-zero.
  2. There is a difference in the expected blood pressures.

## 2. Construct the Test Statistic

- The test statistic summarizes the extent to which our data are consistent with  $H_0$ .

## 2. Construct the Test Statistic

- The test statistic summarizes the extent to which our data are consistent with  $H_0$ .
- Let  $\hat{\mu}_t / \hat{\mu}_c$  respectively denote the average blood pressure for the  $n_t / n_c$  mice in the treatment and control groups.

## 2. Construct the Test Statistic

- The test statistic summarizes the extent to which our data are consistent with  $H_0$ .
- Let  $\hat{\mu}_t / \hat{\mu}_c$  respectively denote the average blood pressure for the  $n_t / n_c$  mice in the treatment and control groups.
- To test  $H_0 : \mu_t = \mu_c$ , we use a two-sample  $t$ -statistic

$$T = \frac{\hat{\mu}_t - \hat{\mu}_c}{s \sqrt{\frac{1}{n_t} + \frac{1}{n_c}}}$$

### 3. Compute the $p$ -Value

- The  $p$ -value is the probability of observing a test statistic at least as extreme as the observed statistic, *under the assumption that  $H_0$  is true.*

### 3. Compute the $p$ -Value

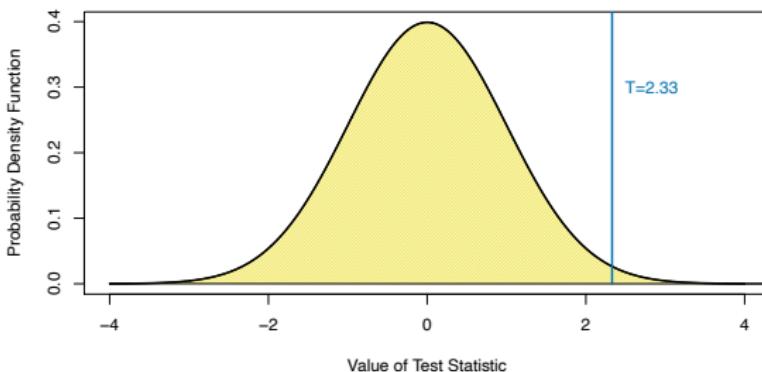
- The  $p$ -value is the probability of observing a test statistic at least as extreme as the observed statistic, *under the assumption that  $H_0$  is true.*
- A small  $p$ -value provides evidence *against*  $H_0$ .

### 3. Compute the $p$ -Value

- The  $p$ -value is the probability of observing a test statistic at least as extreme as the observed statistic, *under the assumption that  $H_0$  is true.*
- A small  $p$ -value provides evidence *against*  $H_0$ .
- Suppose we compute  $T = 2.33$  for our test of  $H_0 : \mu_t = \mu_c$ .

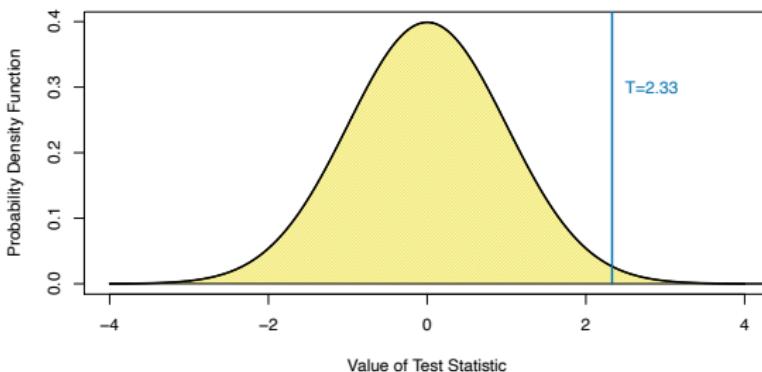
### 3. Compute the $p$ -Value

- The  $p$ -value is the probability of observing a test statistic at least as extreme as the observed statistic, *under the assumption that  $H_0$  is true*.
- A small  $p$ -value provides evidence *against*  $H_0$ .
- Suppose we compute  $T = 2.33$  for our test of  $H_0 : \mu_t = \mu_c$ .
- Under  $H_0$ ,  $T \sim N(0, 1)$  for a two-sample  $t$ -statistic.



### 3. Compute the $p$ -Value

- The  $p$ -value is the probability of observing a test statistic at least as extreme as the observed statistic, *under the assumption that  $H_0$  is true*.
- A small  $p$ -value provides evidence *against*  $H_0$ .
- Suppose we compute  $T = 2.33$  for our test of  $H_0 : \mu_t = \mu_c$ .
- Under  $H_0$ ,  $T \sim N(0, 1)$  for a two-sample  $t$ -statistic.



- The  $p$ -value is 0.02 because, if  $H_0$  is true, we would only see  $|T|$  this large 2% of the time.

## 4. Decide Whether to Reject $H_0$ , Part 1

- A small  $p$ -value indicates that such a large value of the test statistic is unlikely to occur under  $H_0$ .

## 4. Decide Whether to Reject $H_0$ , Part 1

- A small  $p$ -value indicates that such a large value of the test statistic is unlikely to occur under  $H_0$ .
- So, a small  $p$ -value provides evidence against  $H_0$ .

## 4. Decide Whether to Reject $H_0$ , Part 1

- A small  $p$ -value indicates that such a large value of the test statistic is unlikely to occur under  $H_0$ .
- So, a small  $p$ -value provides evidence against  $H_0$ .
- If the  $p$ -value is sufficiently small, then we will want to *reject*  $H_0$  (and, therefore, make a potential “discovery”).

## 4. Decide Whether to Reject $H_0$ , Part 1

- A small  $p$ -value indicates that such a large value of the test statistic is unlikely to occur under  $H_0$ .
- So, a small  $p$ -value provides evidence against  $H_0$ .
- If the  $p$ -value is sufficiently small, then we will want to *reject*  $H_0$  (and, therefore, make a potential “discovery”).
- *But how small is small enough?* To answer this, we need to understand the *Type I error*.

## 4. Decide Whether to Reject $H_0$ , Part 2

		Truth	
		$H_0$	$H_a$
Decision	Reject $H_0$	Type I Error	Correct
	Do Not Reject $H_0$	Correct	Type II Error

#### 4. Decide Whether to Reject $H_0$ , Part 2

		Truth	
		$H_0$	$H_a$
Decision	Reject $H_0$	Type I Error	Correct
	Do Not Reject $H_0$	Correct	Type II Error

#### 4. Decide Whether to Reject $H_0$ , Part 2

		The null hypothesis doesn't hold, and we rejected it!	
		$H_0$	$H_a$
Decision	Reject $H_0$	Type I Error	Correct
	Do Not Reject $H_0$	Correct	Type II Error

Truth

#### 4. Decide Whether to Reject $H_0$ , Part 2

The null hypothesis doesn't hold, and we didn't reject it!

