

EECS E6690: Statistical Learning for Biological and Information Systems

Lecture 1: Introduction

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm
303 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

E6690 Statistical Learning: Brief Description

- ▶ **Deluge of Data in Biology and Information Systems:** Ongoing advancements in information systems as well as the emerging revolution in microbiology and neuroscience are creating a deluge of data, whose mining, inference and prediction will have an enormous economic, social, scientific and medical/therapeutic impact.
- ▶ **Biology:** For example, in biology, microarray technology is creating vast amounts of gene expression data, whose understanding could lead to better diagnostics and cures of diseases, e.g., cancer.
Personalized medicine: designing treatments for individual patients.
- ▶ **Information Systems:** Similarly, in information systems, companies like Google, Amazon, Facebook, etc., are facing various problems on massive data sets, e.g., ranking, association and community detection.

E6690 Statistical Learning: Brief Description

This course will cover a variety of fundamental statistical (machine) learning techniques that are suitable for the emerging problems in these application areas, but also applicable in general

- ▶ Basics of Statistics and Optimization
- ▶ Introduction to Statistical/Machine Learning Techniques
 - ▶ Supervised versus unsupervised learning
 - ▶ Inference and prediction
 - ▶ Linear versus nonlinear models
 - ▶ Training, testing and validation
 - ▶ Regularization
 - ▶ And many more
- ▶ Specifics of Biological and Information Systems Data
 - ▶ High dimensionality and need for regularization
 - ▶ Large sparse graphs
 - ▶ Community detection
 - ▶ Ranking
 - ▶ Association rules (Market basket analysis)

E6690 Statistical Learning: Course Logistics

Prerequisites: Calculus. Some knowledge of probability/statistics and optimization is strongly encouraged, but not required. Familiarity with a programming language, say Matlab, is highly desirable.

Textbooks: The following two books will represent the supporting references for the course. The books are available online:

- ESL** Hastie, T., Tibshirani, R. and Friedman, J.
The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd Edition.
Springer, 2009. <https://hastie.su.domains/Papers/ESLII.pdf>.
- ISL** James, G., Witten, D. Hastie, T. and Tibshirani, R. An Introduction to Statistical Learning,
Springer, 2014. Available online at <https://www.statlearning.com>. R code can be found at:
<https://hastie.su.domains/ISLR2/Labs/>

In addition, lecture notes as well as occasionally other books and research papers will be used.

Homework: Biweekly homework will be assigned (about 3-4)

Programming: The course uses R language. Pointers to its free download and resources, as well as basic examples of programming in R will be covered in class.

Grading: Homework (20%) + Midterm (35%) + Final Proj (45%).

E6690 Statistical Learning: Course Logistics

Midterm: In class, closed book; 2 page cheat-sheet allowed; 2 1/2 hours

- ▶ Mixture of problem solving and descriptive answers
(This might change since the course is online.)

Final Project: Done in groups of 3 (maybe 4) students

- ▶ First, select a paper(s) from a data repository, e.g.:
 - ▶ GEO (Gene Expression Omnibus) Data Repository
<https://www.ncbi.nlm.nih.gov/geo/>
 - ▶ UC Irvine Machine Learning Repository
<https://archive.ics.uci.edu/ml/datasets.php>
- ▶ General Project Outline
 1. **Introduction:** e.g., describe the application area, problems considered, etc
 2. **Data set(s) and paper(s):** e.g., describe data in detail, what was done in the paper(s), common stat/machine learning tools, etc
 3. **Reproduce the results from the paper(s)**
 4. **Try different techniques learned in class, or propose new ones**
 5. **Discussion and conclusion:** e.g., compare different techniques, pros and cons, future work, etc

Statistical Learning: What Does It Involve?

In general, Statistical (Machine) Learning (supervised) problems typically can be posed as

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

where ϵ is the noise.

Problem: Estimate f from training data $\{(x_i, y_i)\}$, and then use it as a general solution. Typically: $y \in \mathbb{R}$: **regression**; or y -discrete: **classification**
Two main setups:

- ▶ Noiseless case ($Y = f(X)$): more common in machine learning
- ▶ Noisy case ($Y = f(X) + \epsilon$): more prevalent in statistics

Areas involved:

- ▶ **Approximation theory** - for picking a class of functions
- ▶ **Optimization** - for fitting the training data
- ▶ **Computing** - fitting and testing
- ▶ **Probability and Statistics** - testing, error estimation

Machine Learning Versus Classical Programming

Interesting Question: What is the difference between classical programming and statistical/machine learning?

$$Y = f(X)$$

- ▶ Classical Programming: f is an algorithm designed by a person
- ▶ Statistical Learning: f is discovered through examples by training

General Course Objectives

- ▶ Focus/motivation - emerging applications in:
 - ▶ Biology and Medicine
 - ▶ Information Technology, e.g., problems arising from: Google, Facebook, Twitter, Amazon, etc.
- ▶ Learn fundamental concepts and techniques in statistical (machine) learning techniques that are
 - ▶ Suitable for these application areas
 - ▶ Useful and applicable in general
- ▶ Develop the necessary knowledge as we go (e.g., Statistics, Optimization, Approximation Theory, etc)
- ▶ Learn R
- ▶ Have a hands-on experience on a real, practical problem through a final project

Overall objective: **Become an expert in Statistical/Machine Learning**

Programming in R: Computing Platform

- ▶ Language and environment for statistical computing and graphics
- ▶ Free software
- ▶ Download
 - ▶ R from <http://cran.r-project.org/>
 - ▶ RStudio, an Integrated Development Environment for R, from <http://www.rstudio.com/products/rstudio/download/>
- ▶ Resources
 - ▶ R for beginners
 - ▶ Quick-R
 - ▶ Cookbook for R
 - ▶ R for Data Science
 - ▶ Try R

Brief Statistics Review

Crash Course in Undergraduate Statistics

Example

The following numbers are particle (contamination) counts for a sample of 10 semiconductor silicon wafers:

50 48 44 56 61 52 53 55 67 51

Over a long run the process average for wafer particle counts has been 50 counts per wafer, and on the basis of the sample, we want to test whether a change has occurred.

Is data consistent with a given hypothesis?

- ▶ Idea: Data → estimate with a known distribution
(Estimates are not unique)
- ▶ Is the estimate consistent the hypothesized distribution?
How likely is the estimate?

Estimates

- ▶ A statistic is a property of sample data taken from a population
- ▶ A point estimate of some unknown parameter is a statistic that provides a best guess at the parameter value
- ▶ A point estimate $\hat{\theta}$ is **unbiased** if $\mathbb{E}\hat{\theta} = \theta$
- ▶ X_1, X_2, \dots, X_n – i.i.d. with population mean μ & variance σ^2
- ▶ Examples

- ▶ Sample mean - estimate of the population mean μ

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

- ▶ Sample variance - estimate of the population variance σ^2

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$$

- ▶ Variability: $\text{Var}(\bar{X}) = \sigma^2/n$, but σ unknown
replace σ^2/n with *standard error (SE)*, $\text{SE}(\bar{X})^2 := S^2/n$
 $\Rightarrow \text{Var}(\bar{X}) = \sigma^2/n \approx \text{SE}(\bar{X})^2$

Variability of estimates: Known variance

- If X_1, \dots, X_n are **i.i.d. normal**, then

- \bar{X} is normal:

$$\frac{\bar{X} - \mu}{\sqrt{\sigma^2/n}} \sim \mathcal{N}(0, 1)$$

- S^2 has a known distribution:

$$\frac{n-1}{\sigma^2} S^2 \sim \chi_{n-1}^2,$$

where χ_{n-1}^2 (Chi - square) is a random variable whose distribution is equal to the sum of $(n - 1)$ squares of independent standard normal random variables

- \bar{X} and S^2 are independent (prove)
- If X_1, \dots, X_n are **not** i.i.d normal, then CLT:

$$\frac{\bar{X} - \mu}{\sqrt{\sigma^2/n}} \Rightarrow \mathcal{N}(0, 1)$$

Variability of estimates: Unknown variance

- ▶ If X_1, \dots, X_n are **i.i.d. normal**, then
 - ▶ t -statistic:

$$\frac{\bar{X} - \mu}{\sqrt{S^2/n}} \sim \frac{\mathcal{N}(0, 1)}{\sqrt{\chi_{n-1}^2/(n-1)}} \sim t_{n-1},$$

t_{n-1} is Student's t -distribution with $(n-1)$ degrees of freedom
Developed by William Gosset; worked for Guinness brewery,
published the paper under the pen name Student

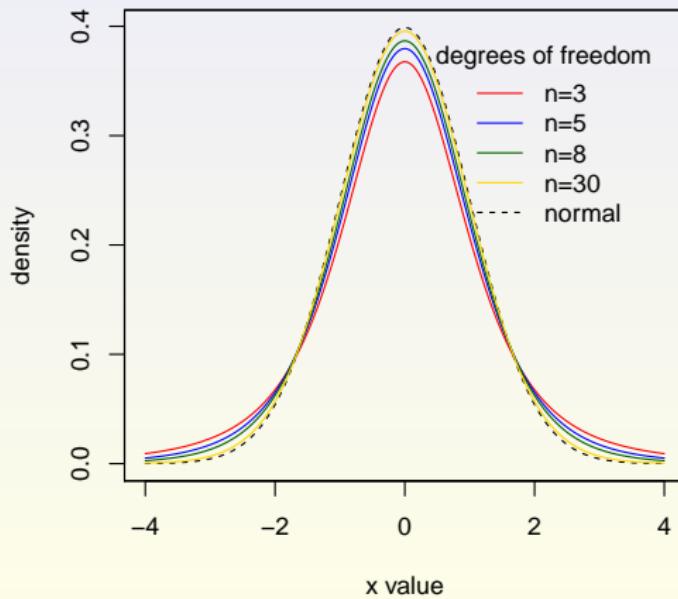
- ▶ Representation of t_n : Let $Z \sim \mathcal{N}(0, 1)$ and $V \sim \chi_n^2$ be independent

$$\frac{Z}{\sqrt{V/n}} \sim t_n$$

t-distribution

- ▶ Zero mean
- ▶ Variance ($n > 2$): $n/(n - 2)$

PDFs of t distributions



t-test

- ▶ Null hypothesis $\mathcal{H}_0 : \mu = \mu_0$
- ▶ Under \mathcal{H}_0 , *t*-statistic:

$$t = \frac{\bar{X} - \mu_0}{\sqrt{S^2/n}} \sim t_{n-1}$$

and the corresponding *p-value* is the probability of observing $|t_{n-1}|$ that is $\geq |t|$, i.e., $p = \mathbb{P}[|t_{n-1}| \geq |t|]$.

- ▶ Large values of *t* unlikely under \mathcal{H}_0
- ▶ Typically:
 - ▶ pick a significance value, say $\alpha = 0.05$ (not unique)
 - ▶ reject if $p < \alpha$, say $p < 0.05$
 - ▶ accept if $p \geq \alpha$, say $p \geq 0.05$

Intro to Statistical Learning

Supervised vs. unsupervised learning

- ▶ **Supervised learning:** there is an input-output relationship

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

- ▶ $X \in \mathbb{R}^p$ - Vector of p predictor measurements
- ▶ $Y \in \mathbb{R}$ - Outcome measurements
- ▶ ϵ : noise
- ▶ Two problems:
 - ▶ Regression: Y is quantitative/real
 - ▶ Classification: Y is categorical/discrete
- ▶ Training data (observations): $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- ▶ Objectives:
 - ▶ Statistics: Prediction, inference
 - ▶ Machine learning: Solve a problem via training
- ▶ **Unsupervised learning:** No outcome variable Y
 - ▶ Objective can be vague - just exploring data
 - ▶ Learn interesting phenomena in data, e.g.:
 - ▶ Clustering, community detection, data association, low dimensional representation

Learning

- ▶ Let $Y \in \mathbb{R}$ be the output variable, and $X \in \mathbb{R}^p$ the input vector $X = (X_1, X_2, \dots, X_p)$. Then

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

- ▶ Want to estimate what f is
- ▶ ϵ is unavoidable noise that is independent of X , zero mean
- ▶ How to estimate f from the data? How to evaluate the estimate?
- ▶ Given an estimate \hat{f} for f , predict unavailable values of Y for known values of X : $\hat{Y} = \hat{f}(X)$
- ▶ Reducible and irreducible errors:
 - ▶ \hat{f} is not exactly f , but f can potentially be learnt given enough data
 - ▶ even if f is known, there is error: $\epsilon = Y - f(X)$

Two approaches to estimate f

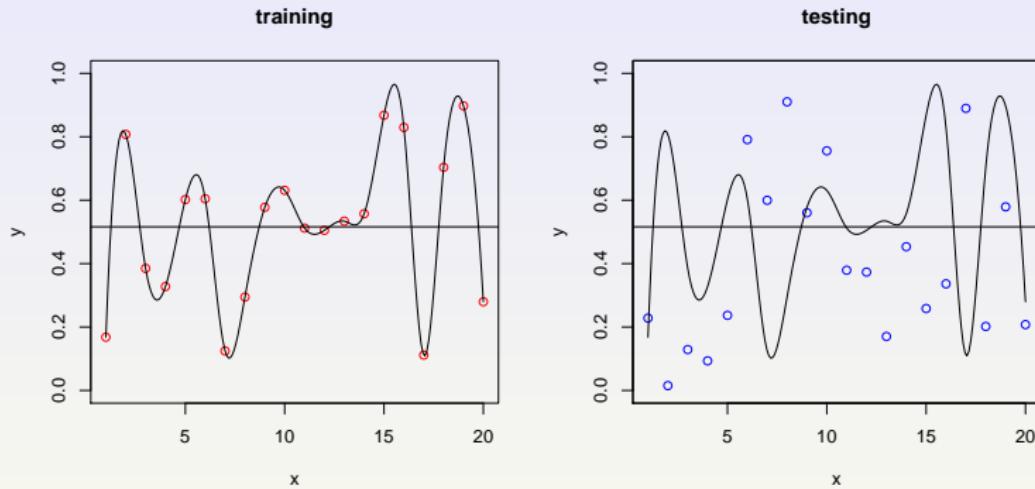
- ▶ Parametric
 - ▶ Assume a specific form of f
 - ▶ Example: the linear model

$$\hat{f}(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

- ▶ Use training data to choose the values of parameters $\beta_0, \beta_1, \dots, \beta_p$
- ▶ Pro: easier to estimate parameters than arbitrary function
- ▶ Con: the choice of f might be (very) wrong

- ▶ Non-parametric
 - ▶ Make the parametric form more flexible
 - ▶ This makes \hat{f} more complex and potentially following the noise too closely, thereby **overfitting**
 - ▶ Get f as close as possible to the data points, subject to not being too non-smooth
 - ▶ Pro: more likely to get f right, especially if f is “strange”
 - ▶ Con: more data is needed to obtain a good estimate for f

Example



- ▶ More complicated models not always better - e.g., **overfitting**
John von Neumann: "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."
(Deep learning: 50,000,000 parameters (!))
- ▶ Amount of available data
- ▶ Interpretability

Linear Regression

Idea

- ▶ Simple approach to supervised learning
- ▶ Assumes linear dependence of quantitative Y on X_1, X_2, \dots, X_p
- ▶ True regression functions are never linear!
 - ▶ But most learning methods linear in parameters (β -s)!
- ▶ Extremely useful both conceptually and practically

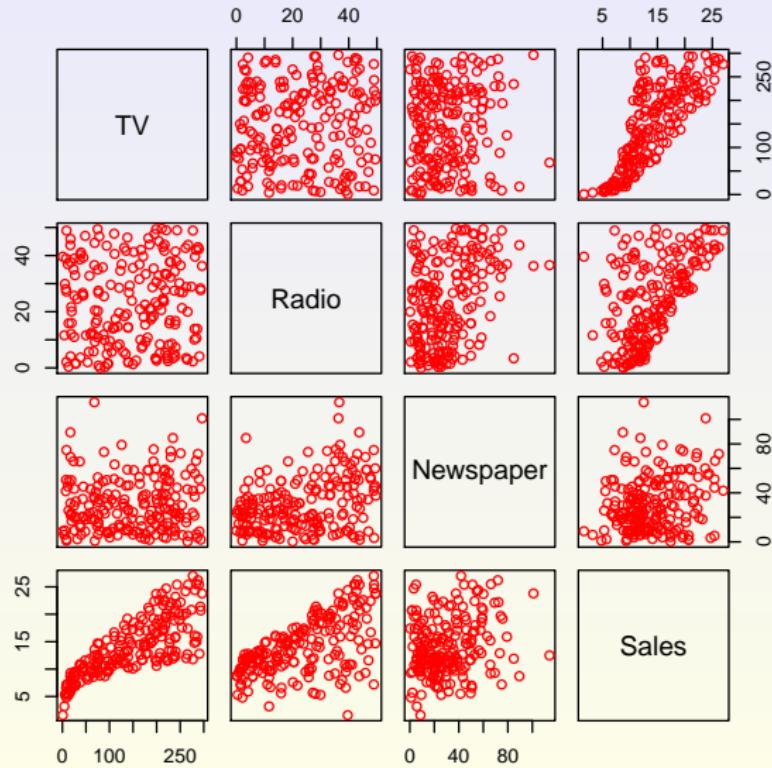
Data set

- ▶ Will use Advertising.csv to illustrate concepts
- ▶ 200 observations:

```
","", "TV", "Radio", "Newspaper", "Sales"  
"1", 230.1, 37.8, 69.2, 22.1  
"2", 44.5, 39.3, 45.1, 10.4  
"3", 17.2, 45.9, 69.3, 9.3  
. . .  
"198", 177, 9.3, 6.4, 12.8  
"199", 283.6, 42, 66.2, 25.5  
"200", 232.1, 8.6, 8.7, 13.4
```

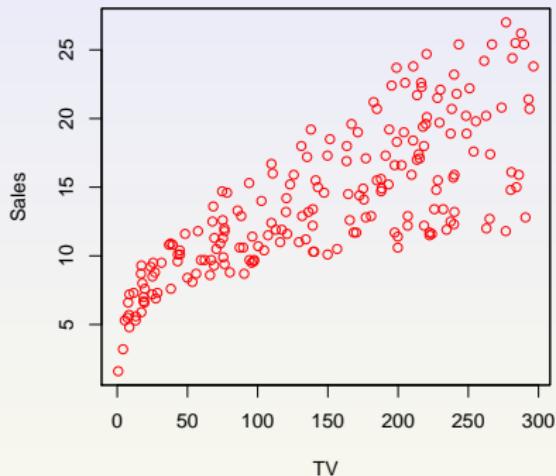
Advertising data set

Visualize data - whenever possible



Single predictor: TV vs. Sales

```
> adv<-read.csv("advertising.csv",header=TRUE,sep=",")  
> plot(adv$TV,adv$Sales,xlab="TV",ylab="Sales",col="red")
```



- ▶ Linear model

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

where

- ▶ β_0 and β_1 : unknown constants/parameters/coefficients (intercept and slope)
- ▶ ϵ : error term

Single predictor: Model selection

- ▶ Estimate β_0 and β_1 based on data
- ▶ Given estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, predict future sales using

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

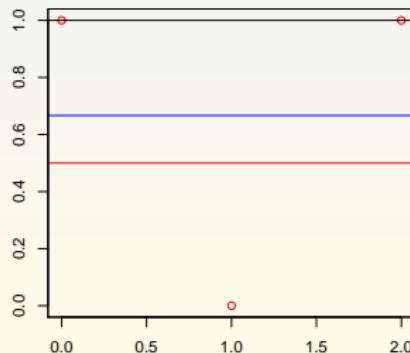
- ▶ \hat{y} : prediction of Y given $X = x$
- ▶ **Residuals:** $y_i - \hat{y}_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$
- ▶ Select $\hat{\beta}_0$ and $\hat{\beta}_1$ to “minimize” residuals
- ▶ How to minimize a vector?

Need to Define Distance: Vector norms

- ▶ Example: l_p norm

$$\|z\|_p = \left(\sum_{i=1}^n |z_i|^p \right)^{1/p}$$

- ▶ Example: 3 data point - $\{(0, 1), (1, 0), (2, 1)\}$
The result depends on the choice of the norm (!)
(parallel to x -axis due to symmetry)



One dimensional l_2 regression: Least squares

- ▶ $\min \|\mathbf{y} - \hat{\mathbf{y}}\|_2$
- ▶ Residual Sum of Squares (RSS):

$$\text{RSS} \equiv \text{RSS}(\beta_0, \beta_1) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- ▶ Least squares approach: $\min_{\beta_0, \beta_1} \text{RSS}$
- ▶ Solution:

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x},\end{aligned}$$

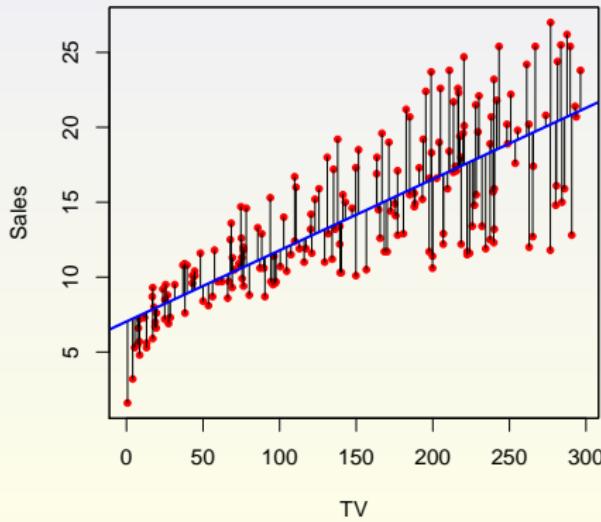
where $\bar{x} = n^{-1} \sum_{i=1}^n x_i$ and $\bar{y} = n^{-1} \sum_{i=1}^n y_i$ are the sample means

Example

```
> lm1<-lm(adv$Sales~adv$TV)
> summary(lm1)
```

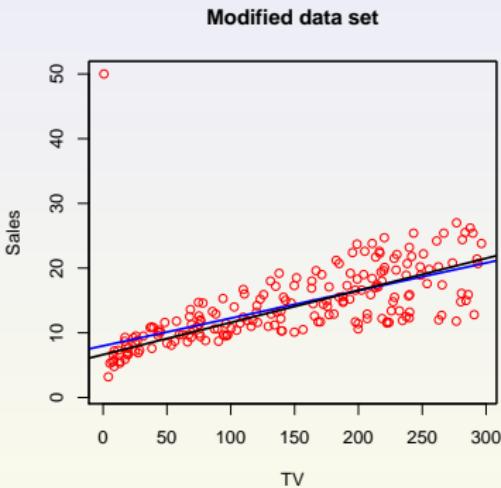
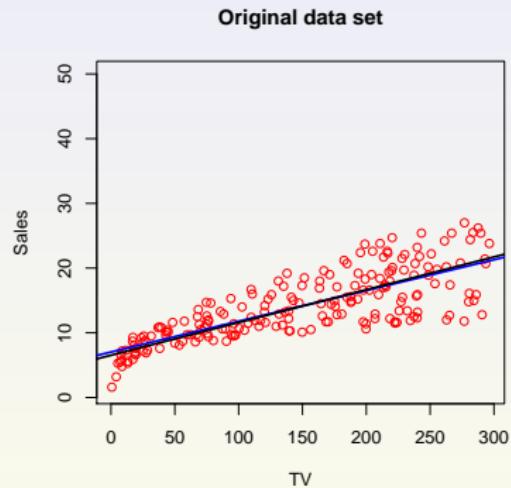
$$\text{Sales} = 7.032594 + 0.047537 \times \text{TV}$$

```
> plot(adv$TV,adv$Sales,xlab="TV",ylab="Sales",col="red",pch=20)
> abline(lm(adv$Sales~adv$TV),col="blue",lwd=2)
> Sales_Predict<-predict(lm1)
> segments(adv$TV, adv$Sales, adv$TV, Sales_Predict)
```



Example: l_2 vs. l_1

- ▶ One point in the data set modified

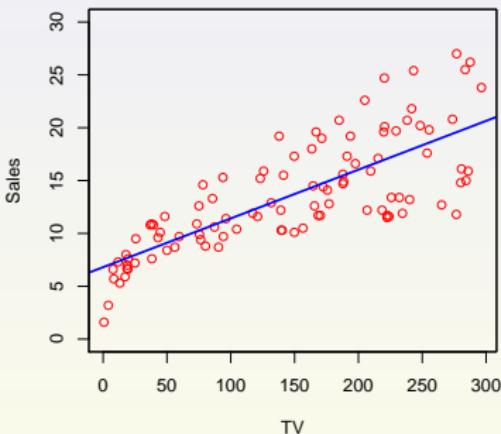
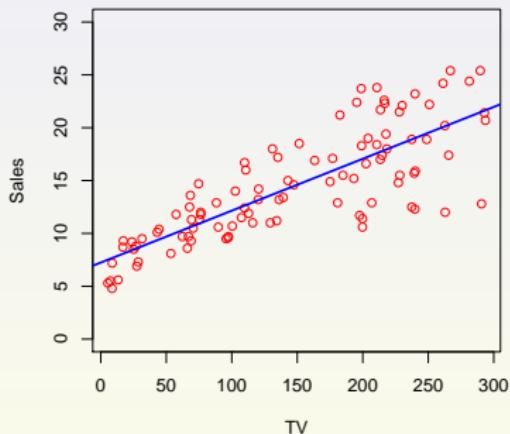


Coefficient estimates

- ▶ Suppose the true model is

$$\text{Sales} = \beta_0 + \beta_1 \times \text{TV} + \epsilon$$

- ▶ How good are estimates $\hat{\beta}_0$ and $\hat{\beta}_1$?



$$i = 1, \dots, 100 : \quad \text{Sales} = 7.241734 + 0.049069 \times \text{TV}$$

$$i = 101, \dots, 200 : \quad \text{Sales} = 6.803818 + 0.046135 \times \text{TV}$$

Properties of $\hat{\beta}_0$ and $\hat{\beta}_1$

- ▶ Repeated sampling
- ▶ $\hat{\beta}_0$ and $\hat{\beta}_1$ vary
- ▶ Means:

$$\mathbb{E}\hat{\beta}_0 = \beta_0 \quad \text{and} \quad \mathbb{E}\hat{\beta}_1 = \beta_1$$

- ▶ Variances:

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

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

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

- ▶ An estimate of σ^2 :

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

where RSE is the Residual Standard Error

Confidence intervals

- ▶ Normality assumption: $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- ▶ t -statistic:

$$\frac{\hat{\beta}_1 - \beta_1}{\text{SE}(\hat{\beta}_1)} \sim t_{n-2},$$

where

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

- ▶ $(1 - \gamma)$ confidence interval (example $\gamma = 5\%$, $1 - \gamma = 95\%$)

$$[\hat{\beta}_1 - \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}, \hat{\beta}_1 + \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}]$$

is such that

$$\mathbb{P}[\beta_1 \in [\hat{\beta}_1 - \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}, \hat{\beta}_1 + \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}]] = 1 - \gamma,$$

where $t_{\gamma/2, n-2}$ is the $(1 - \gamma/2)$ -th quantile of the t_{n-2} distribution

Hypothesis testing

- ▶ Typical testing (null vs. alternative hypothesis):

\mathcal{H}_0 : there is no relationship between X and Y
versus alternative

\mathcal{H}_A : there is some relationship between X and Y

- ▶ Formally:

$$\mathcal{H}_0 : \beta_1 = 0 \quad \text{vs.} \quad \mathcal{H}_A : \beta_1 \neq 0$$

- ▶ To test \mathcal{H}_0 ($\beta_1 = 0$), compute a t -statistic:

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

which is distributed according to a t -distribution with $(n - 2)$ degrees of freedom

- ▶ Compute the p -value – probability of observing any value equal to $|t|$ or larger

Example

```
> summary(lm1)
```

Call:

```
lm(formula = adv$Sales ~ adv$TV)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.3860	-1.9545	-0.1913	2.0671	7.2124

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	7.032594	0.457843	15.36	<2e-16 ***
adv\$TV	0.047537	0.002691	17.67	<2e-16 ***

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Residual standard error: 3.259 on 198 degrees of freedom

Multiple R-squared: 0.6119, Adjusted R-squared: 0.6099

F-statistic: 312.1 on 1 and 198 DF, p-value: < 2.2e-16

```
> qt(0.975,198)
[1] 1.972017
```

Reading:

ISL: Read in detail Chapter 2 and Section 3.1.

Also, looking through the entire Chapters 1-3 is recommended.

EECS E6690: Statistical Learning for Biological and Information Systems

Lecture 2: Multiple Linear Regression

Prof. Predrag R. Jelenković

Time: Tuesday 4:10-6:40pm

303 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA

Office: 812 Schapiro Research Bldg.

Phone: (212) 854-8174

Email: predrag@ee.columbia.edu

URL: <http://www.ee.columbia.edu/~predrag>

Last lecture: Intro to stat learning

Supervised vs. Unsupervised learning

Supervised:

- ▶ Let Y be the output variable, and X the input vector $X = (X_1, X_2, \dots, X_p)$. Then

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

- ▶ Want to estimate f
- ▶ ϵ is unavoidable/irreducible noise that is independent of X , zero mean
- ▶ How to estimate f from the data? How to evaluate the estimate?
- ▶ Errors: irreducible, reducible, bias
- ▶ Overfitting and testing

John von Neumann on overfitting: "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."

Unsupervised: No $f(\cdot)/Y$, just X

Last lecture: Estimation and Testing

Estimation:

- ▶ Select a class of function for f : Hypothesis class \mathcal{H}
say, \mathcal{H} are linear functions, i.e., linear regression
- ▶ Select as distance metric, i.e., **loss function**, which measures the error between $f \in \mathcal{H}$ and data
- ▶ Optimization: find $\hat{f} \in \mathcal{H}$ which minimizes the error/loss function

Testing: How good is \hat{f} on unseen data? Two approaches:

- ▶ Analytical (first 3 lectures)
 - ▶ Make some analytical assumptions, e.g. Gaussian
 - ▶ Compute distributions for the parameters of interest
 - ▶ Develop statistical tests to characterize \hat{f} : t-test, F-test, etc
- ▶ Numerical (rest of the class)
 - ▶ Split data into training and testing
 - ▶ Use training data to find \hat{f}
 - ▶ Use testing data to evaluate how good is \hat{f}

Last lecture

- ▶ Install and get familiar with R (attend the recitation session)

Brief stat review:

- ▶ X_1, X_2, \dots, X_n – i.i.d. with mean μ and variance σ^2
- ▶ Estimators of mean and variance

- ▶ Sample mean

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

- ▶ Sample variance

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$$

- ▶ \bar{X} and S^2 are **unbiased** estimators, i.e.

$$\mathbb{E}\bar{X} = \mu, \quad \text{and} \quad \mathbb{E}S^2 = \sigma^2$$

- ▶ Variability of \bar{X} : $\text{SE}(\bar{X})$ = Standard Error of the mean

$$\text{Var}(\bar{X}) = \sigma^2/n \approx (\text{SE}(\bar{X}))^2 = S^2/n$$

Last lecture: Variability

- ▶ If X_1, \dots, X_n are i.i.d. and **normal/Gaussian**, then
 - ▶ \bar{X} is normal
 - ▶ S^2 has Chi - square distribution:

$$\frac{n-1}{\sigma^2} S^2 \sim \chi_{n-1}^2,$$

χ_{n-1}^2 = sum of $(n-1)$ squares of independent standard normal variables

- ▶ \bar{X} and S^2 are independent
- ▶ t -value and Student's t-distribution:

$$\frac{\bar{X} - \mu}{\sqrt{S^2/n}} \sim \frac{\mathcal{N}(0, 1)}{\sqrt{\chi_{n-1}^2/(n-1)}} = t_{n-1},$$

William Gosset, 1908, under pen name Student

- ▶ ... if X is not normal/Gaussian, then use the CLT

Last lecture: Hypothesis testing - t -test

t_n has a known symmetric and bell shaped density
(use $\Gamma(k + 1/2) \approx \sqrt{k}\Gamma(k)$ for large k)

$$f_n(t) = \frac{\Gamma((n+1)/2)}{\sqrt{\pi n} \Gamma(n/2)} \left(1 + \frac{t^2}{n}\right)^{-\frac{n+1}{2}} \approx \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} \quad (\text{large } n)$$

t -test:

- ▶ Null hypothesis $\mathcal{H}_0 : \mu = \mu_0$
- ▶ Under \mathcal{H}_0 , compute t -value and p -value:

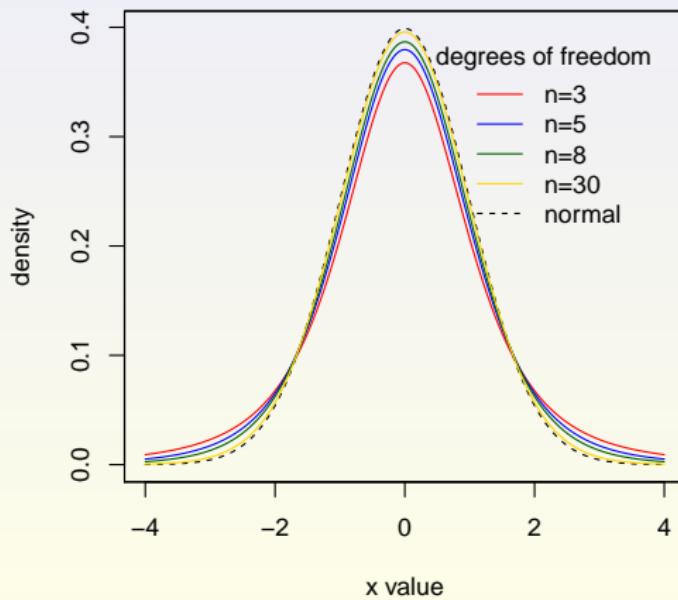
$$t = \frac{\bar{X} - \mu_0}{\sqrt{S^2/n}}, \quad p = \mathbb{P}[|t_{n-1}| \geq |t|]$$

- ▶ Since large values of t unlikely under \mathcal{H}_0 , typically
 - ▶ pick a significance value, say $\alpha = 0.05$
 - ▶ reject \mathcal{H}_0 if $p < \alpha$, say $p < 0.05$
 - ▶ accept \mathcal{H}_0 if $p \geq \alpha$, say $p \geq 0.05$

t-distribution

- ▶ Zero mean
- ▶ Variance ($n > 2$): $n/(n - 2)$

PDFs of t distributions



Last lecture: Linear regression

- ▶ Simple approach to supervised learning
- ▶ Assumes linear dependence of Y on X_1, X_2, \dots, X_p
Almost never true in reality.
- ▶ Extremely useful both conceptually and practically
- ▶ Linear model in 1D ($p = 1$): $X = X_1$

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

- ▶ Estimate β_0 and β_1 by **minimizing residuals**

$$y_i - \hat{y}_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$$

- ▶ Norm selection (distance measure) is important
e.g., l_2 vs. l_1
- ▶ l_2 regression: Least squares (r_{xy} - correlation coefficient)

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2} = r_{xy} \frac{S_y}{S_x}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Last lecture: Statistics of $\hat{\beta}_0$ and $\hat{\beta}_1$

- ▶ **Repeated sampling**
- ▶ $\hat{\beta}_0$ and $\hat{\beta}_1$ vary
- ▶ **Unbiased estimators:**

$$\mathbb{E}\hat{\beta}_0 = \beta_0 \quad \text{and} \quad \mathbb{E}\hat{\beta}_1 = \beta_1$$

- ▶ Variances: (model $Y = f(X) + \epsilon$, ϵ -Gaussian)

$$\begin{aligned}\text{Var}(\hat{\beta}_1) &= \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}, \\ \text{Var}(\hat{\beta}_0) &= \sigma^2 \left(\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right),\end{aligned}$$

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

- ▶ An estimate of σ^2 :

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

where RSE is the Residual Standard Error

Last lecture: Hypothesis testing and confidence intervals

- ▶ Normality assumption: $\epsilon \sim \mathcal{N}(0, \sigma^2)$

- ▶ t -statistic:

$$\frac{\hat{\beta}_1 - \beta_1}{\text{SE}(\hat{\beta}_1)} \sim t_{n-2},$$

where

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

- ▶ Hypothesis testing using t statistics
- ▶ $(1 - \gamma)$ confidence interval (say $\gamma = 5\%, 1 - \gamma = 95\%$):

$$[\hat{\beta}_1 - \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}, \hat{\beta}_1 + \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}]$$

where $t_{\gamma/2, n-2}$ is the $(1 - \gamma/2)$ -th quantile of the t_{n-2} distribution $\mathbb{P}[-t_{\gamma/2, n-2} \leq t_{n-2} \leq t_{\gamma/2, n-2}] = 1 - \gamma$, i.e,

$$\mathbb{P}[\hat{\beta}_1 - \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2} \leq \beta_1 \leq \hat{\beta}_1 + \text{SE}(\hat{\beta}_1) \cdot t_{\gamma/2, n-2}] = 1 - \gamma$$

Example

Recall the 1D model from Lecture 1: $Y = \beta_0 + \beta_1$ (TV advertising) + ϵ , for which we computed the estimates on $n = 200$ data points

$$\hat{\beta}_0 = 0.047537, \quad \hat{\beta}_1 = 7.032594.$$

$\text{SE}(\hat{\beta}_1) = 0.457843$ and degrees of freedom, $\text{DF} = n - 2 = 198$.

Hypothesis testing: $\mathcal{H}_0 : \beta_1 = 0$ vs. $\mathcal{H}_A : \beta_1 \neq 0$ Hence, t-statistics is

$$t = \frac{\hat{\beta}_1 - 0}{\text{SE}(\hat{\beta}_1)} = \frac{7.032594}{0.457843} = 15.36027$$

$$\Rightarrow \text{p-val} = \mathbb{P}[|t_{198}| > 15.36027] = 2 * (1 - \text{pt}(15.36027, \text{df}=198)) \approx 0$$

\Rightarrow Reject \mathcal{H}_0 , ($\text{pt}()$ is a probability distribution of t -variable in R).

Confidence interval (CI): say 95% CI, $\gamma = 5\%$

$$t_{2.5\%, 198} = |\text{qt}(0.025, \text{df}=198)| = 1.972017, \text{ qt}() = \text{t-quantile function in R.}$$

$$\begin{aligned} 95\% \text{ CI for true } \beta_1 &: (\hat{\beta}_1 \pm t_{2.5\%, 198} \times \text{SE}(\hat{\beta}_1)) \\ &= (7.032594 \pm 1.972017 \times 0.457843) \\ &= (6.12972, 7.935468) \end{aligned}$$

Multidimensional linear regression

- ▶ Model

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

- ▶ Example

$$\text{Sales} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{Radio} + \beta_3 \times \text{Newspaper} + \epsilon$$

- ▶ Interpretation: β_i is the average effect on Y of a one unit increase in X_i , holding all other predictors fixed

- ▶ Notes

- ▶ Ideally the predictors are uncorrelated
- ▶ Correlations amongst predictors cause problems
 - ▶ increased variance of coefficients
 - ▶ tricky interpretations (example: $X_1 = X_2^2$)
- ▶ Claims of causality should be avoided for observational data

l_2 regression

- ▶ n observations: $(y_i, x_{i,1}, x_{i,2}, \dots, x_{i,p})$
- ▶ Given estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$, the prediction is

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

or in matrix form

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ 1 & x_{2,1} & \cdots & x_{2,p} \\ \vdots & & & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,p} \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{bmatrix}$$

- ▶ Minimize (over β_1, \dots, β_p) the residual sum of squares

$$\text{RSS}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

l_2 regression: Solution

- Minimize RSS: Differentiating $\text{RSS}(\beta)$, we get

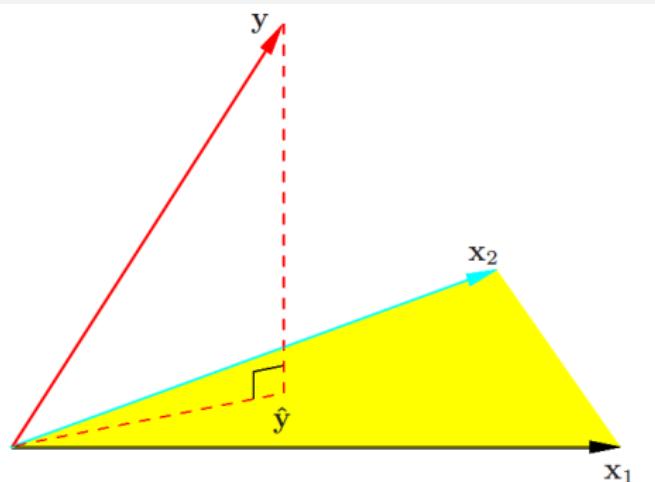
$$\frac{\partial \text{RSS}(\beta)}{\partial \beta} = -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta) = \mathbf{0}$$

- Solution $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_p)$: assuming $\mathbf{X}^\top \mathbf{X}$ is full rank

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \quad \hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$$

If $\mathbf{X}^\top \mathbf{X}$ is singular, use the pseudo-inverse, which finds $\hat{\beta}$ with the smallest l_2 norm, smallest $\|\hat{\beta}\|_2^2$.

- Geometry



Example: Advertising data

```
> lm2<-lm(adv$Sales~adv$TV+adv$Radio+adv$Newspaper)
> summary(lm2)
```

Call:

```
lm(formula = adv$Sales ~ adv$TV + adv$Radio + adv$Newspaper)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.8277	-0.8908	0.2418	1.1893	2.8292

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.938889	0.311908	9.422	<2e-16 ***
adv\$TV	0.045765	0.001395	32.809	<2e-16 ***
adv\$Radio	0.188530	0.008611	21.893	<2e-16 ***
adv\$Newspaper	-0.001037	0.005871	-0.177	0.86

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Residual standard error: 1.686 on 196 degrees of freedom

Multiple R-squared: 0.8972, Adjusted R-squared: 0.8956

F-statistic: 570.3 on 3 and 196 DF, p-value: < 2.2e-16

```
> cor(adv[,2:5])
```

	TV	Radio	Newspaper	Sales
TV	1.00000000	0.05480866	0.05664787	0.7822244
Radio	0.05480866	1.00000000	0.35410375	0.5762226
Newspaper	0.05664787	0.35410375	1.00000000	0.2282990
Sales	0.78222442	0.57622257	0.22829903	1.0000000

Solution: Algebraic/geometric interpretations

- ▶ Ideally $\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\beta}}$ ($\text{RSS} = 0$), but this equation has no solution (except in trivial cases), since $\mathbf{y} \notin C(\mathbf{X})$
 $C(\mathbf{X})$ = column space, i.e., hyperplane formed by columns of \mathbf{X} .
- ▶ Instead, we solve $\mathbf{P}\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\beta}}$, where $\mathbf{P} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ is the l_2 -projection matrix onto $C(\mathbf{X})$
 - ▶ \mathbf{P} is sometimes called "hat" matrix, denoted as \mathbf{H} , since it puts a hat on \mathbf{y} , i.e. $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y} \equiv \mathbf{H}\mathbf{y}$
- ▶ Equivalently, $\hat{\boldsymbol{\beta}}$ satisfies

$$\mathbf{X}^\top\mathbf{y} = \mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}$$

- ▶ A unique solution exists when the columns of \mathbf{X} are linearly independent – in that case, $\mathbf{X}^\top\mathbf{X}$ is full-rank and positive definite
- ▶ Consequences:
 - ▶ $(\mathbf{y} - \hat{\mathbf{y}}) = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})$ is perpendicular to $C(\mathbf{X})$
 - ▶ $\mathbf{0} = (\mathbf{y} - \hat{\mathbf{y}})^\top\mathbf{X} = (\mathbf{y} - \mathbf{P}\mathbf{y})^\top\mathbf{X} = \mathbf{y}^\top(\mathbf{X} - \mathbf{P}\mathbf{X})$
 - ▶ $\sum_{i=1}^n (y_i - \hat{y}_i) = (\mathbf{y} - \hat{\mathbf{y}})\mathbf{1} = 0$
 - ▶ $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|_2^2 + (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top\mathbf{X}^\top\mathbf{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})$

Computational Complexity

Note that $\mathbf{X}^\top \mathbf{X}$ is a $(p + 1) \times (p + 1)$ matrix, and thus finding $\hat{\beta}$ requires solving linear system of $(p + 1)$ equations

$$(\mathbf{X}^\top \mathbf{X}) \hat{\beta} = \mathbf{X}^\top \mathbf{y},$$

which has $O(p^3)$ computational complexity.

High dimensional data:

- ▶ Suppose $p \gg n$
 $p = \#$ of dimensions, $n = \#$ of samples
- ▶ Example: $n = 100$ samples of $p = 10,000$ gene expressions

$$\text{computational complexity} = 10^{12}(!)$$

- ▶ Can we do better than that?