		Truth	
		$H_0$	$H_a$
Decision	Reject $H_0$	Type I Error	Correct
	Do Not Reject $H_0$	Correct	Type II Error

#### 4. Decide Whether to Reject $H_0$ , Part 2

		Truth	
		$H_0$	$H_a$
Decision	Reject $H_0$	Type I Error	Correct
	Do Not Reject $H_0$	Correct	Type II Error

The null hypothesis holds, and we rejected it!

## 4. Decide Whether to Reject $H_0$ , Part 3

- The *Type I error rate* is the probability of making a Type I error.
- We want to ensure a small Type I error rate.

## 4. Decide Whether to Reject $H_0$ , Part 3

- The *Type I error rate* is the probability of making a Type I error.
- We want to ensure a small Type I error rate.
- If we only reject  $H_0$  when the p-value is less than  $\alpha$ , then the Type I error rate will be at most  $\alpha$ .

## 4. Decide Whether to Reject $H_0$ , Part 3

- The *Type I error rate* is the probability of making a Type I error.
- We want to ensure a small Type I error rate.
- If we only reject  $H_0$  when the p-value is less than  $\alpha$ , then the Type I error rate will be at most  $\alpha$ .
- So, *we reject  $H_0$  when the p-value falls below some  $\alpha$* : often we choose  $\alpha$  to equal 0.05 or 0.01 or 0.001.

## Multiple Testing

- Now suppose that we wish to test  $m$  null hypotheses,  $H_{01}, \dots, H_{0m}$ .

## Multiple Testing

- Now suppose that we wish to test  $m$  null hypotheses,  $H_{01}, \dots, H_{0m}$ .
- Can we simply reject all null hypotheses for which the corresponding  $p$ -value falls below (say) 0.01?

## Multiple Testing

- Now suppose that we wish to test  $m$  null hypotheses,  $H_{01}, \dots, H_{0m}$ .
- Can we simply reject all null hypotheses for which the corresponding  $p$ -value falls below (say) 0.01?
- If we reject all null hypotheses for which the  $p$ -value falls below 0.01, then how many Type I errors will we make?

## A Thought Experiment

- Suppose that we flip a fair coin ten times, and we wish to test  $H_0$ : *the coin is fair.*

## A Thought Experiment

- Suppose that we flip a fair coin ten times, and we wish to test  $H_0$ : *the coin is fair.*
  - We'll probably get approximately the same number of heads and tails.
  - The p-value probably won't be small. We do not reject  $H_0$ .

## A Thought Experiment

- Suppose that we flip a fair coin ten times, and we wish to test  $H_0$ : *the coin is fair.*
  - We'll probably get approximately the same number of heads and tails.
  - The p-value probably won't be small. We do not reject  $H_0$ .
- But what if we flip 1,024 fair coins ten times each?

## A Thought Experiment

- Suppose that we flip a fair coin ten times, and we wish to test  $H_0$ : *the coin is fair.*
  - We'll probably get approximately the same number of heads and tails.
  - The p-value probably won't be small. We do not reject  $H_0$ .
- But what if we flip 1,024 fair coins ten times each?
  - We'd expect one coin (on average) to come up all tails.

## A Thought Experiment

- Suppose that we flip a fair coin ten times, and we wish to test  $H_0$ : *the coin is fair.*
  - We'll probably get approximately the same number of heads and tails.
  - The p-value probably won't be small. We do not reject  $H_0$ .
- But what if we flip 1,024 fair coins ten times each?
  - We'd expect one coin (on average) to come up all tails.
  - The p-value for the null hypothesis that this particular coin is fair is less than 0.002!
  - So we would conclude it is not fair, i.e. we *reject  $H_0$* , even though it's a fair coin.

# A Thought Experiment

- Suppose that we flip a fair coin ten times, and we wish to test  $H_0$ : *the coin is fair.*
  - We'll probably get approximately the same number of heads and tails.
  - The p-value probably won't be small. We do not reject  $H_0$ .
- But what if we flip 1,024 fair coins ten times each?
  - We'd expect one coin (on average) to come up all tails.
  - The p-value for the null hypothesis that this particular coin is fair is less than 0.002!
  - So we would conclude it is not fair, i.e. we *reject  $H_0$* , even though it's a fair coin.
- If we test a lot of hypotheses, we are almost certain to get one very small p-value by chance!

# Multiple Testing: Even XKCD Weighs In



<https://xkcd.com/882/>

# The Challenge of Multiple Testing

- Suppose we test  $H_{01}, \dots, H_{0m}$ , all of which are true, and reject any null hypothesis with a p-value below 0.01.

# The Challenge of Multiple Testing

- Suppose we test  $H_{01}, \dots, H_{0m}$ , all of which are true, and reject any null hypothesis with a p-value below 0.01.
- Then we expect to falsely reject approximately  $0.01 \times m$  null hypotheses.

# The Challenge of Multiple Testing

- Suppose we test  $H_{01}, \dots, H_{0m}$ , all of which are true, and reject any null hypothesis with a p-value below 0.01.
- Then we expect to falsely reject approximately  $0.01 \times m$  null hypotheses.
- If  $m = 10,000$ , then we expect to falsely reject 100 null hypotheses by chance!

# The Challenge of Multiple Testing

- Suppose we test  $H_{01}, \dots, H_{0m}$ , all of which are true, and reject any null hypothesis with a p-value below 0.01.
- Then we expect to falsely reject approximately  $0.01 \times m$  null hypotheses.
- If  $m = 10,000$ , then we expect to falsely reject 100 null hypotheses by chance!
- *That's a lot of Type I errors, i.e. false positives!*

## The Family-Wise Error Rate

- The family-wise error rate (FWER) is the probability of making *at least one* Type I error when conducting  $m$  hypothesis tests.

## The Family-Wise Error Rate

- The family-wise error rate (FWER) is the probability of making *at least one* Type I error when conducting  $m$  hypothesis tests.
- $\text{FWER} = \Pr(V \geq 1)$

	$H_0$ is True	$H_0$ is False	Total
Reject $H_0$	$V$	$S$	$R$
Do Not Reject $H_0$	$U$	$W$	$m - R$
Total	$m_0$	$m - m_0$	$m$

## Challenges in Controlling the Family-Wise Error Rate

$$\begin{aligned}\text{FWER} &= 1 - \Pr(\text{do not falsely reject any null hypotheses}) \\ &= 1 - \Pr\left(\bigcap_{j=1}^m \{\text{do not falsely reject } H_{0j}\}\right).\end{aligned}$$

## Challenges in Controlling the Family-Wise Error Rate

$$\begin{aligned}\text{FWER} &= 1 - \Pr(\text{do not falsely reject any null hypotheses}) \\ &= 1 - \Pr\left(\bigcap_{j=1}^m \{\text{do not falsely reject } H_{0j}\}\right).\end{aligned}$$

If the tests are independent and all  $H_{0j}$  are true then

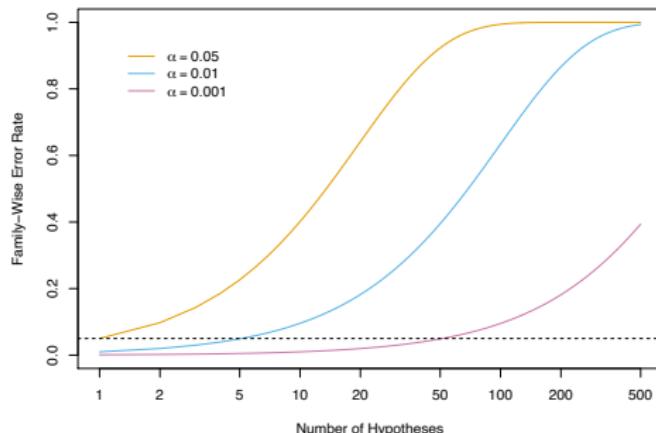
$$\text{FWER} = 1 - \prod_{j=1}^m (1 - \alpha) = 1 - (1 - \alpha)^m.$$

# Challenges in Controlling the Family-Wise Error Rate

$$\begin{aligned}\text{FWER} &= 1 - \Pr(\text{do not falsely reject any null hypotheses}) \\ &= 1 - \Pr\left(\bigcap_{j=1}^m \{\text{do not falsely reject } H_{0j}\}\right).\end{aligned}$$

If the tests are independent and all  $H_{0j}$  are true then

$$\text{FWER} = 1 - \prod_{j=1}^m (1 - \alpha) = 1 - (1 - \alpha)^m.$$



## The Bonferroni Correction

$$\begin{aligned}\text{FWER} &= \Pr(\text{falsely reject at least one null hypothesis}) \\ &= \Pr(\cup_{j=1}^m A_j) \\ &\leq \sum_{j=1}^m \Pr(A_j)\end{aligned}$$

where  $A_j$  is the event that we falsely reject the  $j$ th null hypothesis.

## The Bonferroni Correction

$$\begin{aligned}\text{FWER} &= \Pr(\text{falsely reject at least one null hypothesis}) \\ &= \Pr(\cup_{j=1}^m A_j) \\ &\leq \sum_{j=1}^m \Pr(A_j)\end{aligned}$$

where  $A_j$  is the event that we falsely reject the  $j$ th null hypothesis.

- If we only reject hypotheses when the p-value is less than  $\alpha/m$ , then

$$\text{FWER} \leq \sum_{j=1}^m \Pr(A_j) \leq \sum_{j=1}^m \frac{\alpha}{m} = m \times \frac{\alpha}{m} = \alpha,$$

because  $\Pr(A_j) \leq \alpha/m$ .

- This is the *Bonferroni Correction*: to control FWER at level  $\alpha$ , reject any null hypothesis with  $p$ -value below  $\alpha/m$ .

## Fund Manager Data

Manager	Mean, $\bar{x}$	$s$	$t$ -statistic	$p$ -value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

## Fund Manager Data

Manager	Mean, $\bar{x}$	$s$	$t$ -statistic	$p$ -value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

- $H_{0j}$ : the  $j$ th manager's expected excess return equals zero.
- If we reject  $H_{0j}$  if the p-value is less than  $\alpha = 0.05$ , then we will conclude that the *first* and *third* managers have significantly non-zero excess returns.

## Fund Manager Data

Manager	Mean, $\bar{x}$	$s$	$t$ -statistic	$p$ -value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

- $H_{0j}$ : the  $j$ th manager's expected excess return equals zero.
- If we reject  $H_{0j}$  if the p-value is less than  $\alpha = 0.05$ , then we will conclude that the *first* and *third* managers have significantly non-zero excess returns.
- However, we have tested multiple hypotheses, so the FWER is *greater* than 0.05.

## Fund Manager Data with Bonferroni Correction

Manager	Mean, $\bar{x}$	$s$	$t$ -statistic	$p$ -value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

- Using a Bonferroni correction, we reject for p-values less than  $\alpha/m = 0.05/5 = 0.01$ .

## Fund Manager Data with Bonferroni Correction

Manager	Mean, $\bar{x}$	$s$	$t$ -statistic	$p$ -value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

- Using a Bonferroni correction, we reject for p-values less than  $\alpha/m = 0.05/5 = 0.01$ .
- Consequently, we will reject the null hypothesis only for the *first* manager.

## Fund Manager Data with Bonferroni Correction

Manager	Mean, $\bar{x}$	$s$	$t$ -statistic	$p$ -value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

- Using a Bonferroni correction, we reject for  $p$ -values less than  $\alpha/m = 0.05/5 = 0.01$ .
- Consequently, we will reject the null hypothesis only for the *first* manager.
- Now the FWER is at most 0.05.

## Holm's Method for Controlling the FWER

## Holm's Method for Controlling the FWER

1. Compute  $p$ -values,  $p_1, \dots, p_m$ , for the  $m$  null hypotheses  $H_{01}, \dots, H_{0m}$ .

## Holm's Method for Controlling the FWER

1. Compute  $p$ -values,  $p_1, \dots, p_m$ , for the  $m$  null hypotheses  $H_{01}, \dots, H_{0m}$ .
2. Order the  $m$   $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .

## Holm's Method for Controlling the FWER

1. Compute  $p$ -values,  $p_1, \dots, p_m$ , for the  $m$  null hypotheses  $H_{01}, \dots, H_{0m}$ .
2. Order the  $m$   $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .
3. Define

$$L = \min \left\{ j : p_{(j)} > \frac{\alpha}{m + 1 - j} \right\}.$$

## Holm's Method for Controlling the FWER

1. Compute  $p$ -values,  $p_1, \dots, p_m$ , for the  $m$  null hypotheses  $H_{01}, \dots, H_{0m}$ .
2. Order the  $m$   $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .
3. Define

$$L = \min \left\{ j : p_{(j)} > \frac{\alpha}{m + 1 - j} \right\}.$$

4. Reject all null hypotheses  $H_{0j}$  for which  $p_j < p_{(L)}$ .

## Holm's Method for Controlling the FWER

1. Compute  $p$ -values,  $p_1, \dots, p_m$ , for the  $m$  null hypotheses  $H_{01}, \dots, H_{0m}$ .
2. Order the  $m$   $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .
3. Define

$$L = \min \left\{ j : p_{(j)} > \frac{\alpha}{m + 1 - j} \right\}.$$

4. Reject all null hypotheses  $H_{0j}$  for which  $p_j < p_{(L)}$ .
- Holm's method controls the FWER at level  $\alpha$ .

## Holm's Method on the Fund Manager Data

Manager	Mean, $\bar{x}$	$s$	t-statistic	p-value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

- The ordered  $p$ -values are  $p_{(1)} = 0.006$ ,  $p_{(2)} = 0.012$ ,  $p_{(3)} = 0.601$ ,  $p_{(4)} = 0.756$  and  $p_{(5)} = 0.918$ .