Dual solution: dot products and kernels

Note that $\hat{\beta}$ can be represented as a linear combination of data

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top (\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-2} \mathbf{X}^\top \mathbf{y}) =: \mathbf{X}^\top \boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i \mathbf{x}_i,$$

where $\mathbf{x}_i = (1, x_{i,1}, \dots, x_{i,p})$ is the i th data point, which implies

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}\mathbf{X}^\top \boldsymbol{\alpha} =: \mathbf{K}\boldsymbol{\alpha},$$

where \mathbf{K} is a matrix of dot products, also known as Kernel or Gram matrix, which is symmetric and positive definite

$$K_{kj} = \langle \mathbf{x}_k, \mathbf{x}_j \rangle := \sum_{l=0}^p x_{k,l} x_{j,l}.$$

Hence, by minimizing the dual problem $\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|_2^2$, one finds (assuming \mathbf{K} being non-singular)

$$\boldsymbol{\alpha} = \mathbf{K}^{-1} \mathbf{y}$$

which has computational complexity $O(n^3)$. Direct computation of K requires $O(n^2 p)$ operations, resulting in total complexity $O(n^2(p+n)) \ll O(p^3)$ when $n \ll p$.

We will be back to dual (Kernel) solution throughout the course.

Back to the model: How good is the model fit?

- ▶ Total sum of squares: $TSS = (\mathbf{y} - \bar{y}\mathbf{1})^\top(\mathbf{y} - \bar{y}\mathbf{1})$
- ▶ Explained sum of squares: $ESS = (\hat{\mathbf{y}} - \bar{y}\mathbf{1})^\top(\hat{\mathbf{y}} - \bar{y}\mathbf{1})$
- ▶ Then

$$\begin{aligned} TSS &= (\mathbf{y} - \hat{\mathbf{y}} + \hat{\mathbf{y}} - \bar{y}\mathbf{1})^\top(\mathbf{y} - \hat{\mathbf{y}} + \hat{\mathbf{y}} - \bar{y}\mathbf{1}) \\ &= RSS + ESS + 2(\mathbf{y} - \hat{\mathbf{y}})^\top(\hat{\mathbf{y}} - \bar{y}\mathbf{1}) \\ &= RSS + ESS \end{aligned}$$

- ▶ A measure of quality of the model

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

- ▶ $R^2 \uparrow$ as more explanatory variables are added to the model – need to consider the number of variables

Example: Advertising data

```
> summary(lm(adv$Sales~adv$TV+adv$Radio))
```

Call:

```
lm(formula = adv$Sales ~ adv$TV + adv$Radio)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.7977	-0.8752	0.2422	1.1708	2.8328

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.92110	0.29449	9.919	<2e-16 ***
adv\$TV	0.04575	0.00139	32.909	<2e-16 ***
adv\$Radio	0.18799	0.00804	23.382	<2e-16 ***

Signif. codes:	0 ***	0.001 **	0.01 * 0.05 . 0.1	1

Residual standard error: 1.681 on 197 degrees of freedom

Multiple R-squared: **0.8972**, Adjusted R-squared: 0.8962

F-statistic: 859.6 on 2 and 197 DF, p-value: < 2.2e-16

► R^2

► Are all predictors useful? Which are?

Distribution of $\hat{\beta}$

- ▶ Normality assumption: i.i.d. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
- ▶ $\mathbf{y} = \mathbf{X}\beta + \epsilon$ is also normally distributed, with mean $\mu = \mathbf{X}\beta$, covariance matrix $\Sigma = \sigma^2 \mathbf{I}$ and density

$$f_{\mathbf{y}}(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)}$$

- ▶ $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ is also normal with

$$\mathbb{E}\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X} \beta + (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbb{E}\epsilon = \beta$$

and

$$\begin{aligned}\text{Cov}(\hat{\beta}) &= \mathbb{E}(\hat{\beta} - \beta)(\hat{\beta} - \beta)^\top \\ &= \mathbb{E}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \epsilon ((\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \epsilon)^\top = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}\end{aligned}$$

- ▶ Hence $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1})$

Residuals

- Residuals $\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$ are also **normal** with zero mean

$$\mathbb{E}[\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}] = \mathbb{E}[\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} - \mathbf{X}\hat{\boldsymbol{\beta}}] = \mathbf{0}$$

and covariance

$$\begin{aligned}\mathbb{E}(\mathbf{y} - \hat{\mathbf{y}})(\mathbf{y} - \hat{\mathbf{y}})^\top &= \mathbb{E}(\mathbf{y} - \mathbf{P}\mathbf{y})(\mathbf{y} - \mathbf{P}\mathbf{y})^\top \\ &= \mathbb{E}(\mathbf{I} - \mathbf{P})\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top(\mathbf{I} - \mathbf{P})^\top = \sigma^2(\mathbf{I} - \mathbf{P})\end{aligned}$$

since $\mathbf{P}^2 = \mathbf{P}$ and

$$\mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I} - \mathbf{P})\mathbf{y} = (\mathbf{I} - \mathbf{P})(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) = (\mathbf{I} - \mathbf{P})\boldsymbol{\epsilon}$$

Estimating σ

- ▶ RSS = $(\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}}) = \boldsymbol{\epsilon}^\top (\mathbf{I} - \mathbf{P}) \boldsymbol{\epsilon}$, and

$$\begin{aligned}\text{rank}(\mathbf{I} - \mathbf{P}) &= \text{tr}(\mathbf{I} - \mathbf{P}) \\ &= \text{tr}(\mathbf{I} - \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top) \\ &= n - \text{tr}(\mathbf{X}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}) = n - p - 1\end{aligned}$$

- ▶ Then it can be shown (Cochran's Theorem - next class)

$$\frac{\text{RSS}}{\sigma^2} \sim \chi_{n-p-1}^2$$

- ▶ Estimator (since $\chi_{n-p-1}^2 = n - p - 1$)

$$\hat{\sigma} = \sqrt{\frac{\text{RSS}}{n - p - 1}}$$

Back to testing

- ▶ $\mathcal{H}_0 : \beta_j = 0$
- ▶ Intuition: reject \mathcal{H}_0 if $\hat{\beta}_j$ is “large”
- ▶ How large?
- ▶ Under \mathcal{H}_0 , $\hat{\beta}_j$ is $\mathcal{N}(0, \sigma^2(\mathbf{X}^\top \mathbf{X})_{j,j}^{-1})$
- ▶ Consider t -statistic

$$\frac{\hat{\beta}_j}{\hat{\sigma} \sqrt{(\mathbf{X}^\top \mathbf{X})_{j,j}^{-1}}} = \frac{\frac{\hat{\beta}_j}{\sigma \sqrt{(\mathbf{X}^\top \mathbf{X})_{j,j}^{-1}}}}{\sqrt{\frac{\text{RSS}}{\sigma^2(n-p-1)}}} \sim t_{n-p-1}$$

F-test

- ▶ Better idea: use RSS to test instead of $\hat{\beta}$
- ▶ $\mathcal{H}_0 : \beta_1 = \beta_2 = \dots = \beta_p = 0$
- ▶ \mathcal{H}_1 : exists j such that $\beta_j \neq 0$
- ▶ Under \mathcal{H}_0 , we have a null model: $Y = \beta_0 + \epsilon$
- ▶ Let RSS_0 be the residual sum of squares under \mathcal{H}_0
- ▶ Under \mathcal{H}_0 :

$$\frac{\text{RSS}_0 - \text{RSS}}{\sigma^2} = \frac{\text{TSS} - \text{RSS}}{\sigma^2} \sim \chi_p^2$$

and

$$\frac{\frac{\text{TSS} - \text{RSS}}{p}}{\frac{\text{RSS}}{n-p-1}} \sim F_{p, n-p-1}$$

- ▶ Back to the example:

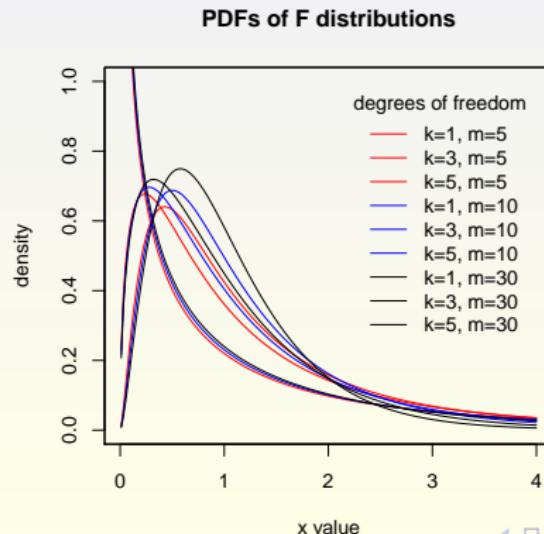
Residual standard error: 1.686 on 196 degrees of freedom
Multiple R-squared: 0.8972, Adjusted R-squared: 0.8956
F-statistic: 570.3 on 3 and 196 DF, p-value: < 2.2e-16

F-distribution

- ▶ $F_{k,m}$: independent $V \sim \chi_k^2$ and $W \sim \chi_m^2$

$$\frac{V/k}{W/m} \sim F_{k,m}$$

- ▶ Mean ($m > 2$): $m/(m - 2)$
- ▶ Variance ($m > 4$): $\frac{2m^2(k+m-2)}{k(m-2)^2(m-4)}$



F-test

- ▶ $\mathcal{H}_0 : \beta_j = 0$
- ▶ Under \mathcal{H}_0 , we have a reduced model:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_{j-1} X_{j-1} + \beta_{j+1} X_{j+1} + \cdots + \beta_p X_p + \epsilon$$

- ▶ Refer to the reduced model by index $-j$
- ▶ Intuition: while $\text{RSS} \leq \text{RSS}_{-j}$...
 - ▶ if $\text{RSS} \ll \text{RSS}_{-j}$, then reject \mathcal{H}_0
 - ▶ if $\text{RSS} \approx \text{RSS}_{-j}$, then accept \mathcal{H}_0
- ▶ Under \mathcal{H}_0 :

$$\frac{\text{RSS}_{-j} - \text{RSS}}{\sigma^2} \sim \chi_1^2$$

and

$$\frac{\text{RSS}_{-j} - \text{RSS}}{\frac{\text{RSS}}{n-p-1}} \sim F_{1,n-p-1}$$

F-test

- ▶ (m) denotes a sub-model obtained by a linear constraint on β
- ▶ Examples
 - ▶ $\beta_1 = \beta_2 = \dots = \beta_p$: $Y = \beta_0 + \beta_1(X_1 + X_2 + \dots + X_p) + \epsilon$
 - ▶ $\beta_1 = \beta_2$: $Y = \beta_0 + \beta_1(X_1 + X_2) + \beta_3X_3 + \dots + \beta_pX_p + \epsilon$
- ▶ Testing: \mathcal{H}_0 (reduced model) vs. \mathcal{H}_1 (complete model)
- ▶ $q < p$ is the number of explanatory variables in the reduced model
- ▶ Under \mathcal{H}_0 :

$$\frac{\text{RSS}_{(m)} - \text{RSS}}{\sigma^2} \sim \chi_{p-q}^2 \quad \Rightarrow \quad \frac{\frac{\text{RSS}_{(m)} - \text{RSS}}{p-q}}{\frac{\text{RSS}}{n-p-1}} \sim F_{p-q, n-p-1}$$

Qualitative predictors

- ▶ Credit.csv data set
- ▶ 400 observations:

```
","", "Income", "Limit", "Rating", "Cards", "Age", "Education", "Gender", "Student", "Married", "Ethnicity", "Balance"  
"1", 14.891, 3606, 283, 2, 34, 11, " Male", "No", "Yes", "Caucasian", 333  
"2", 106.025, 6645, 483, 3, 82, 15, "Female", "Yes", "Yes", "Asian", 903  
"3", 104.593, 7075, 514, 4, 71, 11, " Male", "No", "No", "Asian", 580  
"4", 148.924, 9504, 681, 3, 36, 11, "Female", "No", "No", "Asian", 964  
"5", 55.882, 4897, 357, 2, 68, 16, " Male", "No", "Yes", "Caucasian", 331  
.  
.  
.  
"398", 57.872, 4171, 321, 5, 67, 12, "Female", "No", "Yes", "Caucasian", 138  
"399", 37.728, 2525, 192, 1, 44, 13, " Male", "No", "Yes", "Caucasian", 0  
"400", 18.701, 5524, 415, 5, 64, 7, "Female", "No", "No", "Asian", 966
```

- ▶ Quantitative predictors: Income (in thousands), Limit (credit), Rating, Cards (number of), Age, Education (years of), Balance
- ▶ Qualitative predictors (factors): Gender, Student, Married, Ethnicity

Incorporating qualitative predictors

- ▶ Dependency of Balance on Gender
- ▶ Ignore all other variables
- ▶ Gender has two levels:

$$X_i = \begin{cases} 1, & \text{if } i\text{th individual is female} \\ 0, & \text{if } i\text{th individual is male} \end{cases}$$

- ▶ Model: $Y = \beta_0 + \beta_1 X + \epsilon$
- ▶ Interpretation
 - ▶ β_0 : average Balance among males
 - ▶ $\beta_0 + \beta_1$: average Balance among females
 - ▶ β_1 : average difference in Balance between females and males
- ▶ The 1/0 encoding is arbitrary. Can use another scheme – only the interpretation changes

Example

```
> credit <- read.csv("credit.csv", header=TRUE, sep=",")  
> lm3<-lm(Balance~Gender, data=credit)  
> summary(lm3)
```

Call:

```
lm(formula = Balance ~ Gender, data = credit)
```

Residuals:

Min	1Q	Median	3Q	Max
-529.54	-455.35	-60.17	334.71	1489.20

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	509.80	33.13	15.389	<2e-16 ***
GenderFemale	19.73	46.05	0.429	0.669

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Residual standard error: 460.2 on 398 degrees of freedom

Multiple R-squared: 0.0004611, Adjusted R-squared: -0.00205

F-statistic: 0.1836 on 1 and 398 DF, p-value: 0.6685

Incorporating qualitative predictors

- ▶ When a factor has more than two levels, a single dummy variable can not represent all possible values
- ▶ In that case, create additional dummy variables
- ▶ Example: Ethnicity

$$X_{i,1} = \begin{cases} 1, & \text{if } i\text{th individual is Asian} \\ 0, & \text{if } i\text{th individual is not Asian} \end{cases}$$

$$X_{i,2} = \begin{cases} 1, & \text{if } i\text{th individual is Caucasian} \\ 0, & \text{if } i\text{th individual is not Caucasian} \end{cases}$$

- ▶ Model:

$$Y = \begin{cases} \beta_0 + \beta_1 + \epsilon, & \text{if } i\text{th individual is Asian} \\ \beta_0 + \beta_2 + \epsilon, & \text{if } i\text{th individual is Caucasian} \\ \beta_0 + \epsilon, & \text{otherwise} \end{cases}$$

Example

```
> lm4<-lm(Balance~Ethnicity,data=credit)
> summary(lm4)
```

Call:

```
lm(formula = Balance ~ Ethnicity, data = credit)
```

Residuals:

Min	1Q	Median	3Q	Max
-531.00	-457.08	-63.25	339.25	1480.50

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	531.00	46.32	11.464	<2e-16 ***
EthnicityAsian	-18.69	65.02	-0.287	0.774
EthnicityCaucasian	-12.50	56.68	-0.221	0.826

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Residual standard error: 460.9 on 397 degrees of freedom

Multiple R-squared: 0.0002188, Adjusted R-squared: -0.004818

F-statistic: 0.04344 on 2 and 397 DF, p-value: 0.9575

Extensions: Interactions

- ▶ Relax additivity
- ▶ Back to Advertising data set
- ▶ Suppose spending money on radio advertising increases the effectiveness of TV advertising
- ▶ New model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$

- ▶ $X_1 X_2$ – just multiply observations
- ▶ Hierarchical principle: if interaction is in the model, main effects are in the model, even if main effects are not significant

Example

```
> lm5<-lm(Sales~TV+Radio+TV*Radio,data=adv)
> summary(lm5)

Call:
lm(formula = Sales ~ TV + Radio + TV * Radio, data = adv)

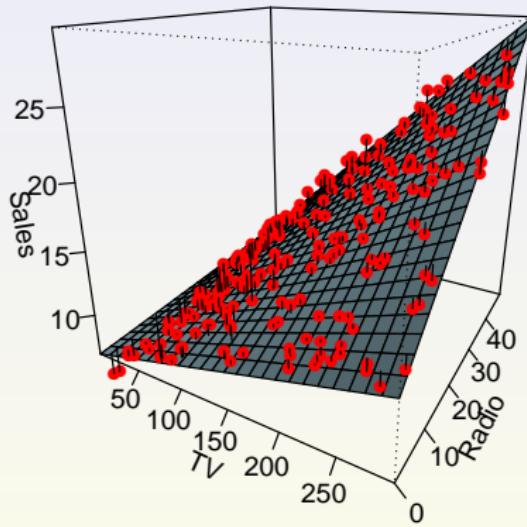
Residuals:
    Min      1Q  Median      3Q     Max 
-6.3366 -0.4028  0.1831  0.5948  1.5246 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 6.750e+00  2.479e-01  27.233 <2e-16 ***
TV          1.910e-02  1.504e-03 12.699 <2e-16 ***
Radio       2.886e-02  8.905e-03  3.241  0.0014 **  
TV:Radio    1.086e-03  5.242e-05 20.727 <2e-16 ***  
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.9435 on 196 degrees of freedom
Multiple R-squared:  0.9678, Adjusted R-squared:  0.9673 
F-statistic:  1963 on 3 and 196 DF,  p-value: < 2.2e-16
```

- ▶ $\hat{\beta}_3$: the increase in the effectiveness of TV advertising for a one unit increase in radio advertising (or vice-versa)

Example

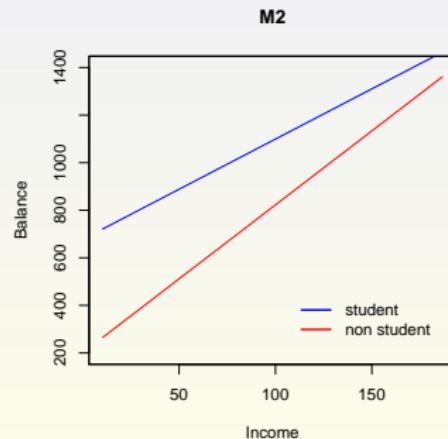
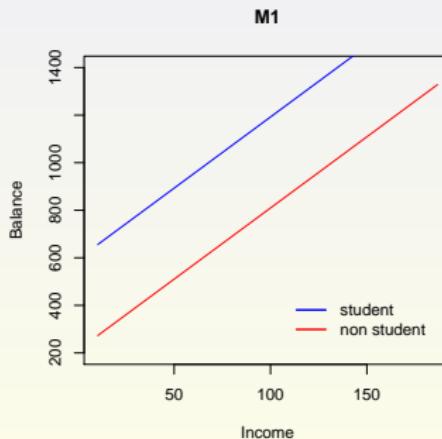


Example

- ▶ Credit data set
- ▶ $Y = \text{Balance}$, $X_1 = \text{Income}$, $X_2 = \text{Student} \in \{0, 1\}$
- ▶ Two models:

$$M_1 : Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

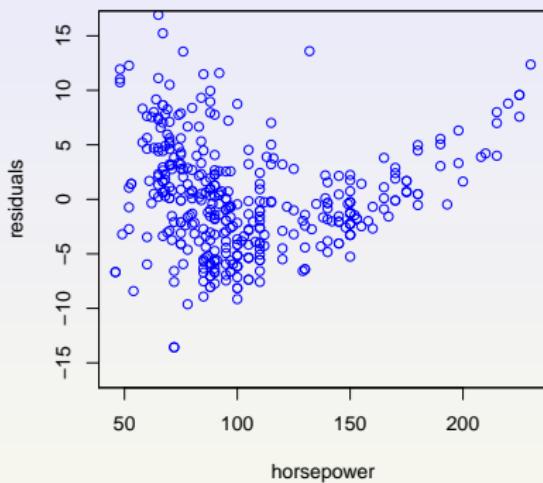
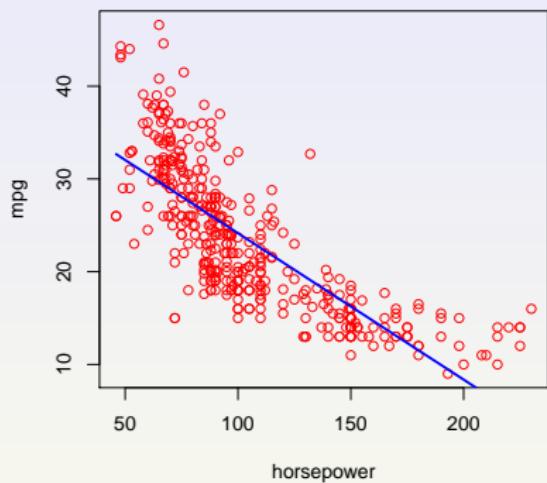
$$M_2 : Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$



- ▶ Changes in income affect students and non-students differently

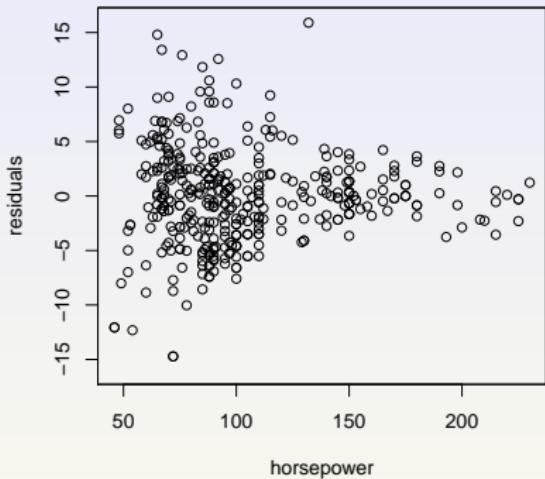
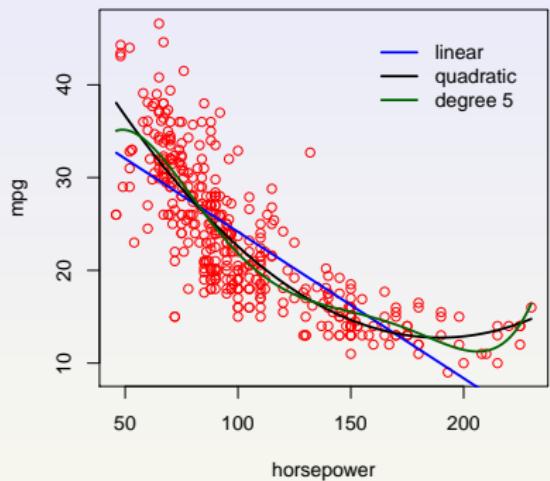
Extensions: Nonlinearities - Basis expansion

- ▶ Auto data set: mpg vs. horsepower



- ▶ Polynomial regression
- ▶ Model: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
- ▶ The model is still linear in β : treat X^2 as a variable
- ▶ We will have a comprehensive lecture on basis expansions later

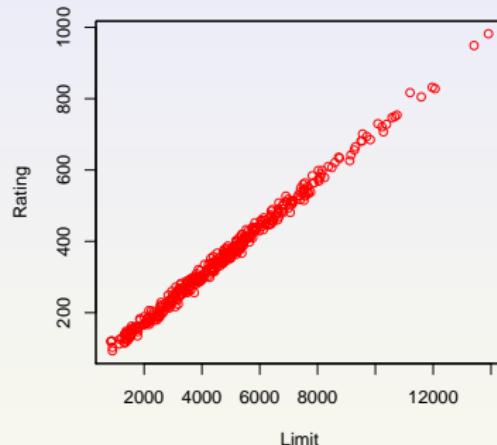
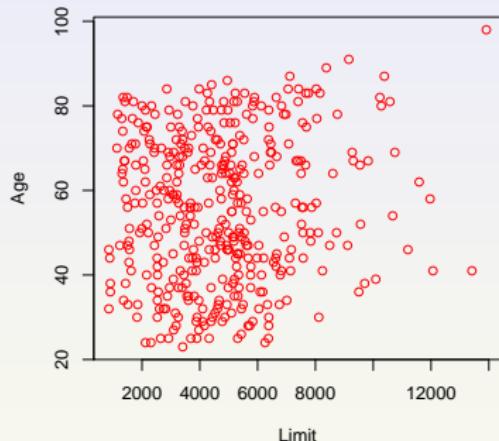
Example



- ▶ Increasing the degree can cause overfitting
- ▶ Alternative transformations

Colinearity

- ▶ Credit data set:

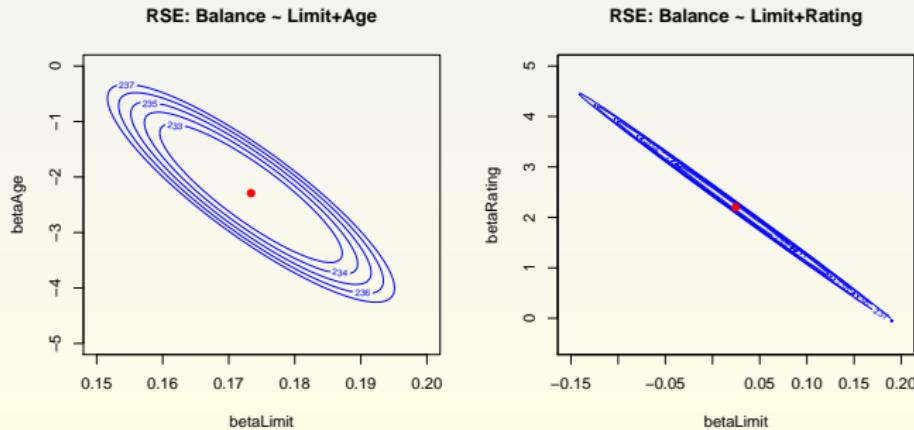


- ▶ Rating and Limit are co-linear
- ▶ Difficult to assess individual impact on Balance
- ▶ $\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{X} \hat{\boldsymbol{\beta}}$ can be numerically unstable

Example

```
> lm8a<-lm(Balance ~ Limit + Age,data=credit)
> summary(lm8a)$coefficients
    Estimate Std. Error t value Pr(>|t|) 
(Intercept) -173.410901 43.828387048 -3.956589 9.005366e-05
Limit         0.173365  0.005025662 34.495944 1.627198e-121
Age          -2.291486  0.672484540 -3.407492 7.226468e-04
```

```
> lm8b<-lm(Balance ~ Limit + Rating,data=credit)
> summary(lm8b)$coefficients
    Estimate Std. Error t value Pr(>|t|) 
(Intercept) -377.53679536 45.25417619 -8.3425846 1.213565e-15
Limit         0.02451438  0.06383456  0.3840298 7.011619e-01
Rating        2.20167217  0.95229387  2.3119672 2.129053e-02
```



Prediction considerations

- ▶ Examine assumptions
- ▶ Uncertainties/Errors
 - ▶ Regression coefficients are noisy
 - ▶ Measurements are noisy even when the function is known
 - ▶ Bias: The true function might not be linear

K-NN regression: Non-parametric approach

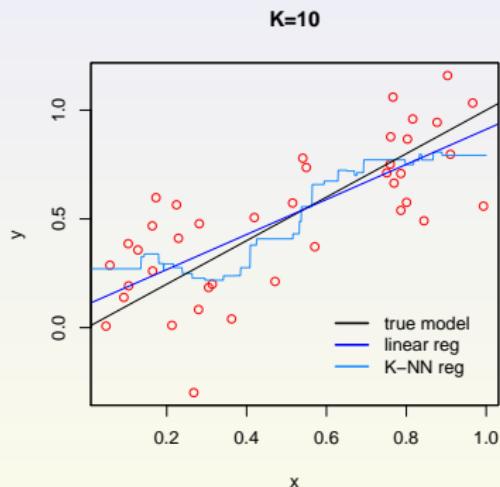
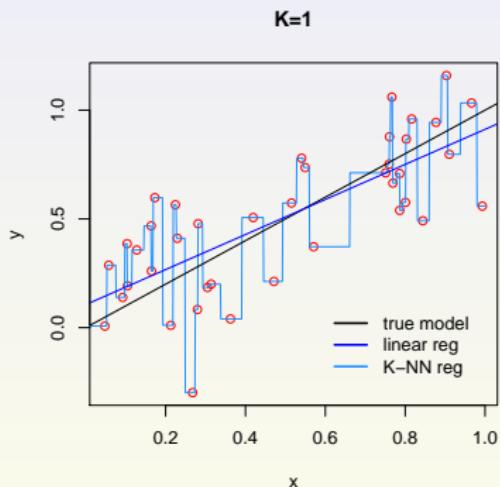
- ▶ Linear regression is not the only approach
- ▶ K-NN: K-nearest neighbors
- ▶ Intuition: “similar” argument values should lead to “similar” function values
 - ▶ distances
 - ▶ data normalization
 - ▶ high dimensionality case
- ▶ Let \mathcal{N}_x^K be the K -nearest neighbors set of observations for x :

$$\hat{f}(x) = \frac{1}{K} \sum_{i \in \mathcal{N}_x^K} y_i$$

- ▶ K is a parameter of the algorithm
 - ▶ Small K – flexible fit, high variance
 - ▶ Large K – smooth, high bias
 - ▶ How to select K ?

Examples

- ▶ Linear model:



Reading:

ISL: Finish reading Chapter 3

ESL: Chapter 3

Homework: Homework 1 due in 2 weeks - Sep 27.

EECS E6690: Statistical Learning for Biological and Information Systems

Lecture 3: Model Selection and Regularization

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm
303 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Big Picture: Estimation, Testing and Model Selection

Estimation:

- ▶ Select a class of function for f : Hypothesis class \mathcal{H}
say, \mathcal{H} are linear functions, i.e., linear regression
- ▶ Optimization: find $\hat{f} \in \mathcal{H}$ which minimizes the error/loss function

Testing: How good is \hat{f} on unseen data?

Two approaches:

- ▶ *Analytical*: Make some analytical assumptions, e.g. Gaussian, and compute distributions for the parameters of interest.
Develop statistical tests to characterize \hat{f} : t-test, F-test, etc.
- ▶ *Numerical* (coming soon): Split data into training and testing.
Use training data to find \hat{f} and testing data to evaluate it.

Model selection and regularization: Find the smallest/simplest model?

- ▶ *Analytical*: Use F/t-tests to select the smallest model
- ▶ *Numerical*:
 - ▶ **Model selection**: Fit and test models with less predictors
 - ▶ **Regularization**: Modify the loss function such that it penalizes more complex models. This also helps with overfitting.

Last lecture: Multidimensional linear regression

- ▶ p predictors (features, independent variables)
- ▶ n observations: $(y_i, x_{i,1}, x_{i,2}, \dots, x_{i,p})$
- ▶ Given estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$, the prediction is

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

or in matrix form

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ 1 & x_{2,1} & \cdots & x_{2,p} \\ \vdots & & & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,p} \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{bmatrix}$$

- ▶ Minimize (over β_1, \dots, β_p) the residual sum of squares (l_2 norm)

$$\text{RSS}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Last lecture: l_2 solution

- ▶ Differentiating RSS(β), we get

$$-2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = \mathbf{0}$$

- ▶ If $\mathbf{X}^\top \mathbf{X}$ has a full rank

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- ▶ *Hat matrix:* $\mathbf{P} := \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$; it puts a hat on \mathbf{y}
 \mathbf{P} is the l_2 -projection matrix of \mathbf{y} onto $C(\mathbf{X})$
- ▶ $\hat{\mathbf{y}}$ and $(\mathbf{y} - \hat{\mathbf{y}})$ are orthogonal; $\hat{\mathbf{y}}$ is in $C(\mathbf{X})$
- ▶ $\sum_{i=1}^n (y_i - \hat{y}_i) = (\mathbf{y} - \hat{\mathbf{y}})\mathbf{1} = 0$
(since $\mathbf{1} \in C(\mathbf{X})$ and $(\mathbf{y} - \hat{\mathbf{y}})$ orthogonal to $C(\mathbf{X})$)

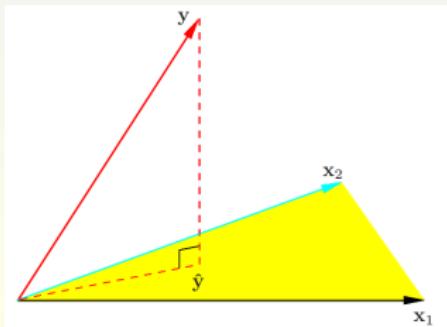
Geometry of 1D linear regression

Consider a data set $(x_1, y_1), \dots, (x_n, y_n)$, $n \geq 2$, and let

$$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{X} = [\mathbf{1}, \mathbf{x}], \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$$

Then, $\hat{\mathbf{y}}$ is projection of vector \mathbf{y} onto a plane spanned by vectors $(\mathbf{1}, \mathbf{x})$, and can be computed algebraically as

$$\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}, \quad \text{where } \mathbf{P} := \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top.$$



Geometry of 1D linear regression

Let us now compute this projection geometrically. First let us convert vectors $(\mathbf{1}, \mathbf{x})$ into orthonormal basis $(\mathbf{u}_1, \mathbf{u}_2)$ using Gram-Schmidt method. Assume that $\mathbf{1}$ and \mathbf{x} are linearly independent, i.e., $\mathbf{x} \neq c\mathbf{1}, c \neq 0$. First, we normalize $\mathbf{u}_1 = \mathbf{1}/\sqrt{n}$, and then

$$\mathbf{u}_2 = \frac{\mathbf{x} - (\mathbf{x} \cdot \mathbf{u}_1)\mathbf{u}_1}{\|\mathbf{x} - (\mathbf{x} \cdot \mathbf{u}_1)\mathbf{u}_1\|} = \frac{\mathbf{x} - \bar{x}\mathbf{1}}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}}, \quad \text{where } \bar{x} = \frac{1}{n} \sum x_i.$$

Now, we project \mathbf{y} onto $(\mathbf{u}_1, \mathbf{u}_2)$ using

$$\begin{aligned}\hat{\mathbf{y}} &= (\mathbf{y} \cdot \mathbf{u}_1)\mathbf{u}_1 + (\mathbf{y} \cdot \mathbf{u}_2)\mathbf{u}_2 \\ &= \bar{y}\mathbf{1} + \frac{\mathbf{x} \cdot \mathbf{y} - \bar{x}\mathbf{1} \cdot \mathbf{y}}{\sum(x_i - \bar{x})^2}(\mathbf{x} - \bar{x}\mathbf{1}), \quad \text{where } \bar{y} = \frac{1}{n} \sum y_i \\ &= \hat{\beta}_0\mathbf{1} + \hat{\beta}_1\mathbf{x}, \quad \text{where } \hat{\beta}_1 = \frac{\sum(y_i - \bar{y})(x_i - \bar{x})}{\sum(x_i - \bar{x})^2}, \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}.\end{aligned}\tag{1}$$

Moreover, using (1), argue that the projection matrix $\mathbf{P} = \mathbf{Q}\mathbf{Q}^\top$, where \mathbf{Q} is a matrix with columns $(\mathbf{u}_1, \mathbf{u}_2)$, i.e., $\mathbf{Q} = [\mathbf{u}_1 \mathbf{u}_2]$. Show that $\mathbf{P} = \mathbf{Q}\mathbf{Q}^\top = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$.

Dual solution: dot products and kernels

Note that $\hat{\beta}$ can be represented as a linear combination of data

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top (\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-2} \mathbf{X}^\top \mathbf{y}) =: \mathbf{X}^\top \boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i \mathbf{x}_i,$$

where $\mathbf{x}_i = (1, x_{i,1}, \dots, x_{i,p})$ is the i th data point, which implies

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}\mathbf{X}^\top \boldsymbol{\alpha} =: \mathbf{K}\boldsymbol{\alpha},$$

where \mathbf{K} is a matrix of dot products, also known as Kernel or Gram matrix, which is symmetric and positive definite

$$K_{kj} = \langle \mathbf{x}_k, \mathbf{x}_j \rangle := \sum_{l=0}^p x_{k,l} x_{j,l}.$$

Hence, by minimizing the dual problem $\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|_2^2$, one finds (assuming \mathbf{K} being non-singular)

$$\boldsymbol{\alpha} = \mathbf{K}^{-1} \mathbf{y}$$

which has computational complexity $O(n^3)$. Direct computation of K requires $O(n^2 p)$ operations, resulting in total complexity $O(n^2(p+n)) \ll O(p^3)$ when $n \ll p$.

We will be back to dual (Kernel) solution throughout the course.

Last lecture: Goodness of fit

- ▶ Total sum of squares: $TSS = (\mathbf{y} - \bar{y}\mathbf{1})^\top(\mathbf{y} - \bar{y}\mathbf{1})$
- ▶ Explained sum of squares: $ESS = (\hat{\mathbf{y}} - \bar{y}\mathbf{1})^\top(\hat{\mathbf{y}} - \bar{y}\mathbf{1})$
- ▶ Then

$$\begin{aligned} TSS &= (\mathbf{y} - \hat{\mathbf{y}} + \hat{\mathbf{y}} - \bar{y}\mathbf{1})^\top(\mathbf{y} - \hat{\mathbf{y}} + \hat{\mathbf{y}} - \bar{y}\mathbf{1}) \\ &= RSS + ESS + 2(\mathbf{y} - \hat{\mathbf{y}})^\top(\hat{\mathbf{y}} - \bar{y}\mathbf{1}) \\ &= RSS + ESS \end{aligned}$$

- ▶ A measure of quality of the model

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

- ▶ R^2 - *Coefficient of determination*: better fit as $R^2 \uparrow 1$
i.e., more data explained by the model

$R^2 = 1$ perfect linear fit

Last lecture: Computing distributions

- ▶ Normal/Gaussian assumption: i.i.d. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
- ▶ **Normal/Gaussian + Linear:** Can compute anything
Check EC in HW1 for the derivation of χ^2, t, F distributions.
- ▶ $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2(\mathbf{X}^\top \mathbf{X})^{-1})$
- ▶ $\text{RSS} = (\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}}) = \boldsymbol{\epsilon}^\top (\mathbf{I} - \mathbf{P})\boldsymbol{\epsilon}$, with zero-mean, normal residuals

$$\mathbb{E}[\mathbf{y} - \mathbf{X}\hat{\beta}] = \mathbb{E}[\mathbf{X}\beta + \boldsymbol{\epsilon} - \mathbf{X}\hat{\beta}] = \mathbf{0}$$

- ▶ Then it can be shown

$$\frac{\text{RSS}}{\sigma^2} \sim \chi_{n-p-1}^2$$

- ▶ Estimator: when n is large, $\chi_{n-p-1}^2 \approx n - p - 1$, and

$$\hat{\sigma} = \sqrt{\frac{\text{RSS}}{n - p - 1}}$$

Last lecture: F -test

- ▶ Could use $\hat{\beta}$ and t-test, but F (Fisher)-test is easier
- ▶ F -test idea: **use RSS** to test instead of $\hat{\beta}$
- ▶ $\mathcal{H}_0 : \beta_1 = \beta_2 = \dots = \beta_p = 0$
- ▶ \mathcal{H}_1 : exists j such that $\beta_j \neq 0$
- ▶ Under \mathcal{H}_0 , we have a null model: $Y = \beta_0 + \epsilon$
- ▶ Let RSS_0 be the residual sum of squares under \mathcal{H}_0
- ▶ Under \mathcal{H}_0 :

$$\frac{\text{RSS}_0 - \text{RSS}}{\sigma^2} = \frac{\text{TSS} - \text{RSS}}{\sigma^2} \sim \chi_p^2$$

and

$$\frac{\frac{\text{TSS} - \text{RSS}}{p}}{\frac{\text{RSS}}{n-p-1}} \sim F_{p,n-p-1}$$

F -distribution computed explicitly in Extra Credit, HW1.

Last lecture: F -test general

- ▶ (m) denotes a sub-model obtained by a linear constraint on β
- ▶ Examples
 - ▶ $\beta_1 = \beta_2 = \dots = \beta_p$: $Y = \beta_0 + \beta_1(X_1 + X_2 + \dots + X_p) + \epsilon$
 - ▶ $\beta_1 = \beta_2$: $Y = \beta_0 + \beta_1(X_1 + X_2) + \beta_3X_3 + \dots + \beta_pX_p + \epsilon$
- ▶ Testing: \mathcal{H}_0 (reduced model) vs. \mathcal{H}_1 (complete model)
- ▶ $q < p$ is the number of explanatory variables in the reduced model
- ▶ Under \mathcal{H}_0 :

$$\frac{\text{RSS}_{(m)} - \text{RSS}}{\sigma^2} \sim \chi_{p-q}^2$$

and

$$\frac{\frac{\text{RSS}_{(m)} - \text{RSS}}{p-q}}{\frac{\text{RSS}}{n-p-1}} \sim F_{p-q, n-p-1}$$

Small digression: RSS, χ^2 and Cochran's Theorem

- ▶ Several times in the class we said that the distribution of RSS/σ^2 for linear regression has χ^2 distribution for Gaussian noise ϵ
 $(Y = f(X) + \epsilon)$

$$\frac{RSS}{\sigma^2} \sim \chi_{n-p-1}^2,$$

where $n - p - 1$ represents the degrees of freedom and p is the number of predictors

- ▶ How can we show/prove this? (Not needed for the grade.)
- ▶ **Cochran's Theorem (1934)** If $y_i, 1 \leq i \leq n$ are i.i.d. $\mathcal{N}(0, \sigma^2)$ gaussian and $A_j, j = 1, 2$ are idempotent ($A_j^2 = A_j$) and symmetric ($A_j^\top = A_j$) matrices such that $\text{rank}(A_j) = r_j$, $r_1 + r_2 = n$ and $A_1 + A_2 = I_n$, where I_n is $n \times n$ identity matrix, then the following variables are independent and have χ^2 distribution

$$\mathbf{y}^\top A_j \mathbf{y} \sim \sigma^2 \chi_{r_j}^2$$

Proof: For example, it can be found [here](#).

Application of Cochran's Theorem: Distribution of RSS

- Recall that $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$, where \mathbf{P} is the hat matrix

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$$

- Now, it is easy to check that \mathbf{P} is symmetric and idempotent, i.e. $\mathbf{P} = \mathbf{P}^\top$ and $\mathbf{P}^2 = \mathbf{P}$
- Next, the same is true for $\mathbf{I} - \mathbf{P}$ since it is a difference of 2 symmetric matrices and

$$(\mathbf{I} - \mathbf{P})^2 = \mathbf{I} - 2\mathbf{P} + \mathbf{P}^2 = \mathbf{I} - 2\mathbf{P} + \mathbf{P} = \mathbf{I} - \mathbf{P}$$

- Rank:** From linear algebra, it is known that rank of symmetric and idempotent matrices is equal to their trace ($\text{tr}(AB) = \text{tr}(BA)$)

$$\begin{aligned}\text{rank}(\mathbf{P}) &= \text{tr}(\mathbf{P}) = \text{trace}(\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top) \\ &= \text{trace}(\mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1}) = \text{trace}(\mathbf{I}_{p+1}) = p + 1\end{aligned}$$

Application of Cochran's Theorem: Distribution of RSS

- ▶ Then,

$$\text{rank}(\mathbf{I} - \mathbf{P}) = \text{tr}(\mathbf{I} - \mathbf{P}) = \text{tr}(\mathbf{I}) - \text{tr}(\mathbf{P}) = n - p - 1$$

- ▶ Finally, by applying Cochran's Theorem,

$$\begin{aligned} RSS &= (\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}}) \\ &= (\mathbf{y} - \mathbf{P}\mathbf{y})^\top (\mathbf{y} - \mathbf{P}\mathbf{y}) \\ &= \mathbf{y}^\top (\mathbf{I} - \mathbf{P})^\top (\mathbf{I} - \mathbf{P})\mathbf{y} \\ &= \mathbf{y}^\top (\mathbf{I} - \mathbf{P})^2 \mathbf{y} \\ &= \mathbf{y}^\top (\mathbf{I} - \mathbf{P})\mathbf{y} \sim \sigma^2 \chi_{n-p-1}^2 \end{aligned}$$

- ▶ Similarly, we can apply Cochran's Theorem to compute the distribution of TSS and ESS

Linear Model Selection: Analytical Approach

- ▶ Recall advertising example from last lecture
- ▶ Now, that we know the meaning of: standard error, t-value, F-value, p-value, R^2 , we can completely understand the output of the linear model fit function, $\text{lm}(\cdot)$.