## Holm's Method on the Fund Manager Data

Manager	Mean, $\bar{x}$	$s$	t-statistic	p-value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

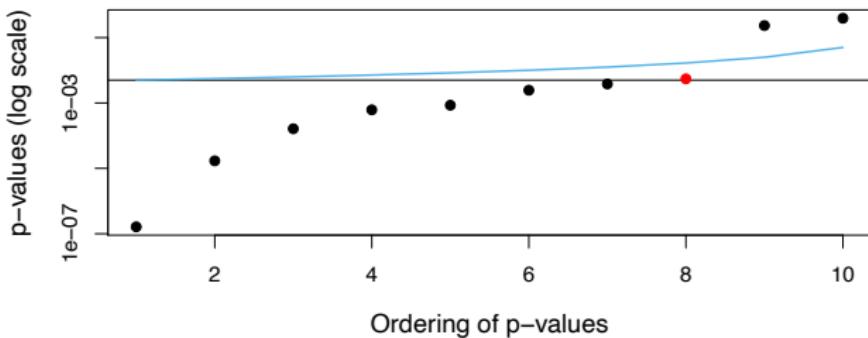
- The ordered  $p$ -values are  $p_{(1)} = 0.006$ ,  $p_{(2)} = 0.012$ ,  $p_{(3)} = 0.601$ ,  $p_{(4)} = 0.756$  and  $p_{(5)} = 0.918$ .
- The Holm procedure rejects the first two null hypotheses, because
  - $p_{(1)} = 0.006 < 0.05/(5 + 1 - 1) = 0.0100$
  - $p_{(2)} = 0.012 < 0.05/(5 + 1 - 2) = 0.0125$ ,
  - $p_{(3)} = 0.601 > 0.05/(5 + 1 - 3) = 0.0167$ .

## Holm's Method on the Fund Manager Data

Manager	Mean, $\bar{x}$	$s$	t-statistic	p-value
One	3.0	7.4	2.86	0.006
Two	-0.1	6.9	-0.10	0.918
Three	2.8	7.5	2.62	0.012
Four	0.5	6.7	0.53	0.601
Five	0.3	6.8	0.31	0.756

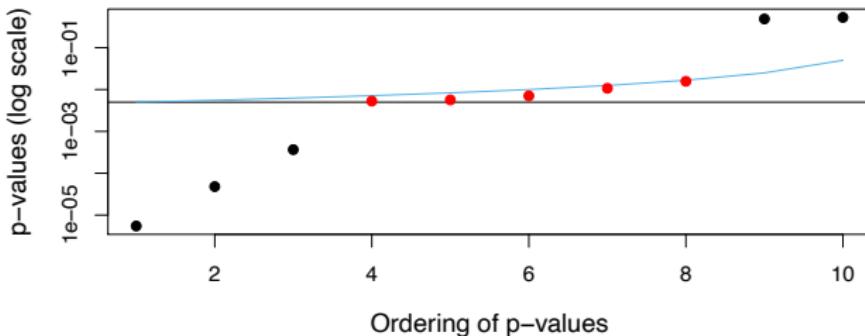
- The ordered  $p$ -values are  $p_{(1)} = 0.006$ ,  $p_{(2)} = 0.012$ ,  $p_{(3)} = 0.601$ ,  $p_{(4)} = 0.756$  and  $p_{(5)} = 0.918$ .
- The Holm procedure rejects the first two null hypotheses, because
  - $p_{(1)} = 0.006 < 0.05/(5 + 1 - 1) = 0.0100$
  - $p_{(2)} = 0.012 < 0.05/(5 + 1 - 2) = 0.0125$ ,
  - $p_{(3)} = 0.601 > 0.05/(5 + 1 - 3) = 0.0167$ .
- Holm rejects  $H_0$  for the *first* and *third* managers, but Bonferroni only rejects  $H_0$  for the *first* manager.

## A Comparison with $m = 10$ p-values



- Aim to control FWER at 0.05.
- p-values below the black horizontal line are rejected by Bonferroni.
- p-values below the blue line are rejected by Holm.
- Holm and Bonferroni make the same conclusion on the black points, but only Holm rejects for the red point.

## A More Extreme Example



- Now five hypotheses are rejected by Holm but not by Bonferroni ....
- .... even though both control FWER at 0.05.

## Holm or Bonferroni?

- Bonferroni is simple ... reject any null hypothesis with a p-value below  $\alpha/m$ .
- Holm is slightly more complicated, but it will lead to more rejections while controlling FWER!!
- So, *Holm is a better choice!*

## Other Methods

- There are lots of specialized approaches to control FWER.

## Other Methods

- There are lots of specialized approaches to control FWER.
- For example:
  - *Tukey's Method*: for pairwise comparisons of the difference in expected means among a number of groups.

## Other Methods

- There are lots of specialized approaches to control FWER.
- For example:
  - *Tukey's Method*: for pairwise comparisons of the difference in expected means among a number of groups.
  - *Scheffé's Method*: for testing arbitrary linear combinations of a set of expected means, e.g.

$$H_0 : \frac{1}{2} (\mu_1 + \mu_3) = \frac{1}{3} (\mu_2 + \mu_4 + \mu_5).$$

## Other Methods

- There are lots of specialized approaches to control FWER.
- For example:
  - *Tukey's Method*: for pairwise comparisons of the difference in expected means among a number of groups.
  - *Scheffé's Method*: for testing arbitrary linear combinations of a set of expected means, e.g.

$$H_0 : \frac{1}{2} (\mu_1 + \mu_3) = \frac{1}{3} (\mu_2 + \mu_4 + \mu_5).$$

- Bonferroni and Holm are general procedures that will work in most settings. However, in certain special cases, methods such as Tukey and Scheffé can give better results: *i.e. more rejections while maintaining FWER control.*

# The False Discovery Rate

## The False Discovery Rate

- Back to this table:

	$H_0$ is True	$H_0$ is False	Total
Reject $H_0$	$V$	$S$	$R$
Do Not Reject $H_0$	$U$	$W$	$m - R$
Total	$m_0$	$m - m_0$	$m$

## The False Discovery Rate

- Back to this table:

	$H_0$ is True	$H_0$ is False	Total
Reject $H_0$	$V$	$S$	$R$
Do Not Reject $H_0$	$U$	$W$	$m - R$
Total	$m_0$	$m - m_0$	$m$

- The FWER rate focuses on controlling  $\Pr(V > 1)$ , i.e., the probability of falsely rejecting *any* null hypothesis.

## The False Discovery Rate

- Back to this table:

	$H_0$ is True	$H_0$ is False	Total
Reject $H_0$	$V$	$S$	$R$
Do Not Reject $H_0$	$U$	$W$	$m - R$
Total	$m_0$	$m - m_0$	$m$

- The FWER rate focuses on controlling  $\Pr(V > 1)$ , i.e., the probability of falsely rejecting *any* null hypothesis.
- This is a tough ask when  $m$  is large! It will cause us to be super conservative (i.e. to very rarely reject).