```
> lm2<-lm(adv$Sales~adv$TV+adv$Radio+adv$Newspaper)
> summary(lm2)

Call:
lm(formula = adv$Sales ~ adv$TV + adv$Radio + adv$Newspaper)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.8277 -0.8908  0.2418  1.1893  2.8292 

Coefficients:
            Estimate Std. Error t value  Pr(>|t|)    
(Intercept) 2.938889  0.311908   9.422 <2e-16 ***
adv$TV       0.045765  0.001395  32.809 <2e-16 ***
adv$Radio    0.188530  0.008611  21.893 <2e-16 ***
adv$Newspaper -0.001037  0.005871  -0.177    0.86  
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.686 on 196 degrees of freedom
Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956 
F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

Linear Model Selection: Analytical Approach

- ▶ Hence, the model with one less predictor (without the newspaper) might be just as good

```
> summary(lm(adv$Sales~adv$TV+adv$Radio))

Call:
lm(formula = adv$Sales ~ adv$TV + adv$Radio)

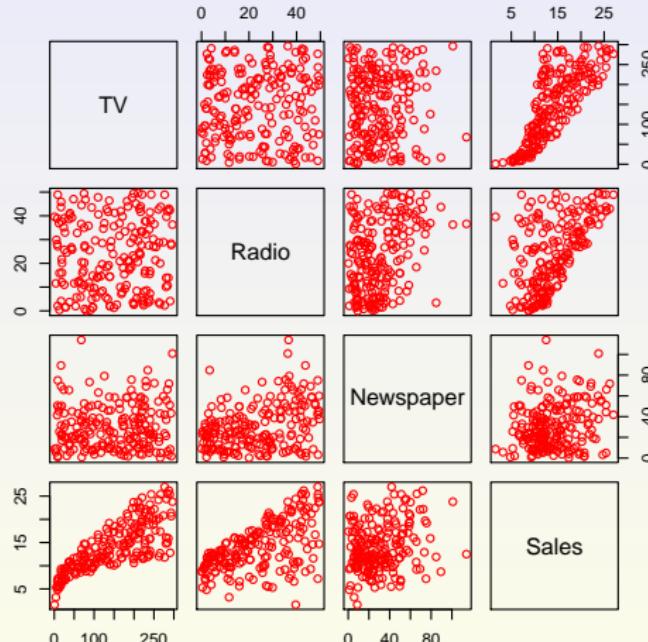
Residuals:
    Min      1Q  Median      3Q     Max 
-8.7977 -0.8752  0.2422  1.1708  2.8328 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 2.92110   0.29449   9.919 <2e-16 ***
adv$TV      0.04575   0.00139  32.909 <2e-16 ***
adv$Radio    0.18799   0.00804  23.382 <2e-16 ***  
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.681 on 197 degrees of freedom
Multiple R-squared:  0.8972, Adjusted R-squared:  0.8962 
F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

Linear Model Selection: Visual Examination

- ▶ We made a lot of assumptions in this model
- ▶ Note: examine the data visually to see if the model makes sense



- ▶ How do we find the best model in general?

Linear Model Selection and Regularization

Additional motivation:

- ▶ Prediction
 - ▶ High-dimensional data, $p \gtrsim n$ – overfitting to the training data
 - ▶ Cannot use the plain vanilla least squares
- ▶ Model interpretability
 - ▶ Hard to interpret model with many predictors
 - ▶ Focus on most important variables
- ▶ Idea: Modify least squares
- ▶ Agenda:
 - ▶ Subset selection
 - ▶ Shrinkage methods
 - ▶ Dimension reduction techniques (next class)

Model Selection: Bias-Variance Trade-off

Test Error, aka Generalization Error can be decomposed as:

- Let x_0 be a test (unseen) point and $y_0 = f(x_0) + \epsilon_0$

$$\begin{aligned}\text{Err}(x_0) &= \mathbb{E} \left(y_0 - \hat{f}(x_0) \right)^2 \\ &= \mathbb{E} \left(f(x_0) + \epsilon_0 - \hat{f}(x_0) \right)^2 \\ &= \sigma^2 + \mathbb{E} \left(f(x_0) - \mathbb{E} \hat{f}(x_0) - \hat{f}(x_0) + \mathbb{E} \hat{f}(x_0) \right)^2 \\ &= \sigma^2 + \left(f(x_0) - \mathbb{E} \hat{f}(x_0) \right)^2 + \text{Var}(\hat{f}(x_0)) \\ &= \sigma^2 + \left(\text{Bias}(\hat{f}(x_0)) \right)^2 + \text{Var}(\hat{f}(x_0))\end{aligned}$$

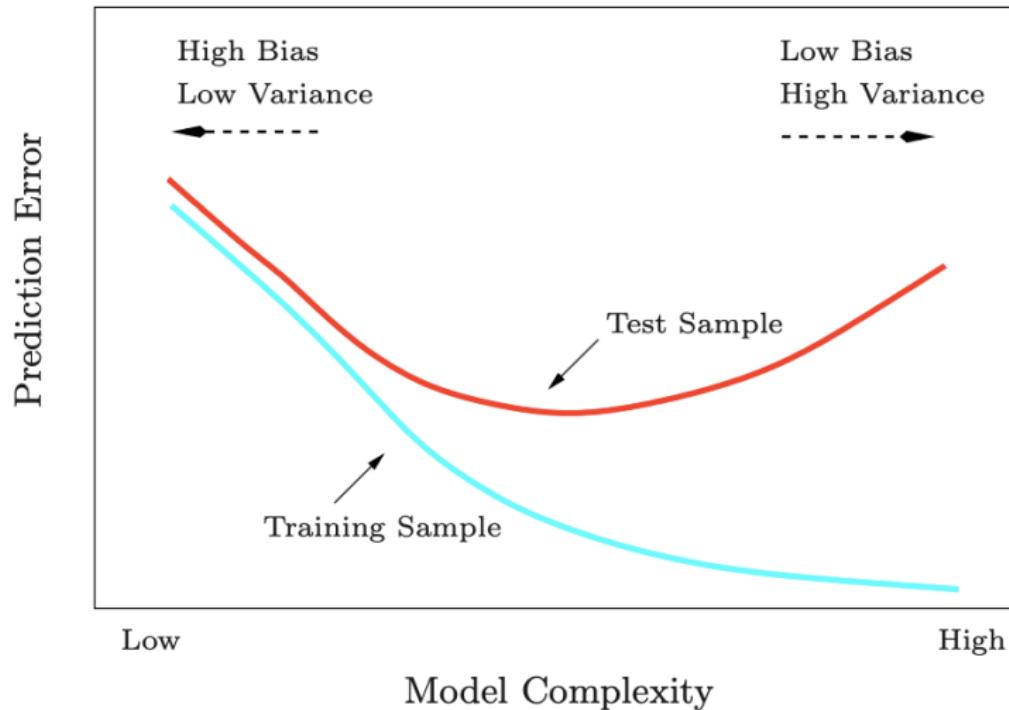
- Linear regression: recall $\hat{f}(x_0) = x_0^\top \hat{\beta} = x_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$

$$\text{Var}(\hat{f}(x_0)) = \sigma^2 \mathbb{E} \left(x_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} x_0 \right) \approx \frac{p}{n} \sigma^2$$

assuming \mathbf{X} random, zero mean and n large: $\mathbf{X}^\top \mathbf{X} \approx n \text{Cov}(X)$.

Increasing p increases the variance and in general reduces the bias

Testing and Training Error versus Model Complexity



Model Selection: Deciding on the important variables

We have seen analytical approaches via F/t-statistics

Numerical approaches:

- ▶ Need a criteria that balance training error and model size
- ▶ Several approaches
 - ▶ Best subsets selection
 - ▶ Consider all 2^p models
 - ▶ Infeasible when p is large
 - ▶ Forward selection
 - ▶ Start with a null model – no predictors
 - ▶ Add predictors one-by-one
 - ▶ Stopping criterion
 - ▶ Backward selection
 - ▶ Start with a full model – p predictors
 - ▶ Eliminate predictors one-by-one
 - ▶ Stopping criterion

Best Subset Selection

- ▶ Algorithm
 - ▶ Let \mathcal{M}_0 denote the null model (no predictors, sample mean prediction)
 - ▶ For $k = 1, 2, \dots, p$:
 - ▶ Fit all $\binom{p}{k}$ models that contain exactly k predictors
 - ▶ Let \mathcal{M}_k be the best of these $\binom{p}{k}$ models in terms of the smallest RSS (equivalently the largest R^2)
 - ▶ Select the best model from among $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$ using some criterion
- ▶ Number of models is 2^p
- ▶ Example: if $p = 20$, number of models to check
 $2^{20} = 1,048,576$

Prostate cancer data set

Data frame with 97 observations on the following 10 variables:

- ▶ **lcavol** - log cancer volume
- ▶ **lweight** - log prostate weight
- ▶ **age** in years
- ▶ **lbph** - log of the amount of benign prostatic hyperplasia
- ▶ **svi** - seminal vesicle invasion
- ▶ **lcp** - log of capsular penetration
- ▶ **gleason** - a numeric vector
- ▶ **pgg45** - percent of Gleason score 4 or 5
- ▶ **lpsa** - **response**: log of prostate specific antigen (PSA)
- ▶ **train** logical True/False vector

Loading libraries and prostate data

```
# Loading libraries/packages and data:  
  
library(ISLR)  
library(ElemStatLearn)  
data(prostate)  
  
# Checking "prostate" data  
# Data manuals:  
# https://cran.r-project.org/web/packages/ElemStatLearn/ElemStatLearn.pdf  
# https://cran.r-project.org/web/packages/ISLR/ISLR.pdf  
  
fix(prostate)  
str(prostate)  
cor(prostate[,1:8])  
pairs(prostate[,1:9], col="violet")  
  
# Separating the training and test data:  
train <- subset( prostate, train==TRUE )[,1:9]  
test <- subset( prostate, train=FALSE )[,1:9]
```

Best Subset Selection

```
# Loading "leaps" library/package for the best subset selection
library(leaps)

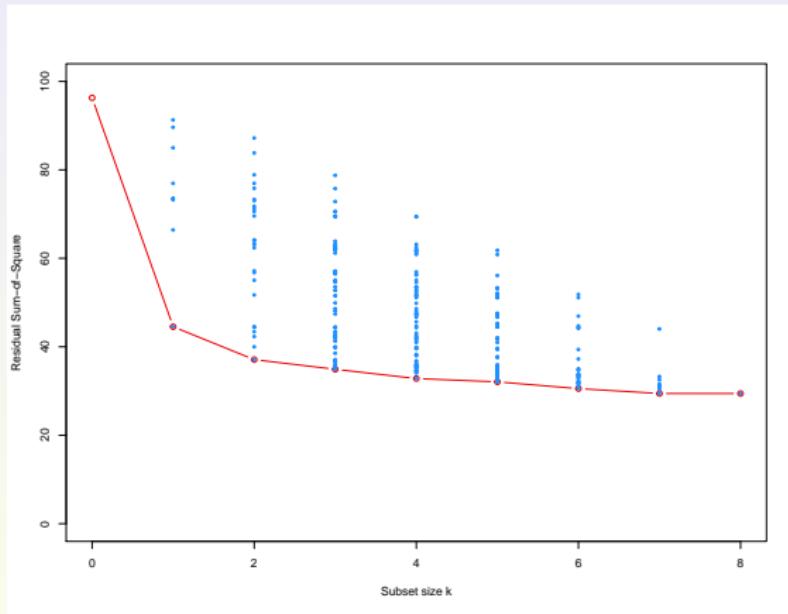
# Computing all combinations using "regsubsets"
prostate.leaps <- regsubsets( lpsa ~ . , data=train, nbest=70, really.big=TRUE )
prostate.leaps.sum <- summary( prostate.leaps )
prostate.models <- prostate.leaps.sum$which
prostate.models.size <- as.numeric(attr(prostate.models, "dimnames")[[1]])
hist( prostate.models.size )

#Extracting all and the best RSS
prostate.models.rss <- prostate.leaps.sum$rss
prostate.models.best.rss <- tapply( prostate.models.rss, prostate.models.size, min )
prostate.models.best.rss

# Adding the result with no X-s, only intercept (beta0) model
prostate.dummy <- lm( lpsa ~ 1, data=train )
prostate.models.best.rss <- c(sum(resid(prostate.dummy)^2),prostate.models.best.rss)
```

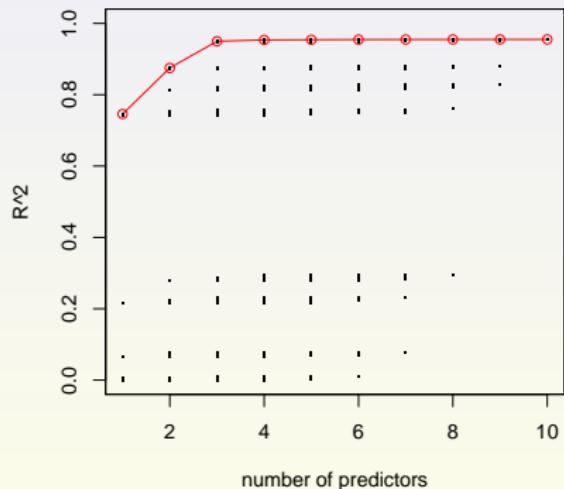
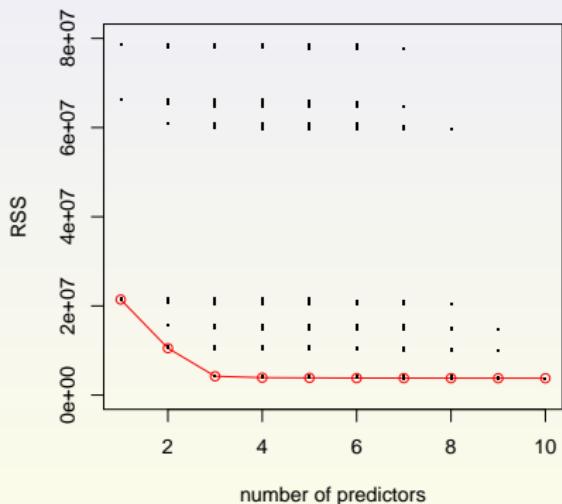
Best Subset Selection

```
# Making the plot  
plot( 0:8, prostate.models.best.rss, ylim=c(0, 100),  
      type="b", xlab="Subset size k", ylab="Residual Sum-of-Square", col="red2" )  
points( prostate.models.size, prostate.models.rss, pch=20, col="dodgerblue",cex=0.7 )
```



Another Example

- ▶ Credit card data set



More Examples

```
> library(ISLR)
> names(Hitters)
[1] "AtBat"      "Hits"       "HmRun"      "Runs"       "RBI"        "Walks"      "Years"       "CAtBat"
[9] "CHits"      "CHmRun"     "CRuns"      "CRBI"       "CWalks"     "League"     "Division"    "PutOuts"
[17] "Assists"    "Errors"     "Salary"     "NewLeague"
> dim(Hitters)
[1] 322 20
> sum(is.na(Hitters$Salary))
[1] 59
> Hitters<-na.omit(Hitters)
> library(leaps)
> regfit.full<-regsubsets(Salary~.,data=Hitters)
> summary(regfit.full)
```

1 subsets of each size up to 8

Selection Algorithm: exhaustive

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	LeagueN	DivisionW			
1	(1)	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" *	" "	" "	" "	" "
2	(1)	" "	" *"	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" *	" "	" "	" "	" "
3	(1)	" "	" *"	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" *	" "	" "	" "	" "
4	(1)	" "	" *"	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" *	" "	" "	" "	" *
5	(1)	" *"	" *"	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" *	" "	" "	" "	" *
6	(1)	" *"	" *"	" "	" "	" *	" "	" "	" "	" "	" "	" "	" "	" *	" "	" "	" "	" *
7	(1)	" "	" *"	" "	" "	" *	" "	" *	" *	" *	" *	" *	" "	" "	" "	" "	" "	" *
8	(1)	" *"	" *"	" "	" "	" *	" "	" "	" "	" "	" *	" *	" *	" "	" *	" "	" "	" *
	PutOuts	Assists	Errors	NewLeagueN														
1	(1)	" "	" "	" "	" "													
2	(1)	" "	" "	" "	" "													
3	(1)	" *"	" "	" "	" "													
4	(1)	" *"	" "	" "	" "													
5	(1)	" *"	" "	" "	" "													
6	(1)	" *"	" "	" "	" "													
7	(1)	" *"	" "	" "	" "													
8	(1)	" *"	" "	" "	" "													

Forward Selection

- ▶ Reduce computational complexity by forfeiting optimality
- ▶ Algorithm
 - ▶ Let \mathcal{M}_0 denote the null model (no predictors, sample mean prediction)
 - ▶ for $k = 0, 1, \dots, p - 1$
 - ▶ Fit all $p - k$ models that augment the predictors in \mathcal{M}_k with one additional predictor
 - ▶ Let \mathcal{M}_{k+1} be the best of these $p - k$ models in terms of the smallest RSS (equivalently the largest R^2)
 - ▶ Select the best model from among $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$ using some criterion
- ▶ Greedy : not optimal, but tractable
- ▶ Number of models is $1 + p(p + 1)/2 = O(p^2) \ll 2^p$
- ▶ If $p > n$, we can construct $\mathcal{M}_0, \dots, \mathcal{M}_n$ models only

Example: Forward Selection is not optimal

```
> regfit.fwd<-regsubsets(Salary~.,data=Hitters,nvmax=19,method = "forward")
> summary(regfit.fwd)
.

Selection Algorithm: forward
   AtBat Hits HmRun Runs RBI Walks Years CatBat CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW
1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " "
2 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " " " "
3 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " " " "
4 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " " " "
5 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " " "
6 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " " "
7 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
8 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
9 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
10 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
11 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
12 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
13 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
14 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
15 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
16 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
17 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
18 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "
19 ( 1 ) "*" " " " " " " " " " " " " " " " " " " " "

   PutOuts Assists Errors NewLeagueN
1 ( 1 ) " " " " " "
2 ( 1 ) " " " " " "
3 ( 1 ) "*" " " " " "
4 ( 1 ) "*" " " " " "
5 ( 1 ) "*" " " " " "
6 ( 1 ) "*" " " " " "
7 ( 1 ) "*" " " " " "
8 ( 1 ) "*" " " " " "
9 ( 1 ) "*" " " " " "
10 ( 1 ) "*" " " " " "
11 ( 1 ) "*" " " " " "
12 ( 1 ) "*" " " " " "
13 ( 1 ) "*" " " " " "
14 ( 1 ) "*" " " " " "
15 ( 1 ) "*" " " " " "
16 ( 1 ) "*" " " " " "
17 ( 1 ) "*" " " " " "
18 ( 1 ) "*" " " " " "
19 ( 1 ) "*" " " " " "
```

Feature selection measures

- ▶ $p + 1$ models: $\mathcal{M}_0, \dots, \mathcal{M}_p$. Which one is the best?
- ▶ RSS and R^2 estimate the training error, not the testing error
- ▶ Two approaches:
 - ▶ Indirect (adjust the training error)
 - ▶ Direct: validation, cross-validation (next week)

Indirect measures: C_p , AIC and BIC

- ▶ Model with $d \leq p$ predictors
- ▶ Mallow's C_p :

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

where $\hat{\sigma}$ is an estimator for the variance of noise (estimated on a model containing all predictors)

- ▶ Akaike information criteria (AIC):

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

- ▶ Bayesian AIC (BIC):

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

- ▶ Heuristic: select a model with the lowest C_p , AIC, BIC

Indirect measures: Adjusted R^2

- ▶ Recall

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

- ▶ Adjusted R^2

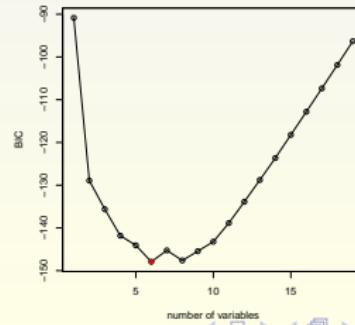
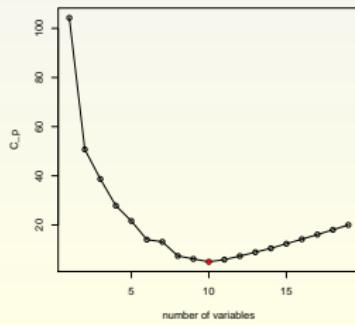
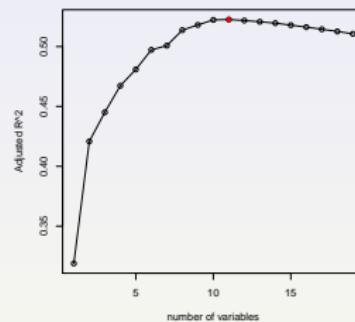
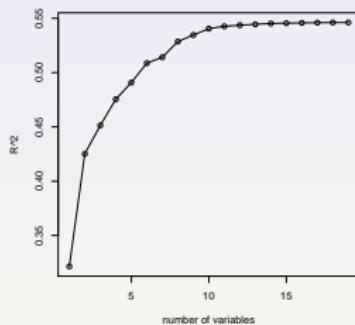
$$\text{Adj}R^2 = 1 - \frac{\text{RSS}/(n - d - 1)}{\text{TSS}/(n - 1)}$$

- ▶ Heuristic: $\max \text{Adj}R^2$
- ▶ Equivalent to

$$\min \frac{\text{RSS}}{n - d - 1}$$

Example

```
> regfit.full<-regsubsets(Salary~.,data=Hitters,nvmax = 19)
> reg.summary<-summary(regfit.full)
> names(reg.summary)
[1] "which"      "rsq"        "rss"        "adjr2"      "cp"         "bic"        "outmat"    "obj"
> reg.summary$rsq
[1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227 0.5285569 0.5346124 0.5404950
[11] 0.5426153 0.5436302 0.5444570 0.5452164 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```



Shrinkage methods

- ▶ An alternative to subset selection
- ▶ Idea: Regularize/constrain coefficients

- ▶ Two widely-used methods:
 - ▶ Ridge regression
 - ▶ LASSO (Least Absolute Shrinkage and Selection Operator)

Ridge Regression

- ▶ OLS: minimize RSS

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

- ▶ Ridge Regression: minimize $(\text{RSS} + \text{shrinkage penalty})$

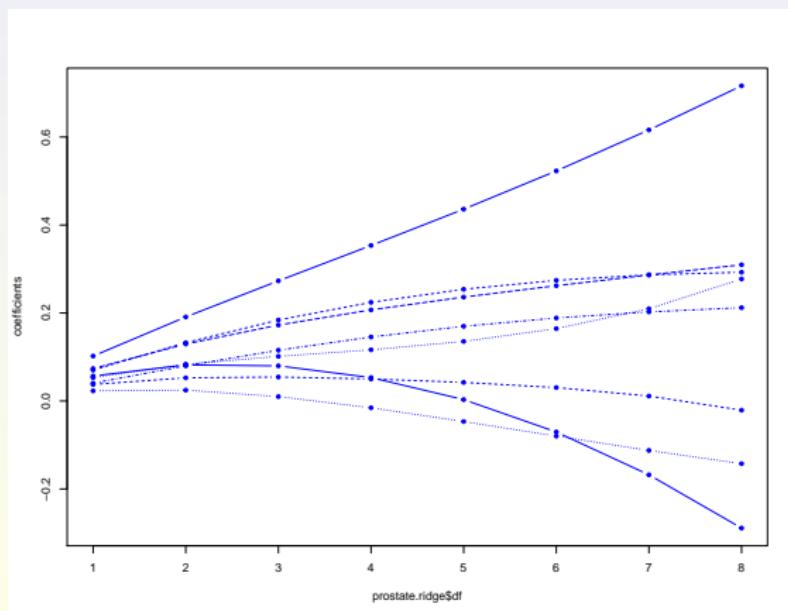
$$\text{RSS} + \lambda \sum_{j=1}^p \beta_j^2$$

- ▶ λ is a tuning parameter: β_j^λ for each λ
- ▶ β_0 is not in the penalty
- ▶ Data normalization:

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

Prostate: Ridge regression

```
# Calling simple.ridge() function which is part of "ElemStatLearn" package  
# Ridge functions in other packages:  
# MASS: lm.ridge()  
# mda : gen.ridge()  
prostate.ridge <- simple.ridge( train[,1:8], train[,9], df=1:8 )  
  
# plot  
matplot( prostate.ridge$df, t(prostate.ridge$beta), type="b",  
         col="blue", pch=20, ylab="coefficients" )
```



Another example: Credit

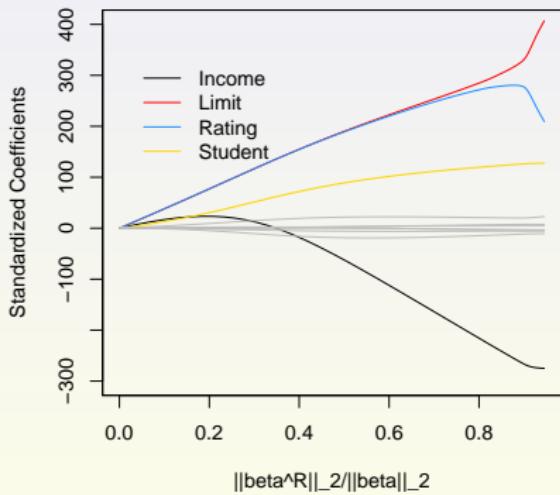
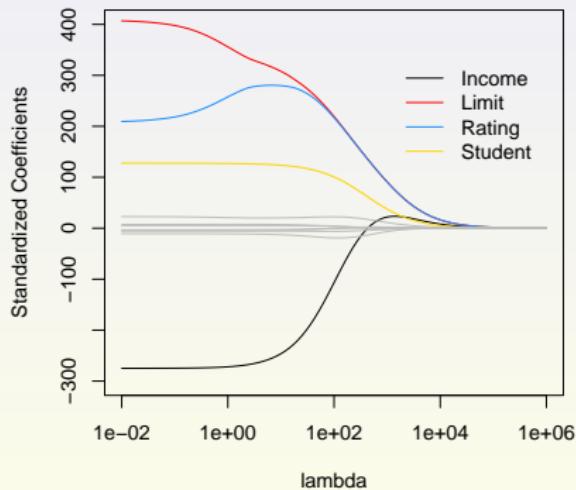
- ▶ `glmnet` function performs ridge regression, Lasso, ...
- ▶ Inputs should be numerical variables

```
> head(credit)
   X Income Limit Rating Cards Age Education Gender Student Married Ethnicity Balance
1 1 14.891 3606 283 2 34 11 Male No Yes Caucasian 333
2 2 106.025 6645 483 3 82 15 Female Yes Yes Asian 903
3 3 104.593 7075 514 4 71 11 Male No No Asian 580
4 4 148.924 9504 681 3 36 11 Female No No Asian 964
5 5 55.882 4897 357 2 68 16 Male No Yes Caucasian 331
6 6 80.180 8047 569 4 77 10 Male No No Caucasian 1151
> library(glmnet)
> y<-credit$Balance
> x<-model.matrix(Balance~.,credit)[,-c(1,2)]
> head(x)
   Income Limit Rating Cards Age Education GenderFemale StudentYes MarriedYes EthnicityAsian EthnicityCaucasian
1 14.891 3606 283 2 34 11 0 0 1 0 1
2 106.025 6645 483 3 82 15 1 1 1 1 0
3 104.593 7075 514 4 71 11 0 0 0 0 0
4 148.924 9504 681 3 36 11 1 0 0 1 0
5 55.882 4897 357 2 68 16 0 0 0 1 1
6 80.180 8047 569 4 77 10 0 0 0 0 1
> grid<-10^seq(-2,6,length=100)
> credit.ridge<-glmnet(x,y,alpha=0,lambda=grid)

> credit.ridge$lambda[50]
[1] 109.7499
> coef(credit.ridge)[,50]
   (Intercept)           Income          Limit          Rating          Cards          Age
-295.64236303 -2.80221111  0.09282459  1.37158760  16.34674532 -1.10359021
   Education      GenderFemale  StudentYes  MarriedYes EthnicityAsian EthnicityCaucasian
   -0.14925505  0.13149217  328.63578560 -12.37346466  8.29714568  7.20848754
> sqrt(sum(coef(credit.ridge)[-1,50]^2))
[1] 329.4747
> credit.ridge$lambda[80]
[1] 0.4132012
> coef(credit.ridge)[,80]
   (Intercept)           Income          Limit          Rating          Cards          Age
-488.0819858 -7.7661932  0.1634360  1.5382779  15.7906711 -0.6229323
   Education      GenderFemale  StudentYes  MarriedYes EthnicityAsian EthnicityCaucasian
   -0.9901752 -10.5859576  423.9301301 -9.5031779  17.4513942  10.1743680
> sqrt(sum(coef(credit.ridge)[-1,80]^2))
[1] 425.0184
```

Example: Credit

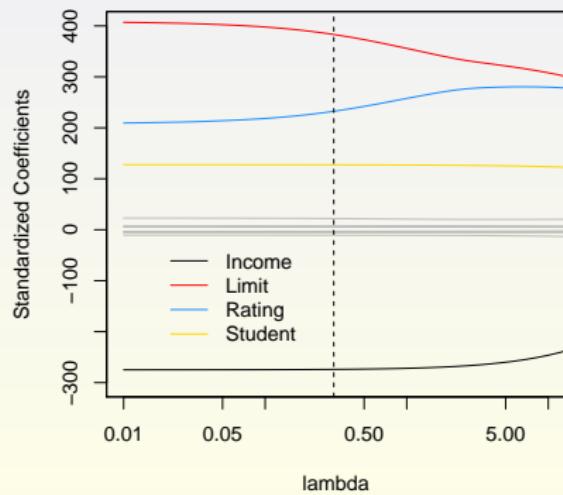
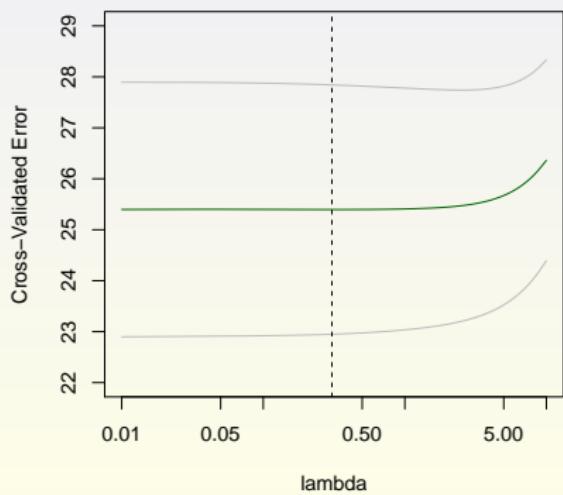
- ▶ Standardized coefficients



Example: Credit

► Optimal λ

```
> set.seed(200)
> grid<-10^seq(1,-2,length=100)
> cvridge.out<-cv.glmnet(x,y,alpha=0,lambda = grid)
> cvridge.out$lambda.min
[1] 0.3053856
```



Lasso

- ▶ Ridge regression: Still p (shrunk) predictors
- ▶ Inference

- ▶ Lasso: minimize

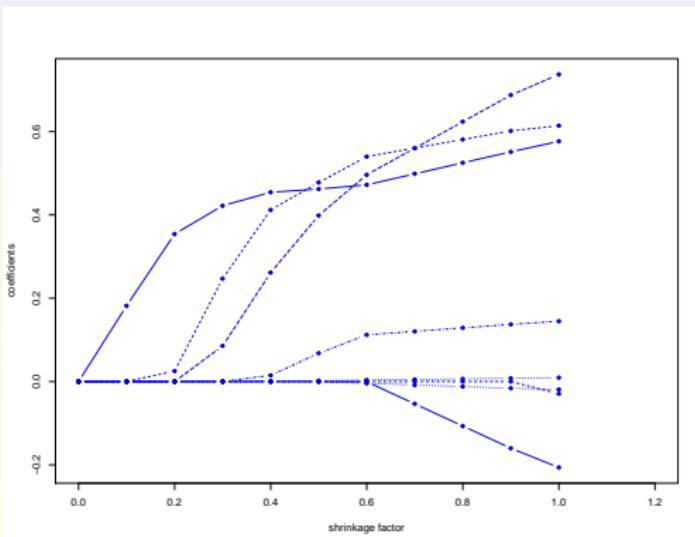
$$\text{RSS} + \lambda \sum_{j=1}^p |\beta_j|$$

- ▶ Rationale for absolute values: Some of β_j 's will be equal to 0
- ▶ Data normalization

Prostate: Lasso

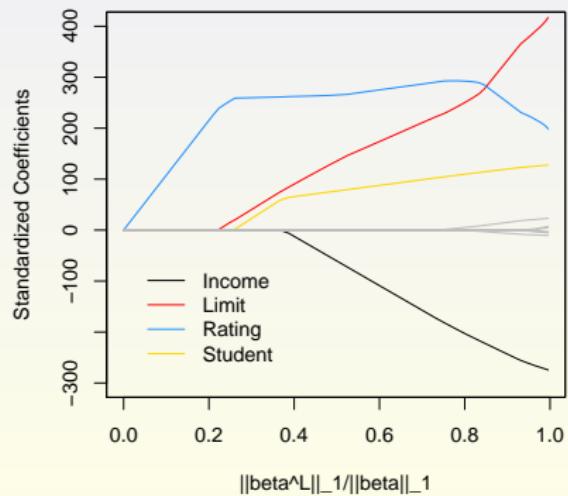
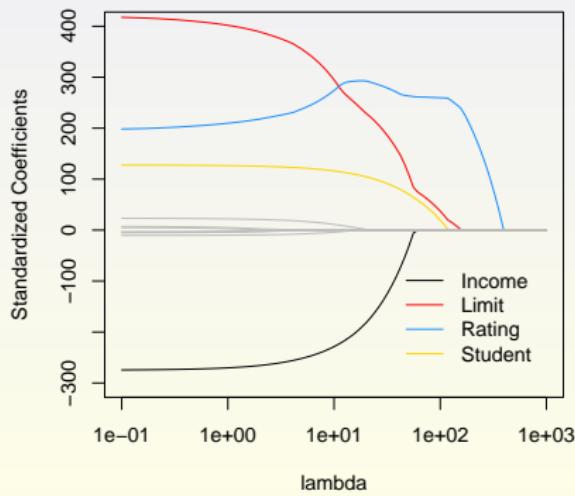
```
# loading package "lasso2"
library(lasso2)
prostate.lasso <- l1ce( lpsa ~ ., data=train, trace=TRUE, sweep.out=~1,
                      bound=seq(0,1,by=0.1) )
prostate.lasso.coef <- sapply(prostate.lasso, function(x) x$coef)
colnames(prostate.lasso.coef) <- seq( 0,1,by=0.1 )

# plot
matplot( seq(0,1,by=0.1), t(prostate.lasso.coef[-1,]), type="b",
          xlab="shrinkage factor", ylab="coefficients",
          xlim=c(0, 1.2), col="blue", pch=20 )
```



Another example: Credit

```
> grid<-10^seq(-1,3,length=100)
> credit.lasso<-glmnet(x,y,alpha=1,lambda=grid)
> credit.lasso$lambda[50]
[1] 10.47616
> coef(credit.lasso)[,50]
(Intercept)           Income          Limit        Rating        Cards         Age
-468.5205454      -6.4263083      0.1254060     1.7922956    7.7155425   -0.2187646
Education       GenderFemale  StudentYes
0.0000000      0.0000000      384.5532767
EthnicityAsian EthnicityCaucasian
0.0000000      0.0000000
Age
```



Ridge vs. Lasso vs. Best subset

- ▶ Equivalent formulations

- ▶ Ridge:

$$\min_{\beta} \text{RSS} \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 \leq s$$

- ▶ Lasso:

$$\min_{\beta} \text{RSS} \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq s$$

- ▶ Best subset selection:

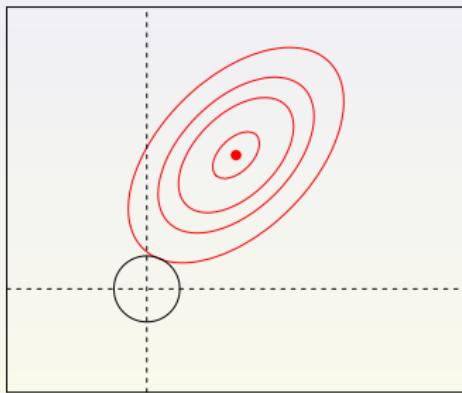
$$\min_{\beta} \text{RSS} \quad \text{subject to} \quad \sum_{j=1}^p 1_{\{\beta_j \neq 0\}} \leq s$$

Lasso vs. Ridge regression

- ▶ Geometry

Ridge regression

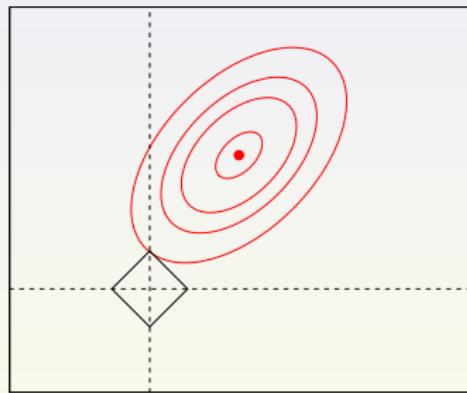
β_2



β_1

Lasso

β_2



β_1

Bias-variance trade-off for linear ridge regression

- ▶ Normality assumption: i.i.d. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
- ▶ Least squares: $\beta_{\text{LS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ is unbiased, but might have high variance:

$$\mathbb{E}\beta_{\text{LS}} = \boldsymbol{\beta} \quad \text{and} \quad \text{Cov}(\beta_{\text{LS}}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}$$

- ▶ Ridge regression: $\beta_{\text{RR}} = (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ is biased, but might have lower variance:

$$\mathbb{E}\beta_{\text{RR}} = (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta}$$

and

$$\text{Cov}(\beta_{\text{RR}}) = \sigma^2 \mathbf{Z} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{Z}^\top,$$

where

$$\mathbf{Z} = (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X}$$

- ▶ Note: $(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})$ is full rank: unique solution β_{RR} always exists
Can be solved in dual dot product/kernel formulation for high-dim data ($p \gg n$): $\boldsymbol{\alpha} = (\lambda \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}$ - more on that in the future.

Bias-variance tradeoff for linear regression

- ▶ Least squares or Ridge regression?
- ▶ How well our solution generalizes to new data?
Let (\mathbf{x}_0, y_0) be future data: \mathbf{x}_0 is known, but not y_0
- ▶ Predictions:
 - ▶ Least squares: $\mathbf{x}_0^\top \boldsymbol{\beta}_{\text{LS}}$
 - ▶ Ridge regression: $\mathbf{x}_0^\top \boldsymbol{\beta}_{\text{RR}}$
- ▶ Expected squared error of the prediction:

$$\mathbb{E} \left[(y_0 - \mathbf{x}_0^\top \hat{\boldsymbol{\beta}})^2 \mid \mathbf{X}, \mathbf{x}_0 \right]$$

- ▶ y and y_0 are Gaussian with the true (but unknown) $\boldsymbol{\beta}$

Bias-variance tradeoff for linear regression

- ▶ Assuming conditioning on \mathbf{X} and \mathbf{x}_0 :

$$\mathbb{E}(y_0 - \mathbf{x}_0^\top \hat{\boldsymbol{\beta}})^2 = \mathbb{E}y_0^2 - 2\mathbb{E}y_0 \mathbf{x}_0^\top \mathbb{E}\hat{\boldsymbol{\beta}} + \mathbf{x}_0^\top \mathbb{E}[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^\top] \mathbf{x}_0$$

- ▶ Since $\mathbb{E}y_0^2 = (\boldsymbol{\beta}^\top \mathbf{x}_0)^2 + \sigma^2$ and

$$\mathbb{E}\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^\top = \mathbb{E}\hat{\boldsymbol{\beta}}\mathbb{E}\hat{\boldsymbol{\beta}}^\top + \text{Cov}(\hat{\boldsymbol{\beta}})$$

- ▶ We have

$$\begin{aligned}\mathbb{E}(y_0 - \mathbf{x}_0^\top \hat{\boldsymbol{\beta}})^2 &= \sigma^2 + \mathbf{x}_0^\top (\boldsymbol{\beta} - \mathbb{E}\hat{\boldsymbol{\beta}})(\boldsymbol{\beta} - \mathbb{E}\hat{\boldsymbol{\beta}})^\top \mathbf{x}_0 + \mathbf{x}_0^\top \text{Cov}(\hat{\boldsymbol{\beta}}) \mathbf{x}_0 \\ &= \text{noise} + \text{bias}^2 + \text{variance}\end{aligned}$$

- ▶ LS: $\mathbb{E}(y_0 - \mathbf{x}_0^\top \hat{\boldsymbol{\beta}})^2 = \sigma^2 + \sigma^2 \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_0$

Reading:

ISL: Finish reading Chapter 6

ESL: Chapter 3: Sections 3.3 - end of chapter.

Homework: Homework 1 next Tue - Sep 27.

EECS E6690: Statistical Learning for Biological and Information Systems Lecture 4: PCA, Nonlinear Models and Model Validation

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.

Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Last lecture: Linear Model Selection and Regularization

Motivation: Find the smallest/simplest model

- ▶ Problem with complex models
 - ▶ Overfitting (especially for $n \leq p$)
 - ▶ High testing error
- ▶ Model interpretability
 - ▶ Hard to interpret model with many predictors
 - ▶ Focus on most important variables
- ▶ Approaches for reduction:
 - ▶ Subset selection
 - ▶ Shrinkage methods
 - ▶ Dimension reduction techniques (today)

Last lecture: Bias-Variance trade-off

- ▶ “Test error”: test variable $y_0 = f(x_0) + \epsilon_0$

$$\begin{aligned}\text{Err}(x_0) &= \mathbb{E} (y_0 - \hat{f}(x_0))^2 \\ &= \mathbb{E} (f(x_0) + \epsilon - \hat{f}(x_0))^2 \\ &= \sigma^2 + \mathbb{E} (f(x_0) - \mathbb{E} \hat{f}(x_0) - \hat{f}(x_0) + \mathbb{E} \hat{f}(x_0))^2 \\ &= \sigma^2 + (f(x_0) - \mathbb{E} \hat{f}(x_0))^2 + \text{Var}(\hat{f}(x_0)) \\ &= \sigma^2 + (\text{Bias}(\hat{f}(x_0)))^2 + \text{Var}(\hat{f}(x_0))\end{aligned}$$

- ▶ For linear models, sample estimation leads to
(see equation (7.12) in the [ESL] book)

$$\frac{1}{n} \sum_{i=1}^n \text{Err}(x_i) = \sigma^2 + \frac{1}{n} \sum_{i=1}^n (\text{Bias}(\hat{f}(x_i)))^2 + \frac{p}{n} \sigma^2$$

- ▶ Increasing p reduces the bias but increases the variance

Last lecture: Subset selection

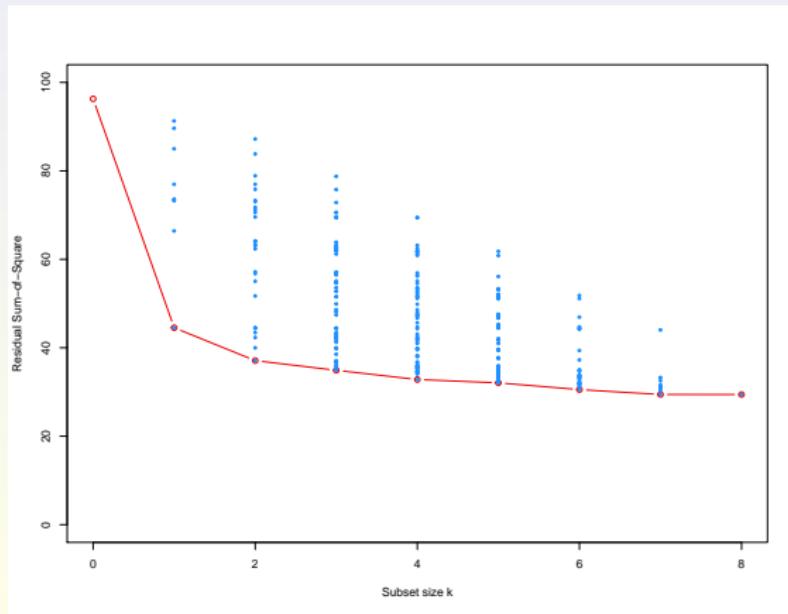
Deciding on the important variables:

- ▶ Need a criteria that balance training error and model size
- ▶ Several approaches
 - ▶ All (or best) subsets selection
 - ▶ Consider all 2^p models
 - ▶ Infeasible when p is large
 - ▶ Forward selection
 - ▶ Start with a null model – no predictors
 - ▶ Add predictors one-by-one
 - ▶ Stopping criterion
 - ▶ Backward selection
 - ▶ Start with a full model – p predictors
 - ▶ Eliminate predictors one-by-one
 - ▶ Stopping criterion

Last lecture: Best subset selection

```
# Making the plot  
plot( 0:8, prostate.models.best.rss, ylim=c(0, 100),  
      type="b", xlab="Subset size k", ylab="Residual Sum-of-Square", col="red2" )  
points( prostate.models.size, prostate.models.rss, pch=20, col="dodgerblue", cex=0.7 )
```

Redo this with "glmnet" library: `glmnet()` function does both ridge ($\alpha=0$) and lasso ($\alpha=1$), and in-between.