## The False Discovery Rate

- Back to this table:

	$H_0$ is True	$H_0$ is False	Total
Reject $H_0$	$V$	$S$	$R$
Do Not Reject $H_0$	$U$	$W$	$m - R$
Total	$m_0$	$m - m_0$	$m$

- The FWER rate focuses on controlling  $\Pr(V > 1)$ , i.e., the probability of falsely rejecting *any* null hypothesis.
- This is a tough ask when  $m$  is large! It will cause us to be super conservative (i.e. to very rarely reject).
- Instead, we can control the *false discovery rate*:

$$\text{FDR} = \mathbb{E}(V/R).$$

## Intuition Behind the False Discovery Rate

$$\text{FDR} = E\left(\frac{V}{R}\right) = E\left(\frac{\text{number of false rejections}}{\text{total number of rejections}}\right)$$

## Intuition Behind the False Discovery Rate

$$\text{FDR} = \text{E} \left( \frac{V}{R} \right) = \text{E} \left( \frac{\text{number of false rejections}}{\text{total number of rejections}} \right)$$

- A scientist conducts a hypothesis test on each of  $m = 20,000$  drug candidates.

## Intuition Behind the False Discovery Rate

$$\text{FDR} = \text{E} \left( \frac{V}{R} \right) = \text{E} \left( \frac{\text{number of false rejections}}{\text{total number of rejections}} \right)$$

- A scientist conducts a hypothesis test on each of  $m = 20,000$  drug candidates.
- She wants to identify a smaller set of promising candidates to investigate further.

## Intuition Behind the False Discovery Rate

$$\text{FDR} = \text{E} \left( \frac{V}{R} \right) = \text{E} \left( \frac{\text{number of false rejections}}{\text{total number of rejections}} \right)$$

- A scientist conducts a hypothesis test on each of  $m = 20,000$  drug candidates.
- She wants to identify a smaller set of promising candidates to investigate further.
- She wants reassurance that this smaller set is really “promising”, i.e. not too many falsely rejected  $H_0$ ’s.

## Intuition Behind the False Discovery Rate

$$\text{FDR} = \text{E} \left( \frac{V}{R} \right) = \text{E} \left( \frac{\text{number of false rejections}}{\text{total number of rejections}} \right)$$

- A scientist conducts a hypothesis test on each of  $m = 20,000$  drug candidates.
- She wants to identify a smaller set of promising candidates to investigate further.
- She wants reassurance that this smaller set is really “promising”, i.e. not too many falsely rejected  $H_0$ ’s.
- FWER controls  $\text{Pr}(\text{at least one false rejection})$ .

## Intuition Behind the False Discovery Rate

$$\text{FDR} = \text{E} \left( \frac{V}{R} \right) = \text{E} \left( \frac{\text{number of false rejections}}{\text{total number of rejections}} \right)$$

- A scientist conducts a hypothesis test on each of  $m = 20,000$  drug candidates.
- She wants to identify a smaller set of promising candidates to investigate further.
- She wants reassurance that this smaller set is really “promising”, i.e. not too many falsely rejected  $H_0$ ’s.
- FWER controls  $\text{Pr}(\text{at least one false rejection})$ .
- FDR controls the fraction of candidates in the smaller set that are really false rejections. This is what she needs!

## Benjamini-Hochberg Procedure to Control FDR

## Benjamini-Hochberg Procedure to Control FDR

1. Specify  $q$ , the level at which to control the FDR.

## Benjamini-Hochberg Procedure to Control FDR

1. Specify  $q$ , the level at which to control the FDR.
2. Compute  $p$ -values  $p_1, \dots, p_m$  for the null hypotheses  $H_{01}, \dots, H_{0m}$ .

## Benjamini-Hochberg Procedure to Control FDR

1. Specify  $q$ , the level at which to control the FDR.
2. Compute  $p$ -values  $p_1, \dots, p_m$  for the null hypotheses  $H_{01}, \dots, H_{0m}$ .
3. Order the  $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .

## Benjamini-Hochberg Procedure to Control FDR

1. Specify  $q$ , the level at which to control the FDR.
2. Compute  $p$ -values  $p_1, \dots, p_m$  for the null hypotheses  $H_{01}, \dots, H_{0m}$ .
3. Order the  $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .
4. Define  $L = \max \{j : p_{(j)} < qj/m\}$ .

## Benjamini-Hochberg Procedure to Control FDR

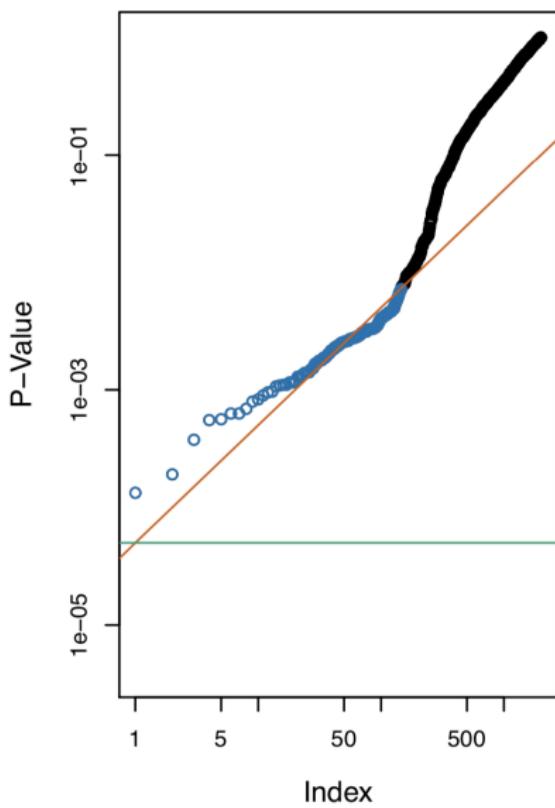
1. Specify  $q$ , the level at which to control the FDR.
2. Compute  $p$ -values  $p_1, \dots, p_m$  for the null hypotheses  $H_{01}, \dots, H_{0m}$ .
3. Order the  $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .
4. Define  $L = \max \{j : p_{(j)} < qj/m\}$ .
5. Reject all null hypotheses  $H_{0j}$  for which  $p_j \leq p_{(L)}$ .

## Benjamini-Hochberg Procedure to Control FDR

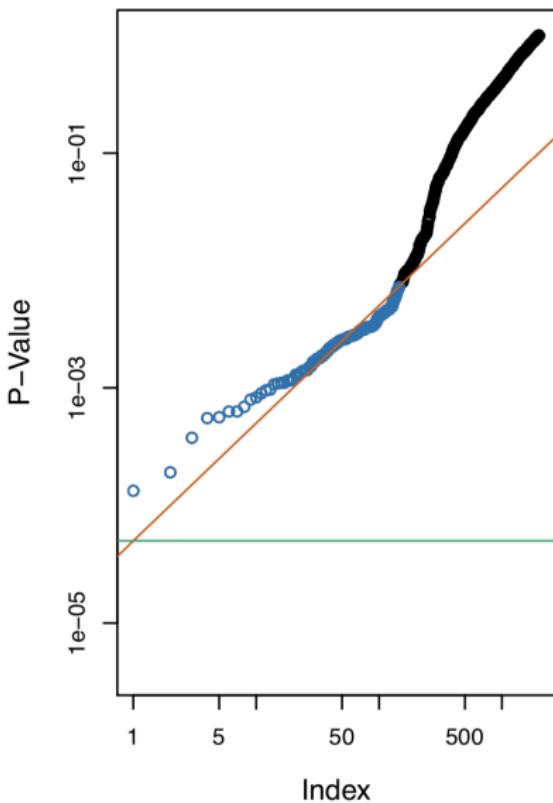
1. Specify  $q$ , the level at which to control the FDR.
2. Compute  $p$ -values  $p_1, \dots, p_m$  for the null hypotheses  $H_{01}, \dots, H_{0m}$ .
3. Order the  $p$ -values so that  $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$ .
4. Define  $L = \max \{j : p_{(j)} < qj/m\}$ .
5. Reject all null hypotheses  $H_{0j}$  for which  $p_j \leq p_{(L)}$ .

Then,  $\text{FDR} \leq q$ .

# A Comparison of FDR Versus FWER, Part 1

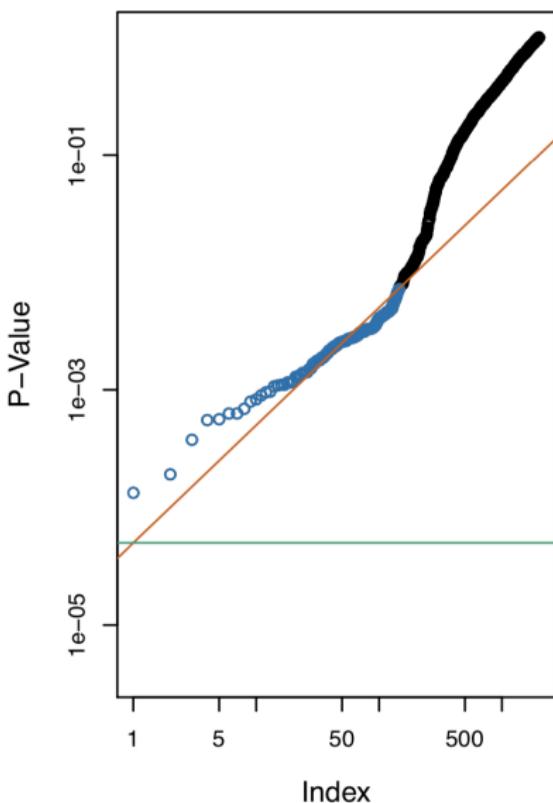


# A Comparison of FDR Versus FWER, Part 1



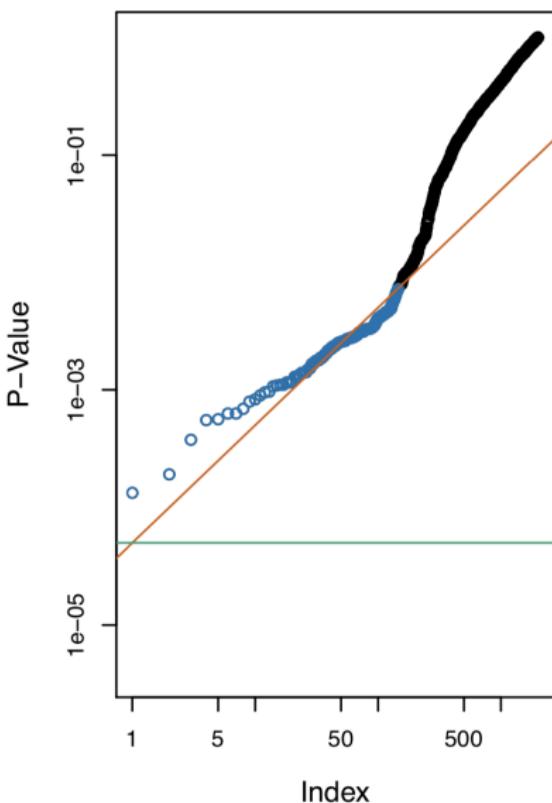
- Here,  $p$ -values for  $m = 2,000$  null hypotheses are displayed.

# A Comparison of FDR Versus FWER, Part 1



- Here,  $p$ -values for  $m = 2,000$  null hypotheses are displayed.
- To control FWER at level  $\alpha = 0.1$  with Bonferroni: reject hypotheses below green line. (*No rejections!*)

# A Comparison of FDR Versus FWER, Part 1



- Here,  $p$ -values for  $m = 2,000$  null hypotheses are displayed.
- To control FWER at level  $\alpha = 0.1$  with Bonferroni: reject hypotheses below green line. (*No rejections!*)
- To control FDR at level  $q = 0.1$  with Benjamini-Hochberg: reject hypotheses shown in blue.

# A Comparison of FDR Versus FWER, Part 2

## A Comparison of FDR Versus FWER, Part 2

- Consider  $m = 5$   $p$ -values from the *Fund* data:  
 $p_1 = 0.006, p_2 = 0.918, p_3 = 0.012, p_4 = 0.601, p_5 = 0.756.$

## A Comparison of FDR Versus FWER, Part 2

- Consider  $m = 5$   $p$ -values from the *Fund* data:  
 $p_1 = 0.006, p_2 = 0.918, p_3 = 0.012, p_4 = 0.601, p_5 = 0.756$ .
- Then  $p_{(1)} = 0.006, p_{(2)} = 0.012, p_{(3)} = 0.601, p_{(4)} = 0.756$ ,  
and  $p_{(5)} = 0.918$ .

## A Comparison of FDR Versus FWER, Part 2

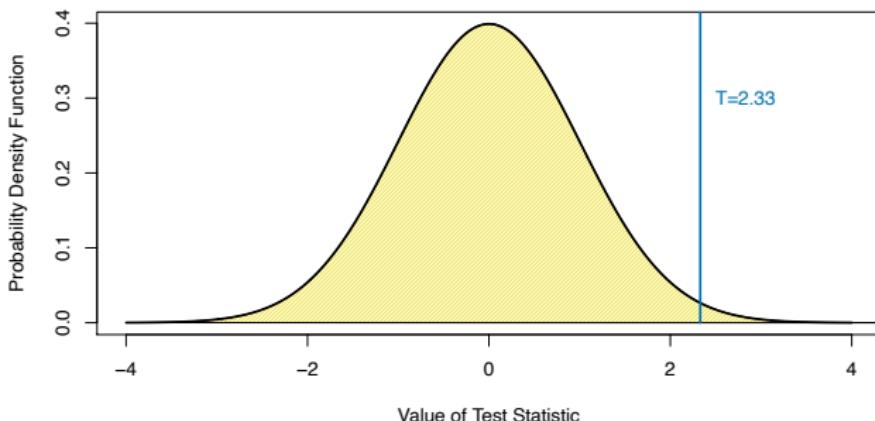
- Consider  $m = 5$   $p$ -values from the *Fund* data:  
 $p_1 = 0.006, p_2 = 0.918, p_3 = 0.012, p_4 = 0.601, p_5 = 0.756$ .
- Then  $p_{(1)} = 0.006, p_{(2)} = 0.012, p_{(3)} = 0.601, p_{(4)} = 0.756$ ,  
and  $p_{(5)} = 0.918$ .
- To control FDR at level  $q = 0.05$  using  
Benjamini-Hochberg:
  - Notice that  $p_{(1)} < 0.05/5, p_{(2)} < 2 \times 0.05/5$ ,  
 $p_{(3)} > 3 \times 0.05/5, p_{(4)} > 4 \times 0.05/5$ , and  $p_{(5)} > 5 \times 0.05/5$ .
  - So, we reject  $H_{01}$  and  $H_{03}$ .