Last lecture: Suboptimal (greedy) selection

Forward (or backward) stepwise selection

- ▶ Reduce computational complexity by forfeiting optimality
- ▶ Algorithm
 - ▶ Let \mathcal{M}_0 denote the null model (no predictors, sample mean prediction)
 - ▶ for $k = 0, 1, \dots, p - 1$
 - ▶ Fit all $p - k$ models that augment the predictors in \mathcal{M}_k with one additional predictor
 - ▶ Let \mathcal{M}_{k+1} be the best of these $p - k$ models in terms of the smallest RSS (equivalently the largest R^2)
 - ▶ Select the best model from among $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$ using some criterion
- ▶ Greedy
- ▶ Number of models is $1 + p(p + 1)/2$
- ▶ If $p > n$, we can construct $\mathcal{M}_0, \dots, \mathcal{M}_n$ models only

Last lecture: Regularization (shrinkage) methods

- ▶ An alternative to subset selection
- ▶ Idea: Regularize/constrain coefficients

- ▶ Two widely-used methods:
 - ▶ Ridge regression
 - ▶ LASSO (least absolute shrinkage and selection operator)

Last lecture: Ridge

- ▶ OLS: minimize RSS

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

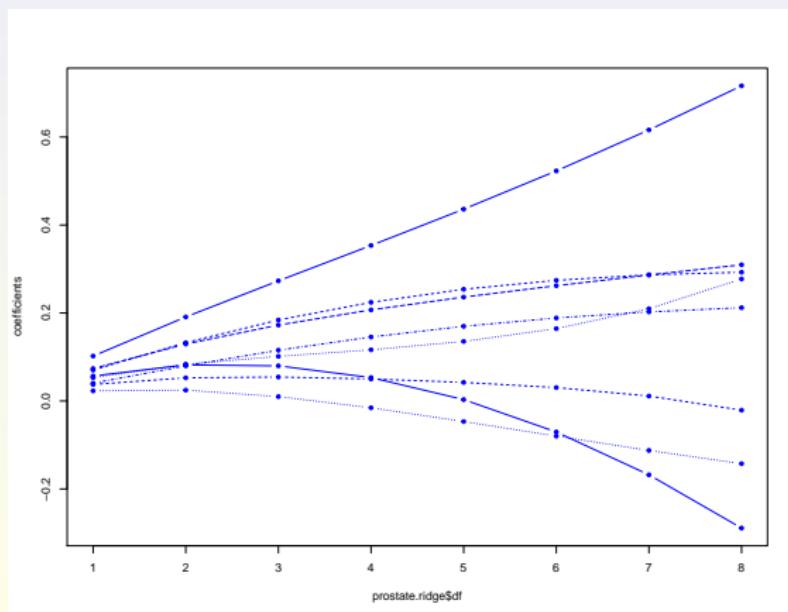
- ▶ Ridge regression: minimize (RSS + shrinkage penalty)

$$\text{RSS} + \lambda \sum_{j=1}^p \beta_j^2$$

- ▶ λ is a tuning parameter: β_j^λ for each λ
- ▶ β_0 is not in the penalty

Prostate: Ridge regression

```
# Calling simple.ridge() function which is part of "ElemStatLearn" package  
# Ridge functions in other packages:  
# MASS: lm.ridge()  
# mda : gen.ridge()  
prostate.ridge <- simple.ridge( train[,1:8], train[,9], df=1:8 )  
  
# plot  
matplot( prostate.ridge$df, t(prostate.ridge$beta), type="b",  
         col="blue", pch=20, ylab="coefficients" )
```



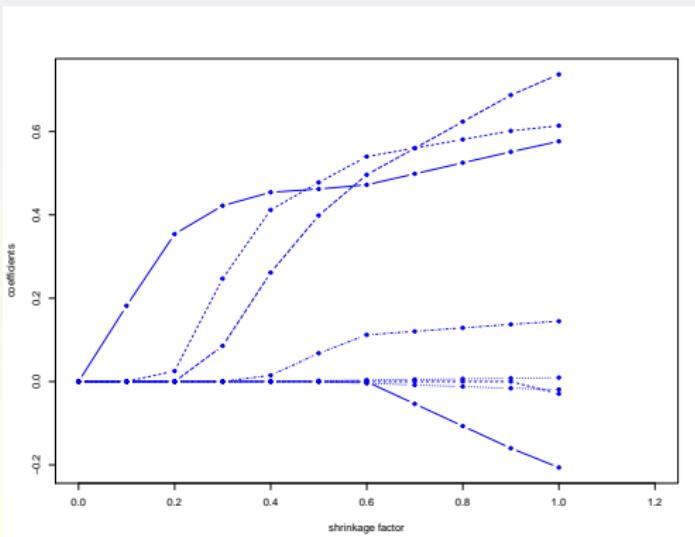
Last lecture: Lasso

- ▶ Ridge regression: Still p (shrunk) predictors
 - ▶ Inference
-
- ▶ Lasso: minimize
$$\text{RSS} + \lambda \sum_{j=1}^p |\beta_j|$$
 - ▶ Rationale for absolute values: Some of β_j 's will be equal to 0
 - ▶ Data normalization

Prostate: Lasso

```
# loading package "lasso2"
library(lasso2)
prostate.lasso <- l1ce( lpsa ~ ., data=train, trace=TRUE, sweep.out=~1,
                      bound=seq(0,1,by=0.1) )
prostate.lasso.coef <- sapply(prostate.lasso, function(x) x$coef)
colnames(prostate.lasso.coef) <- seq( 0,1,by=0.1 )

# plot
matplot( seq(0,1,by=0.1), t(prostate.lasso.coef[-1,]), type="b",
          xlab="shrinkage factor", ylab="coefficients",
          xlim=c(0, 1.2), col="blue", pch=20 )
```



Lasso vs. Ridge regression

- ▶ Equivalent formulations

- ▶ Ridge (ℓ_2 penalty):

$$\min_{\boldsymbol{\beta}} \text{RSS} \quad \text{subject to } \sum_{j=1}^p \beta_j^2 \leq s$$

- ▶ Lasso (ℓ_1 penalty):

$$\min_{\boldsymbol{\beta}} \text{RSS} \quad \text{subject to } \sum_{j=1}^p |\beta_j| \leq s$$

- ▶ Best subset selection (ℓ_0 penalty):

$$\min_{\boldsymbol{\beta}} \text{RSS} \quad \text{subject to } \sum_{j=1}^p 1_{\{\beta_j \neq 0\}} \leq s$$

Last lecture: Model selection measures

Indirect measures: C_p , AIC and BIC

- ▶ Model with d predictors
- ▶ Mallow's C_p :

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

where $\hat{\sigma}$ is an estimator for the variance of noise (estimated on a model containing all predictors)

- ▶ Akaike information criteria (AIC):

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

- ▶ Bayesian AIC (BIC):

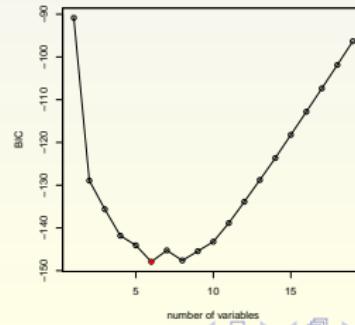
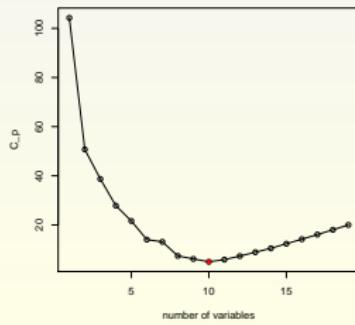
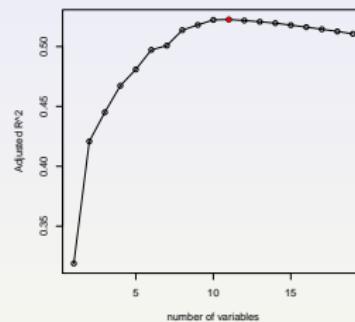
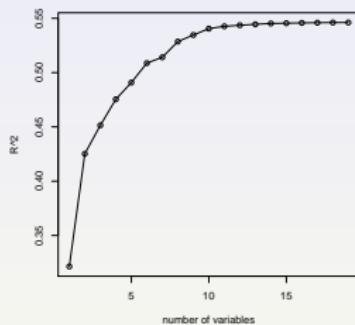
$$\text{BIC} = \frac{1}{n}(\text{RSS} + d\hat{\sigma}^2 \log n)$$

- ▶ **Heuristic:** select a model with the lowest C_p , AIC, BIC, or

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

Example

```
> regfit.full<-regsubsets(Salary~.,data=Hitters,nvmax = 19)
> reg.summary<-summary(regfit.full)
> names(reg.summary)
[1] "which"      "rsq"        "rss"        "adjr2"      "cp"         "bic"        "outmat"    "obj"
> reg.summary$rsq
[1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227 0.5285569 0.5346124 0.5404950
[11] 0.5426153 0.5436302 0.5444570 0.5452164 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```



Dimension Reduction

- ▶ Subset selection, shrinkage methods:
 - ▶ Original predictors used
- ▶ Dimensionality reduction idea
 - ▶ Represent/approximate X with a vector Z having less dimensions
 - ▶ Then, apply regression to Z
- ▶ Many approaches for doing this
- ▶ Common approach: Principal Component Analysis (PCA)

Dimension Reduction: PCA

- ▶ ▶ *Transform the predictors.* Let $Z = (Z_1, \dots, Z_q)$ represent $q < p$ linear combinations of the original p predictors:

$$Z_m = \sum_{j=1}^p \phi_{m,j} X_j$$

for some constants $\phi_{m,1}, \dots, \phi_{m,p}$

- ▶ *Use Ordinary Least Squares (OLS):* fit the linear regression

$$\hat{y}_i = \theta_0 + \sum_{m=1}^q \theta_m z_{i,m}$$

- ▶ If $\{\phi_{m,i}\}$ are chosen appropriately, dimension reduction can outperform the OLS regression

Dimension reduction

- ▶ Regression coefficients

$$\begin{aligned}\sum_{m=1}^q \theta_m z_{i,m} &= \sum_{m=1}^q \theta_m \sum_{j=1}^p \phi_{m,j} x_{i,j} \\ &= \sum_{j=1}^p \sum_{m=1}^q \theta_m \phi_{m,j} x_{i,j} \\ &= \sum_{j=1}^p \beta_j x_{i,j}\end{aligned}$$

where

$$\beta_j = \sum_{m=1}^q \theta_m \phi_{m,j}$$

- ▶ PCA is equivalent to imposing constraints on β_j 's in OLS

Finding Principal Components

PCA: Unsupervised method - will be covered more after the midterm
Good for high-dimensional data

Few words here: Finding the first principal component

- ▶ Look for the linear combination of the sample feature of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$

- ▶ *First loading vector/principal component:* $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$
- ▶ Assume x_i -s are **centered** ($\sum x_i = 0$)
- ▶ Look for ϕ_1 that has the largest sample variance, i.e.

$$\max_{\phi_1} \frac{1}{n} \sum_{i=1}^n z_{i1}^2 = \max_{\phi_1} \frac{1}{n} \sum_{i=1}^n (\phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip})^2$$

subject to $\sum_{j=1}^p \phi_{j1}^2 = 1$ (i.e., ϕ_1 is a unit vector)

- ▶ This optimization problem can be solved via eigen-decomposition

First Principal Components: Geometric interpretation

- ▶ Loading vector ϕ_1 represents the direction along which the data varies the most
- ▶ If we project x_1, \dots, x_n onto ϕ_1 , the projected values are the PC scores z_{i1} since

$$z_{i1} = \langle x_i, \phi_1 \rangle$$

Second and higher principal components

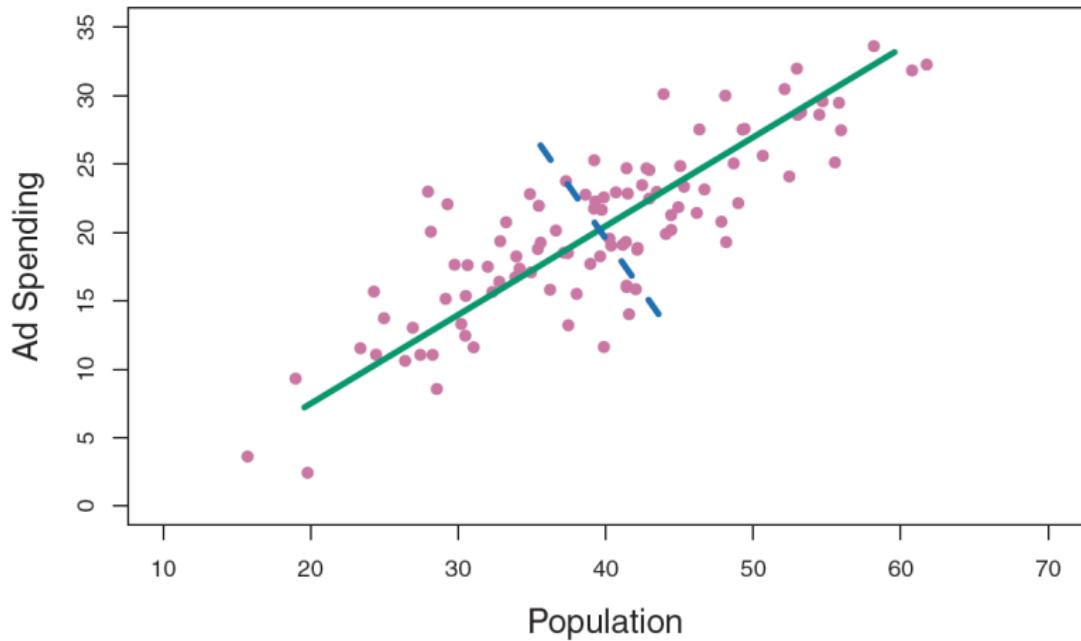
- ▶ After ϕ_1 has been determined, we look for ϕ_2 in a similar way, but with the additional constraint that ϕ_1, ϕ_2 are uncorrelated

$$\langle \phi_1, \phi_2 \rangle = 0$$

- ▶ We continue this procedure until we find as many PC as we want

PCA: Example

Two PC-s: Solid line: First PC; Dashed line: Second PC



PCA as Eigenvalue-Eigenvector Decomposition

Consider finding $k \leq p$ principal components: $\phi_1, \phi_2, \dots, \phi_k$

$$\mathbf{U} = [\phi_1 \quad \phi_2 \quad \cdots \quad \phi_k], \quad \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k.$$

Then, the projection of a data point $\mathbf{x}_i, 1 \leq i \leq n$ is given by

$$\hat{\mathbf{x}}_i = (\mathbf{x}_i \cdot \phi_1) \phi_1 + \cdots + (\mathbf{x}_i \cdot \phi_k) \phi_k = \mathbf{U} \mathbf{U}^\top \mathbf{x}_i,$$

implying

$$\|\hat{\mathbf{x}}_i\|^2 = \mathbf{x}_i^\top \mathbf{U} \mathbf{U}^\top \mathbf{U} \mathbf{U}^\top \mathbf{x}_i = \mathbf{x}_i^\top \mathbf{U} \mathbf{U}^\top \mathbf{x}_i.$$

Note that $\hat{\mathbf{x}}_i$ minimizes the distance $\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|$, and thus, finding k principle components is equivalent to finding \mathbf{U} that maximizes

$$\begin{aligned} M &= \max_{\mathbf{U}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \sum_{i=1}^n \mathbf{x}_i^\top \mathbf{U} \mathbf{U}^\top \mathbf{x}_i \\ &= \text{trace}\left(\mathbf{U}^\top \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \mathbf{U}\right), \quad (\text{using } \mathbf{x}^\top \mathbf{y} = \text{trace}(\mathbf{x}\mathbf{y}^\top)) \end{aligned}$$

PCA as Eigenvalue-Eigenvector Decomposition

Now, if $\mathbf{A} = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$, the optimization problem becomes

$$M = \max_{\mathbf{U}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{trace}(\mathbf{U}^\top \mathbf{A} \mathbf{U})$$

Note that \mathbf{A} is a symmetric matrix, and therefore orthogonally diagonalizable with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$. If \mathbf{U} is composed of k eigenvectors that correspond to the k largest eigenvalues, then

$$M \geq \sum_{i=1}^k \lambda_i.$$

On the other hand, we can diagonalize $\mathbf{A} = \mathbf{V}^\top \boldsymbol{\Lambda} \mathbf{V}$, where \mathbf{V} is an orthogonal matrix, yielding

$$\begin{aligned} M &= \max_{\mathbf{U}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{trace}(\mathbf{U}^\top \mathbf{V}^\top \boldsymbol{\Lambda} \mathbf{V} \mathbf{U}) = \max_{\mathbf{B}: \mathbf{B}^\top \mathbf{B} = \mathbf{I}_k} \text{trace}(\mathbf{B}^\top \boldsymbol{\Lambda} \mathbf{B}) \\ &= \sum_{i=1}^p \sum_{j=1}^k b_{ij}^2 \lambda_i = \sum_{i=1}^p \lambda_i \sum_{j=1}^k b_{ij}^2 \leq \sum_{i=1}^k \lambda_i \quad (\text{Note: } \sum_{i=1}^p b_{ij}^2 = 1, \sum_{j=1}^k b_{ij}^2 \leq 1) \end{aligned}$$

$\Rightarrow M = \sum_{i=1}^k \lambda_i$: Hence, the first k principal components correspond to the eigenvectors of the k largest eigenvalues of \mathbf{A} .

Beyond Linear - More Linear ☺: Basis Expansion

Basis expansion

- ▶ Map data \mathbf{x} into higher dimensional space $\mathbb{R}^d, d > p$:
 $\phi(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$, i.e.,

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_d(\mathbf{x}))$$

- ▶ Fit linear regression on $\phi(\mathbf{x})$ in the higher dimensional space

Example: Polynomial regression of degree q

- ▶ Map data $\mathbf{x} = (x_1, \dots, x_p)$ into

$$\phi(\mathbf{x}) = (x_1, x_1^2, \dots, x_1^q, x_2, x_2^2, \dots, x_2^q, \dots, x_p, x_p^2, \dots, x_p^q, \dots)$$

- ▶ Fit linear regression in the higher dimensional, \mathbb{R}^{pq} , space
 - ▶ Exponential growth of dimensionality: if $p = 10$ and $q = 10$

Quadratic Expansion

Example:

- ▶ Consider data with two predictors: $x_i = (x_{i1}, x_{i2})$
- ▶ We can obtain the preceding kernel by considering feature map

$$\phi(x_i) = (1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2)$$

if $x \in \mathbb{R}^p$, then $\phi(x) \in \mathbb{R}^{2p}$

- ▶ Then, the inner product

$$\begin{aligned}\langle \phi(x_i), \phi(x_j) \rangle &= 1 + 2x_{i1}x_{j1} + 2x_{i1}x_{j1} + x_{i1}^2x_{j1}^2 \\ &\quad + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2x_{j2}^2 \\ &= 1 + 2\langle x_i, x_j \rangle + \langle x_i, x_j \rangle^2 \\ &= (1 + \langle x_j, x_i \rangle)^2 = K(x_i, x_j)\end{aligned}$$

Quadratic Kernel: $K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^2$

K remains $n \times n$ matrix: does not grow with basis expansion

Polynomial Kernel and Dual Solution for Ridge

Polynomial Kernel:

$$K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^d$$

K is $n \times n$, doesn't grow with d ; recall, n is the number of data points x_i . Computing K is of the order $O(n^2 p)$.

- ▶ Dual solution - recall dual form for \hat{y}

$$\hat{y} = K\alpha,$$

- ▶ Dual Ridge: find α that minimizes

$$\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_2^2 = \|\mathbf{y} - K\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_2^2$$

- ▶ Solution

$$\boldsymbol{\alpha} = (\lambda I + K)^{-1} \mathbf{y}$$

which has computational complexity $O(n^3)$, and does not depend on the polynomial basis expansion parameter d . Hence the total complexity is $O((p+n)n^2) \approx O(n^3)$, when $p \ll n$, and does not depend on d .

Piecewise Polynomial: Splines

Example: Piecewise linear

- ▶ Assume $x \in \mathbb{R}$, i.e., $p = 1$
- ▶ Consider points: $w_1 \leq w_2 \leq \dots \leq w_n$ - called **knots**
- ▶ Consider basis

$$b_0(x) = 1, b(x) = x, b_i = (x - w_i)^+$$

- ▶ Then, fitting linear regression to these bases will result in piecewise linear approximation
- ▶ Relation to Neural Nets: Two layer neural network corresponds to **free knot linear spline** (studied in statistics since the 70s):
 - ▶ By free knot, one means that knots/weights w_i and linear regression coefficients β_j are optimized at the same time
 - ▶ This problem is not convex

In general, **basis can be chosen to be anything**: trigonometric functions, wavelets, etc.

Where are we now?

Recall, that we started with a general supervised Statistical (Machine) Learning problems

$$Y = f(X)$$

Problem: Estimate f from training data $\{(x_i, y_i)\}$, and then use it in general for inference and prediction

- ▶ First, we assumed some characteristics of the approximation function. e.g.: \hat{f} is linear or nonlinear via basis expansion
- ▶ Then, we used the training data $\{(x_i, y_i)\}$ to find the best fit of \hat{f} to the training data by minimizing the square error, RSS, or some other error function.
- ▶ For linear regression and Gaussian noise, we used classical tools from statistics, χ^2 , t-statistics and F-statistics, to characterize the confidence of our model.

Where are we now and what are we doing next?

- ▶ Searching for simpler models:
 - ▶ Selecting a model using t-statistics and F-statistics
 - ▶ Trying all combinations of predictors - not scalable - and then using indirect measures (C_p , AIC, BIC, etc.).
 - ▶ Introducing penalties: Ridge or Lasso - scalable
 - ▶ Preprocessing data - dimensionality reduction
- ▶ Bias - Variance: in general the preceding procedures introduce bias, but the hope is that at an expense of a small bias we have a bigger reduction in variance
- ▶ How do we select the best model?
 - ▶ For linear and Gaussian: we could use the t -statistics and F -statistics
 - ▶ We could use the indirect measures: C_p , AIC, BIC, etc.
 - ▶ Are there more direct general procedures to test and select the best model?

Model validation: Training vs. test error

- ▶ Select a statistical learning method, e.g.: linear model
- ▶ Training error: the average error from using the method to predict the response on the observations used in its training
- ▶ **Test error:** the average error from using the method to predict the response on a **new observation**
- ▶ Ideally: a large designated test set – seldomly available

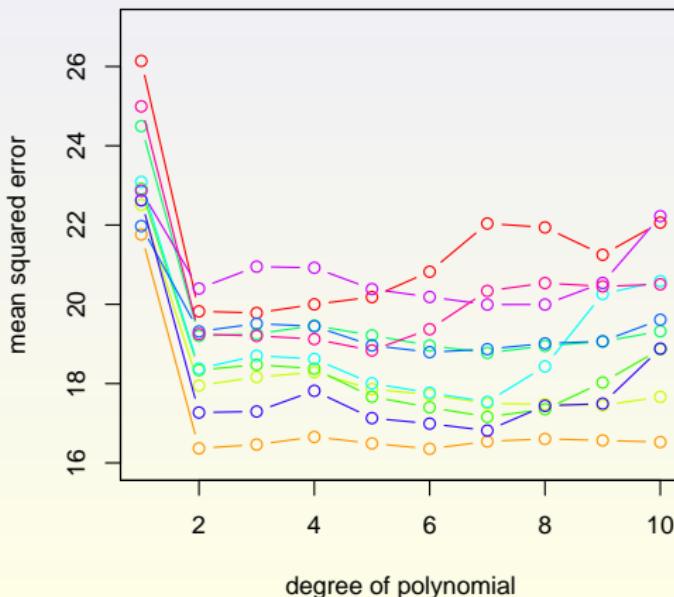
Validation-set approach

- ▶ Randomly divide the available samples into:
 - ▶ training set
 - ▶ validation set
- ▶ Random split into two halves
- ▶ Fit a model using the training set
- ▶ Use the model to predict the responses in the validation set
- ▶ The validation-set error is an estimate of the test error
- ▶ Drawbacks
 - ▶ The error estimate can be variable – depends on the split
 - ▶ Only a subset of observations used to fit the model
 - ▶ Tends to overestimate the test error for the model fit on the entire data set

Example: Auto data set

```
> set.seed(1)
> train<-sample(392,196) #pick randomly 1/2 sample
> err<-rep(0,10)
> for(i in 1:10) {
+   lm.fit<-lm(mpg~poly(horsepower,i),data = auto, subset = train)
+   err[i]<-mean((auto$mpg-predict(lm.fit,auto))[-train]^2)}
```

Different "train" set produces different results



K -fold cross-validation

- ▶ Popular approach
- ▶ Used in model selection

- ▶ Procedure
 - ▶ Randomly divide observations into K equal-sized parts
 - ▶ Leave out part k , fit a model using the remaining $K - 1$ parts
 - ▶ Use the left-out part to estimate the error
 - ▶ Repeat for all k
 - ▶ Combine results

K-fold cross-validation

- ▶ K parts: C_1, C_2, \dots, C_K
- ▶ $\cup_k C_k = \{1, \dots, n\}$
- ▶ n_k : the number of observations in part k
- ▶ Compute

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

where

$$\text{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i)^2$$

and \hat{y}_i is the prediction for observation i obtained from the data without part k

- ▶ $K = n$: leave-one out cross-validation (LOOCV)

LOOCV

Linear model example: we can compute CV error

- ▶ **No randomness** – all subsets of size $(n - 1)$ considered
- ▶ \mathbf{X} and \mathbf{y} :
 - ▶ observation i : \mathbf{X}_i and y_i
 - ▶ no observation i : $\mathbf{X}_{(i)}$ and $\mathbf{y}_{(i)}$
- ▶ Coefficients
 - ▶ observation i omitted:

$$\mathbf{X}_{(i)}^\top \mathbf{X}_{(i)} \hat{\boldsymbol{\beta}}_{(i)} = \mathbf{X}_{(i)}^\top \mathbf{y}_{(i)}$$

- ▶ all observations used:

$$(\mathbf{X}_{(i)}^\top \mathbf{X}_{(i)} + \mathbf{X}_i^\top \mathbf{X}_i) \hat{\boldsymbol{\beta}} = \mathbf{X}_{(i)}^\top \mathbf{y}_{(i)} + \mathbf{X}_i^\top y_i$$

and

$$\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_{(i)} + (\mathbf{X}_{(i)}^\top \mathbf{X}_{(i)})^{-1} \mathbf{X}_i^\top (y_i - \hat{y}_i),$$

where \hat{y}_i is the prediction for y_i using all observations

$$\hat{y}_i = \mathbf{X}_i \hat{\boldsymbol{\beta}} = \mathbf{X}_i (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

LOOCV

- ▶ Some algebra and

$$\begin{aligned} (\mathbf{X}_{(i)}^\top \mathbf{X}_{(i)})^{-1} &= (\mathbf{X}^\top \mathbf{X} - \mathbf{X}_i^\top \mathbf{X}_i)^{-1} \\ &= (\mathbf{X}^\top \mathbf{X})^{-1} + \frac{(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}_i^\top \mathbf{X}_i (\mathbf{X}^\top \mathbf{X})^{-1}}{1 - \mathbf{X}_i (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}_i^\top} \end{aligned}$$

yield

$$\begin{aligned} \text{CV}_{(n)} &= \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{X}_i \hat{\boldsymbol{\beta}}_{(i)})^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2 \end{aligned}$$

where $h_i = \mathbf{X}_i (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}_i^\top$

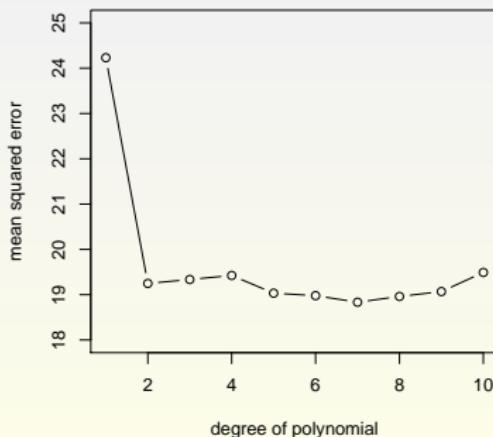
- ▶ **Weighted sum of squared residuals** (Provides validation for using modified RSS for some of the indirect measures).

Example: Auto data set

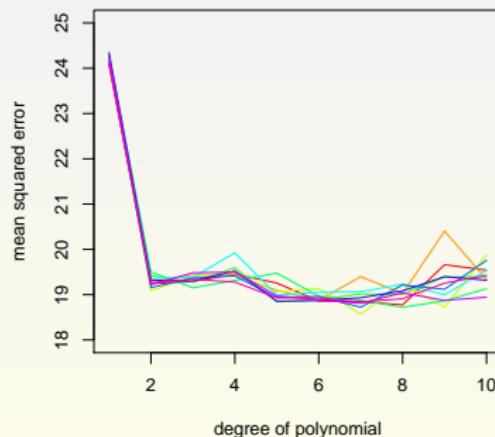
```
> library(boot)
> cv.err<-rep(0,10)
> for(i in 1:10) {
+   glm.fit<-glm(mpg~poly(horsepower,i), data=auto)
+   cv.err[i]<-cv.glm(auto,glm.fit)$delta[1]
+ }

> set.seed(1)
> for(i in 1:10) {
+   glm.fit<-glm(mpg~poly(horsepower,i), data=auto)
+   cv.err[i]<-cv.glm(auto,glm.fit,K=10)$delta[1]
+ }
```

LOOCV



10-fold CV



What if we are interested in computing other statistics, not just the test error, for the non-Gaussian or nonlinear case?

Bootstrap

- ▶ Setup
 - ▶ Population model that produces an outcome Y
 - ▶ Observations Z from this population model
 - ▶ Statistic $T(Z)$
 - ▶ Distribution of $T(Z)$
- ▶ Idea
 - ▶ The distribution of $T(Z)$ can be estimated by sampling Z from the population model
 - ▶ Resample with replacement from Z to “approximate” sampling from the population model
- ▶ Why?
 - ▶ Only samples Z available
 - ▶ No information on the population model

Bootstrap: Basic algorithm

- ▶ Input
 - ▶ A sample of data $Z = (Z_1, \dots, Z_n)$
 - ▶ An estimation rule \hat{T} for Statistic T
- ▶ Algorithm
 1. Generate bootstrap samples $Z^{*1}, Z^{*2}, \dots, Z^{*B}$
 - ▶ Create Z^{*b} by selecting n points from Z
 - ▶ A particular Z_i can appear in Z^{*b} multiple times
 2. Evaluate the estimator on each Z^{*b} :

$$\hat{T}_b = \hat{T}(Z^{*b})$$

- ▶ The empirical distribution of $\{\hat{T}_1, \dots, \hat{T}_B\}$ is an estimate of the distribution of $T(Z)$
- ▶ Bootstrap distribution
- ▶ Overlap between Z and Z^{*b} ?

Example: Variance estimation of the median

- ▶ The median of x_1, \dots, x_n , $x_i \in \mathbb{R}$, is found by sorting the numbers and taking the middle one (or averaging the two middle ones)
 - ▶ How good is the estimate $\text{median}(x_1, \dots, x_n)$?
 - ▶ Find it's variance
 - ▶ How?
1. Generate bootstrap datasets Z^{*1}, \dots, Z^{*B}
 2. Calculate:

$$\hat{T}_{\text{mean}} = \frac{1}{B} \sum_{b=1}^B \text{median}(Z^{*b})$$

and

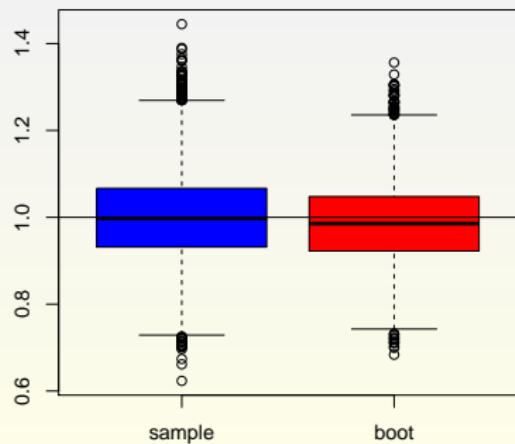
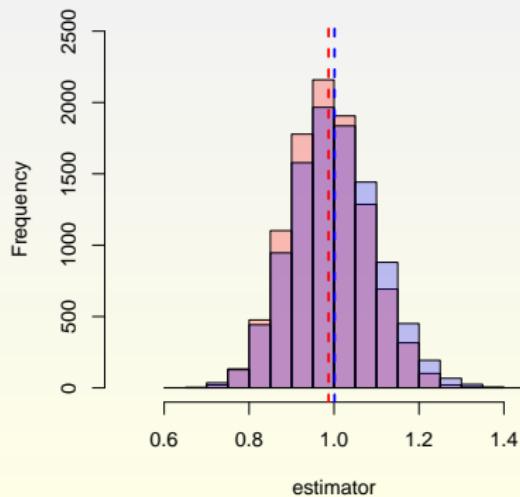
$$\hat{T}_{\text{var}} = \frac{1}{B-1} \sum_{b=1}^B (\text{median}(Z^{*b}) - \hat{T}_{\text{mean}})^2$$

When it works?

- ▶ The bootstrap distribution should be close to the sampling distribution
 - ▶ The number of bootstrap samples B should be large enough
 - ▶ The original data sample Z should be large enough to be “representative” of the population model
- ▶ Few assumptions about the population model
- ▶ Can yield inaccurate results (e.g., “extreme value” statistics)
- ▶ Usually reliable for estimating standard errors for estimators
 - ▶ Standard errors
 - ▶ The bootstrap estimate of the standard error is the standard deviation of the bootstrap distribution
 - ▶ Confidence intervals
 - ▶ $(1 - \gamma)$ confidence interval
 - ▶ t_α – the α th quantile of the bootstrap distribution
 - ▶ $(1 - \gamma)$ percentile bootstrap interval: $[t_{\gamma/2}, t_{1-\gamma/2}]$

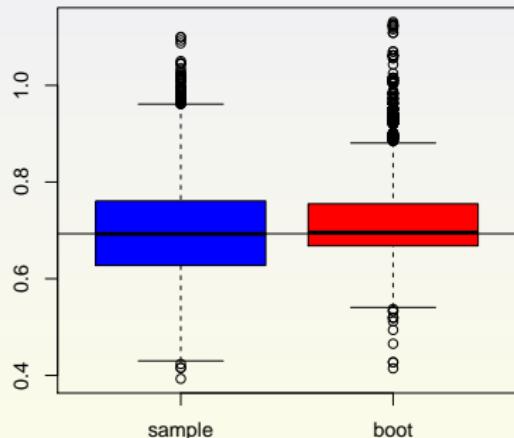
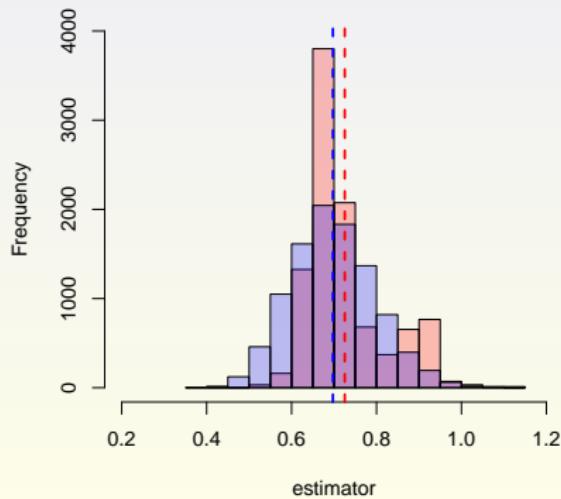
Example: Mean

```
> library(stats)
> set.seed(100)
> z<-rexp(100,rate=1)
> boot.fn=function(data,index)
+   return(mean(data[index]))
> boot.fn(z,1:100)
[1] 0.9874761
> boot.fn(z,sample(100,100,replace=T))
[1] 1.184723
> boot.fn(z,sample(100,100,replace=T))
[1] 1.131222
> library(boot)
> boot.out<-boot(z,boot.fn,1000)
```



Example: Median

```
> boot.fn=function(data,index)
+   return(median(data[index]))
> boot.fn(z,1:100)
[1] 0.6955635
> boot.fn(z,sample(100,100,replace=T))
[1] 0.6373702
> boot.fn(z,sample(100,100,replace=T))
[1] 0.652271
> boot.out<-boot(z,boot.fn,1000)
```



Bootstrap: Regression modeling

- ▶ n observations, response \mathbf{y} , covariates \mathbf{X}
- ▶ Bootstrap standard errors for OLS coefficients using case resampling:
 - ▶ For $b = 1, \dots, B$
 - ▶ Draw sample uniformly at random, with replacement, from observations (\mathbf{X}, \mathbf{y}) . Let the i th outcome in the b th sample be $(\mathbf{X}_i^{*b}, y_i^{*b})$
 - ▶ Compute $\hat{\beta}^{*b}$ given $(\mathbf{X}^{*b}, \mathbf{y}^{*b})$
- ▶ Bootstrap distribution of $\hat{\beta}$ to compute standard errors

Example: When analytics is unavailable, use bootstrap

Example: Heteroskedasticity

- ▶ Suppose $X_i \sim \mathcal{N}(0, 1)$ and $y_i = X_i + \epsilon_i$, for $i = 1, \dots, n$, where $\epsilon_i \sim \mathcal{N}(0, X_i^4)$
- ▶ Non-constant error variance
- ▶ Standard assumptions violated
- ▶ Standard errors?

Example: Heteroskedasticity

```
> n<-1000
> set.seed(1)
> x<-rnorm(n); y<-x+x^2*rnorm(n); df<-data.frame(x,y)
>
> lm.fit<-summary(lm(y~x,data = df))
> lm.fit$coefficients
  Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.02807759 0.06502742 0.4317807 6.659940e-01
x           1.18461640 0.06286099 18.8450180 5.229894e-68
>
> coef.boot=function(data,indices) {
+   coef.fit<-lm(y~x,data=data[indices,])
+   return(coef(coef.fit))
+ }
>
> boot.out<-boot(df,coef.boot, 50000)
> boot.out
```

ORDINARY NONPARAMETRIC BOOTSTRAP

```
Call:
boot(data = df, statistic = coef.boot, R = 50000)
```

```
Bootstrap Statistics :
      original     bias    std. error
t1* 0.02807759 -0.0008122518  0.06504296
t2* 1.18461640 -0.0014536288  0.14104734
```

Reading:

ISL: Read Chapters 5 and 7

ESL: Read Chapter 5, 7, and specifically Sections 5.1-5.2 and 7.10-7.11 for this lecture.

Homework 1: Due today, Sep 27, by 11:59pm.

Homework 2: Due Fri, Oct 7th, by 11:59pm.

Midterm planned for Oct 25th

EECS E6690: Statistical Learning for Biological and Information Systems

Lecture 5: Classification

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm
303 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Last lecture: Dimension reduction

- ▶ Dimensionality reduction idea
 - ▶ Represent/approximate X with a vector Z having less dimensions
 - ▶ Then, apply regression to Z
- ▶ Many approaches for doing this
- ▶ Common approach: Principal Component Analysis (PCA)
Finding the first principal component: projection of x onto ϕ

$$z_{1i} = \langle \mathbf{x}_i, \boldsymbol{\phi} \rangle = \phi_{11}x_{1i} + \phi_{21}x_{2i} + \cdots + \phi_{p1}x_{pi}$$

Assume x_i -s are centered ($\sum x_i = 0$)

- ▶ Look for $\boldsymbol{\phi}_1$ that has the largest sample variance, i.e.

$$\max_{\boldsymbol{\phi}_1} \frac{1}{n} \sum_{i=1}^n z_{1i}^2 = \max_{\boldsymbol{\phi}_1} \frac{1}{n} \sum_{i=1}^n (\phi_{11}x_{1i} + \phi_{21}x_{2i} + \cdots + \phi_{p1}x_{pi})^2$$

subject to $\sum_{j=1}^p \phi_{j1}^2 = 1$ (i.e., $\boldsymbol{\phi}_1$ is a unit vector)

Last lecture: Second principal component

- ▶ Loading vector ϕ_1 represents the direction along which the data varies the most
- ▶ If we project x_1, \dots, x_n onto ϕ_1 , the projected values are the PC scores z_{i1} since

$$z_{i1} = \langle \mathbf{x}_i, \phi_1 \rangle$$

Second and higher principal components

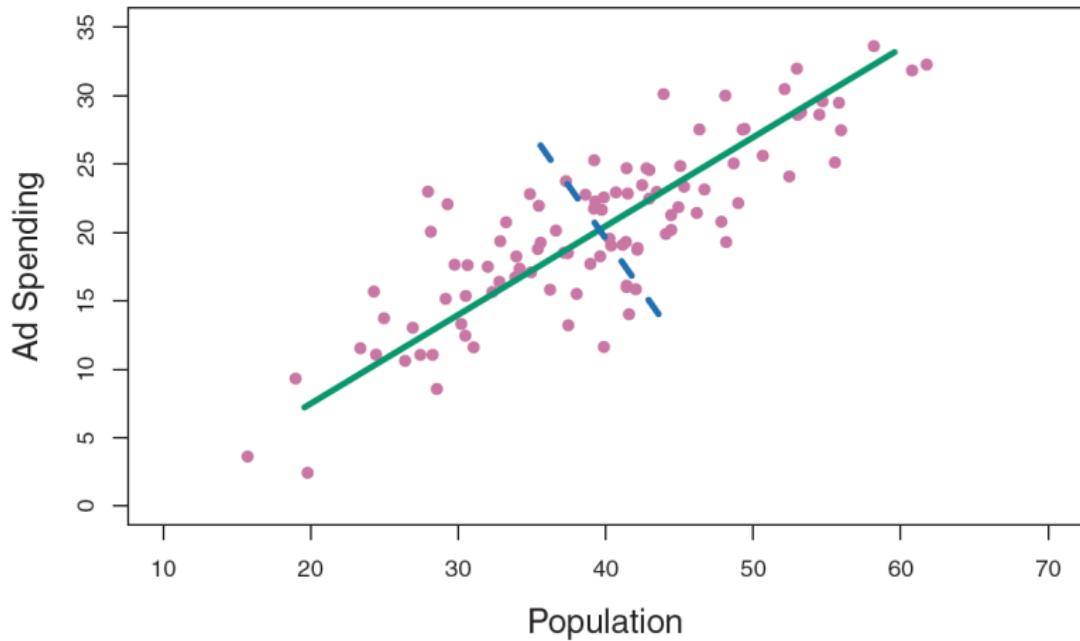
- ▶ After ϕ_1 has been determined, we look for ϕ_2 in a similar way, but with the additional constraint that ϕ_1, ϕ_2 are uncorrelated

$$\langle \phi_1, \phi_2 \rangle = 0$$

- ▶ We continue this procedure until we find as many PC as we want
- ▶ This optimization problem can be solved via eigen-decomposition

PCA Example

Two PC-s: Solid line: First PC; Dashed line: Second PC



Last lecture: PCA as Eigenvalue-Eigenvector Decomposition

Consider finding $k \leq p$ principal components: $\phi_1, \phi_2, \dots, \phi_k$

$$\mathbf{U} = [\phi_1 \quad \phi_2 \quad \cdots \quad \phi_k], \quad \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k.$$

Then, the projection of a data point $\mathbf{x}_i, 1 \leq i \leq n$ is given by

$$\hat{\mathbf{x}}_i = (\mathbf{x}_i \cdot \phi_1) \phi_1 + \cdots + (\mathbf{x}_i \cdot \phi_k) \phi_k = \mathbf{U} \mathbf{U}^\top \mathbf{x}_i$$

$$\mathbf{z}_i = (z_{1i}, z_{2i}, \dots, z_{ki}) = (\mathbf{x}_i \cdot \phi_1, \mathbf{x}_i \cdot \phi_2, \dots, \mathbf{x}_i \cdot \phi_k)$$

implying

$$\|\hat{\mathbf{x}}_i\|^2 = \mathbf{x}_i^\top \mathbf{U} \mathbf{U}^\top \mathbf{U} \mathbf{U}^\top \mathbf{x}_i = \mathbf{x}_i^\top \mathbf{U} \mathbf{U}^\top \mathbf{x}_i.$$

Note that $\hat{\mathbf{x}}_i$ minimizes the distance $\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|$, and thus, finding k principle components is equivalent to finding \mathbf{U} that maximizes

$$\begin{aligned} M &= \max_{\mathbf{U}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \sum_{i=1}^n \mathbf{x}_i^\top \mathbf{U} \mathbf{U}^\top \mathbf{x}_i \\ &= \text{trace} \left(\mathbf{U}^\top \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \mathbf{U} \right), \quad (\text{using } \mathbf{x}^\top \mathbf{y} = \text{trace}(\mathbf{x}\mathbf{y}^\top)) \end{aligned}$$

Last lecture: PCA as Eigenvalue-Eigenvector Decomposition

Now, if $\mathbf{A} = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$, the optimization problem becomes

$$M = \max_{\mathbf{U}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{trace}(\mathbf{U}^\top \mathbf{A} \mathbf{U})$$

Note that \mathbf{A} is a symmetric matrix, and therefore orthogonally diagonalizable with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$. If \mathbf{U} is composed of k eigenvectors that correspond to the k largest eigenvalues, then

$$M \geq \sum_{i=1}^k \lambda_i.$$

On the other hand, we can diagonalize $\mathbf{A} = \mathbf{V}^\top \boldsymbol{\Lambda} \mathbf{V}$, where \mathbf{V} is an orthogonal matrix, yielding

$$\begin{aligned} M &= \max_{\mathbf{U}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{trace}(\mathbf{U}^\top \mathbf{V}^\top \boldsymbol{\Lambda} \mathbf{V} \mathbf{U}) = \max_{\mathbf{B}: \mathbf{B}^\top \mathbf{B} = \mathbf{I}_k} \text{trace}(\mathbf{B}^\top \boldsymbol{\Lambda} \mathbf{B}) \\ &= \sum_{i=1}^p \sum_{j=1}^k b_{ij}^2 \lambda_i = \sum_{i=1}^p \lambda_i \sum_{j=1}^k b_{ij}^2 \leq \sum_{i=1}^k \lambda_i \quad (\text{Note: } \sum_{i=1}^p b_{ij}^2 = 1, \sum_{j=1}^k b_{ij}^2 \leq 1) \end{aligned}$$

$\Rightarrow M = \sum_{i=1}^k \lambda_i$: Hence, the first k principal components correspond to the eigenvectors of the k largest eigenvalues of \mathbf{A} .

Last lecture: Basis expansions

- ▶ Map data \mathbf{x} into higher dimensional space $\mathbb{R}^d, d > p$:
 $\phi(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$, i.e.,

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_d(\mathbf{x}))$$

- ▶ Fit linear regression on $\phi(\mathbf{x})$ in the higher dimensional space

Example: Polynomial regression of degree q

- ▶ Map data $\mathbf{x} = (x_1, \dots, x_p)$ into

$$\phi(\mathbf{x}) = (x_1, x_1^2, \dots, x_1^q, x_2, x_2^2, \dots, x_2^q, \dots, x_p, x_p^2, \dots, x_p^q, \dots)$$

- ▶ Fit linear regression in the higher dimensional, \mathbb{R}^{pq} , space
 - ▶ Exponential growth of dimensionality: if $p = 10$ and $q = 10$

Polynomial Kernel and Dual Solution for Ridge

Polynomial Kernel: dot products can have closed form

$$K(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle = (1 + \langle x_j, x_i \rangle)^d$$

K is $n \times n$, doesn't grow with d ; recall, n is the number of data points x_i .

- ▶ Dual solution - recall dual form for \hat{y}

$$\hat{y} = \mathbf{K}\boldsymbol{\alpha},$$

- ▶ Dual Ridge: find $\boldsymbol{\alpha}$ that minimizes

$$\|\mathbf{y} - \hat{y}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_2^2 = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_2^2$$

- ▶ Solution

$$\boldsymbol{\alpha} = (\lambda \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}$$

which has computational complexity $O(n^3)$, and does not depend on the polynomial basis expansion parameter d .

Other basis: piecewise polynomial, Fourier, wavelets, etc.

Last lecture: Model validation

Training versus Test Error

- ▶ Select a statistical learning method, e.g.: linear model, polynomial, piecewise polynomial/splines, etc.
- ▶ Training error: the average error from using the method to predict the response on the observations used in its training
- ▶ **Test error:** the average error from using the method to predict the response on a **new observation**
- ▶ Ideally: a large designated test set – rarely available

Last lecture: Validation-set approach

- ▶ Randomly divide the available samples into:
 - ▶ training set
 - ▶ validation set
- ▶ Random split into two halves
- ▶ Fit a model using the training set
- ▶ Use the model to predict the responses in the validation set
- ▶ The validation-set error is an estimate of the test error
- ▶ Drawbacks
 - ▶ The error estimate can be variable – depends on the split
 - ▶ Only a subset of observations used to fit the model
 - ▶ Tends to overestimate the test error

Last lecture: K -fold cross-validation

- ▶ Popular approach
 - ▶ Pro: scales well with data size
 - ▶ Con: there is still randomness
- ▶ Procedure
 - ▶ Randomly divide observations into K equal-sized parts
 - ▶ Leave out part k , fit a model using the remaining $K - 1$ parts
 - ▶ Use the left-out part to estimate the error
 - ▶ Repeat for all k
 - ▶ Combine results

Last lecture: K -fold cross-validation

- ▶ K parts: C_1, C_2, \dots, C_K
- ▶ $\cup_k C_k = \{1, \dots, n\}$
- ▶ n_k : the number of observations in part k
- ▶ Compute

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

where

$$\text{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i)^2$$

and \hat{y}_i is the prediction for observation i obtained from the data without part k

- ▶ $K = n$: leave-one out cross-validation (LOOCV)

Last lecture: LOOCV

Linear model example: we can compute CV error

- ▶ Pro: **No randomness** – all subsets of size $(n - 1)$ considered
- ▶ Con: Doesn't scale with data size
- ▶ Linear regression
 - ▶ \mathbf{X} and y :
 - ▶ observation i : \mathbf{X}_i and y_i
 - ▶ no observation i : $\mathbf{X}_{(i)}$ and $\mathbf{y}_{(i)}$
 - ▶ CV error

$$\begin{aligned}\text{CV}_{(n)} &= \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{X}_i \hat{\boldsymbol{\beta}}_{(i)})^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2\end{aligned}$$

where $h_i = \mathbf{X}_i(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}_i^\top$

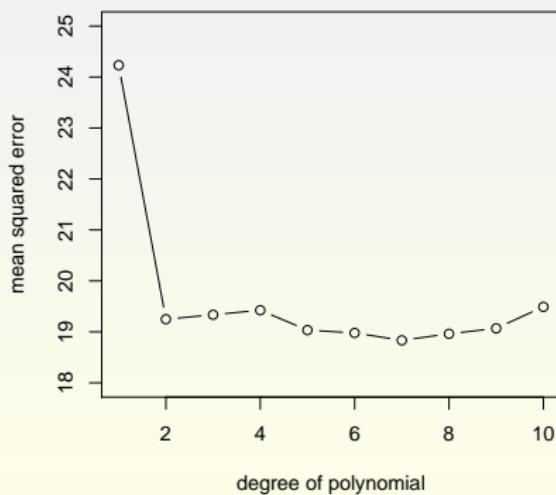
- ▶ **Weighted sum of squared residuals**

Example: Auto data set

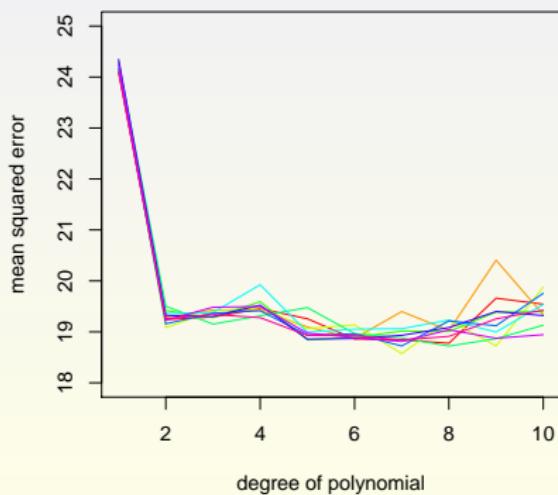
```
> library(boot)
> cv.err<-rep(0,10)
> for(i in 1:10) {
+   glm.fit<-glm(mpg~poly(horsepower,i), data=auto)
+   cv.err[i]<-cv.glm(auto,glm.fit)$delta[1]
+ }