## A Comparison of FDR Versus FWER, Part 2

- Consider  $m = 5$   $p$ -values from the *Fund* data:  
 $p_1 = 0.006, p_2 = 0.918, p_3 = 0.012, p_4 = 0.601, p_5 = 0.756$ .
- Then  $p_{(1)} = 0.006, p_{(2)} = 0.012, p_{(3)} = 0.601, p_{(4)} = 0.756$ , and  $p_{(5)} = 0.918$ .
- To control FDR at level  $q = 0.05$  using Benjamini-Hochberg:
  - Notice that  $p_{(1)} < 0.05/5, p_{(2)} < 2 \times 0.05/5, p_{(3)} > 3 \times 0.05/5, p_{(4)} > 4 \times 0.05/5$ , and  $p_{(5)} > 5 \times 0.05/5$ .
  - So, we reject  $H_{01}$  and  $H_{03}$ .
- To control FWER at level  $\alpha = 0.05$  using Bonferroni:
  - We reject any null hypothesis for which the  $p$ -value is less than  $0.05/5$ .
  - So, we reject only  $H_{01}$ .

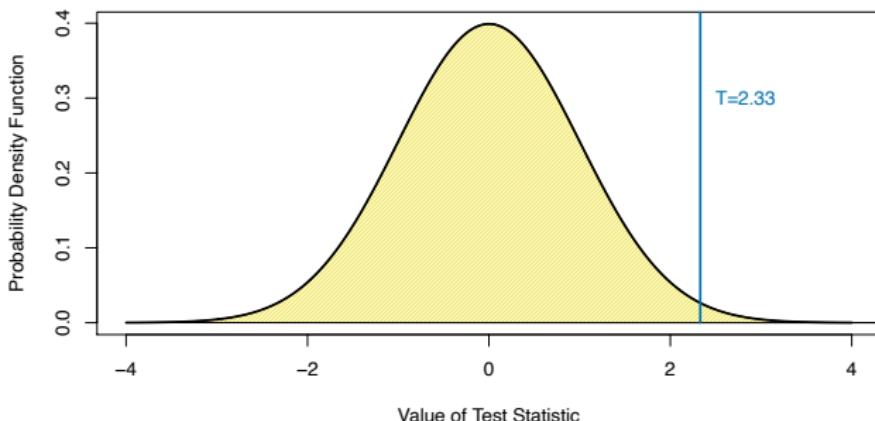
## Re-Sampling Approaches

- So far, we have assumed that we want to test some null hypothesis  $H_0$  with some test statistic  $T$ , and that we know (or can assume) the distribution of  $T$  under  $H_0$ .
- This allows us to compute the  $p$ -value.



## Re-Sampling Approaches

- So far, we have assumed that we want to test some null hypothesis  $H_0$  with some test statistic  $T$ , and that we know (or can assume) the distribution of  $T$  under  $H_0$ .
- This allows us to compute the  $p$ -value.



- What if this *theoretical null distribution* is unknown?

# A Re-Sampling Approach for a Two-Sample t-Test, Part 1

- Suppose we want to test  $H_0 : E(X) = E(Y)$  versus  $H_a : E(X) \neq E(Y)$ , using  $n_X$  independent observations from  $X$  and  $n_Y$  independent observations from  $Y$ .
- The two-sample t-statistic takes the form

$$T = \frac{\hat{\mu}_X - \hat{\mu}_Y}{s\sqrt{1/n_X + 1/n_Y}}.$$

# A Re-Sampling Approach for a Two-Sample t-Test, Part 1

- Suppose we want to test  $H_0 : E(X) = E(Y)$  versus  $H_a : E(X) \neq E(Y)$ , using  $n_X$  independent observations from  $X$  and  $n_Y$  independent observations from  $Y$ .
- The two-sample t-statistic takes the form

$$T = \frac{\hat{\mu}_X - \hat{\mu}_Y}{s\sqrt{1/n_X + 1/n_Y}}.$$

- If  $n_X$  and  $n_Y$  are large, then  $T$  approximately follows a  $N(0, 1)$  distribution under  $H_0$ .

# A Re-Sampling Approach for a Two-Sample t-Test, Part 1

- Suppose we want to test  $H_0 : E(X) = E(Y)$  versus  $H_a : E(X) \neq E(Y)$ , using  $n_X$  independent observations from  $X$  and  $n_Y$  independent observations from  $Y$ .
- The two-sample t-statistic takes the form

$$T = \frac{\hat{\mu}_X - \hat{\mu}_Y}{s\sqrt{1/n_X + 1/n_Y}}.$$

- If  $n_X$  and  $n_Y$  are large, then  $T$  approximately follows a  $N(0, 1)$  distribution under  $H_0$ .
- If  $n_X$  and  $n_Y$  are small, then we don't know the theoretical null distribution of  $T$ .

# A Re-Sampling Approach for a Two-Sample t-Test, Part 1

- Suppose we want to test  $H_0 : E(X) = E(Y)$  versus  $H_a : E(X) \neq E(Y)$ , using  $n_X$  independent observations from  $X$  and  $n_Y$  independent observations from  $Y$ .
- The two-sample t-statistic takes the form

$$T = \frac{\hat{\mu}_X - \hat{\mu}_Y}{s\sqrt{1/n_X + 1/n_Y}}.$$

- If  $n_X$  and  $n_Y$  are large, then  $T$  approximately follows a  $N(0, 1)$  distribution under  $H_0$ .
- If  $n_X$  and  $n_Y$  are small, then we don't know the theoretical null distribution of  $T$ .
- Let's take a *permutation* or *re-sampling* approach....

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

1. Compute the two-sample  $t$ -statistic  $T$  on the original data  $x_1, \dots, x_{n_X}$  and  $y_1, \dots, y_{n_Y}$ .

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

1. Compute the two-sample  $t$ -statistic  $T$  on the original data  $x_1, \dots, x_{n_X}$  and  $y_1, \dots, y_{n_Y}$ .
2. For  $b = 1, \dots, B$  (where  $B$  is a large number, like 1,000):

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

1. Compute the two-sample  $t$ -statistic  $T$  on the original data  $x_1, \dots, x_{n_X}$  and  $y_1, \dots, y_{n_Y}$ .
2. For  $b = 1, \dots, B$  (where  $B$  is a large number, like 1,000):
  - 2.1 Randomly shuffle the  $n_x + n_Y$  observations.

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

1. Compute the two-sample  $t$ -statistic  $T$  on the original data  $x_1, \dots, x_{n_X}$  and  $y_1, \dots, y_{n_Y}$ .
2. For  $b = 1, \dots, B$  (where  $B$  is a large number, like 1,000):
  - 2.1 Randomly shuffle the  $n_x + n_Y$  observations.
  - 2.2 Call the first  $n_X$  shuffled observations  $x_1^*, \dots, x_{n_X}^*$  and call the remaining observations  $y_1^*, \dots, y_{n_Y}^*$ .

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

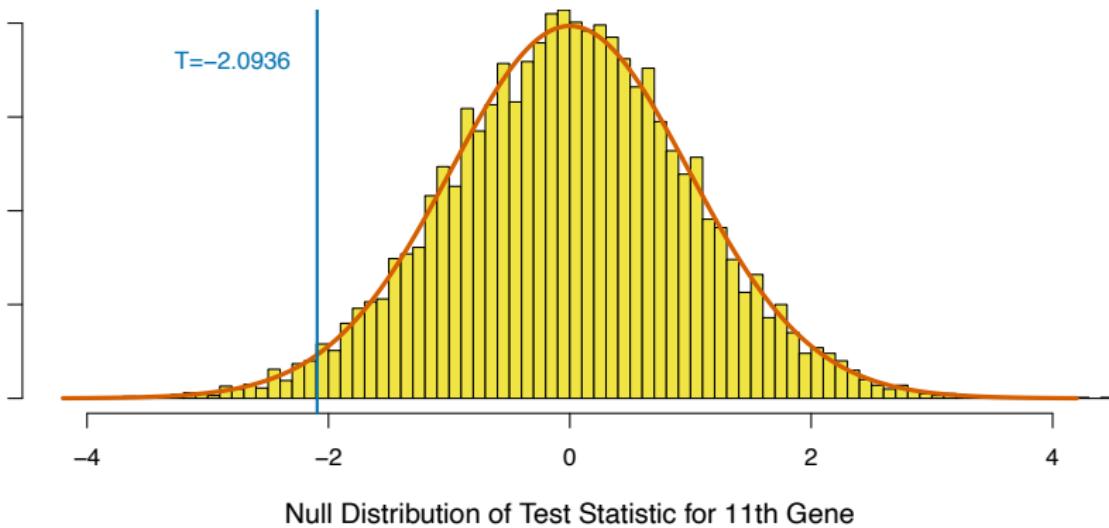
1. Compute the two-sample  $t$ -statistic  $T$  on the original data  $x_1, \dots, x_{n_X}$  and  $y_1, \dots, y_{n_Y}$ .
2. For  $b = 1, \dots, B$  (where  $B$  is a large number, like 1,000):
  - 2.1 Randomly shuffle the  $n_x + n_Y$  observations.
  - 2.2 Call the first  $n_X$  shuffled observations  $x_1^*, \dots, x_{n_X}^*$  and call the remaining observations  $y_1^*, \dots, y_{n_Y}^*$ .
  - 2.3 Compute a two-sample  $t$ -statistic on the shuffled data, and call it  $T^{*b}$ .

# A Re-Sampling Approach for a Two-Sample t-Test, Part 2

1. Compute the two-sample  $t$ -statistic  $T$  on the original data  $x_1, \dots, x_{n_X}$  and  $y_1, \dots, y_{n_Y}$ .
2. For  $b = 1, \dots, B$  (where  $B$  is a large number, like 1,000):
  - 2.1 Randomly shuffle the  $n_x + n_Y$  observations.
  - 2.2 Call the first  $n_X$  shuffled observations  $x_1^*, \dots, x_{n_X}^*$  and call the remaining observations  $y_1^*, \dots, y_{n_Y}^*$ .
  - 2.3 Compute a two-sample  $t$ -statistic on the shuffled data, and call it  $T^{*b}$ .
3. The  $p$ -value is given by

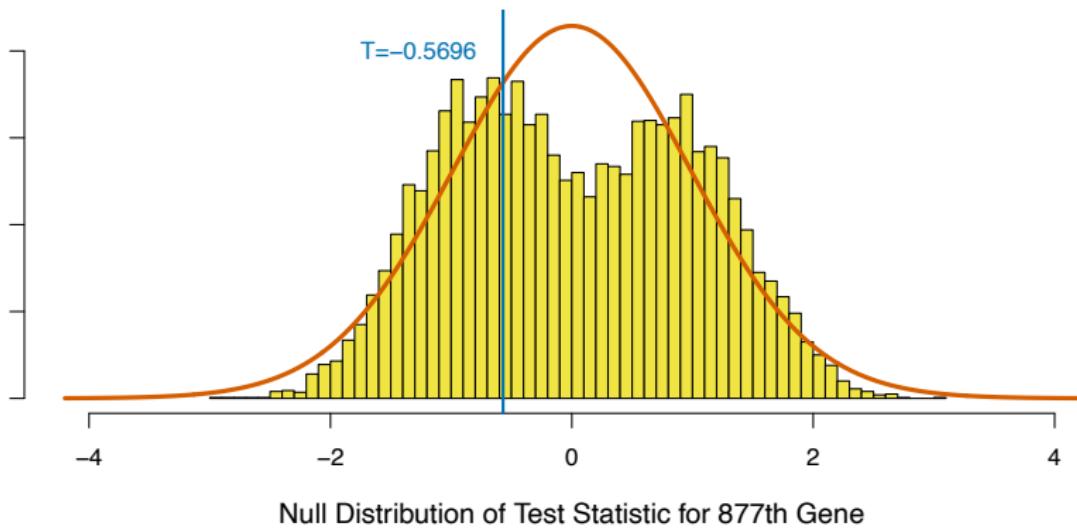
$$\frac{\sum_{b=1}^B \mathbf{1}_{(|T^{*b}| \geq |T|)}}{B}.$$

## Application to Gene Expression Data, Part 1



Theoretical  $p$ -value is 0.041. Re-sampling  $p$ -value is 0.042.

## Application to Gene Expression Data, Part 2



Theoretical  $p$ -value is 0.571. Re-sampling  $p$ -value is 0.673.

## More on Re-Sampling Approaches

## More on Re-Sampling Approaches

- Re-sampling approaches are useful if the theoretical null distribution is unavailable, or requires stringent assumptions. (*So, they're always useful!*)

## More on Re-Sampling Approaches

- Re-sampling approaches are useful if the theoretical null distribution is unavailable, or requires stringent assumptions. (*So, they're always useful!*)
- An extension of the re-sampling approach to compute a  $p$ -value can be used to control FDR.

## More on Re-Sampling Approaches

- Re-sampling approaches are useful if the theoretical null distribution is unavailable, or requires stringent assumptions. (*So, they're always useful!*)
- An extension of the re-sampling approach to compute a  $p$ -value can be used to control FDR.
- This example involved a two-sample  $t$ -test, but similar approaches can be developed for other test statistics.