> set.seed(1)
> for(i in 1:10) {
+   glm.fit<-glm(mpg~poly(horsepower,i), data=auto)
+   cv.err[i]<-cv.glm(auto,glm.fit,K=10)$delta[1]
+ }
```

LOOCV



10-fold CV



Last lecture: Bootstrap

- ▶ Setup
 - ▶ Population model that produces an outcome Y
 - ▶ Observations Z from this population model
 - ▶ Statistic $T(Z)$
 - ▶ Distribution of $T(Z)$
- ▶ Idea
 - ▶ The distribution of $T(Z)$ can be estimated by sampling Z from the population model
 - ▶ Resample with replacement from Z to “approximate” sampling from the population model
- ▶ Why?
 - ▶ Only samples Z available
 - ▶ No information on the population model

Last lecture: Bootstrap

Basic algorithm

- ▶ Input
 - ▶ A sample of data $\mathbf{Z} = (\mathbf{Z}_1, \dots, \mathbf{Z}_n)$
 - ▶ An estimation rule \hat{T} for Statistic T
- ▶ Algorithm
 1. Generate bootstrap samples $\mathbf{Z}^{*1}, \mathbf{Z}^{*2}, \dots, \mathbf{Z}^{*B}$
 - ▶ Create \mathbf{Z}^{*b} by selecting n points from \mathbf{Z}
 - ▶ A particular \mathbf{Z}_i can appear in \mathbf{Z}^{*b} multiple times
 2. Evaluate the estimator on each \mathbf{Z}^{*b} :

$$\hat{T}_b = \hat{T}(\mathbf{Z}^{*b})$$

- ▶ The empirical distribution of $\{\hat{T}_1, \dots, \hat{T}_B\}$ is an estimate of the distribution of $T(\mathbf{Z})$
- ▶ Bootstrap distribution
- ▶ Overlap between \mathbf{Z} and \mathbf{Z}^{*b} ?

Last lecture: Bootstrap regression modeling

- ▶ n observations, response \mathbf{y} , covariates \mathbf{X}
- ▶ Bootstrap standard errors for OLS coefficients using case resampling:
 - ▶ For $b = 1, \dots, B$
 - ▶ Draw sample uniformly at random, with replacement, from observations (\mathbf{X}, \mathbf{y}) . Let the i th outcome in the b th sample be $(\mathbf{X}_i^{*b}, y_i^{*b})$
 - ▶ Compute $\hat{\beta}^{*b}$ given $(\mathbf{X}^{*b}, \mathbf{y}^{*b})$
- ▶ Bootstrap distribution of $\hat{\beta}$ to compute standard errors

Few more words on regressions

Regression: Estimate real valued/quantitative f

$$Y = f(X)$$

from training data $\{(x_i, y_i)\}$, and then use it for inference and prediction.

- ▶ Assume something on \hat{f} , e.g.: linear or polynomial

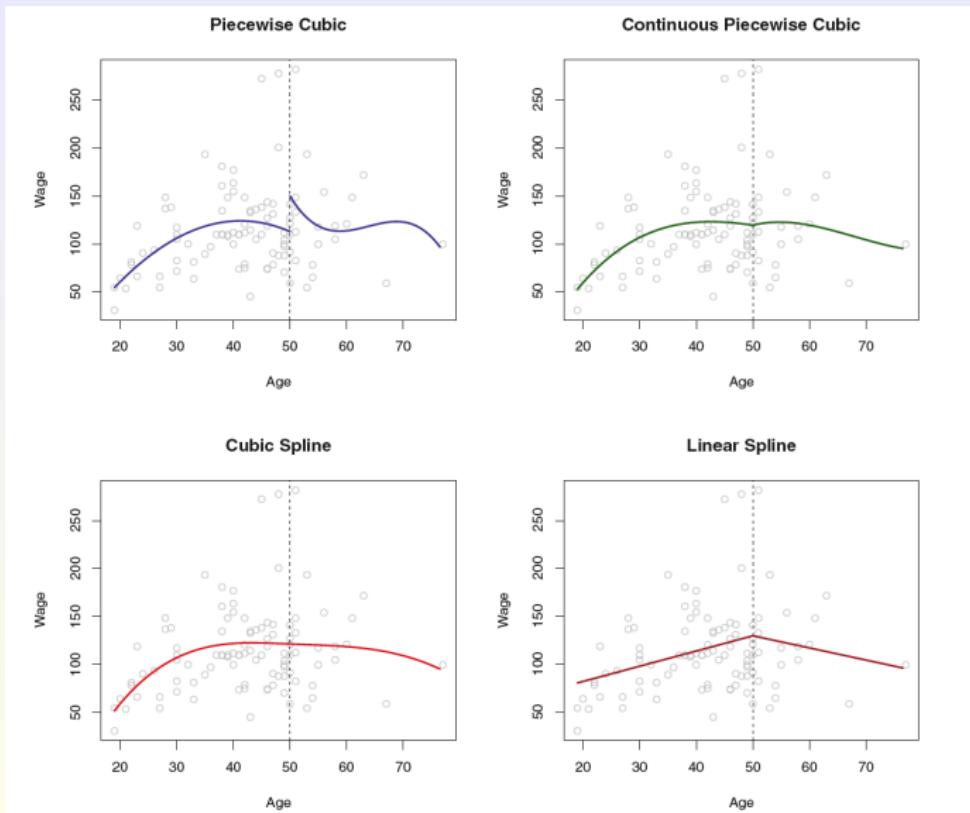
Many other options for the approximation function \hat{f} , e.g.

- ▶ Piecewise-polynomial/splines, e.g., piecewise-constant/linear; see Sec. 5.2 in ESL, Sec. 7.2-74 in ISL
- ▶ Smoothing splines: impose smoothness at the boundaries, e.g. continuity, continuous derivatives, etc.; see Sec. 5.4 in ESL, Sec. 7.5 in ISL

- ▶ After selecting a class of approximation functions, we go through all the steps we did before:

- ▶ Training (fitting)
- ▶ Simplifying: model selection, regularization (e.g., Ridge, Lasso)
- ▶ Testing: analytical, cross-validation, bootstrap
keep an eye on overfitting

Example: Piecewise-polynomial, i.e., splines



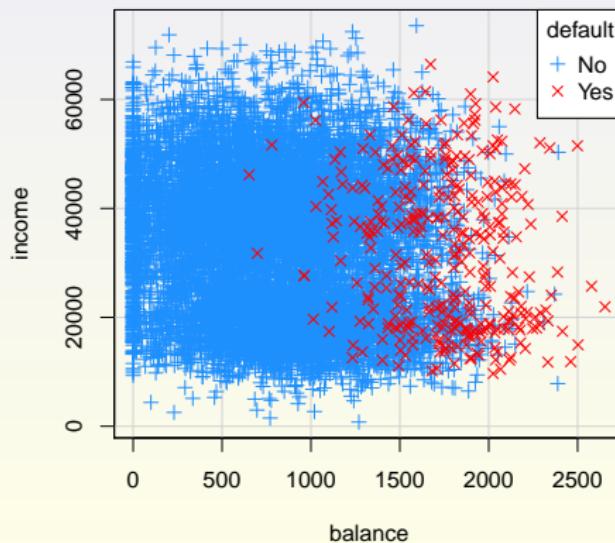
Classification

- ▶ Regression: real-valued/quantitative response
- ▶ Classification: categorical response
- ▶ Probability that a data point belongs to a class $c \in C$
- ▶ Example: Medical diagnosis
 - ▶ cancer, stroke, drug overdose, epileptic seizure
 - ▶ unordered set

Default data set

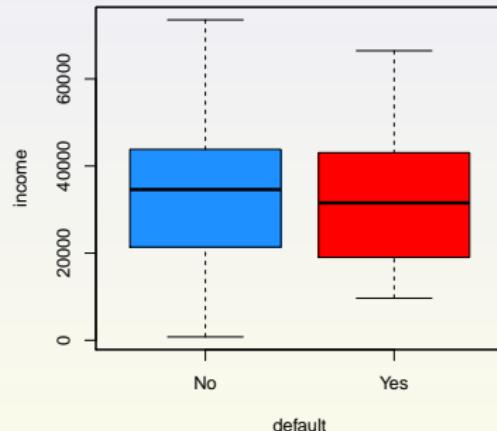
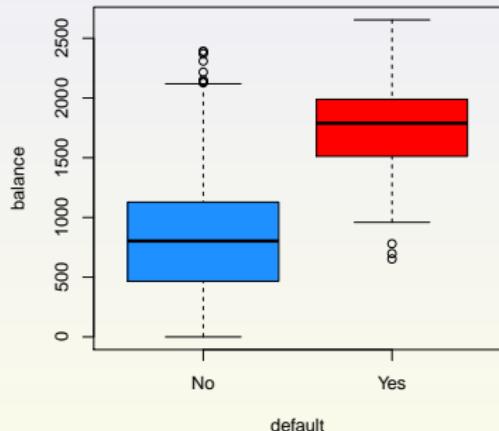
- Available in the ISLR package

```
> summary(Default)
   default    student      balance       income
No :9667    No :7056    Min.   : 0.0   Min.   : 772
Yes: 333   Yes:2944   1st Qu.: 481.7  1st Qu.:21340
              Median : 823.6  Median :34553
              Mean   : 835.4  Mean   :33517
              3rd Qu.:1166.3  3rd Qu.:43808
              Max.   :2654.3  Max.   :73554
```



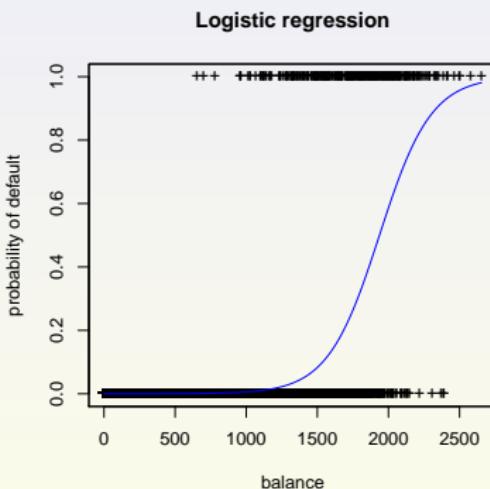
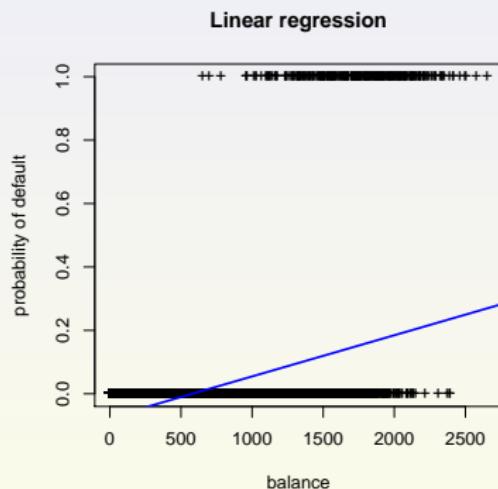
Default data set

```
> par(mfrow=c(1,2))
> boxplot(balance~default,data=Default,col=c("dodgerblue","red"),xlab="default",ylab="balance")
> boxplot(income~default,data=Default,col=c("dodgerblue","red"),xlab="default",ylab="income")
```



Linear regression

- ▶ Binary variables
- ▶ Predicted values not always in $[0, 1]$



Classification setting

- ▶ Loss function: Error rate

$$\frac{1}{n} \sum_{i=1}^n 1_{\{y_i \neq \hat{y}_i\}}$$

- ▶ Bayes Classifier - Optimal: Assign an observation x_0 to a class j for which the conditional probability

$$\mathbb{P}[Y = j | X = x_0]$$

is the largest. We will prove this later.

- ▶ Bayes error rate

$$1 - \mathbb{E}(\max_j \mathbb{P}[Y = j | X])$$

- ▶ We look for ways to approximate the conditional probabilities
They are difficult to estimate/compute from data. **Why?**

Logistic regression

- ▶ Model the conditional probability of $\mathbb{P}[Y = j | X = x]$
- ▶ Example: $\mathbb{P}[\text{default}=\text{Yes} | \text{balance}] = p(\text{balance})$
- ▶ Need a function with values in $[0, 1]$
- ▶ Logistic function (for binary variables, can be extended)

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

- ▶ Estimate β via Maximum Likelihood Estimation (MLE)
- ▶ Odds:

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

- ▶ A unit increase in X multiplies odds by e^{β_1}
- ▶ $\text{logit } p(X) = \beta_0 + \beta_1 X$

Example

```
> glm1a<-glm(default~balance,data = Default,family = binomial())
> summary(glm1a)
```

Call:

```
glm(formula = default ~ balance, family = binomial(), data = Default)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.2697	-0.1465	-0.0589	-0.0221	3.7589

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)		
(Intercept)	-1.065e+01	3.612e-01	-29.49	<2e-16 ***		
balance	5.499e-03	2.204e-04	24.95	<2e-16 ***		

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .	0.1	1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom
Residual deviance: 1596.5 on 9998 degrees of freedom
AIC: 1600.5

Number of Fisher Scoring iterations: 8

► Example:

$$\hat{P}[\text{default} = \text{Yes} | \text{balance} = 1000] = \frac{e^{-10.65 + 0.0055 \cdot 1000}}{1 + e^{-10.65 + 0.0055 \cdot 1000}} = 0.006$$

Example

```
> glm1b<-glm(default~student,data = Default,family = binomial())
> summary(glm1b)

Call:
glm(formula = default ~ student, family = binomial(), data = Default)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-0.2970 -0.2970 -0.2434 -0.2434  2.6585 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -3.50413   0.07071 -49.55 < 2e-16 ***
studentYes   0.40489   0.11502    3.52 0.000431 ***  
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom
Residual deviance: 2908.7 on 9998 degrees of freedom
AIC: 2912.7

Number of Fisher Scoring iterations: 6
```

- ▶ Example:

$$\hat{\mathbb{P}}[\text{default} = \text{Yes} | \text{student} = \text{Yes}] = \frac{e^{-3.504+0.405 \cdot 1}}{1 + e^{-3.504+0.405 \cdot 1}} = 0.043$$

MLE

- ▶ The probability of observed data under the model specified by β is given by the likelihood function

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

- ▶ MLE: select a model that maximizes likelihood of data

$$\max_{\beta} \ell(\beta)$$

or equivalently

$$\max_{\beta} \sum_{i=1}^n \left\{ y_i(\beta_0 + \beta_1 x_i) - \ln \left(1 + e^{\beta_0 + \beta_1 x_i} \right) \right\}$$

- ▶ First-order conditions
- ▶ Newton's method

Multiple logistic regression

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

```
> glm2<-glm(default~balance+income+student,data = Default,family = binomial())
> summary(glm2)
```

Call:
glm(formula = default ~ balance + income + student, family = binomial(),
data = Default)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.4691	-0.1418	-0.0557	-0.0203	3.7383

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)		
(Intercept)	-1.087e+01	4.923e-01	-22.080	< 2e-16 ***		
balance	5.737e-03	2.319e-04	24.738	< 2e-16 ***		
income	3.033e-06	8.203e-06	0.370	0.71152		
studentYes	-6.468e-01	2.363e-01	-2.738	0.00619 **		

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .	0.1	1

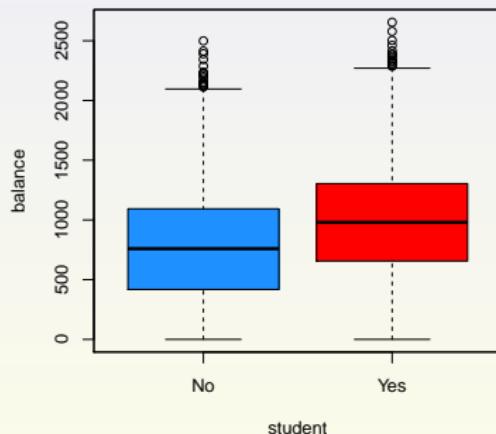
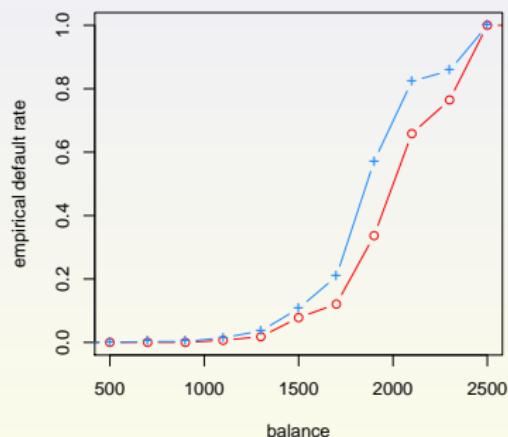
(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom
Residual deviance: 1571.5 on 9996 degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8

Confounding

- ▶ Dependency among predictors
- ▶ Similar to colinearity



South African heart disease data set

```
> head(SAheart)
```

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
1	160	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
2	144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
3	118	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
4	170	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
5	134	13.60	3.50	27.78	Present	60	25.99	57.34	49	1
6	132	6.20	6.47	36.21	Present	62	30.77	14.14	45	0

```
> summary(SAheart)
```

	sbp	tobacco	ldl	adiposity	famhist	typea
Min.	:101.0	Min. : 0.0000	Min. : 0.980	Min. : 6.74	Absent:270	Min. :13.0
1st Qu.	:124.0	1st Qu.: 0.0525	1st Qu.: 3.283	1st Qu.:19.77	Present:192	1st Qu.:47.0
Median	:134.0	Median : 2.0000	Median : 4.340	Median :26.11		Median :53.0
Mean	:138.3	Mean : 3.6356	Mean : 4.740	Mean :25.41		Mean :53.1
3rd Qu.	:148.0	3rd Qu.: 5.5000	3rd Qu.: 5.790	3rd Qu.:31.23		3rd Qu.:60.0
Max.	:218.0	Max. :31.2000	Max. :15.330	Max. :42.49		Max. :78.0
	obesity	alcohol	age	chd		
Min.	:14.70	Min. : 0.00	Min. :15.00	Min. :0.0000		
1st Qu.	:22.98	1st Qu.: 0.51	1st Qu.:31.00	1st Qu.:0.0000		
Median	:25.80	Median : 7.51	Median :45.00	Median :0.0000		
Mean	:26.04	Mean : 17.04	Mean :42.82	Mean : 0.3463		
3rd Qu.	:28.50	3rd Qu.: 23.89	3rd Qu.:55.00	3rd Qu.:1.0000		
Max.	:46.58	Max. :147.19	Max. :64.00	Max. :1.0000		

- ▶ Prevalence in the region (not the data set) ≈ 0.05
- ▶ Adjust β_0 :

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

Example

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

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

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7781	-0.8213	-0.4387	0.8889	2.5435

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-6.1507209	1.3082600	-4.701	2.58e-06 ***
sbp	0.0065040	0.0057304	1.135	0.256374
tobacco	0.0793764	0.0266028	2.984	0.002847 **
ldl	0.1739239	0.0596617	2.915	0.003555 **
adiposity	0.0185866	0.0292894	0.635	0.525700
famhistPresent	0.9253704	0.2278940	4.061	4.90e-05 ***
typea	0.0395950	0.0123202	3.214	0.001310 **
obesity	-0.0629099	0.0442477	-1.422	0.155095
alcohol	0.0001217	0.0044832	0.027	0.978350
age	0.0452253	0.0121298	3.728	0.000193 ***

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11 on 461 degrees of freedom
Residual deviance: 472.14 on 452 degrees of freedom
AIC: 492.14

Number of Fisher Scoring iterations: 5

Discriminant classification

- ▶ Model the distribution of X for each class separately
 - ▶ Will use Gaussian distribution
 - ▶ Leads to linear or quadratic discriminant analysis
 - ▶ Possible to use other distributions
- ▶ Bayes theorem

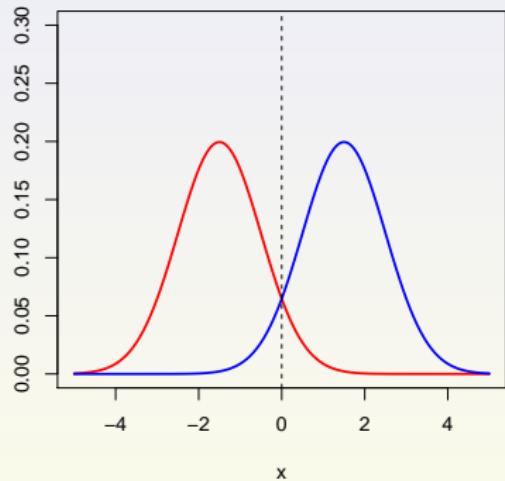
$$p_k(x) = \mathbb{P}[Y = k \mid X = x] = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

where

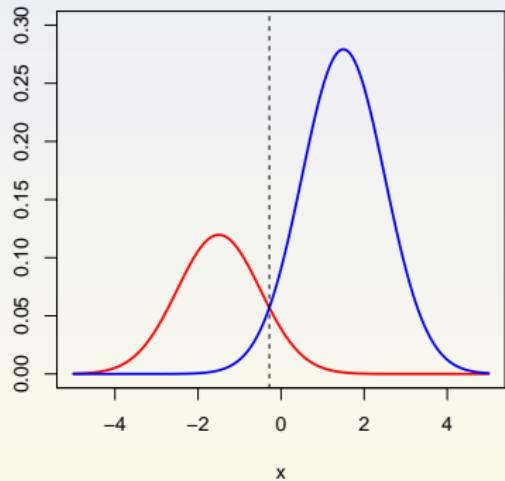
- ▶ f_k is the density of X in class k
- ▶ π_k is the prior probability for class k
- ▶ Classify to the most likely class – the highest $\pi_k f_k(x)$

Example

$\pi_1=0.5, \pi_2=0.5$



$\pi_1=0.3, \pi_2=0.7$



Discriminant analysis

- ▶ Start with $p = 1$
- ▶ Gaussian density (mean μ_k , variance σ_k^2):

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

- ▶ Discriminant function δ_k is quadratic (in x):

$$p_k(x) \propto \delta_k(x) = -x^2 \frac{1}{2\sigma_k^2} + x \frac{\mu_k}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - \log \sigma_k + \log \pi_k$$

- ▶ Probabilities:

$$\mathbb{P}[Y = k \mid X = x] = \frac{e^{\delta_k(x)}}{\sum_{l=1}^K e^{\delta_l(x)}}$$

Linear discriminant analysis

- ▶ Special case: $\sigma_1 = \sigma_2 = \dots = \sigma_K = \sigma$
- ▶ Discriminant function δ_k is linear (in x):

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

- ▶ Example: $K = 2$, $\pi_1 = \pi_2$ – decision boundary is at

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

- ▶ Parameter estimation:

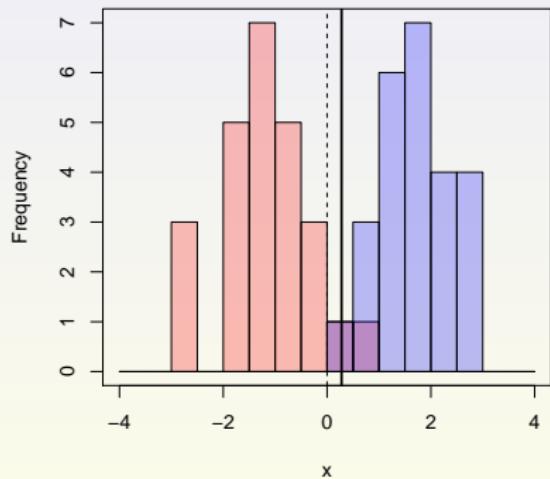
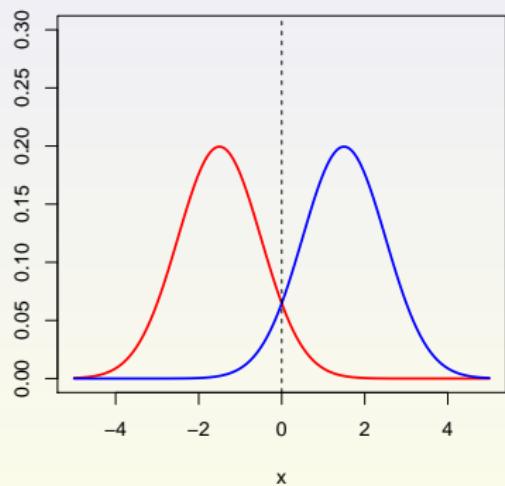
$$\hat{\pi}_k = \frac{n_k}{n}$$

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

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

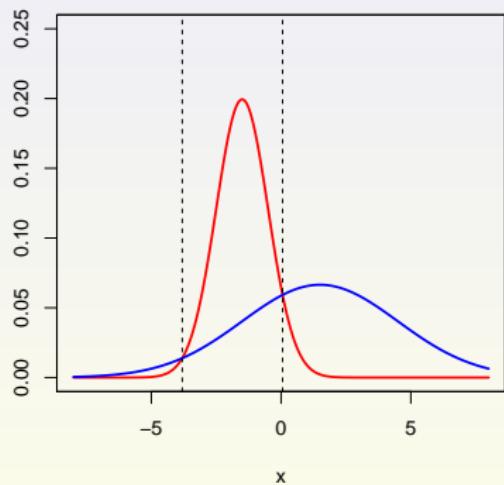
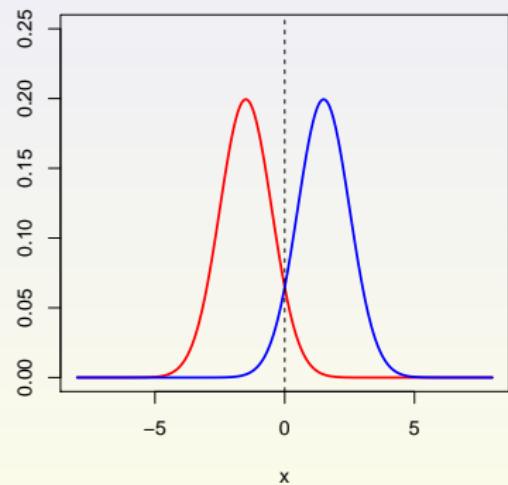
Example

In practice - finite number of samples = deviations



Linear vs. quadratic

Decision region more complicated



Discriminant analysis for $p > 1$

Quadratic - sigma unequal

- ▶ Density:

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_k|^{1/2}} e^{-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)}$$

- ▶ Linear discriminant function (equal $\boldsymbol{\Sigma}_k$):

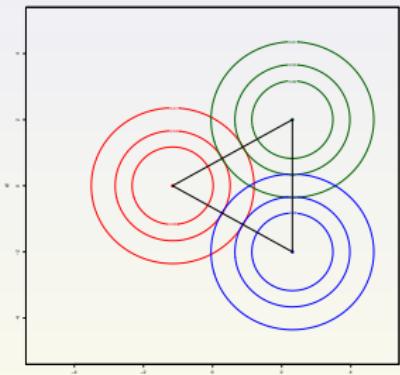
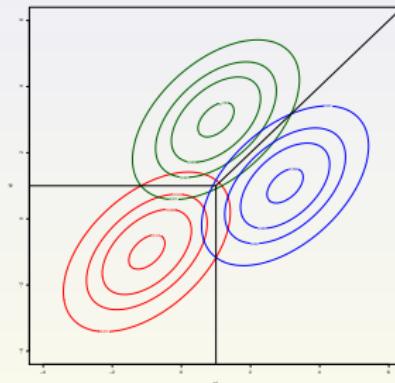
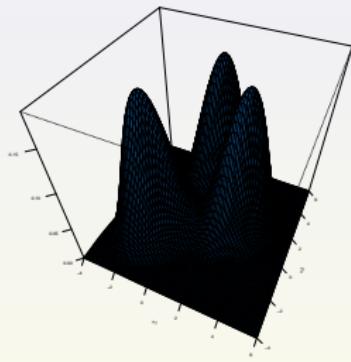
$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k$$

- ▶ Quadratic discriminant function (different $\boldsymbol{\Sigma}_k$):

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k) - \frac{1}{2} \log |\boldsymbol{\Sigma}_k| + \log \pi_k$$

Example

- ▶ $p = 2, K = 3, \pi_1 = \pi_2 = \pi_3 = 1/3$
- ▶ Coordinate transformation



- ▶ Sufficient to consider a $(K - 1)$ -dimensional hyperplane

Logistic regression vs. LDA

- ▶ Two classes
- ▶ Logistic regression

$$\log \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

- ▶ LDA

$$\log \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} = \left(\log \frac{\pi_1}{\pi_2} - \frac{1}{2} \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 \right) + \mathbf{x}^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)$$

- ▶ Same linear form
- ▶ Different way to estimate parameters

Naïve Bayes

- ▶ Assumes features are independent in each class
- ▶ Covariance matrices Σ_k are diagonal:

$$\pi_k f_k(\mathbf{x}) = \pi_k \prod_{i=1}^p f_{ki}(x_i) = \pi_k \prod_{i=1}^p \frac{1}{\sqrt{2\pi}\sigma_{ki}} e^{-\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2}}$$

and

$$\delta_k(\mathbf{x}) = - \sum_{i=1}^p \left[\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} + \log \sigma_{ki} \right] + \log \pi_k$$

- ▶ Advantages
 - ▶ much easier to estimate parameters for $p \gg 1$
 - ▶ can use both qualitative and categorical features (use PMFs instead of PDFs)
 - ▶ often produces good results

Naïve Bayes: Example

Name	Over 170cm	Eye	Hair length	Sex
Drew	No	Blue	Short	Male
Claudia	Yes	Brown	Long	Female
Drew	No	Blue	Long	Female
Drew	No	Blue	Long	Female
Alberto	Yes	Brown	Short	Male
Karin	No	Blue	Long	Female
Nina	Yes	Brown	Short	Female
Sergio	Yes	Blue	Long	Male

Sex	Name	\hat{p}
Male	Drew	1/3
	!Drew	2/3
Female	Drew	2/5
	!Drew	3/5

Sex	Over 170cm	\hat{p}
Male	Yes	2/3
	No	1/3
Female	Yes	2/5
	No	3/5

Sex	Eye	\hat{p}
Male	Blue	2/3
	Brown	1/3
Female	Blue	3/5
	Brown	2/5

Sex	Hair length	\hat{p}
Male	Short	2/3
	Long	1/3
Female	Short	1/5
	Long	4/5

{Name = Drew, Over 170cm = Yes, Eye = Blue, Hair length = Long} = ?

Proof of optimality of the Bayes Classifier

- ▶ Consider the case of two classes $Y \in \{0, 1\}$ and general $X, X \in \mathbb{R}^p$
- ▶ The proof is not needed for the grade.

Definition (Bayes Classifier)

Let $\eta(x) = \mathbb{P}[Y = 1 | X = x]$ and let the Bayes Classifier be defined as

$$f^*(x) = \begin{cases} 1, & \text{if } \eta(x) \geq 1/2 \\ 0, & \text{otherwise,} \end{cases}$$

i.e., it assigns x to a class k for which $\mathbb{P}[Y = k | X = x]$ has maximum value.

Theorem (Optimality)

For any classifier $g(x) \in \{0, 1\}$,

$$\mathbb{P}[g(X) \neq Y] \geq \mathbb{P}[f^*(X) \neq Y],$$

i.e., the Bayes classifier is optimal.

Proof of optimality of the Bayes Classifier

Proof. We will actually prove a stronger statement that

$$\mathbb{P}[g(X) \neq Y | X = x] \geq \mathbb{P}[f^*(X) \neq Y | X = x],$$

which by taking the expectation with respect to X yields the theorem.

$$\begin{aligned}\mathbb{P}[g(X) \neq Y | X = x] &= 1 - \mathbb{P}[g(X) = Y | X = x] \\&= 1 - (\mathbb{P}[Y = 1, g(X) = 1 | X = x] + \mathbb{P}[Y = 0, g(X) = 0 | X = x]) \\&= 1 - (\mathbb{E}[1_{\{Y=1\}} 1_{\{g(X)=1\}} | X = x] + \mathbb{E}[1_{\{Y=0\}} 1_{\{g(X)=0\}} | X = x]) \\&= 1 - (1_{\{g(x)=1\}} \mathbb{E}[1_{\{Y=1\}} | X = x] + 1_{\{g(X)=0\}} \mathbb{E}[1_{\{Y=0\}} | X = x]) \\&= 1 - (1_{\{g(x)=1\}} \mathbb{P}[Y = 1 | X = x] + 1_{\{g(x)=0\}} \mathbb{P}[Y = 0 | X = x]) \\&= 1 - (1_{\{g(x)=1\}} \eta(x) + 1_{\{g(x)=0\}} (1 - \eta(x)))\end{aligned}$$

Proof of optimality of the Bayes Classifier

Next, consider the difference

$$\begin{aligned} & \mathbb{P}[g(X) \neq Y | X = x] - \mathbb{P}[f^*(X) \neq Y | X = x] \\ &= \eta(x) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}) + (1 - \eta(x)) (1_{\{f^*(x)=0\}} - 1_{\{g(x)=0\}}) \\ &= (2\eta(x) - 1) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}), \end{aligned}$$

where the last equality follows from $1_{\{g(x)=0\}} = 1 - 1_{\{g(x)=1\}}$.

Finally, we show that the last expression is nonnegative. To this end, consider the following two cases:

1. $f^*(x) = 1 \Leftrightarrow \eta(x) \geq 1/2$, and therefore

$$(2\eta(x)-1) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}) = (2\eta(x)-1) (1 - 1_{\{g(x)=1\}}) \geq 0$$

2. $f^*(x) = 0 \Leftrightarrow \eta(x) < 1/2$, which also implies

$$(2\eta(x)-1) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}) = (2\eta(x)-1) (0 - 1_{\{g(x)=1\}}) \geq 0$$



Reading:

ISL: Read Chapter 4

ESL: Read Chapter 4

Homework 2: Due Fri, Oct 7th, by 11:59pm.

Midterm planned for Oct 25th

EECS E6690: Statistical Learning for Biological and Information Systems

Lecture 6: Tree-Based Methods

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm
303 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Last lecture: Classification

- ▶ Regression: quantitative response
- ▶ Classification: categorical response
- ▶ Probability that a data point belongs to a class $c \in C$
- ▶ Example: Medical diagnosis
 - ▶ cancer, stroke, drug overdose, epileptic seizure
 - ▶ unordered set

Last lecture: General Bayes approach

The Optimal Bayes Classifier

- ▶ Assign x to a class k for which

$$\mathbb{P}[Y = k \mid X = x]$$

has the **maximum value**

- ▶ Bayes formula (assume X is discrete; otherwise, replace $X = x$ with $X \in (x, x + dx)$)

$$p_k(x) = \mathbb{P}[Y = k \mid X = x] = \frac{\mathbb{P}[Y = k, X = x]}{\mathbb{P}[X = x]}$$
$$= \frac{\mathbb{P}[Y = k]\mathbb{P}[X = x \mid Y = k]}{\sum_{l=1}^K \mathbb{P}[Y = l]\mathbb{P}[X = x \mid Y = l]} =: \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

where

- ▶ $\pi_k = \mathbb{P}[Y = k]$ - **prior** probability for class k
- ▶ $p_k(x) = \mathbb{P}[Y = k \mid X = x]$ - **posterior** probability
- ▶ $f_k(x) = \mathbb{P}[X = x \mid Y = k]$ - likelihood function: the density of X in class k
- ▶ Problem: $\mathbb{P}[Y = k \mid X = x], \mathbb{P}[X = x \mid Y = k]$ unknown
 - ▶ Make assumptions: logistic, Gaussian, independence, etc.

Proof of optimality of the Bayes classifier

- We consider the case of two classes $Y \in \{0, 1\}$ and general X , say $X \in \mathbb{R}^p$
- The proof is not needed for the grade.

Definition (Bayes classifier)

Let $\eta(x) = \mathbb{P}[Y = k | X = x]$ and

$$f^*(x) = \begin{cases} 1, & \text{if } \eta(x) \geq 1/2 \\ 0, & \text{otherwise,} \end{cases}$$

i.e., it assigns x to a class k for which $\mathbb{P}[Y = k | X = x]$ has maximum value.

Theorem (Optimality)

For any classifier $g(x) \in \{0, 1\}$,

$$\mathbb{P}[g(X) \neq Y] \geq \mathbb{P}[f^*(X) \neq Y],$$

i.e., the Bayes classifier is optimal.

Proof of optimality of the Bayes Classifier

Proof. We will actually prove a stronger statement that

$$\mathbb{P}[g(X) \neq Y | X = x] \geq \mathbb{P}[f^*(X) \neq Y | X = x],$$

which by taking the expectation with respect to X yields the theorem.

$$\begin{aligned}\mathbb{P}[g(X) \neq Y | X = x] &= 1 - \mathbb{P}[g(X) = Y | X = x] \\&= 1 - (\mathbb{P}[Y = 1, g(X) = 1 | X = x] + \mathbb{P}[Y = 0, g(X) = 0 | X = x]) \\&= 1 - (\mathbb{E}[1_{\{Y=1\}} 1_{\{g(X)=1\}} | X = x] + \mathbb{E}[1_{\{Y=0\}} 1_{\{g(X)=0\}} | X = x]) \\&= 1 - (1_{\{g(x)=1\}} \mathbb{E}[1_{\{Y=1\}} | X = x] + 1_{\{g(x)=0\}} \mathbb{E}[1_{\{Y=0\}} | X = x]) \\&= 1 - (1_{\{g(x)=1\}} \mathbb{P}[Y = 1 | X = x] + 1_{\{g(x)=0\}} \mathbb{P}[Y = 0 | X = x]) \\&= 1 - (1_{\{g(x)=1\}} \eta(x) + 1_{\{g(x)=0\}} (1 - \eta(x)))\end{aligned}$$

Similarly, we can express

$$\mathbb{P}[f^*(X) \neq Y | X = x] = 1 - (1_{\{f^*(x)=1\}} \eta(x) + 1_{\{f^*(x)=0\}} (1 - \eta(x)))$$

Proof of optimality of the Bayes Classifier

Next, consider the difference

$$\begin{aligned} & \mathbb{P}[g(X) \neq Y | X = x] - \mathbb{P}[f^*(X) \neq Y | X = x] \\ &= \eta(x) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}) + (1 - \eta(x)) (1_{\{f^*(x)=0\}} - 1_{\{g(x)=0\}}) \\ &= (2\eta(x) - 1) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}), \end{aligned}$$

where the last equality uses

$$1_{\{g(x)=0\}} = 1 - 1_{\{g(x)=1\}}, 1_{\{f^*(x)=0\}} = 1 - 1_{\{f^*(x)=1\}}.$$

Finally, we show that the last expression is nonnegative. To this end, consider the following two cases:

1. $f^*(x) = 1 \Leftrightarrow \eta(x) \geq 1/2$, and therefore

$$(2\eta(x)-1) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}) = (2\eta(x)-1) (1 - 1_{\{g(x)=1\}}) \geq 0$$

2. $f^*(x) = 0 \Leftrightarrow \eta(x) < 1/2$, which also implies

$$(2\eta(x)-1) (1_{\{f^*(x)=1\}} - 1_{\{g(x)=1\}}) = (2\eta(x)-1) (0 - 1_{\{g(x)=1\}}) \geq 0$$



Last lecture: Logistic regression

- ▶ Model $p_k(X)$, $k = 0, 1$, as logistic function
- ▶ Example:

$$\mathbb{P}[\text{default}=\text{Yes} \mid \text{balance}] = p_1(\text{balance})$$

- ▶ Logistic function (for binary variables, can be extended)

$$\mathbb{P}[\text{default}=\text{Yes} \mid X] \equiv p_1(X) \equiv p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

- ▶ Estimate β via Maximum Likelihood Estimation (MLE)
- ▶ Odds:

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

- ▶ A unit increase in X multiplies odds by e^{β_1}
- ▶ $\text{logit } p(X) = \beta_0 + \beta_1 X$

Example

```
> glm1a<-glm(default~balance,data = Default,family = binomial())
> summary(glm1a)
```

Call:

```
glm(formula = default ~ balance, family = binomial(), data = Default)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.2697	-0.1465	-0.0589	-0.0221	3.7589

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)		
(Intercept)	-1.065e+01	3.612e-01	-29.49	<2e-16 ***		
balance	5.499e-03	2.204e-04	24.95	<2e-16 ***		

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .	0.1	1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom
Residual deviance: 1596.5 on 9998 degrees of freedom
AIC: 1600.5

Number of Fisher Scoring iterations: 8

► Example:

$$\hat{P}[\text{default} = \text{Yes} | \text{balance} = 1000] = \frac{e^{-10.65 + 0.0055 \cdot 1000}}{1 + e^{-10.65 + 0.0055 \cdot 1000}} = 0.006$$

Last lecture: MLE fit

- ▶ Assume that $\mathbb{P}[Y_i = y_i | X_i = x_i]$ follows the logistic function $p(x)$, and the conditional independence

$$\begin{aligned}\mathbb{P}[Y_1 = y_1, \dots, Y_n = y_n | X_1 = x_1, \dots, X_n = x_n] \\ = \mathbb{P}[Y_1 = y_1 | X_1 = x_1] \cdots \mathbb{P}[Y_n = y_n | X_n = x_n]\end{aligned}$$

- ▶ Hence the preceding conditional probability is maximized on observed data for β that maximizes the likelihood function

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

- ▶ MLE: select a model that maximizes likelihood of data

$$\max_{\beta} \ell(\beta)$$

or equivalently

$$\max_{\beta} \sum_{i=1}^n \left\{ y_i (\beta_0 + \beta_1 x_i) - \ln (1 + e^{\beta_0 + \beta_1 x_i}) \right\}$$

- ▶ First-order conditions: Newton's method

Last lecture: Multiple logistic regression

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

```
> glm2<-glm(default~balance+income+student,data = Default,family = binomial())
> summary(glm2)
```

Call:
glm(formula = default ~ balance + income + student, family = binomial(),
data = Default)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.4691	-0.1418	-0.0557	-0.0203	3.7383

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)		
(Intercept)	-1.087e+01	4.923e-01	-22.080	< 2e-16 ***		
balance	5.737e-03	2.319e-04	24.738	< 2e-16 ***		
income	3.033e-06	8.203e-06	0.370	0.71152		
studentYes	-6.468e-01	2.363e-01	-2.738	0.00619 **		

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .	0.1	1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2920.6 on 9999 degrees of freedom
Residual deviance: 1571.5 on 9996 degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8

Last lecture: Discriminant classification

Gaussian assumptions

- ▶ Start with $p = 1$
- ▶ Gaussian density (mean μ_k , variance σ_k^2):

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

- ▶ Discriminant function δ_k is quadratic (in x):

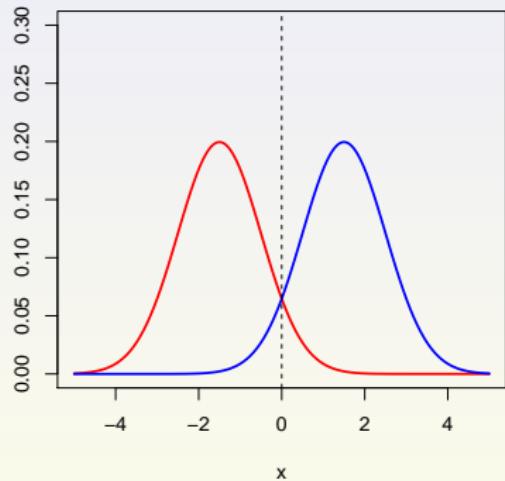
$$p_k(x) \propto \delta_k(x) = -x^2 \frac{1}{2\sigma_k^2} + x \frac{\mu_k}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - \log \sigma_k + \log \pi_k$$

- ▶ Probabilities:

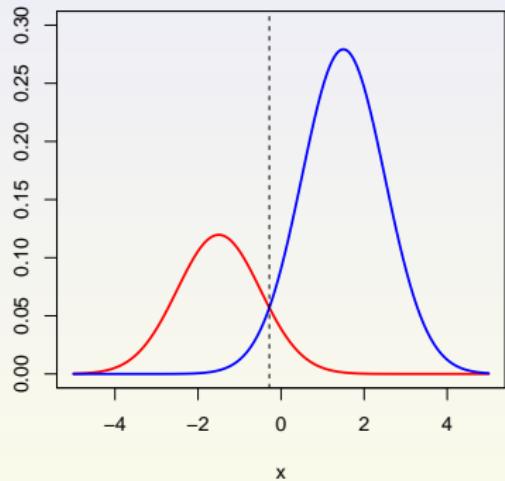
$$\mathbb{P}[Y = k \mid X = x] = \frac{e^{\delta_k(x)}}{\sum_{l=1}^K e^{\delta_l(x)}}$$

Example

$\pi_1=0.5, \pi_2=0.5$



$\pi_1=0.3, \pi_2=0.7$



Last lecture: Linear discriminant analysis

- ▶ **Special case:** $\sigma_1 = \sigma_2 = \dots = \sigma_K = \sigma$
- ▶ Discriminant function δ_k is linear (in x):

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

- ▶ Example: $K = 2$, $\pi_1 = \pi_2$ – decision boundary is at

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

- ▶ Parameter estimation:

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

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

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

Last lecture: Quadratic discriminant analysis

Quadratic - sigma unequal

- ▶ Density:

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_k|^{1/2}} e^{-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)}$$

- ▶ Linear discriminant function (equal $\boldsymbol{\Sigma}_k$):

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k$$

- ▶ Quadratic discriminant function (different $\boldsymbol{\Sigma}_k$):

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k) - \frac{1}{2} \log |\boldsymbol{\Sigma}_k| + \log \pi_k$$

Last lecture: Logistic regression vs. LDA

- ▶ Two classes
- ▶ Logistic regression

$$\log \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

- ▶ LDA

$$\log \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} = \left(\log \frac{\pi_1}{\pi_2} - \frac{1}{2} \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 \right) + \mathbf{x}^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)$$

- ▶ Same linear form
- ▶ Different way to estimate parameters

Last lecture: Naïve Bayes

- ▶ Assumes features are independent in each class
- ▶ Covariance matrices Σ_k are diagonal:

$$\pi_k f_k(\mathbf{x}) = \pi_k \prod_{i=1}^p f_{ki}(x_i) = \pi_k \prod_{i=1}^p \frac{1}{\sqrt{2\pi}\sigma_{ki}} e^{-\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2}}$$

and

$$\delta_k(\mathbf{x}) = - \sum_{i=1}^p \left[\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} + \log \sigma_{ki} \right] + \log \pi_k$$

- ▶ Advantages
 - ▶ much easier to estimate parameters for $p \gg 1$
 - ▶ can use both qualitative and categorical features (use PMFs instead of PDFs)
 - ▶ often produces good results

Predictability versus Interpretability

- ▶ **Predictability:** Determined by how good is the model in predicting the future values, i.e., minimizing the prediction error.
- ▶ **Interpretability:** Determined by how well the model interprets/explains data.

Often, these important questions don't go hand in hand

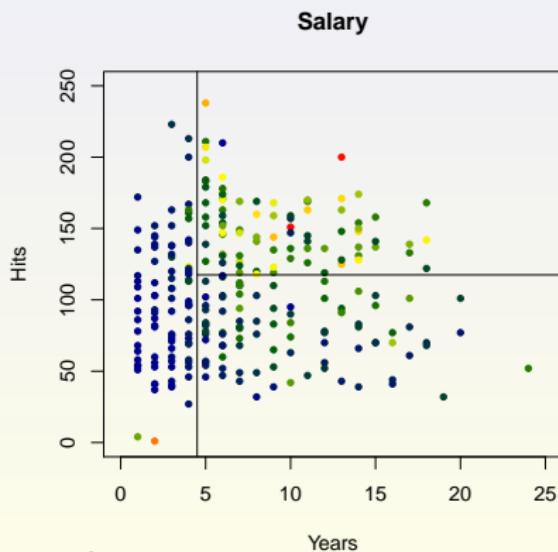
- ▶ LDA is easier to interpret than the logistic classification.
- ▶ Today, we'll see **tree based methods that have good interpretability** for both regression and classification.

Tree-Based Methods: Decision trees

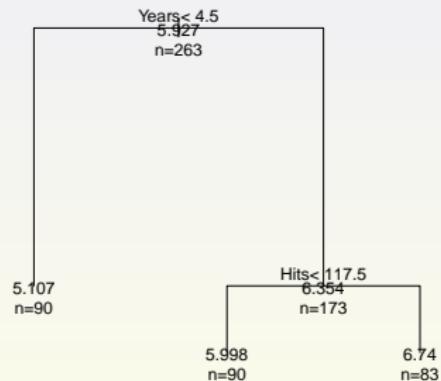
- ▶ Models for regression and classification
- ▶ Idea:
 - ▶ Segment the predictor space (X_1, \dots, X_p) into distinct and non-overlapping regions, R_1, \dots, R_j
 - ▶ Prediction (classification) based on:
 - ▶ Average (majority vote) over segments

Example: Hitters

- Predict Salary based on Years and Hits
- Remove missing values and apply log-transform
- Salary encoding from low to high: blue - green - yellow - red



Classification tree for Salary



- Interpretation
- Prediction

Regression Trees: Segmentation

- ▶ In general, the regions can have any shape
- ▶ Focus on high-dimensional rectangles (boxes)
- ▶ Goal: Find R_1, \dots, R_J that minimize the RSS:

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

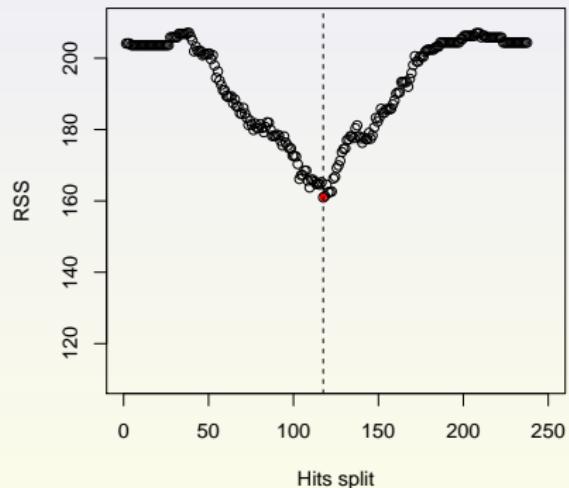
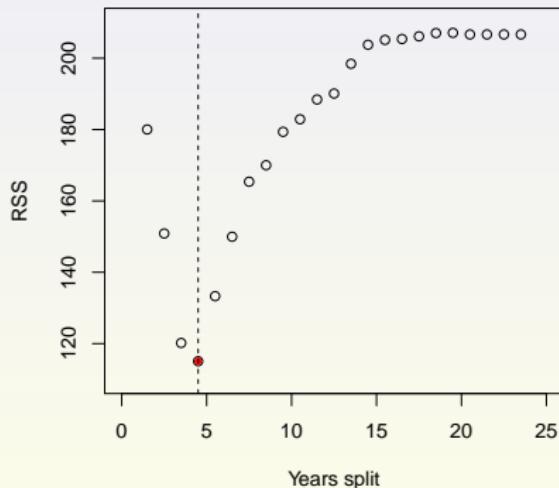
where \hat{y}_{R_j} is the mean response for the observations in R_j

- ▶ Computationally infeasible to consider all partitions
- ▶ Top-down, greedy approach: Binary splitting
- ▶ Stopping criteria (e.g., max number of observations in a box)

Segmentation: Example

- ▶ $R^-(j, s) = \{X : X_j \leq s\}$ and $R^+(j, s) = \{X : X_j > s\}$
- ▶ Minimize

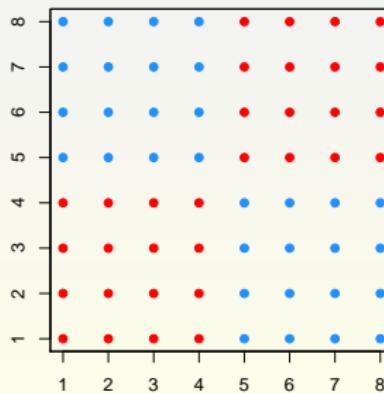
$$\sum_{i: x_i \in R^-(j, s)} (y_i - \hat{y}_{R^-})^2 + \sum_{i: x_i \in R^+(j, s)} (y_i - \hat{y}_{R^+})^2$$



Year-split gives the minimal RSS

Overfitting

- ▶ Optimal tree size?
 - ▶ training error decreases as the size increases
 - ▶ testing error decreases, but then increases
- ▶ Grow the tree only if RSS decreases – poor results
- ▶ Example: 2 values: red and blue
always same RSS regardless of the cut
but for the next cut - there is (!)



- ▶ Alternative: Grow the tree to a large size and then trim it back

Tree pruning

- ▶ Start with a large tree T_0
- ▶ Cost complexity pruning (weakest link pruning)
- ▶ Sequence of trees indexed by α
- ▶ For each α :

$$\min_{T \subseteq T_0} \left\{ \sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \right\},$$

where

- ▶ $|T|$ is the number of leafs in T
- ▶ R_m is the box corresponding to the m th leaf
- ▶ \hat{y}_{R_m} is the mean of training observations in R_m
- ▶ Parameter α
 - ▶ Controls the complexity/fit tradeoff
 - ▶ Select $\hat{\alpha}$ using cross-validation

Fitting a tree

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 α .
3. Use K -fold cross-validation to choose α . For each $k = 1, \dots, K$:
 - 3.1. Repeat Steps 1 and 2 on the $(K - 1)$ 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 α .
4. Return the subtree from Step 2 that corresponds to the chosen value of α .

Example: Hitters

```
> library(tree)
> hitters.fit<-tree(Salary~Years+Hits, data=myHitters)
> summary(hitters.fit)

Regression tree:
tree(formula = Salary ~ Years + Hits, data = myHitters)
Number of terminal nodes:  8
Residual mean deviance:  0.2708 = 69.06 / 255
Distribution of residuals:
   Min. 1st Qu. Median 3rd Qu. Max.
-2.2400 -0.2980 -0.0365  0.0000  0.3233  2.1520
> cv.hitters<-cv.tree(hitters.fit)
> cv.hitters
$size
[1] 8 7 6 5 4 3 2 1

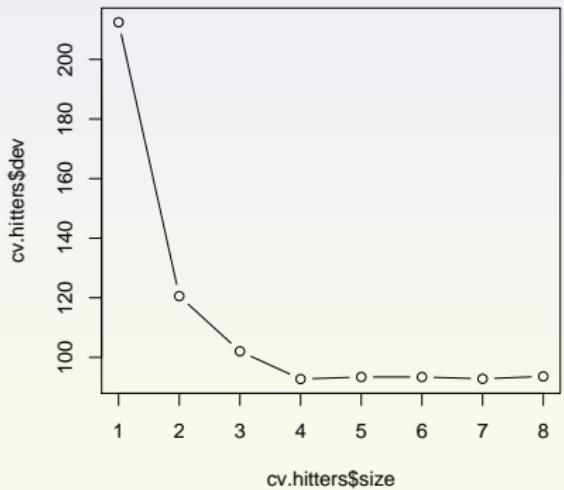
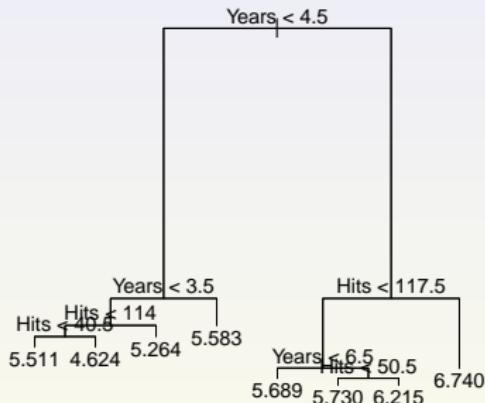
$dev
[1]  95.23044  91.91239  95.49769  95.49769  90.07986  96.01860 117.07588 211.16929

$k
[1]      -Inf  2.293634  3.470318  3.501308  3.793540  9.210099 23.728527 92.095258

$method
[1] "deviance"

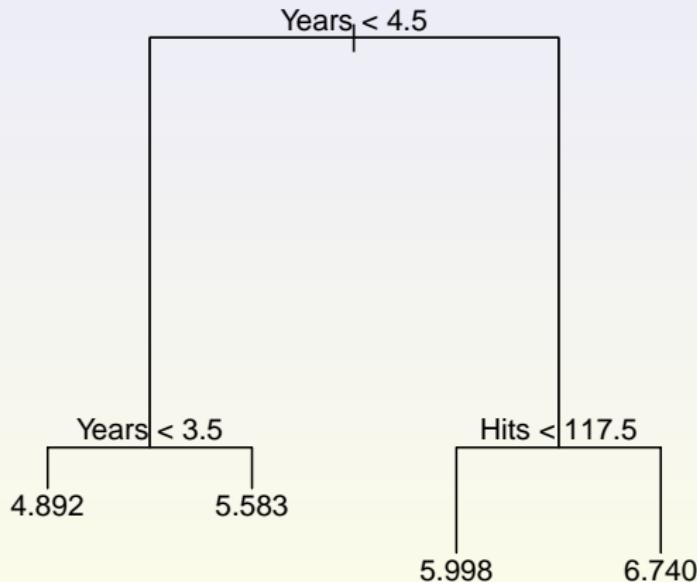
attr(,"class")
[1] "prune"           "tree.sequence"
```

Example: Hitters



Example: Hitters

```
> prune.hitters<-prune.tree(hitters.fit,best=cv.hitters$size[which.min(cv.hitters$dev)])
```



Classification trees

- ▶ Similar to regression trees
- ▶ Predict a qualitative response
- ▶ **Prediction within a box:** most commonly occurring class
- ▶ Need an alternative to RSS

Objective

- ▶ $\hat{p}_{m,k}$ – proportion of training observations in the m th box that are from class k
- ▶ Minimize one of the following measures
 - ▶ Classification error rate

$$E = 1 - \max_k \hat{p}_{m,k}$$

- ▶ Gini index

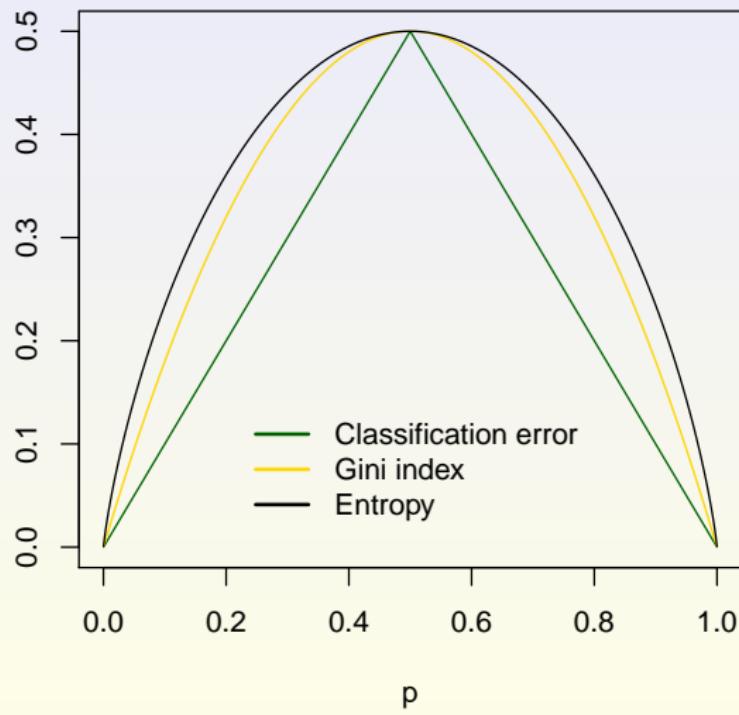
$$G = \sum_{k=1}^K \hat{p}_{m,k} (1 - \hat{p}_{m,k})$$

- ▶ Entropy

$$D = - \sum_{k=1}^K \hat{p}_{m,k} \log \hat{p}_{m,k}$$

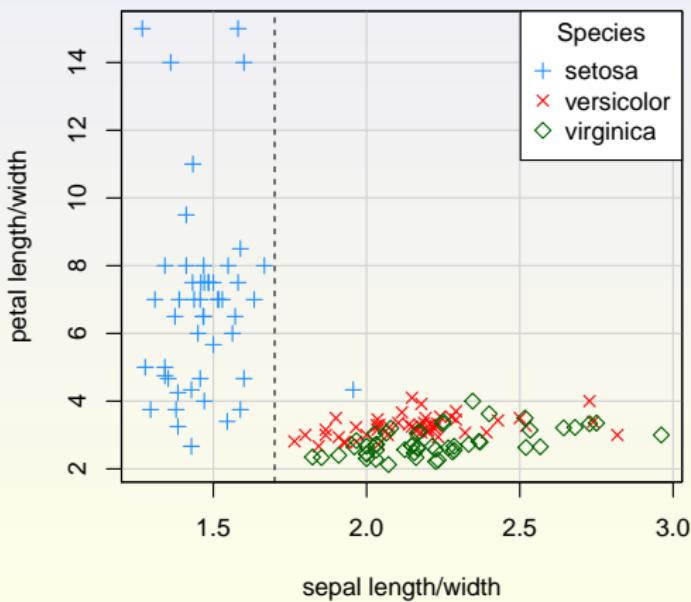
Measures

- $K = 2$



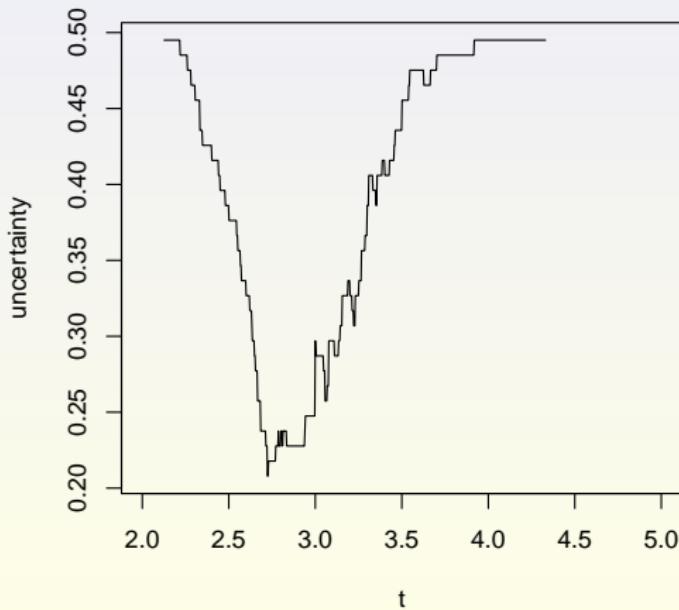
Example: Irises

- ▶ Classifying irises using sepal and petal measurements:
 - ▶ $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
 - ▶ $x_1 = \text{ratio of sepal length to width}$
 - ▶ $x_2 = \text{ratio of petal length to width}$



Example: Irises

- ▶ Split R_2 using $1_{\{x_2 \leq t\}}$
 - ▶ $u(R_2^-)$
 - ▶ $u(R_2^+)$
 - ▶ $p_{R_2^-} u(R_2^-) + p_{R_2^+} u(R_2^+)$



Example: South African heart disease set

```
> heart.tree<-tree(chd~.,data=SAheart)
> summary(heart.tree)

Classification tree:
tree(formula = chd ~ ., data = SAheart)
Variables actually used in tree construction:
[1] "age"      "tobacco"   "alcohol"   "typea"     "famhist"   "adiposity" "ldl"
Number of terminal nodes:  15
Residual mean deviance:  0.8733 = 390.3 / 447
Misclassification error rate: 0.2078 = 96 / 462

> set.seed(1)
> cv.heart<-cv.tree(heart.tree,FUN=prune.misclass)
> cv.heart
$size
[1] 15 10  9  6  5  4  1

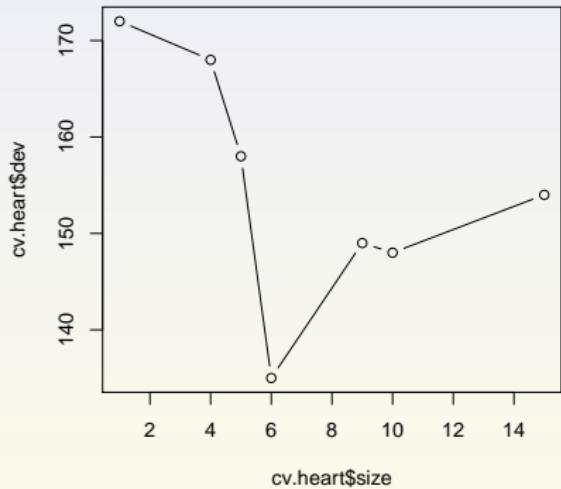
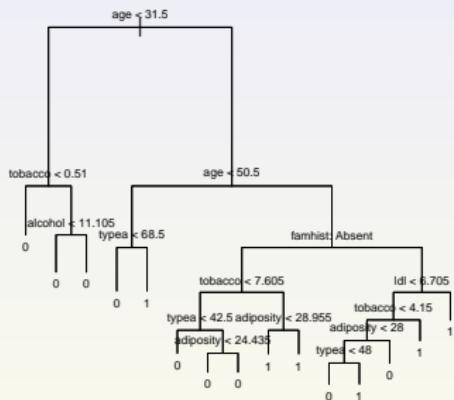
$dev
[1] 154 148 149 135 158 168 172

$k
[1] -Inf    0     1     3     8    10    12

$method
[1] "misclass"

attr(,"class")
[1] "prune"           "tree.sequence"
```

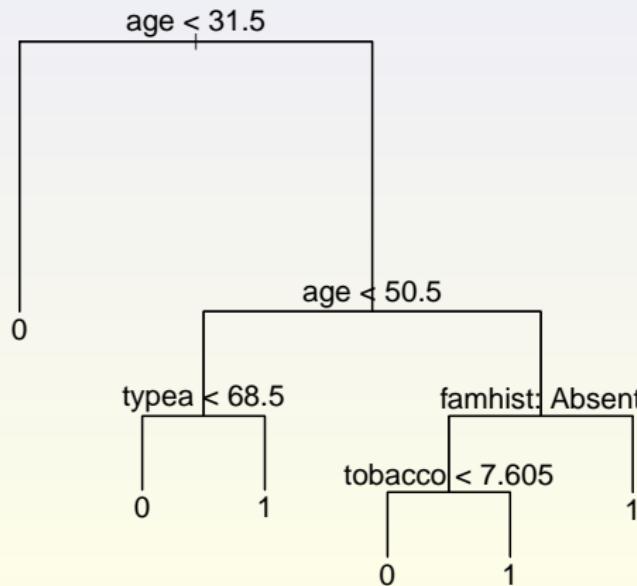
Example: South African heart disease set



Example: South African heart disease set

```
> heart.prune<-prune.misclass(heart.tree,best=cv.heart$size[which.min(cv.heart$dev)])
> heart.predict<-predict(heart.prune,data=SAheart,type="class")
> table(heart.predict,SAheart$chd)
```

```
heart.predict 0 1
              0 266 70
              1 36  90
```



Recall Bootstrap: Basic algorithm

- ▶ Input

- ▶ A sample of data $\mathbf{Z} = (\mathbf{Z}_1, \dots, \mathbf{Z}_n)$
- ▶ An estimation rule \hat{T} for Statistic T

- ▶ Algorithm

1. Generate bootstrap samples $\mathbf{Z}^{*1}, \mathbf{Z}^{*2}, \dots, \mathbf{Z}^{*B}$
 - ▶ Create \mathbf{Z}^{*b} by selecting points from \mathbf{Z}
 - ▶ A particular \mathbf{Z}_i can appear in \mathbf{Z}^{*b} multiple times
2. Evaluate the estimator on each \mathbf{Z}^{*b} :

$$\hat{T}_b = \hat{T}(\mathbf{Z}^{*b})$$

- ▶ The empirical distribution of $\{\hat{T}_1, \dots, \hat{T}_B\}$ is an estimate of the distribution of $T(\mathbf{Z})$
- ▶ Bootstrap distribution
- ▶ Overlap between \mathbf{Z} and \mathbf{Z}^{*b} ?

Bumping

Works for both: classifiers or regressions

- ▶ Stochastic search

 avoids getting stuck in a poor solution/local minimum

- ▶ Train a classifier or regression model \hat{f}_0 on Z
- ▶ For $b = 1, \dots, B$:
 1. Draw a bootstrap sample Z^{*b} of size n from training data
 2. Train a classifier or regression model \hat{f}_b on Z^{*b}
- ▶ Select the best model, e.g.,

$$\hat{b} = \arg \min_{0 \leq b \leq B} \sum_{i=1}^n (y_i - \hat{f}_b(z_i))^2$$

Bagging

Works for both: classifiers or regressions

- ▶ Bootstrap aggregation/averaging
 - reduces the variance/overfitting
- ▶ For $b = 1, \dots, B$:
 1. Draw a bootstrap sample Z^{*b} of size n from training data
 2. Train a classifier or regression model \hat{f}_b on Z^{*b}
- ▶ For a “new” point x_0 , compute:

$$\hat{f}_{\text{avg}}(\mathbf{x}_0) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{x}_0)$$

- ▶ Regression: $\hat{f}_{\text{avg}}(\mathbf{x}_0)$ is the prediction
- ▶ Classification: Pick majority
- ▶ Example: Bagging trees

Random Forests

Works for both: classifiers or regressions

- ▶ Improvement over bagged trees
- ▶ Idea: Decorrelated trees
 - ▶ Still learn a tree on each bootstrap set
 - ▶ To split a region, consider only a subset of predictors/covariates
- ▶ Input parameter: $m \leq p$, often $m \approx \sqrt{p}$
- ▶ For $b = 1, \dots, B$
 - ▶ Draw a bootstrap sample Z^{*b} of size n from the training data
 - ▶ Train a tree classifier on Z^{*b} , each split is computed as:
 - ▶ Randomly select m predictors/covariates, newly chosen for each b
 - ▶ Make the best split restricted to that subsets of covariates
- ▶ Similarly as in bagging: for regression prediction

$$\hat{f}_{\text{avg}}(\mathbf{x}_0) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{x}_0);$$

for classification: choose the majority vote among B classifiers.

Example: South African heart disease set

```
> library(randomForest)
> set.seed(10)
> train<-sample(1:nrow(SAheart),nrow(SAheart)/2)
> bag.heart<-randomForest(chd~., data = SAheart, subset=train, mtry=9, importance=TRUE)
> bag.heart

Call:
randomForest(formula = chd ~ ., data = SAheart, mtry = 9, importance = TRUE,      subset = train)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 9

OOB estimate of  error rate: 36.8%
Confusion matrix:
  0  1 class.error
0 123 31  0.2012987
1  54 23  0.7012987
> table(SAheart$chd[-train],predict(bag.heart, newdata = SAheart[-train,]))

   0   1
0 118 30
1  54 29
>
> bag.heart<-randomForest(chd~., data = SAheart, subset=train, mtry=3, importance=TRUE)
> bag.heart

Call:
randomForest(formula = chd ~ ., data = SAheart, mtry = 3, importance = TRUE,      subset = train)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 3

OOB estimate of  error rate: 33.33%
Confusion matrix:
  0  1 class.error
0 134 20  0.1298701
1  57 20  0.7402597
> table(SAheart$chd[-train],predict(bag.heart, newdata = SAheart[-train,]))

   0   1
0 128 20
1  58 25
```

Reading:

ISL: Read Chapter 8

ESL: Section 9.2

Homework: Homework 3 due Wed, Oct 19.

No late submission allowed in order to give you enough time to study the solutions before the midterm, which is planned for Oc 25.

EECS E6690: SL for Bio & Info

Lecture 7: Boosting and Intro to Support Vector Machines

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm
303 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Last lecture: Tree-based methods

Decision trees

- ▶ Models for both: regression and classification
- ▶ Idea:
 - ▶ Segment the predictor space (the set of possible values for X_1, \dots, X_p) into distinct and non-overlapping regions, R_1, \dots, R_j
 - ▶ Regression (classification): Average (majority vote) over segments

Advantages and Disadvantages of Trees

Interpretation vs. Prediction accuracy

Advantages:

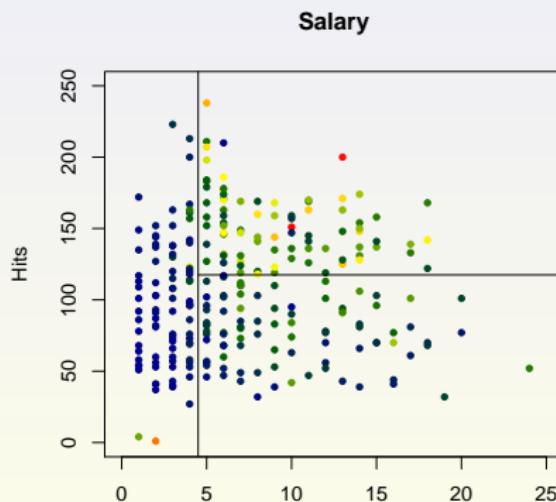
- ▶ Easy to explain to people.
Even easier than linear regression!
- ▶ Some believe that decision trees mirror human decision-making.
- ▶ Can be graphically displayed and interpreted by non-experts.

Disadvantages:

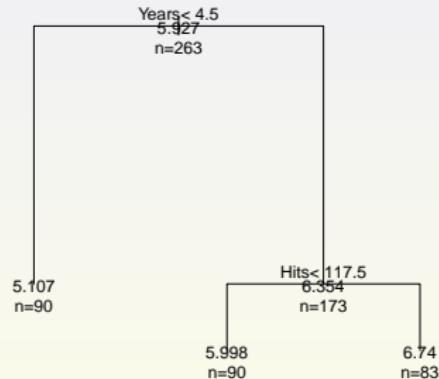
- ▶ Lower predictive accuracy than other methods.
- ▶ Not robust/sensitive: small changes in data can result in large changes in the final decision tree.

Example: Hitters

- ▶ Predict Salary based on Years and Hits
- ▶ Remove missing values and apply log-transform
- ▶ Salary encoding from low to high: blue - green - yellow - red



Classification tree for Salary



- ▶ Intuitive interpretation
- ▶ Prediction

Last lecture: Segmentation

- ▶ In general, the regions can have any shape
- ▶ Focus on high-dimensional rectangles (boxes)
- ▶ Goal: Find R_1, \dots, R_J that minimize the RSS:

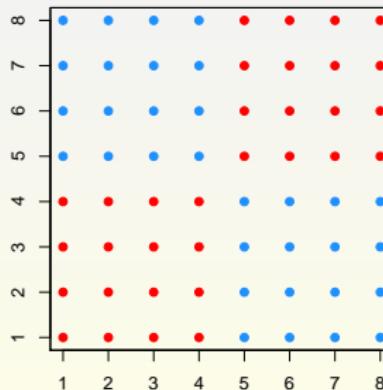
$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where \hat{y}_{R_j} is the mean response for the observations in R_j

- ▶ Computationally infeasible to consider all partitions
- ▶ Top-down, greedy approach: Binary splitting
- ▶ Stopping criteria (e.g., max number of observations in a box)

Last lecture: Cannot grow trees sequentially

- ▶ Optimal tree size?
 - ▶ Training error decreases as the size increases
 - ▶ Testing error decreases, but then increases
- ▶ Grow the tree only if RSS decreases – poor results
- ▶ Example: 2 values: red and blue
always same RSS regardless of the cut
but the next cut - there is (!)



- ▶ Alternative: Grow the tree to a large size and then trim/prune it

Last lecture: Tree pruning

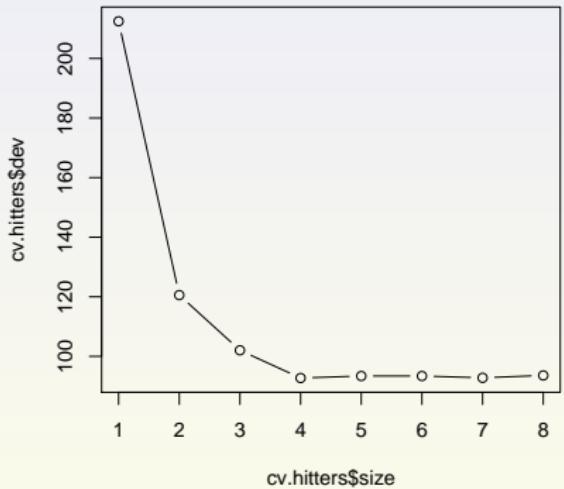
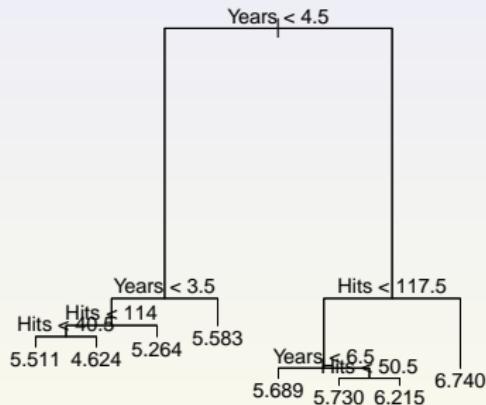
- ▶ Start with a large tree T_0
- ▶ Cost complexity pruning (weakest link pruning)
- ▶ Sequence of trees indexed by α
- ▶ For each α :

$$\min_{T \subseteq T_0} \left\{ \sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \right\},$$

where

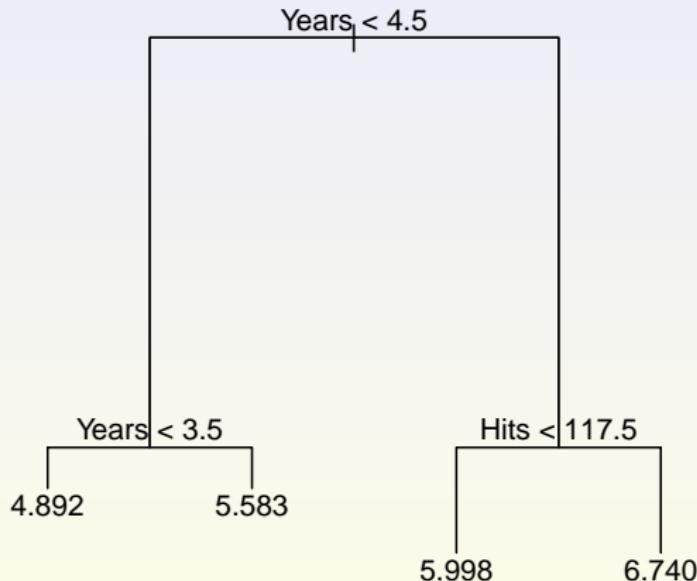
- ▶ $|T|$ is the number of leafs in T
- ▶ R_m is the box corresponding to the m th leaf
- ▶ \hat{y}_{R_m} is the mean of training observations in R_m
- ▶ Parameter α
 - ▶ Controls the complexity/fit tradeoff
 - ▶ Select $\hat{\alpha}$ using cross-validation

Example: Hitters - build a large tree



Example: Hitters - pruning

```
> prune.hitters<-prune.tree(hitters.fit,best=cv.hitters$size[which.min(cv.hitters$dev)])
```



Last lecture: Classification trees

- ▶ Similar to regression trees
- ▶ Predict a qualitative response
- ▶ **Prediction within a box:** most commonly occurring class
- ▶ Need an alternative to RSS

Last lecture: Error measures for classification

- ▶ $\hat{p}_{m,k}$ – proportion of training observations in the m th box that are from class k
- ▶ Minimize one of the following measures
 - ▶ Classification error rate

$$E = 1 - \max_k \hat{p}_{m,k}$$

- ▶ Gini index

$$G = \sum_{k=1}^K \hat{p}_{m,k}(1 - \hat{p}_{m,k})$$

- ▶ Entropy (or, deviance= $2D$)

$$D = - \sum_{k=1}^K \hat{p}_{m,k} \log \hat{p}_{m,k}$$

Example: SA heart disease data - build a large tree

```
> heart.tree<-tree(chd~.,data=SAheart)
> summary(heart.tree)

Classification tree:
tree(formula = chd ~ ., data = SAheart)
Variables actually used in tree construction:
[1] "age"      "tobacco"   "alcohol"   "typea"     "famhist"   "adiposity" "ldl"
Number of terminal nodes:  15
Residual mean deviance:  0.8733 = 390.3 / 447
Misclassification error rate: 0.2078 = 96 / 462

> set.seed(1)
> cv.heart<-cv.tree(heart.tree,FUN=prune.misclass)
> cv.heart
$size
[1] 15 10  9  6  5  4  1

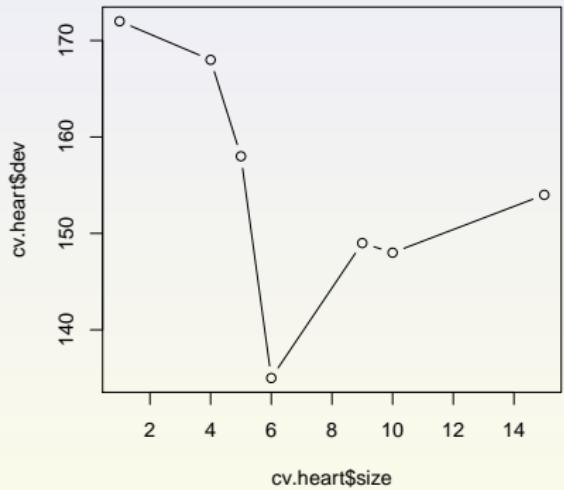
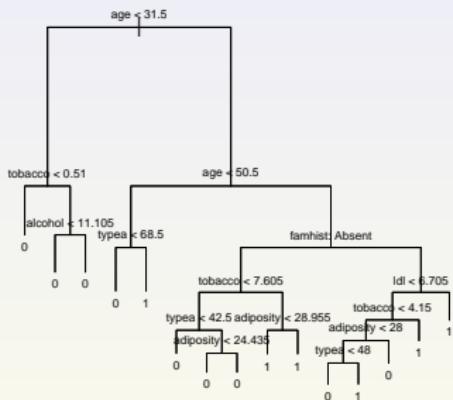
$dev
[1] 154 148 149 135 158 168 172

$k
[1] -Inf    0     1     3     8    10    12

$method
[1] "misclass"

attr(,"class")
[1] "prune"           "tree.sequence"
```

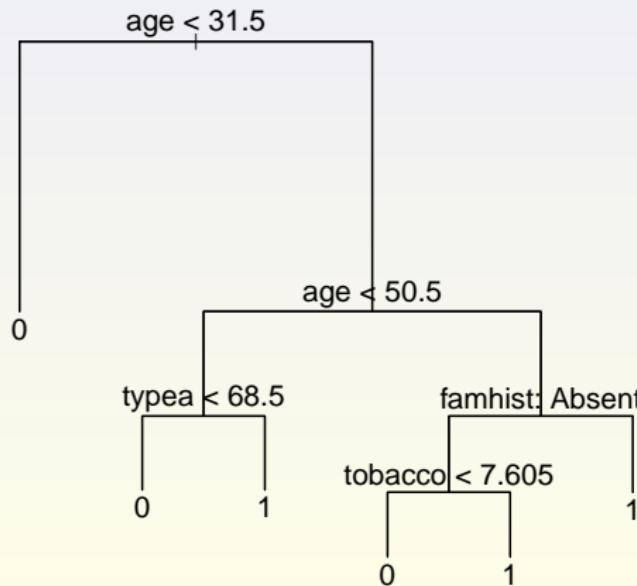
Example: SA heart disease data - after pruning



Example: South African heart disease set

```
> heart.prune<-prune.misclass(heart.tree,best=cv.heart$size[which.min(cv.heart$dev)])
> heart.predict<-predict(heart.prune,data=SAheart,type="class")
> table(heart.predict,SAheart$chd)
```

```
heart.predict 0 1
              0 266 70
              1 36  90
```



Bumping

Works for both: classifiers or regressions

- ▶ Stochastic search

 avoids getting stuck in a poor solution/local minimum

- ▶ Train a classifier or regression model \hat{f}_0 on Z
- ▶ For $b = 1, \dots, B$:
 1. Draw a bootstrap sample Z^{*b} of size n from training data
 2. Train a classifier or regression model \hat{f}_b on Z^{*b}
- ▶ Select the best model, e.g.,

$$\hat{b} = \arg \min_{0 \leq b \leq B} \sum_{i=1}^n (y_i - \hat{f}_b(z_i))^2$$

Bagging

Works for both: classifiers or regressions

- ▶ Bootstrap aggregation/averaging
 - reduces the variance/overfitting
- ▶ For $b = 1, \dots, B$:
 1. Draw a bootstrap sample Z^{*b} of size n from training data
 2. Train a classifier or regression model \hat{f}_b on Z^{*b}
- ▶ For a “new” point x_0 , compute:

$$\hat{f}_{\text{avg}}(\mathbf{x}_0) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{x}_0)$$

- ▶ Regression: $\hat{f}_{\text{avg}}(\mathbf{x}_0)$ is the prediction
- ▶ Classification: Pick majority
- ▶ Example: Bagging trees

Random Forests

Works for both: classifiers or regressions

- ▶ Improvement over bagged trees
- ▶ Idea: Decorrelated trees
 - ▶ Still learn a tree on each bootstrap set
 - ▶ To split a region, consider only a subset of predictors/covariates
- ▶ Input parameter: $m \leq p$, often $m \approx \sqrt{p}$
- ▶ For $b = 1, \dots, B$
 - ▶ Draw a bootstrap sample Z^{*b} of size n from the training data
 - ▶ Train a tree classifier on Z^{*b} , each split is computed as:
 - ▶ Randomly select m predictors/covariates, newly chosen for each b
 - ▶ Make the best split restricted to that subsets of covariates
- ▶ Similarly as in bagging: for regression prediction

$$\hat{f}_{\text{avg}}(\mathbf{x}_0) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{x}_0);$$

for classification: choose the majority vote among B classifiers.

Example: South African heart disease set

```
> library(randomForest)
> set.seed(10)
> train<-sample(1:nrow(SAheart),nrow(SAheart)/2)
> bag.heart<-randomForest(chd~., data = SAheart, subset=train, mtry=9, importance=TRUE)
> bag.heart

Call:
randomForest(formula = chd ~ ., data = SAheart, mtry = 9, importance = TRUE,      subset = train)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 9

OOB estimate of  error rate: 36.8%
Confusion matrix:
  0  1 class.error
0 123 31  0.2012987
1  54 23  0.7012987
> table(SAheart$chd[-train],predict(bag.heart, newdata = SAheart[-train,]))

   0   1
0 118 30
1  54 29
>
> bag.heart<-randomForest(chd~., data = SAheart, subset=train, mtry=3, importance=TRUE)
> bag.heart

Call:
randomForest(formula = chd ~ ., data = SAheart, mtry = 3, importance = TRUE,      subset = train)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 3

OOB estimate of  error rate: 33.33%
Confusion matrix:
  0  1 class.error
0 134 20  0.1298701
1  57 20  0.7402597
> table(SAheart$chd[-train],predict(bag.heart, newdata = SAheart[-train,]))

   0   1
0 128 20
1  58 25
```

Boosting

Slow learning

- ▶ General approach that can be applied to many ML methods
- ▶ Focus on trees
 - ▶ Works for both regression and classification
- ▶ Similar to Random Forests
 - ▶ Make a family of weak learners
 - ▶ Then, the solution is a combination of many of these weak learners, $\hat{f}^1, \dots, \hat{f}^B$
 - ▶ Note, boosting does not use the bootstrap samples
- ▶ Idea:
 - ▶ Build trees sequentially in small increments by adding weak learners
 - ▶ Use the previous tree and residuals to create a new one

Boosting Algorithm for Regression

1. Set $\hat{f}(\mathbf{x}_i) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, \dots, B$, repeat:
 - 2.1 Fit a tree \hat{f}_b with d splits ($d + 1$ terminal nodes) to training data (\mathbf{X}, \mathbf{r}) , $\mathbf{r} = (r_1, \dots, r_n)$
 - 2.2 Update \hat{f} by adding in a shrunken version of the new tree:

$$\hat{f}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda \hat{f}^b(\mathbf{x})$$

- 2.3 Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}_b(\mathbf{x}_i)$$

3. Output the boosted model:

$$\hat{f}(\mathbf{x}) = \sum_{b=1}^B \lambda \hat{f}_b(\mathbf{x})$$

► Notes:

- λ is a small positive number (e.g., 0.01 or 0.001)
- Often $d = 1$ works

Example: Hitters

```
> library(gbm)
> set.seed(1)
> train<-sample(1:nrow(myHitters),nrow(myHitters)/2)
> hitters.boost<-gbm(Salary~,data = myHitters[train,], distribution = "gaussian", n.trees = 5000,
+ interaction.depth = 4 )
> hatSalary<-predict(hitters.boost, newdata=myHitters[-train,], n.trees = 5000)
> mean((myHitters$Salary[-train] - hatSalary)^2)
[1] 0.1802494
>
> hitters.boost<-gbm(Salary~,data = myHitters[train,], distribution = "gaussian", n.trees = 5000,
+ interaction.depth = 4, shrinkage = 0.1 )
> hatSalary<-predict(hitters.boost, newdata=myHitters[-train,], n.trees = 5000)
> mean((myHitters$Salary[-train] - hatSalary)^2)
[1] 0.2967324
```

Classification: AdaBoost

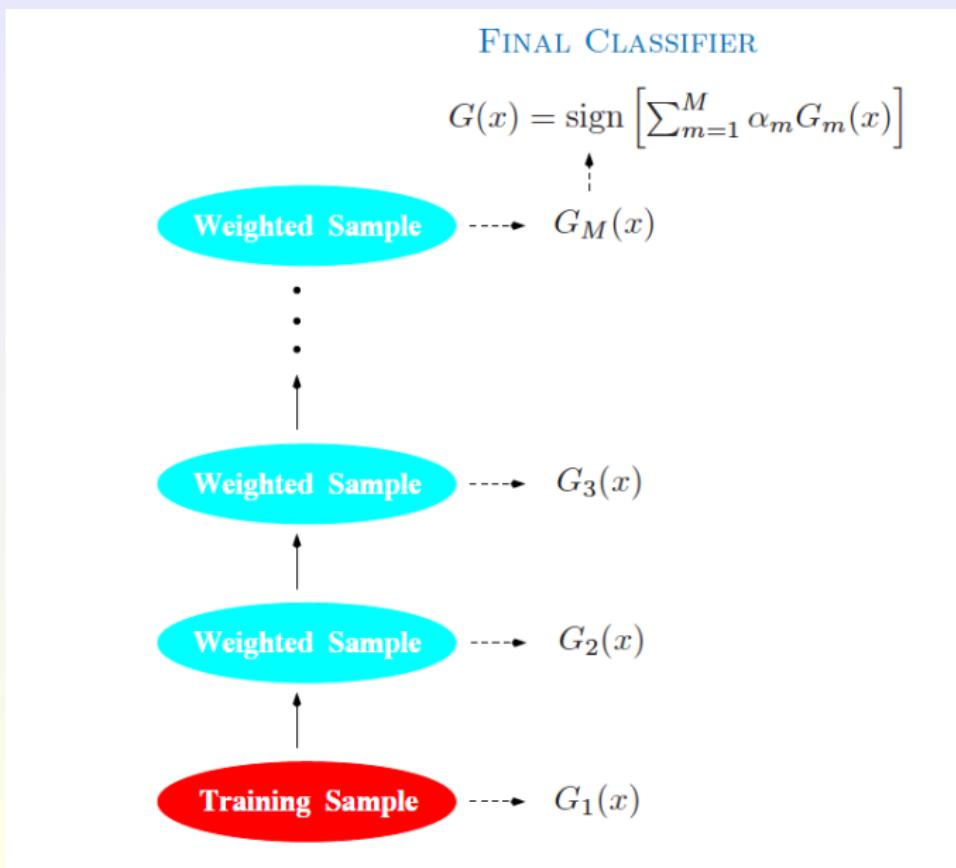
- ▶ Due to Freund and Schapire (1997)
- ▶ Consider two classes: $Y \in \{-1, 1\}$
- ▶ For a classifier $G(x) \in \{-1, 1\}$, the training error is

$$e_m = \frac{1}{n} \sum_{i=1}^n 1_{\{y_i \neq G(x_i)\}}$$

- ▶ Main idea: construct **weak classifiers**
 - ▶ Weak classifier: slightly better than random guessing (50% error)
 - ▶ Sequentially, construct weak classifiers, $G_m(x)$, on modified training data.
- ▶ Final classifier, combination of weak classifiers through a weighted majority vote

$$G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$$

Classification: AdaBoost schematic



Classification: AdaBoost

1. Set $w_i = 1/n, i = 1, 2, \dots, n$, where n is the number of training points.
2. For $m = 1, \dots, M$, repeat:
 - (a) Fit a (weak) classifier $G_m(x)$ to training data using weights w_i .
 - (b) Compute the weighted error

$$e_m = \frac{\sum_{i=1}^n w_i \mathbf{1}_{\{y_i \neq G_m(x_i)\}}}{\sum_{i=1}^n w_i}$$

- (c) Compute $\alpha_m = \log((1 - e_m)/e_m)$.
- (d) Update

$$w_i \leftarrow w_i \exp(\alpha_m \mathbf{1}_{\{y_i \neq G_m(x_i)\}}), \quad , i = 1, 2, \dots, n.$$

3. Final classifier

$$G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$$

Support Vector Machines

Classification methods we learned:

- ▶ Logistic
- ▶ Gaussian discriminant analysis
- ▶ Tree-based

Support Vector Machines (SVM)

- ▶ Idea: separate points using surfaces
 - ▶ linear surfaces - hyperplanes
 - ▶ general nonlinear surfaces

Three types:

- ▶ Maximal Marginal Classifier - separator = hyperplane
- ▶ Support Vector Classifier - separator = "soft" hyperplane
- ▶ Support Vector Machine - separator = nonlinear surface

Hyperplane separator

Hyperplane in p -dimensions is defined by

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

Examples:

- ▶ 2D hyperplane = line

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$$

- ▶ 3D hyperplane = regular plane

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 = 0$$

Classification decision: Assign x in one of the two classes depending on the sign of

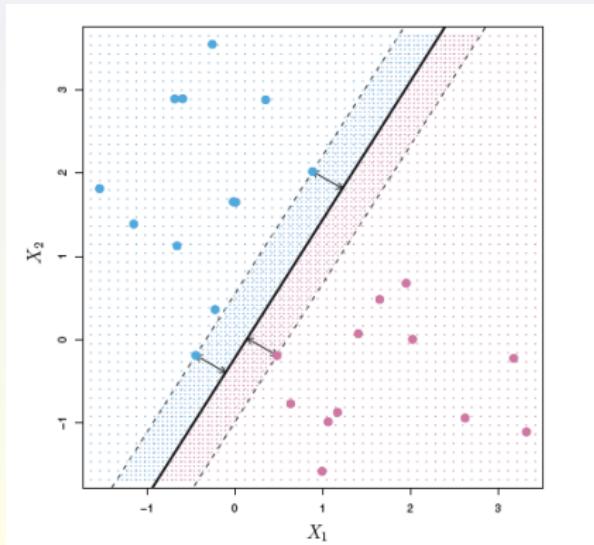
$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p > < 0$$

How do we choose a hyperplane, i.e., β ?

The Maximal Margin Classifier

Optimal hyperplane

- ▶ *Margin*: Distance from an observation to the hyperplane
- ▶ *Maximal margin hyperplane*: One whose smallest margin is maximal
- ▶ *Support vectors*: points that support the maximal margin hyperplane



Understanding geometry: Distance from a hyperplane

Consider a hyperplane in p -dimensions

$$\beta_0 + \langle \boldsymbol{\beta}, \mathbf{x} \rangle = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$ and $\mathbf{x} = (x_1, \dots, x_p)$

- ▶ **Claim:** $\boldsymbol{\beta}$ is a perpendicular vector to this hyperplane

Proof: Let \mathbf{x} and \mathbf{x}' be two points on this hyperplane. Then

$$\beta_0 + \langle \boldsymbol{\beta}, \mathbf{x} \rangle - (\beta_0 + \langle \boldsymbol{\beta}, \mathbf{x}' \rangle) = \langle \boldsymbol{\beta}, (\mathbf{x}' - \mathbf{x}) \rangle = 0$$

- ▶ Perpendicular unit vector

$$\frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}$$

where $\|\boldsymbol{\beta}\| \equiv \|\boldsymbol{\beta}\|_2$ is the usual euclidian norm.

- ▶ **Signed distance** to the hyperplane: Let \mathbf{x}_0 be a point outside the hyperplane and \mathbf{x} inside (note $\langle \boldsymbol{\beta}, \mathbf{x} \rangle = -\beta_0$)

$$\frac{\langle \boldsymbol{\beta}, (\mathbf{x}_0 - \mathbf{x}) \rangle}{\|\boldsymbol{\beta}\|} = \frac{\langle \boldsymbol{\beta}, \mathbf{x}_0 \rangle + \beta_0}{\|\boldsymbol{\beta}\|}$$

Maximum Margin Classifier: Formal computation

Let the two classes be $y_i \in \{1, -1\}$

Then, the Maximal margin hyperplane is the solution to

$$\max_{\beta_j, M} M \quad (1)$$

$$\text{subject to } \sum_1^p \beta_j^2 = 1 \quad (2)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M, \quad \forall i = 1, \dots, n \quad (3)$$

Notes:

- ▶ (3) requires that each observation is on the correct side of the hyperplane
- ▶ The solution M^* is the maximal margin

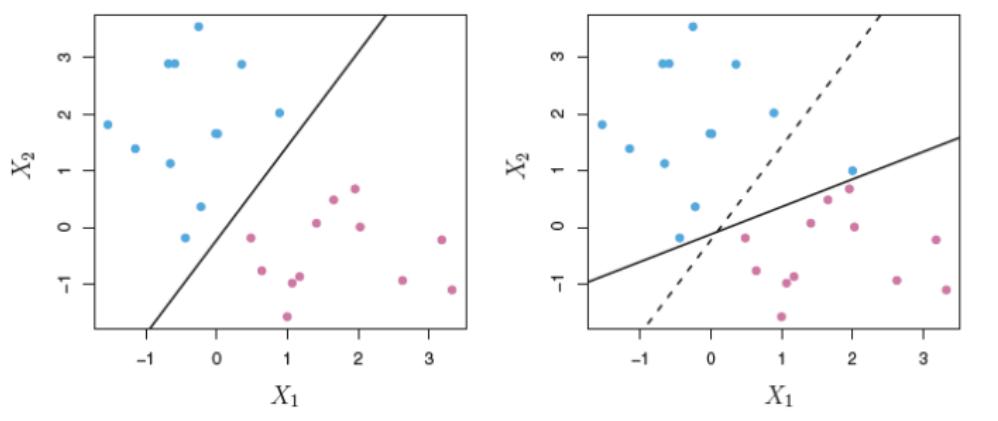
Problems

Non-separable case

- ▶ There is no hyperplane that separates the two classes

Highly sensitive to support vectors

- ▶ The hyperplane moves if one moves or introduces new support vector points



Need a "softer" separator

Support Vector Classifier

- ▶ Greater robustness to individual observations
- ▶ More general: Works for most points

The hyperplane is the solution to

$$\max_{\beta_j, \epsilon_j} M \quad (4)$$

$$\text{subject to } \sum_1^p \beta_j^2 = 1 \quad (5)$$

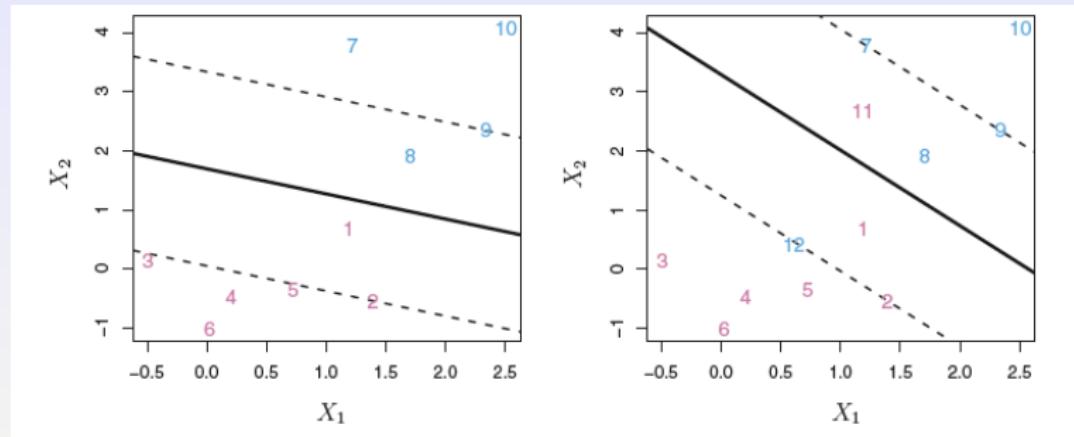
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \quad \forall i \quad (6)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \quad (7)$$

Notes:

- ▶ ϵ_i - slack variables
 - ▶ $\epsilon_i = 0$ - i -th observation on the correct side of the margin
 - ▶ $0 < \epsilon_i < 1$ - i -th observation on the wrong side of the margin
 - ▶ $\epsilon_i > 1$ - i -th observation on the wrong side of the hyperplane
- ▶ C - budget for slackness

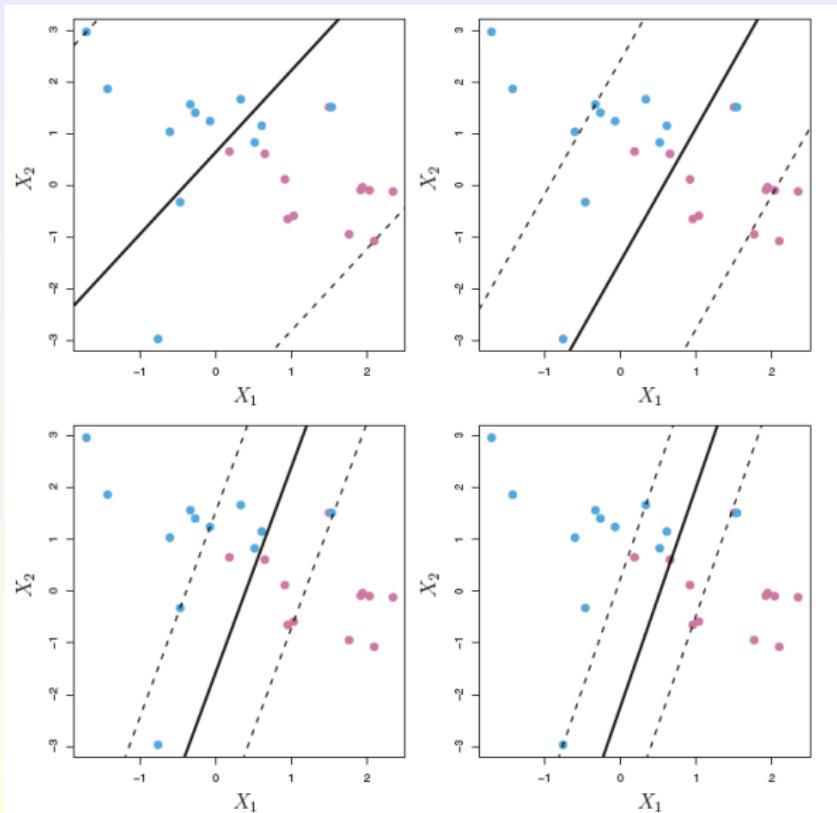
Example



- ▶ Left:
 - ▶ Purple observations: 3,4,5, and 6 = correct side of the margin; 2 is on the margin, and 1 is on the wrong side of the margin.
 - ▶ Blue observations: 7 and 10 = correct side of the margin; 9 is on the margin, and 8 is on the wrong side of the margin.
- ▶ Right: Same as left panel with two additional points, 11 and 12. 11 and 12 = wrong side of both the hyperplane and the margin.
- ▶ **Robustness:** the hyperplane on the right did not move much (!)

Example: Decreasing C

Decreasing C : From top left to bottom right



Support Vector Machines

Nonlinear decision boundary - example

- ▶ p features: X_1, X_2, \dots, X_p
- ▶ expand bases - $2p$ features: $X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$

Fit the hyperplane to the expanded bases

$$\max_{\beta_j, \epsilon_j} M \quad (8)$$

$$\text{subject to } \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1 \quad (9)$$

$$y_i(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2) \geq M(1 - \epsilon_i), \quad \forall i \quad (10)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \quad (11)$$

- ▶ The solutions are in general nonlinear since these are quadratic expressions
- ▶ The SVM explores further this idea (using kernels)

Support Vector Machines

Let $\langle a, b \rangle$ be the inner product

$$\langle a, b \rangle := \sum_{i=1}^p a_i b_i$$

It turns out that the SV classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

with n training parameters, α_i , one per training observation.

- ▶ Estimating α_i and β_0 : requires $\binom{n}{2}$ inner products $\langle x_j, x_i \rangle$
- ▶ α_i is nonzero only for the support vectors (!)
- ▶ Let \mathcal{S} be the set of these support points: $|\mathcal{S}| = \text{small number}$
- ▶ $f(x)$ simplifies to

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

Support Vector Machines: Kernels

Recall, **Kernel** - $K(x_i, x_j)$ is an inner product over expanded basis

$x \rightarrow \phi(x) : \mathbb{R}^p \rightarrow \mathbb{R}^d, d > p$, i.e., $K(x_i, x_j) = \langle \phi(x_j), \phi(x_i) \rangle$

Nice property: we can use Kernel solutions without ever knowing $\phi(x)$

Kernels:

- *Linear*

$$K(x_i, x_j) = \langle x_j, x_i \rangle$$

- *Polynomial* - for positive integer d

$$K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^d$$

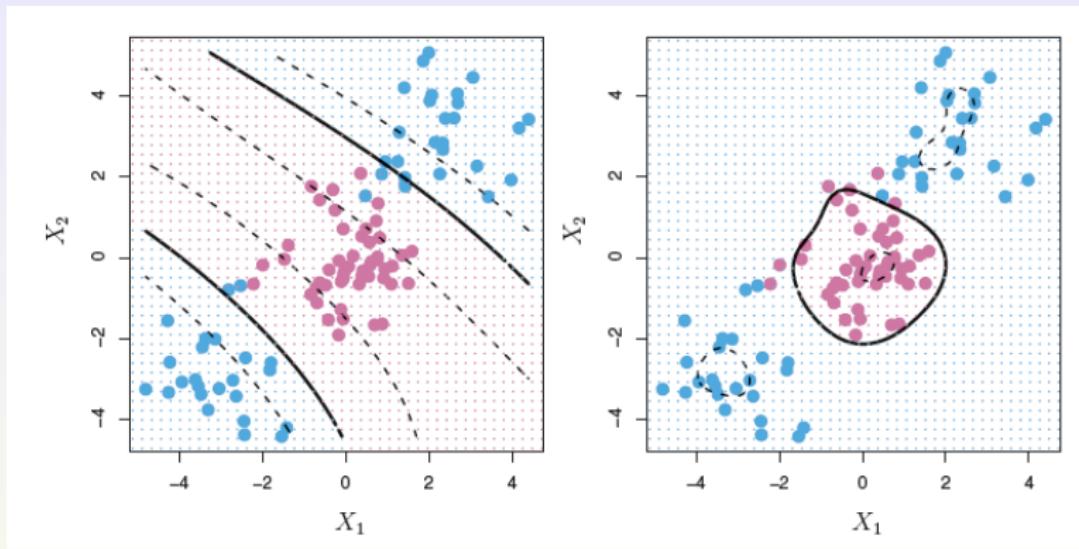
- *Radial* - for $\gamma > 0$

$$K(x_i, x_j) = \exp \left(-\gamma \sum_{k=1}^p (x_{ik} - x_{jk})^2 \right)$$

The resulting classifier is known as **SVM**, with the decision function taking the following nonlinear form in general

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

Example



- ▶ Left: An SVM with a polynomial kernel of degree 3
- ▶ Right: An SVM with a radial kernel
- ▶ Either kernel is capable of capturing the decision boundary

Reading on Bootstrap Improvements and Boosting: Bumping, Bagging, Random forests, Boosting

ISL: Section 8.2

ESL: Sections, 8.7, 8.9, 10.1; More advanced reading: Chapters 10 and 15

Reading on Support Vector Machines

ISL: Sections 9.1-9.3

ESL: Section 12.1-12.3 (more advanced)

Homework: HW3 is due: Wed, Oct 19, 11:59pm.

Since this is the last HW, **no late submission will be permitted**, in order to release the solutions on Thu, Oct 20, to give you time to prepare for **midterm**.

Midterms date: Oct 25, during class time.

Closed book, no electronic devices, but one page - both sides - with **handwritten** formulas/facts is permitted.

EECS E6690: SL for Bio & Info

Lecture 8: Support Vector Machines, Optimization, and Gene Expression Classification

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.

Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Final Project Outline

- ▶ Done in groups of 4 students - assemble the groups
- ▶ Deliverables: 15+ page **paper** & **presentation** with slides
- ▶ **Due:** during the finals week: Dec 16 - 23, very likely **Dec 17**.
One slot for presentations on **Tue, Dec 14, 4:10-6:40pm.**
- ▶ **Data Repositories:** First, select a paper(s) from either:
 - ▶ UC Irvine Machine Learning Repository
<https://archive.ics.uci.edu/ml/datasets.php>
 - ▶ GEO Data Repository <https://www.ncbi.nlm.nih.gov/geo/>,
or Bioconductor Datasets
<http://www.bioconductor.org/packages/release/data/experiment/>
- ▶ **Final Paper Outline:** 5 sections
 1. **Introduction:** e.g., describe the application area, problems considered, etc
 2. **Data set(s) and paper(s):** e.g., describe data in detail, what was done in the paper(s), common stat/machine learning tools, etc
 3. **Reproduce the results from the paper(s)**
 4. **Try different techniques learned in class, or propose new ones**
 5. **Discussion and conclusion:** e.g., compare different techniques, pros and cons, future work, etc

Bioconductor and Additional Datasets

- ▶ Bioconductor provides tools in R for the analysis genomic data:
<https://www.bioconductor.org/>

- ▶ Installing Bioconductor:

<https://www.bioconductor.org/install/>

Run the following code:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.12")
```

- ▶ Then, install Bioconductor packages:

[https://www.bioconductor.org/install/
#install-bioconductor-packages](https://www.bioconductor.org/install/#install-bioconductor-packages)

- ▶ Datasets supported by Bioconductor: <http://www.bioconductor.org/packages/release/data/experiment/>

Last lecture: Bootstrap Methods

- ▶ Stochastic search
- ▶ Works for both: classifiers or regressions
- ▶ Bumping, Bagging, Random forests, Boosting
- ▶ Train a classifier or regression model \hat{f}_0 on Z
- ▶ For $b = 1, \dots, B$:
 1. Draw a bootstrap sample Z^{*b} of size n from training data
 2. Train a classifier or regression model \hat{f}_b on Z^{*b}
- ▶ Bumping: Select the best model, e.g.,

$$\hat{b} = \arg \min_{0 \leq b \leq B} \sum_{i=1}^n (y_i - \hat{f}_b(z_i))^2$$

- ▶ Bagging: average out - reduces variance

For a “new” point x_0 , compute:

$$\hat{f}_{\text{avg}}(x_0) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x_0)$$

Last lecture: Random forests

Works for both: classifiers or regressions

- ▶ Improvement over bagged trees
- ▶ Idea: Decorrelate trees
 - ▶ Still learn a tree on each bootstrap set
 - ▶ To split a region, consider only a subset of predictors

- ▶ Input parameter: $m \leq p$, often $m \approx \sqrt{p}$
- ▶ For $b = 1, \dots, B$
 - ▶ Draw a bootstrap sample Z^{*b} of size n from the training data
 - ▶ Train a tree classifier on Z^{*b} , each split is computed as:
 - ▶ Randomly select m predictors, newly chosen for each b
 - ▶ Make the best split restricted to that subsets of predictors

Last lecture: Boosting for regression

Slow learning

1. Set $\hat{f}(\mathbf{x}_i) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, \dots, B$, repeat:
 - 2.1 Fit a tree \hat{f}_b with d splits ($d + 1$ terminal nodes) to training data (\mathbf{X}, \mathbf{r})
 - 2.2 Update \hat{f} by adding in a shrunken version of the new tree:

$$\hat{f}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda \hat{f}^b(\mathbf{x})$$

- 2.3 Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}_b(\mathbf{x}_i)$$

3. Output the boosted model:

$$\hat{f}(\mathbf{x}) = \sum_{b=1}^B \lambda \hat{f}_b(\mathbf{x})$$

► Notes:

- λ is a small positive number (e.g., 0.01 or 0.001)
- Often $d = 1$ works

Boosting for Classification: AdaBoost

- ▶ Due to Freund and Schapire (1997)
- ▶ Consider two classes: $Y \in \{-1, 1\}$
- ▶ For a classifier $G(x) \in \{-1, 1\}$, the training error is

$$e_m = \frac{1}{n} \sum_{i=1}^n 1_{\{y_i \neq G(x_i)\}}$$

- ▶ Main idea: construct **weak classifiers**
 - ▶ Weak classifier: slightly better than random guessing (50% error)
 - ▶ Sequentially, construct weak classifiers, $G_m(x)$, on modified training data.
- ▶ Final classifier, combination of weak classifiers through a weighted majority vote

$$G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$$

Boosting for Classification: AdaBoost

1. Set $w_i = 1/n, i = 1, 2, \dots, n$, where n is the number of training points.
2. For $m = 1, \dots, M$, repeat:
 - (a) Fit a (weak) classifier $G_m(x)$ to the training data using weights w_i .
 - (b) Compute the weighted error

$$e_m = \frac{\sum_{i=1}^n w_i \mathbf{1}_{\{y_i \neq G_m(x_i)\}}}{\sum_{i=1}^n w_i}$$

- (c) Compute

$$\alpha_m = \log((1 - e_m)/e_m).$$

- (d) Update

$$w_i \leftarrow w_i \exp(\alpha_m \mathbf{1}_{\{y_i \neq G_m(x_i)\}}), \quad , i = 1, 2, \dots, n.$$

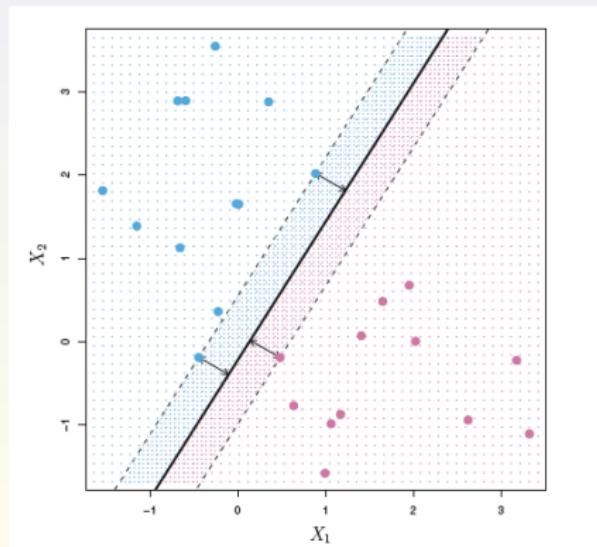
3. Final classifier

$$G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$$

Last lecture: The Maximal Margin Classifier

Optimal hyperplane

- ▶ *Margin*: Distance from an observation to the hyperplane
- ▶ *Maximal margin hyperplane*: One whose smallest margin is maximal
- ▶ *Support vectors*: points that support the maximal margin hyperplane



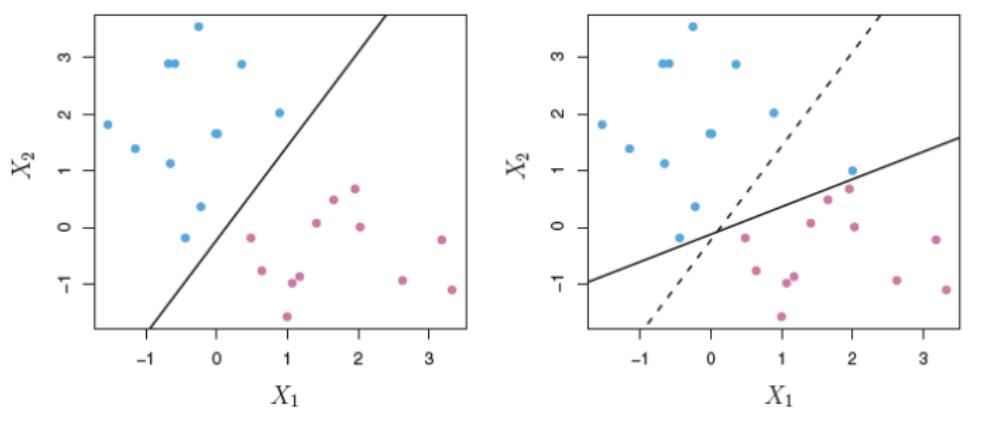
Problems

Non-separable case

- ▶ There is no hyperplane that separates the two classes

Highly sensitive to support vectors

- ▶ The hyperplane moves if one moves or introduces new support vector points



Need a "softer" separator

Last lecture: Support Vector Classifier

- ▶ Greater robustness to individual observations
- ▶ More general: Works for most points

The hyperplane is the solution to

$$\max_{\beta_j, \epsilon_j} M \quad (1)$$

$$\text{subject to } \sum_1^p \beta_j^2 = 1 \quad (2)$$

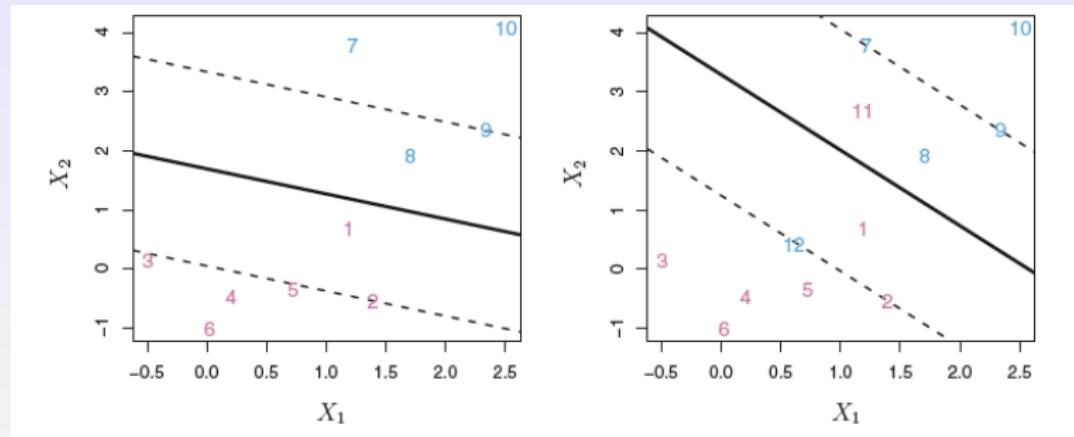
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \quad \forall i \quad (3)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \quad (4)$$

Notes:

- ▶ ϵ_i - *slack variables*
 - ▶ $\epsilon_i = 0$ - i -th observation on the correct side of the margin
 - ▶ $0 < \epsilon_i < 1$ - i -th observation on the wrong side of the margin
 - ▶ $\epsilon_i > 1$ - i -th observation on the wrong side of the hyperplane
- ▶ C - budget for slackness

Example



- ▶ Left:
 - ▶ Purple observations: 3,4,5, and 6 = correct side of the margin; 2 is on the margin, and 1 is on the wrong side of the margin.
 - ▶ Blue observations: 7 and 10 = correct side of the margin; 9 is on the margin, and 8 is on the wrong side of the margin.
- ▶ Right: Same as left panel with two additional points, 11 and 12. 11 and 12 = wrong side of both the hyperplane and the margin.
- ▶ **Robustness:** the hyperplane on the right did not move much (!)

General Support Vector Machines

Nonlinear decision boundary - example

- ▶ p features: X_1, X_2, \dots, X_p
- ▶ expand bases - $2p$ features: $X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$

Fit the hyperplane to the expanded bases

$$\max_{\beta_j, \epsilon_j} M \quad (5)$$

$$\text{subject to } \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1 \quad (6)$$

$$y_i(\beta_0 + \sum_{j=1}^p \beta_{j1}x_{ij} + \sum_{j=1}^p \beta_{j2}x_{ij}^2) \geq M(1 - \epsilon_i), \quad \forall i \quad (7)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \quad (8)$$

- ▶ Nonlinear surface since these are quadratic expressions
Quadratic can be replaced by polynomial of any degree
- ▶ The SVM explores further this idea using kernels

Understanding geometry: Distance from a hyperplane

Consider a hyperplane in p -dimensions

$$\beta_0 + \langle \boldsymbol{\beta}, \mathbf{x} \rangle = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$ and $\mathbf{x} = (x_1, \dots, x_p)$

- ▶ **Claim:** $\boldsymbol{\beta}$ is a perpendicular vector to this hyperplane

Proof: Let \mathbf{x} and \mathbf{x}' be two points on this hyperplane. Then

$$\beta_0 + \langle \boldsymbol{\beta}, \mathbf{x} \rangle - (\beta_0 + \langle \boldsymbol{\beta}, \mathbf{x}' \rangle) = \langle \boldsymbol{\beta}, (\mathbf{x}' - \mathbf{x}) \rangle = 0$$

- ▶ Perpendicular unit vector

$$\frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}$$

where $\|\boldsymbol{\beta}\| \equiv \|\boldsymbol{\beta}\|_2$ is the usual euclidian norm.

- ▶ **Signed distance** to the hyperplane: Let \mathbf{x}_0 be a point outside the hyperplane and \mathbf{x} inside (note $\langle \boldsymbol{\beta}, \mathbf{x} \rangle = -\beta_0$)

$$\frac{\langle \boldsymbol{\beta}, (\mathbf{x}_0 - \mathbf{x}) \rangle}{\|\boldsymbol{\beta}\|} = \frac{\langle \boldsymbol{\beta}, \mathbf{x}_0 \rangle + \beta_0}{\|\boldsymbol{\beta}\|}$$

SVC Geometry

Using the preceding distance formula, we get

$$\begin{aligned} & \max_{\beta_j, \epsilon_j M} M \\ \text{subject to } & y_i \frac{\langle \beta, x \rangle + \beta_0}{\|\beta\|} \geq M(1 - \epsilon_i), \quad \forall i \\ & \epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \end{aligned}$$

Recall that in the last lecture we set $\|\beta\|^2 = \sum_1^p \beta_j^2 = 1$, in which case β is the unit normal vector.

- We can set $M = 1/\|\beta\|$ in the preceding optimization

Reason: If (β_0, β) satisfies the preceding equations, then any scaled version of it satisfies, and in particular, we can set $M = 1/\|\beta\|$, instead of $\|\beta\| = 1$.

SVC: Convex Optimization

Next, optimizing $\max(1/\|\beta\|)$ is equivalent to $\min(\|\beta\|^2/2)$

$$\min_{\beta_j, \epsilon_j} \frac{\|\beta\|^2}{2}$$

subject to $y_i(\langle \beta, x \rangle + \beta_0) \geq (1 - \epsilon_i), \quad \forall i$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C$$

Constrained Optimization: Crash Course

Primal problem

$$\min_{\boldsymbol{x} \in R^n} f(\boldsymbol{x}) \quad (9)$$

subject to $f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \dots, m$
 $h_i(\boldsymbol{x}) = 0, \quad i = 1, \dots, p$

Lagrangian

- ▶ Incorporate the constraints into one equation

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^p \mu_i h_i(\boldsymbol{x}) \quad (10)$$

Lagrange multiplier vectors or dual vectors

$$\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m), \quad \boldsymbol{\mu} = (\mu_1, \dots, \mu_m)$$

Constrained Optimization

Lagrange dual function

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \inf_{\mathbf{x} \in R^n} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad (11)$$

$$= \inf_{\mathbf{x} \in R^n} \left(f(\mathbf{x}) + \sum_{i=1}^m \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^p \mu_i h_i(\mathbf{x}) \right)$$

- ▶ Let f^* = optimal value of the primal problem
- ▶ **Lower bound:** for any $\boldsymbol{\lambda} \geq 0$

$$f^* \geq g(\boldsymbol{\lambda}, \boldsymbol{\mu})$$

since, for any feasible point $\tilde{\mathbf{x}}$ ($\tilde{\mathbf{x}}$ satisfies the primal problem)

$$\sum_{i=1}^m \lambda_i f_i(\tilde{\mathbf{x}}) + \sum_{i=1}^p \mu_i h_i(\tilde{\mathbf{x}}) = \sum_{i=1}^m \lambda_i f_i(\tilde{\mathbf{x}}) \leq 0$$

since all $\lambda_i \geq 0$.

Constrained Optimization

Lagrange dual problem

$$\begin{aligned} & \max g(\lambda, \mu) \\ \text{subject to } & \lambda \geq 0 \end{aligned} \tag{12}$$

- ▶ Let $g^* = g(\lambda^*, \mu^*)$ be the optimal value of the dual problem
 λ^*, μ^* are the optimal Lagrange multipliers
- ▶ Clearly

$$f^* \geq g^*$$

- ▶ When are primal and dual problems equal, i.e., $f^* = g^*$?
- ▶ **Duality gap**

$$f(x) - g(\lambda, \mu)$$

- ▶ Can be used for computation as a **stopping criterion**.

Constrained Optimization

Complementary slackness

- If primal and dual problem values are equal, $f^* = g^*$, and attained at values $f^* = f(\mathbf{x}^*)$ and $g^* = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, then

$$\lambda_i^* f_i(\mathbf{x}^*) = 0, \quad i = 1, \dots, m$$

► Proof

$$\begin{aligned} f^* &= f(\mathbf{x}^*) = g^* = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \\ &= \inf_{\mathbf{x} \in R^n} \left(f(\mathbf{x}) + \sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}) + \sum_{i=1}^p \mu_i h_i^*(\mathbf{x}) \right) \\ &\leq f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}^*) + \sum_{i=1}^p \mu_i h_i^*(\mathbf{x}^*) \\ &\leq f(\mathbf{x}^*) \end{aligned}$$

since $\lambda_i^* \geq 0$, $f_i(\mathbf{x}^*) \leq 0$ and $h_i^*(\mathbf{x}^*) = 0$.

Hence, $\sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}) = 0$, and therefore $\lambda_i^* f_i(\mathbf{x}^*) = 0$.

Karush-Kuhn-Tucker (KKT) Conditions

Necessary conditions for nonconvex problems

- ▶ Let \boldsymbol{x}^* and $\boldsymbol{\lambda}^*, \mu^*$ be any primal and dual points with zero duality gap.
- ▶ Then, the following KKT conditions hold

$$f_i(\boldsymbol{x}^*) \leq 0, \quad (\text{primal feasibility})$$

$$h_i(\boldsymbol{x}^*) = 0, \quad (\text{primal feasibility})$$

$$\lambda_i^* \geq 0, \quad (\text{dual feasibility})$$

$$\lambda_i^* f_i(\boldsymbol{x}^*) = 0, \quad (\text{complement. slackness})$$

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(\boldsymbol{x}^*) + \sum_{i=1}^p \mu_i^* \nabla h_i(\boldsymbol{x}^*) = 0, \quad (\text{stationarity})$$

Recall $\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \dots, \frac{\partial f(\boldsymbol{x})}{\partial x_n} \right)$ is the gradient

- ▶ 4th condition = complementary slackness \Rightarrow
 $\lambda_i = 0, f_i(\boldsymbol{x}^*) < 0$: \boldsymbol{x}^* is **inside**, or
 $\lambda_i > 0, f_i(\boldsymbol{x}^*) = 0$: \boldsymbol{x}^* on the **boundary**

Karush-Kuhn-Tucker (KKT) Conditions

Sufficient conditions for convex problems

- ▶ Assume f, f_i are convex, and h_i are affine (linear)
- ▶ Let $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ be any points that satisfy KKT conditions
- ▶ Then, $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ are primal and dual optimal with zero duality gap
- ▶ **Proof:**
$$\begin{aligned} g(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}) &= L(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}) \\ &= f(\tilde{\boldsymbol{x}}) + \sum_{i=1}^m \tilde{\lambda}_i f_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^p \tilde{\mu}_i h_i(\tilde{\boldsymbol{x}}) \\ &= f(\tilde{\boldsymbol{x}}) \end{aligned}$$

We used in the second equality that $L(\boldsymbol{x}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ is convex in \boldsymbol{x} since $\tilde{\lambda}_i \geq 0$ and h_i are affine, implying, by the last KKT condition, that $\tilde{\boldsymbol{x}}$ minimizes $L(\boldsymbol{x}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$.

In the last line, we used $h_i(\boldsymbol{x}^*) = 0$ and $\lambda_i^* f_i(\boldsymbol{x}^*) = 0$ (complementary slackness)

SVC Continued: Dual Problem

Incorporate the SVC constraints into the Lagrange function

$$L(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{\|\boldsymbol{\beta}\|^2}{2} + c \sum_{i=1}^n \epsilon_i - \sum_{i=1}^n \alpha_i [y_i(\langle \boldsymbol{\beta}, \mathbf{x}_i \rangle + \beta_0) - (1 - \epsilon_i)] - \sum_{i=1}^n \mu_i \epsilon_i$$

Next, we define the **dual function**

$$g(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \inf_{\boldsymbol{\beta}, \beta_0, \epsilon_i} L(\boldsymbol{\beta}, \beta_0, c, \boldsymbol{\alpha}, \boldsymbol{\mu})$$

Then, since $L(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}, \boldsymbol{\mu})$ is quadratic in β_i , we can explicitly compute its derivatives with respect to β_i and ϵ_i

$$\boldsymbol{\beta} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \quad (\partial/\partial \boldsymbol{\beta})$$

$$0 = \sum_{i=1}^n \alpha_i y_i \quad (\partial/\partial \beta_0)$$

$$\alpha_i = c - \mu_i \quad (\partial/\partial \epsilon_i)$$

Plugging the preceding derivatives into L , yields an explicit formula for g , after some algebra.

SVC: Dual Problem

Now, dual problem is to maximize $g(\alpha, \mu)$, i.e.,

$$\max_{\alpha_i} \left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

subject to $0 \leq \alpha_i \leq c, \sum \alpha_i y_i = 0$

KKT conditions to the rescue

$$\alpha_i [y_i (\langle \boldsymbol{\beta}, \mathbf{x}_i \rangle + \beta_0) - (1 - \epsilon_i)] = 0 \quad \text{slackness}$$

$$\mu_i \epsilon_i = 0$$

$$y_i (\langle \boldsymbol{\beta}, \mathbf{x}_i \rangle + \beta_0) - (1 - \epsilon_i) \geq 0$$

plus the preceding derivative equations.

From **slackness**, $\alpha_i > 0$ only for support vectors, when

$$y_i (\langle \boldsymbol{\beta}, \mathbf{x}_i \rangle + \beta_0) = (1 - \epsilon_i)$$

SVC Simplification

From the derivative condition (2 slides ago)

$$\begin{aligned}\hat{\beta} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \\ &= \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{x}_i\end{aligned}$$

And, thus, the classification hyperplane is easy to compute

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

Support Vector Machines: Hilbert spaces

Hilbert spaces: Generalized linear spaces

- ▶ Let $\phi(\mathbf{x}_i)$ be a transformation of feature variables such that
- ▶ **Kernel -** $K(\mathbf{x}_i, \mathbf{x}_j)$ is positive definite
- ▶ **Generalized inner product:**

$$\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = K(\mathbf{x}_i, \mathbf{x}_j)$$

- ▶ Hence, with arbitrary nonlinear transformation, Kernel, we can repeat the preceding optimization, and derive an SVC.

The resulting classifier is known as **SVM**, with the decision function taking the following nonlinear form in general

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, \mathbf{x}_i)$$

Support Vector Machines: Kernels

Kernels:

- *Linear*

$$K(x_i, x_j) = \langle x_j, x_i \rangle$$

- *Polynomial* - for positive integer d

$$K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^d$$

- *Radial* - for $\gamma > 0$ - (or Gaussian $\gamma = 1/(2\sigma^2)$)

$$K(x_i, x_j) = \exp \left(-\gamma \sum_{k=1}^p (x_{ik} - x_{jk})^2 \right)$$

Deriving The Quadratic Kernel

Kernels:

- ▶ Consider data with two predictors: $x_i = (x_{i1}, x_{i2})$
- ▶ *Quadratic Kernel* ($d = 2$): $K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^2$
- ▶ We can obtain the preceding kernel by considering feature map

$$\phi(x_i) = (1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2)$$

- ▶ Then, the inner product

$$\begin{aligned}\langle \phi(x_i), \phi(x_j) \rangle &= 1 + 2x_{i1}x_{j1} + 2x_{i1}x_{j1} + x_{i1}^2x_{j1}^2 \\ &\quad + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2x_{j2}^2 \\ &= 1 + 2\langle x_i, x_j \rangle + \langle x_i, x_j \rangle^2 \\ &= (1 + \langle x_j, x_i \rangle)^2 = K(x_i, x_j)\end{aligned}$$

Finding Feature Maps Is Hard

In general, finding feature maps for a corresponding kernel is hard.

Example: Radial Kernel, $x \in \mathbb{R}$

- ▶ Consider

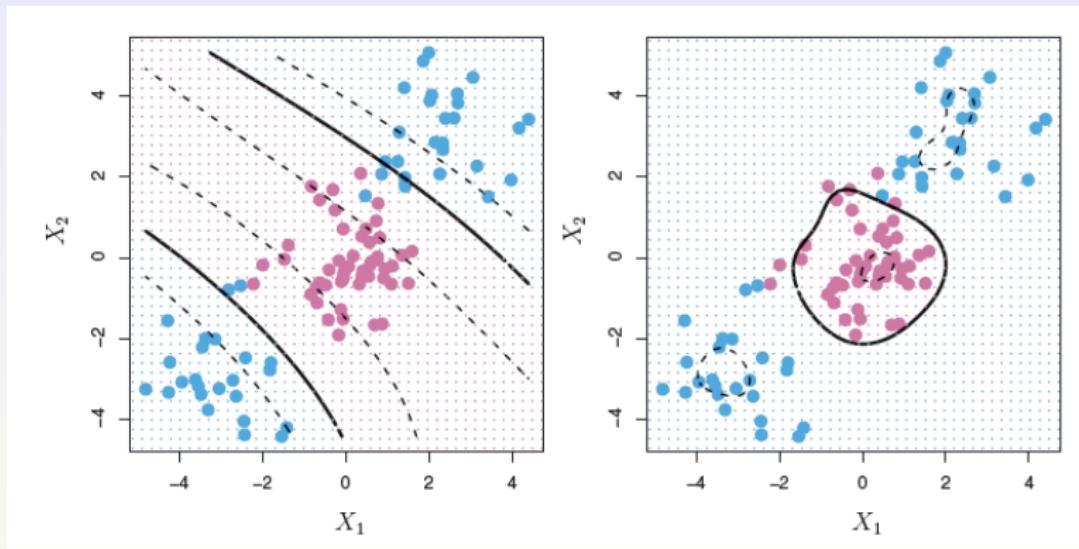
$$\phi(x) = \left(1, \frac{xe^{-x^2/2}}{\sqrt{1!}}, \frac{x^2e^{-x^2/2}}{\sqrt{2!}}, \dots, \frac{x^n e^{-x^2/2}}{\sqrt{n!}}, \dots\right)$$

- ▶ Then

$$\begin{aligned}\langle \phi(x_i), \phi(x_j) \rangle &= \sum_{n=0}^{\infty} \frac{x_i^n e^{-x_i^2/2}}{\sqrt{n!}} \frac{x_j^n e^{-x_j^2/2}}{\sqrt{n!}} \\ &= e^{-x_i^2/2 - x_j^2/2} \sum_{n=0}^{\infty} \frac{x_i^n x_j^n}{n!} = e^{-\|x_i - x_j\|^2/2} = K(x_i, x_j)\end{aligned}$$

Fortunately, we can use Kernels without knowing the feature maps.
We'll talk about this after the midterm.

Example



- ▶ Left: An SVM with a polynomial kernel of degree 3
- ▶ Right: An SVM with a radial kernel
- ▶ Either kernel is capable of capturing the decision boundary

SVM with More than Two Classes

One-Versus-One classification

- ▶ For $K > 2$ classes, consider $\binom{K}{2}$ pairs
- ▶ For example, we might compare the k th class, coded as +1, to the k' th class, coded as -1
- ▶ We **tally the number of times** that the test observation is assigned to each of the K classes
- ▶ The final classification is performed by assigning the test observation to the class to which it was **most frequently assigned** in these $\binom{K}{2}$ pairwise classifications

SVM with More than Two Classes

One-Versus-All classification

- ▶ We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes.
- ▶ Let $\beta_{0k}, \beta_{1k}, \dots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the k th class (coded as +1) to the others (coded as -1).
- ▶ Let x_0 denote a test observation
- ▶ We assign x^* to the class with maximum

$$\beta_{0k} + \beta_{1k}x_1^* + \cdots + \beta_{pk}x_p^*$$

Khan - Gene Expression Data

- ▶ Gene expression measurements for four cancer types of small round blue cell tumors.
- ▶ For each tissue sample, 2308 gene expression measurements are available.

```
> library(ISLR)
> names(Khan)
[1] "xtrain"   "xtest"    "ytrain"   "ytest"
> dim(Khan$xtrain)
[1] 63 2308
> dim(Khan$xtest)
[1] 20 2308
> length(Khan$ytrain)
[1] 63
> length(Khan$ytest)
[1] 20
```

Khan - Gene Expression Data

- ▶ The training and test sets consist of 63 and 20 observations respectively
- ▶ Belong to 4 cancer types

```
> table(Khan$ytrain)
 1   2   3   4
 8 23 12 20
> table(Khan$ytest)
1 2 3 4
3 6 6 5
```

Khan - SVM Classification

Perfect fit with linear kernels (!) - no training errors.

Is this a surprise?

```
> dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
> out=svm(y~, data=dat, kernel="linear",cost=10)
> summary(out)

Call:
svm(formula = y ~ ., data = dat, kernel = "linear",
     cost = 10)

Parameters:

  SVM-Type:  C-classification
  SVM-Kernel:  linear
      cost:  10
      gamma:  0.000433

Number of Support Vectors:  58
( 20 20 11 7 )

Number of Classes:  4

Levels:
  1 2 3 4

> table(out$fitted, dat$y)

      1   2   3   4
1    8   0   0   0
2    0  23   0   0
3    0   0  12   0
4    0   0   0  20
```

Khan - SVM Test Performance

Two testing errors

```
> dat.te=data.frame(x=Khan$xtest, y=as.factor(Khan$ytest))
> pred.te=predict(out, newdata=dat.te)
> table(pred.te, dat.te$y)

pred.te 1 2 3 4
  1 3 0 0 0
  2 0 6 2 0
  3 0 0 4 0
  4 0 0 0 5
```

Reading on Support Vector Machines

ISL: Chapter 9. In particular, read Section 9.6 on experiments in R, including: e1071 library, "Khan" gene expression data (and ROC curves)

ESL: Chapter 12

Paper: Vladimir N. Vapnik (1999), "An Overview of Statistical Learning"

Homework: Start working on the final project.

Optional reading: Optimization - book

Convex Optimization, Stephen Boyd and Lieven Vandenberghe,
Cambridge University Press, 2004

Free download:

http://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

Today's presentation can be found in Chapter 5.

EECS E6690: SL for Bio & Info

Lecture 9: On Unified Supervised Learning Theory, Neural Networks and Deep Learning

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.

Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

On Unified Supervised Learning Theory

Supervised learning: Given training data (\mathbf{x}, \mathbf{y}) , i.e.,

$$(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \rightarrow \boxed{\mathbf{f}(\mathbf{x})} \rightarrow (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$$

Problem:

- ▶ Don't know \mathbf{f}
- ▶ Find the "best" approximation $\hat{\mathbf{f}}$

What have we seen?

- ▶ Linear Ridge: $\hat{\mathbf{f}}(\mathbf{X}) = \beta_0 + \sum_{j=1}^p \mathbf{X}_j \beta_j$

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

- ▶ Lasso: same as above, just change the penalty to ℓ_1 norm:

$$\lambda \sum_{j=1}^p \beta_j^2 \rightarrow \lambda \sum_{j=1}^p |\beta_j|$$

On Unified Supervised Learning Theory

Basis expansion:

- ▶ Polynomial **Ridge** or **Lasso**: Same as before, but more general, polynomial, approximation function

$$\hat{f}(X) = \beta_0 + \sum_{j=1}^p \sum_{l=1}^k \beta_{jl} (X_j)^l$$

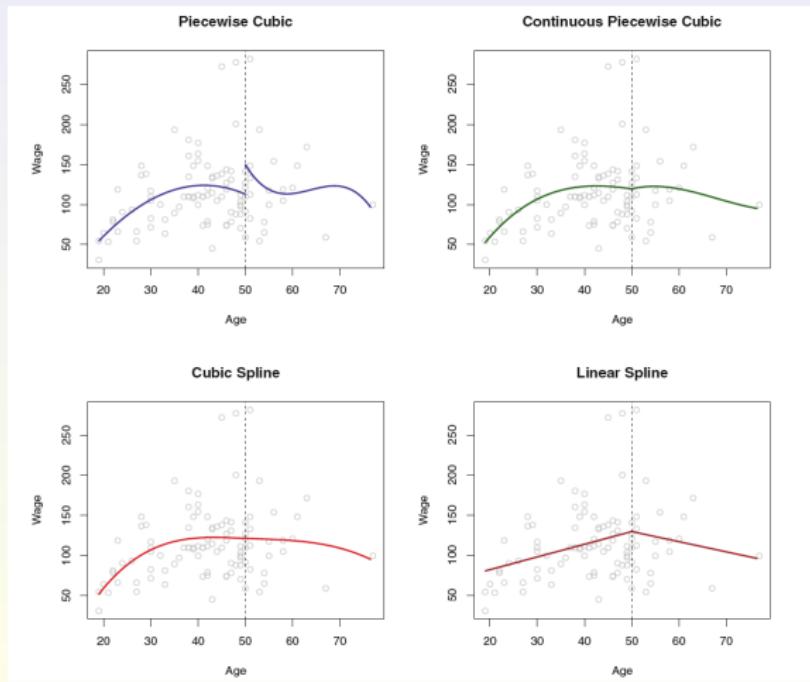
- ▶ **Splines**: Pick $\hat{f}(X)$ to be piecewise polynomial, e.g., piecewise linear. Problem: discontinuities.
- ▶ **Smoothing Splines**: Impose continuity and derivative constraints.

Example: Splines

Top left: cubic - no constraint; Top right: cubic and continuous

Bottom left: cubic - continuous, with continuous \hat{f}' and \hat{f}''

Bottom right: linear continuous



Hinge Loss Formulation of SVC

Recall

$$\min_{\beta_j, \epsilon_j} \frac{\|\boldsymbol{\beta}\|^2}{2}$$

subject to $y_i(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + \beta_0) \geq (1 - \epsilon_i), \quad \forall i$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C$$

Or, equivalently

$$\min_{\beta_i, \epsilon_i} \frac{\|\boldsymbol{\beta}\|^2}{2} + c \sum_{i=1}^n \epsilon_i$$

subject to $\epsilon_i = \max(0, 1 - y_i(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + \beta_0))$

Implying

$$\min_{\boldsymbol{\beta}, \beta_0} \frac{\|\boldsymbol{\beta}\|^2}{2} + c \sum_{i=1}^n \max(0, 1 - y_i(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + \beta_0))$$

$$\Leftrightarrow \min_{\boldsymbol{\beta}, \beta_0} \sum_{i=1}^n \max(0, 1 - y_i(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + \beta_0)) + \lambda \|\boldsymbol{\beta}\|^2$$

On Unified Supervised Learning Theory

Classification: let $y_i \in \{-1, 1\}$

- ▶ **Support Vector Classifier:** can be obtained by solving

$$\min_{\beta_0, \beta} \sum_{i=1}^n \max [0, 1 - y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p)] + \lambda \sum_{j=1}^p \beta_j^2$$

$\max [0, 1 - y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p)]$ is called **hinge loss**

For general **SVM** use a **kernel generalization** of the hyperplane

- ▶ **Logistic regression with ℓ_2 penalty:** replace the hinge loss in the preceding expression with

$$\{y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p) - \ln(1 + e^{\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p})\}$$

On Unified Supervised Learning Theory

In general, all the preceding learning problems can be written as

$$\hat{f}^* = \arg \min_{\hat{f} \in \mathcal{H}} \sum_{i=1}^n L(y_i, \hat{f}(x_i)) + \lambda \Omega(\hat{f}) \quad (1)$$

where L is a general loss function, and $\lambda \Omega(\hat{f})$ is the penalty, and \mathcal{H} is the space of approximation functions that we are considering.

- ▶ Example: linear functions $\hat{f}(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$

- ▶ \mathcal{H} : set off all p -dimensional linear functions
- ▶ Penalty - square of the ℓ_2 norm

$$\Omega(\hat{f}) = \langle \hat{f}, \hat{f} \rangle = \|f\|^2 = \sum_{j=1}^p \beta_j^2$$

- ▶ Loss: $L(y_i, \hat{f}(x_i)) = (y_i - \hat{f}(x_i))^2$

Q: What is the most general that $\hat{f}, \mathcal{H}, L, \Omega$, can be such that the problem has nice analytical and computational properties?

RKHS: Reproducing Kernel Hilbert Spaces

We have already seen it with SVM: here some more details¹

- ▶ Hilbert space $\{\mathcal{H}\}$: natural generalization of Euclidian spaces.
- ▶ It has an **inner product**: $\langle f, g \rangle$, for any $f, g \in \mathcal{H}$, which allows computing angles between the elements of $\{\mathcal{H}\}$.
In particular, $\langle f, g \rangle = 0$, then f, g are **orthogonal**.
- ▶ **Norm/distance**: Norm $\|f\| = \sqrt{\langle f, f \rangle}$, and
distance $d(f, g) = \|f - g\| = \sqrt{\langle f - g, f - g \rangle}, \quad f, g \in \mathcal{H}$
- ▶ Euclidian example: if $x, y \in \mathbb{R}^d$, then $\langle x, y \rangle = x_1y_1 + \cdots + x_py_p$

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}, \quad d(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2}$$

- ▶ Example - matrices: $\langle A, B \rangle = \text{trace}(A^\top B)$
- ▶ Example - random variable: $\langle X, Y \rangle = \text{cov}(X, Y)$

RKHS: Reproducing Kernel Hilbert Spaces

RKHS - \mathcal{H}_k : Subset of Hilbert spaces with really nice properties

- ▶ Kernel is a positive definite function $k(x, y)$
 $(\sum \alpha_i \alpha_j k(x_i, x_j) \geq 0)$
- ▶ **Reproducing property:** if $f \in \mathcal{H}_k$, then

$$\langle f, k(\cdot, x) \rangle = f(x)$$

- ▶ Each kernel, $k(x, y)$, defines uniquely the Hilbert space, \mathcal{H}_k Hence, in this space, \mathcal{H}_k , all we need to know is the kernel!
- ▶ Functions in this space $f(x) \in \mathcal{H}_k$ can be written as

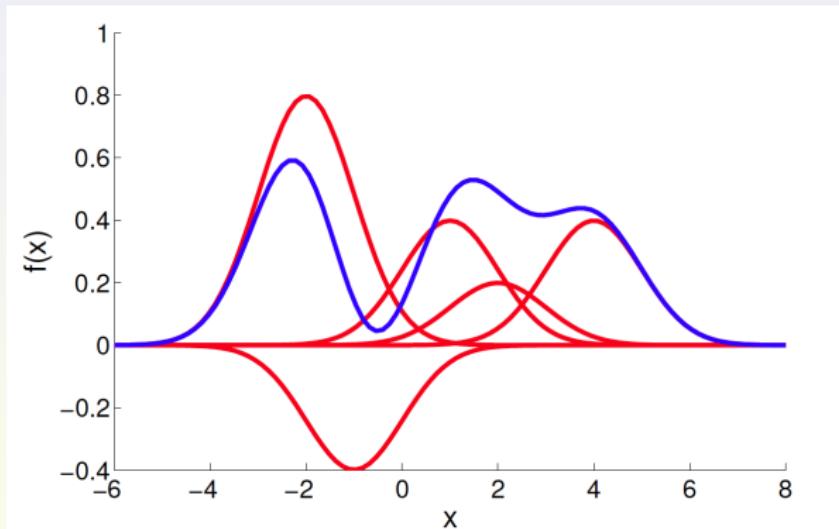
$$f(x) = \sum_i \beta_i k(x, x_i)$$

Example: Gaussian (Radial) Kernel

Gaussian kernel in 1D: $k(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}$

Typical function: sum of weighted Gaussian "blobs" - blue line below

$$f(x) = \sum_i \beta_i k(x, x_i) = \sum_i \beta_i e^{-\frac{(x-x_i)^2}{2\sigma^2}}$$



\mathcal{H}_k is a linear space, but $f(x)$ is very much nonlinear.

Generalized Learning in RKHS

Here we optimize over all approximation function in \mathcal{H}_k

$$\hat{f}^* = \arg \min_{\hat{f} \in \mathcal{H}_k} \sum_{i=1}^n L(y_i, \hat{f}(x_i)) + \lambda \Omega(\|\hat{f}\|_{\mathcal{H}_k}^2) \quad (2)$$

Representer Theorem For any loss function L and any strictly increasing function $\Omega : \mathbb{R} \rightarrow \mathbb{R}$, an optimal solution, \hat{f}^* , of Equation (2) has a form

$$\hat{f}^*(x) = \sum_{i=1}^n \beta_i k(x, x_i)$$

\hat{f}^* has a finite representation (!) even though \mathcal{H}_k can be (is) infinite.
We can put much of the supervised learning under the same umbrella.
Drawback: Theoretical framework works only with ℓ_2 norm, $\|\hat{f}\|_{\mathcal{H}_k}$ - doesn't work for Lasso.

Representer Theorem: Proof

Let's drop "hat" from \hat{f} and let f_s be functions spanned by $k(x, x_i)$

$$f_s(x) = \sum_{i=1}^n \beta_i k(x, x_i)$$

Then, any function $f \in \mathcal{H}_k$ can be decomposed as

$$f(x) = f_s(x) + f_\perp(x) \quad (3)$$

where $f_\perp(x)$ is perpendicular to f_s . Hence, (Pythagoras theorem for \mathcal{H}_k)

$$\|f\|_{\mathcal{H}_k}^2 = \|f_s\|_{\mathcal{H}_k}^2 + \|f_\perp\|_{\mathcal{H}_k}^2 \geq \|f_s\|_{\mathcal{H}_k}^2 \quad (4)$$

Next, by [kernel reproducing property](#) and orthogonality

$$f(x_i) = \langle f, k(\cdot, x_i) \rangle = \langle f_s, k(\cdot, x_i) \rangle = f_s(x) \quad (5)$$

Finally, Equation (4) and Equation (5) imply that an optimizer of Equation (2) must be in the form of $f_s(x)$ from Equation (3) since Ω is increasing and

$$L(y_i, f(x_i)) = L(y_i, f_s(x_i)), \quad \Omega(\|\hat{f}\|_{\mathcal{H}_k}^2) \geq \Omega(\|f_s\|_{\mathcal{H}_k}^2)$$

Generalized Ridge

For quadratic loss function

$$\hat{f}^* = \arg \min_{\hat{f} \in \mathcal{H}_k} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2 + \lambda \|\hat{f}\|_{\mathcal{H}_k}^2 \quad (6)$$

The coefficients, β_i^* , of the optimizer

$$\hat{f}^*(x) = \sum_{i=1}^n \beta_i^* k(x, x_i)$$

are explicitly given by

$$\boldsymbol{\beta}^* = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$$

where \mathbf{K} is a square matrix with elements $K_{ij} = k(x_i, x_j)$, \mathbf{I} is an identity matrix, and \mathbf{y} is the column vector $(y_1, \dots, y_n)^\top$.

Generalized Ridge: Proof

From the Representer Theorem we know that the minimizer has to be of the form

$$f(x) = \sum_{i=1}^n \beta_i k(x, x_i)$$

and their quadratic norm is given by

$$\|f\|_{\mathcal{H}_k}^2 = \langle f, f \rangle = f(x) = \sum_{i=1}^n \sum_{j=1}^n \beta_i \beta_j k(x_j, x_i) = \boldsymbol{\beta}^\top \mathbf{K} \boldsymbol{\beta}$$

since, by reproducing property, $\langle k(\cdot, x_j), k(\cdot, x_i) \rangle = k(x_j, x_i)$. Also, by reproducing property,

$$f(x_i) = \langle f(\cdot), k(\cdot, x_i) \rangle = \sum_{j=1}^n \beta_j k(x_j, x_i)$$

By replacing the preceding 2 equations in the optimization function in Equation (6)

$$\boldsymbol{\beta}^* = \arg \min_{\boldsymbol{\beta}} \sum_{i=1}^n (y_i - \sum_{j=1}^n \beta_j k(x_j, x_i))^2 + \lambda \boldsymbol{\beta}^\top \mathbf{K} \boldsymbol{\beta}$$

which by taking the derivative w.r.t. $\boldsymbol{\beta}$ and setting it to 0 yields

$$\boldsymbol{\beta} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$$

Neural Networks Verus Kernel Methods

- ▶ Similarity: both parametric methods
- ▶ Difference: fixed basis for kernels, while NN is discovering basis/features

Neural networks:

- ▶ Rich \mathcal{H} : Provides a versatile parametric class of functions
 - ▶ Universal function approximation
 - ▶ Difficult to train: non-convex optimization
 - ▶ Often accurate predictions, but difficult to interpret
- ▶ Automatic feature/basis extraction
 - ▶ Traditional Feature Engineering approach: expert constructs feature mapping $\phi : \mathcal{X} \rightarrow \Phi$. Then, apply machine learning to find a linear predictor on $\phi(\mathbf{x})$.
 - ▶ “Deep learning” approach: neurons in hidden layers can be thought of as features that are being learned automatically from data
 - ▶ Shallow neurons corresponds to low level features, while deep neurons correspond to high level features

General Parametric Supervised Learning

- ▶ \mathcal{H} is a parametric class of functions
 $f(w, x), w \in \mathcal{W}, w = (w_1, \dots, w_k)$
 - ▶ Examples: Generalized Ridge, or neural networks
- ▶ Finding $f \in \mathcal{H}$ is equivalent to finding $w \in \mathcal{W}$

Hence, our general supervised learning problem can be formulated in terms of w as (say, $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}^m, \ell : \mathbb{R}^{p+m} \rightarrow \mathbb{R}$ - loss function)

$$\hat{w} = \arg \min_{w \in \mathcal{W}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(w, x_i)) + \lambda R(w) \quad (7)$$

How do we solve the preceding problem?

- ▶ When the problem is convex and we are lucky, we can find an explicit \hat{w} by solving (e.g., generalized (Kernel) Ridge regression)

$$\frac{\partial}{\partial w_i} \left(\frac{1}{n} \sum_{i=1}^n \ell(y_i, f(w, x_i)) + \lambda R(w) \right) = 0, \quad i = 1, \dots, k.$$

Parametric Supervised Learning: Numerical Optimization

Let us denote the objective function

$$F(w) := \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(w, x_i)) + \lambda R(w)$$

In general, we use the following numerical algorithm:

1. Initialization: Pick an initial value w_0 , possibly random
2. Iteration: Keep updating w in small steps Δw

$$w_{n+1} = w_n + \Delta w,$$

such that $F(w_{n+1}) < F(w_n)$.

Stoping criteria: $F(w_n) - F(w_{n+1}) < \epsilon$.

For the preceding procedure to find a local minimum

- We need to **find a direction where F has a maximum decrease/steepest descent**
- Avoid getting stuck on a flat surfaces, say flat saddle point

If F is convex, we can find a global minimum.

Recall Multi Calc: Gradient and Directional Derivatives

- ▶ Let $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}, x = (x_1, \dots, x_n)$
- ▶ **Gradient** is a vector of partial derivatives (assuming they exist)

$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

- ▶ Let $u = (u_1, \dots, u_n)$ be a unit vector ($\|u\|_2^2 = \sum u_i^2 = 1$)
- ▶ **Directional derivative** of $f(x)$ in direction u is

$$D_u f(x) := \frac{d}{dt} f(x+ut) = \sum_{i=1}^n \frac{\partial f}{\partial x_i} u_i = \nabla f(x) \cdot u = \|\nabla f(x)\|_2 \cos \theta,$$

where $y \cdot z = \langle y, z \rangle$ is the dot product and θ is the angle between $\nabla f(x)$ and u .

- ▶ Hence, $-\|\nabla f(x)\|_2 \leq D_u f(x) \leq \|\nabla f(x)\|_2$, i.e.,
 - $\nabla f(x)$: direction of steepest ascent/max increase of $f(x)$
 - $-\nabla f(x)$: direction of steepest descent/max decrease of $f(x)$
 - $\nabla f(x)$ is perpendicular to level curves $f(x) = c$ (prove this)

Path of Gradient Descent

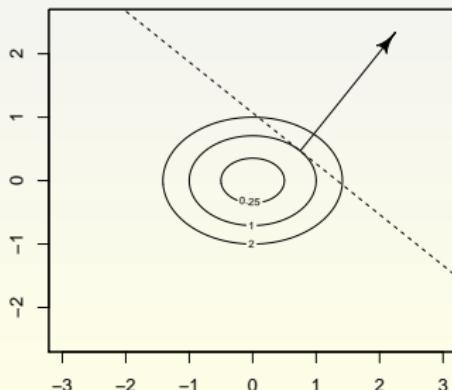
- ▶ $\mathbf{x}(t)$ - path of steepest gradient descent given initial condition $\mathbf{x}(0)$ is given by an ODE

$$\frac{d\mathbf{x}(t)}{dt} = -\eta \nabla f(\mathbf{x}(t)), \quad (8)$$

where η is the rate/speed of descent, a.k.a. learning rate. Note that $x'(t)$ is tangent to the curve $x(t)$, and thus parallel to $\nabla f(\mathbf{x}(t))$.

- ▶ Example: $f(\mathbf{x}) = ax_1^2 + bx_2^2, a, b > 0, \Rightarrow \nabla f(\mathbf{x}) = (2ax_1, 2bx_2) \Rightarrow x'_1(t) = -2\eta a x_1(t), x'_2(t) = -2\eta b x_2(t)$

$$x_1(t) = x_1(0)e^{-2\eta at}, \quad x_2(t) = x_2(0)e^{-2\eta bt}$$



Gradient Descent Algorithm

GD Algorithm is a discrete linear approximation to Equation (8)

$$\frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} \approx \frac{d\mathbf{x}(t)}{dt} = -\eta \nabla f(\mathbf{x}(t))$$

or equivalently (with a small abuse of notation)

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)})$$

- ▶ Hence, after initialization at $\mathbf{x}^{(0)}$, the GD Algorithm follows the preceding iteration to a local minimum
- ▶ Stopping criterion (could be): $|f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})| < \epsilon$

Adaptive GD (AdaGrad): modifies the learning rate η_t in each iteration t

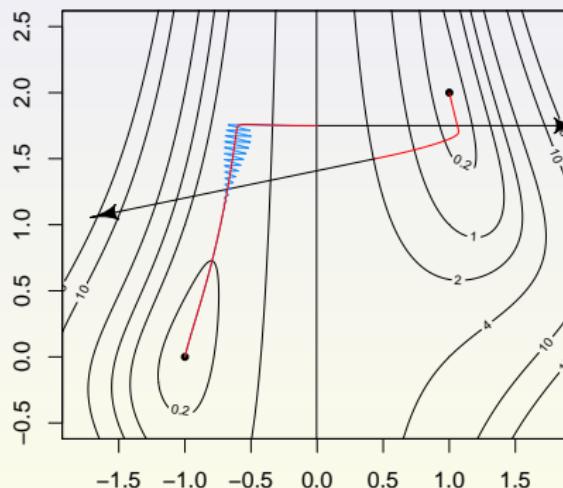
More Interesting Landscape

- ▶ $f(x) = (x_1^2 - 1)^2 + (x_1^2 x_2 - x_1 - 1)^2$

- ▶ Gradient

$$\nabla f(x) = \begin{bmatrix} 4x_1(x_1^2 - 1) + 2(2x_1 x_2 - 1)(x_1^2 x_2 - x_1 - 1) \\ 2x_1^2(x_1^2 x_2 - x_1 - 1) \end{bmatrix}$$

- ▶ Oscillations in "narrow valleys"

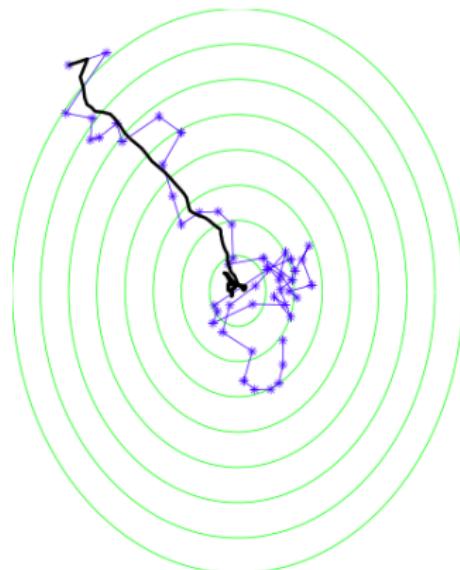
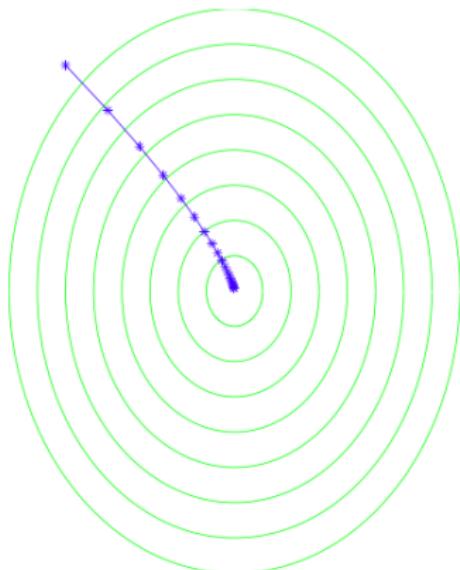


Motivation for momentum: remembers/averages previous Δx

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)}) + \mu_t (\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)})$$

Stochastic Gradient Descent

- ▶ Dates back to Robbins and Monroe (1951).
- ▶ Stochastic gradient is an **unbiased estimator** of the gradient
- ▶ Stochastic versus regular gradient descent



Stochastic Gradient Descent

- ▶ Stochastic approximation of gradient descent
- ▶ Function (typically encountered in learning)

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- ▶ Computationally expensive gradient for large n
- ▶ Approximation: pick a random subset $\mathcal{S} \in [1, n]$

$$\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla f_i(x)$$

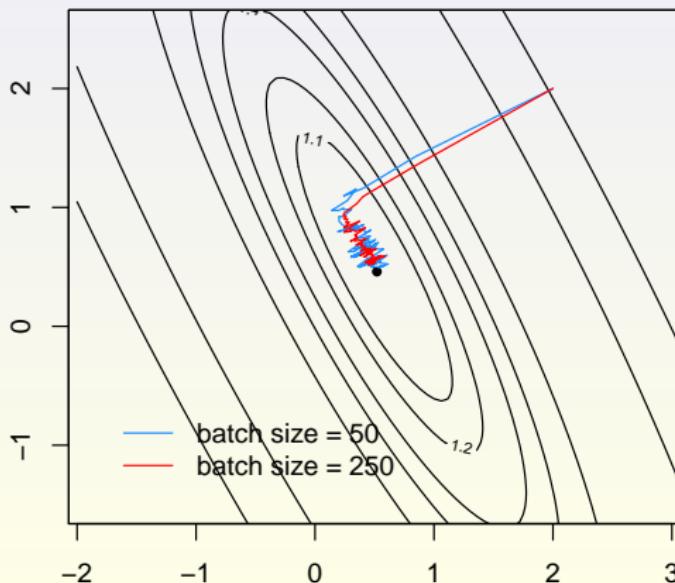
- ▶ \mathcal{S} - called batch/mini-batch
- ▶ Example
 - ▶ n scalar data points x_1, x_2, \dots, x_n
 - ▶ objective

$$\min_c \frac{1}{n} \sum_{i=1}^n (x_i - c)^2$$

SGD Example: Linear Regression

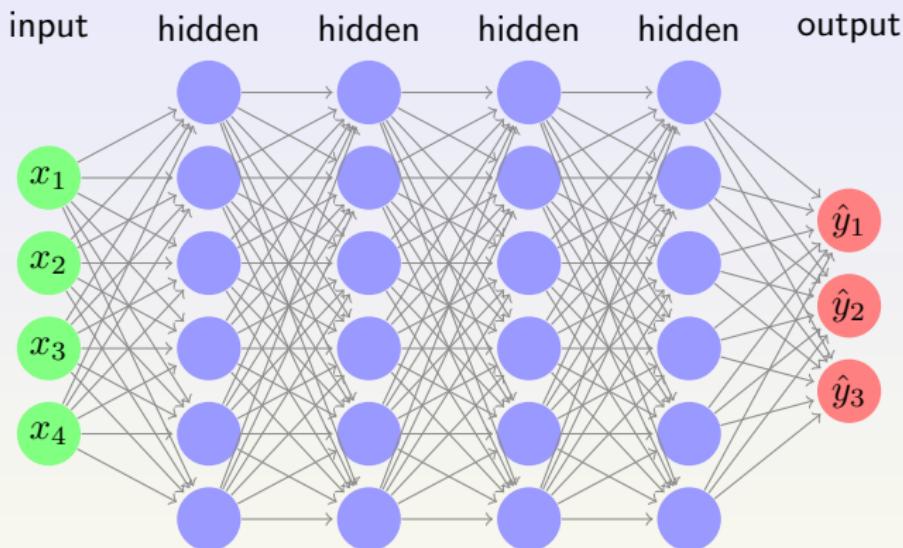
- ▶ Data: $(x_i, y_i)_{i=1}^n, n = 10^5$
- ▶ Loss function: $L(\beta, x, y) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$
- ▶ Stochastic gradient

$$\nabla_{\beta} \hat{L}(\beta, x, y) = \frac{2}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left[\begin{array}{l} (\beta_0 + \beta_1 x_i - y_i) \\ x_i(\beta_0 + \beta_1 x_i - y_i) \end{array} \right]$$



Deep Neural Networks, a.k.a. Multilayer Perceptrons

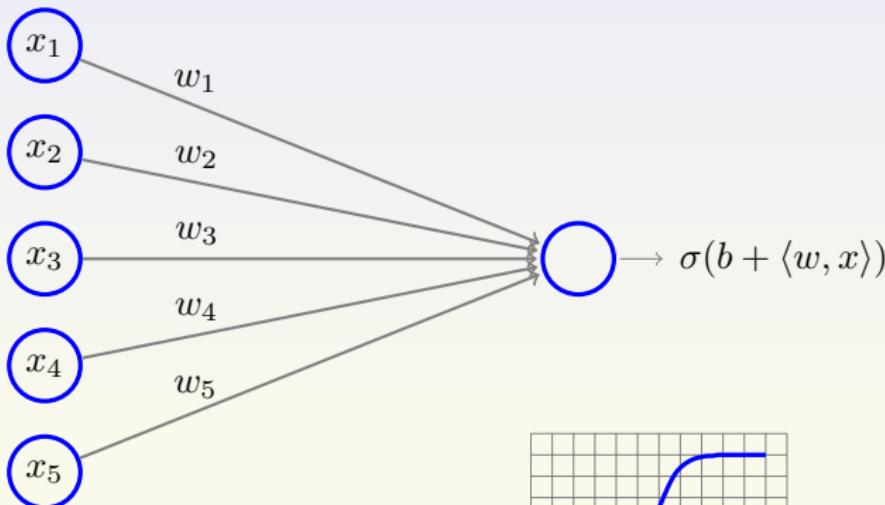
- ▶ Feed-forward network



- ▶ Designed to mimic the function of neurons
- ▶ Blue nodes: activation functions/neurons
- ▶ Depth = lengths of a longest path
- ▶ Deep network: depth ≥ 3
- ▶ Very successful in solving practical problems

A Single Artificial Neuron

- ▶ A single neuron function: $\mathbf{x} \mapsto \sigma(b + \langle \mathbf{w}, \mathbf{x} \rangle)$, where $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is called the activation function of the neuron. Inner/dot product:
$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum x_i w_i.$$
- ▶ More compact notation $\langle \tilde{\mathbf{w}}, \tilde{\mathbf{x}} \rangle$, where $\tilde{\mathbf{x}} = (1, \mathbf{x})$, $\tilde{\mathbf{w}} = (b, \mathbf{w})$



- ▶ E.g., σ is a sigmoidal function

Common Activation Functions/Perceptrons

- ▶ step function

$$\sigma(x) = 1_{\{x>0\}} \quad \sigma'(x) = 0, x \neq 0$$

- ▶ logistic

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \sigma'(x) = \sigma(x)(1 - \sigma(x))$$

- ▶ rectified linear unit (ReLU)

$$\sigma(x) = \max\{x, 0\} \quad \sigma'(x) = 1_{\{x>0\}}, x \neq 0$$

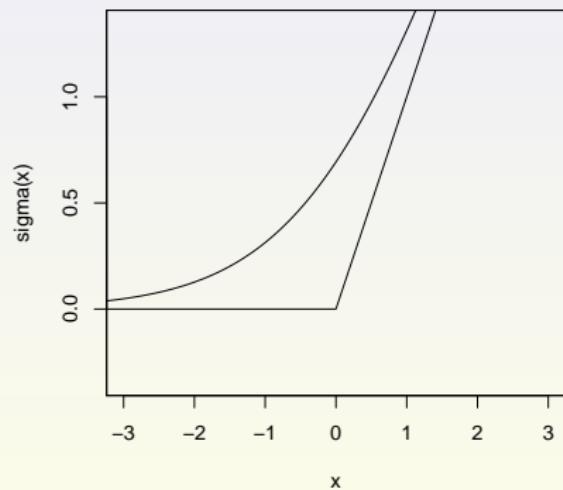
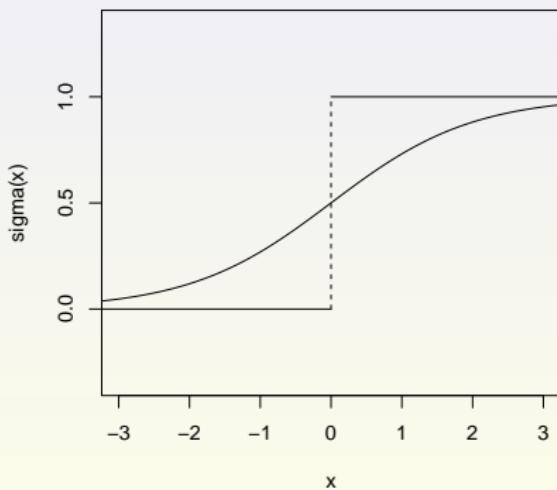
- ▶ soft-plus

$$\sigma(x) = \log(1 + e^x) \quad \sigma'(x) = \frac{1}{1 + e^{-x}}$$

Comparing Activation Functions

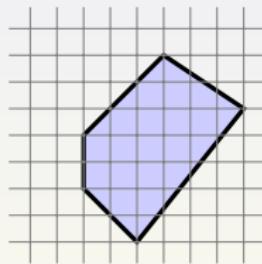
Logistic and soft-plus have continuous derivatives in comparison to step function and ReLU

- ▶ Positive derivative avoids vanishing gradient



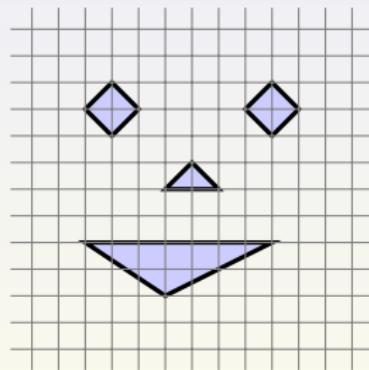
Example

- ▶ Single neuron is a binary half-space classifier: $\text{sign}(w \cdot x + b)$
- ▶ 2 layer networks can express intersection of halfspaces

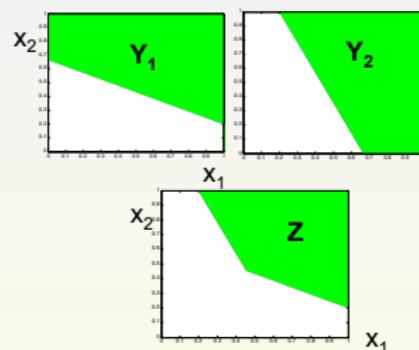
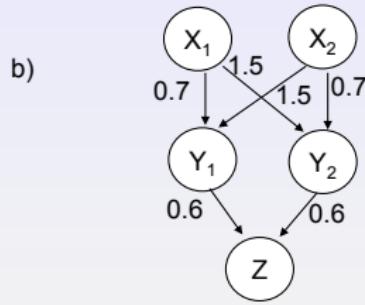
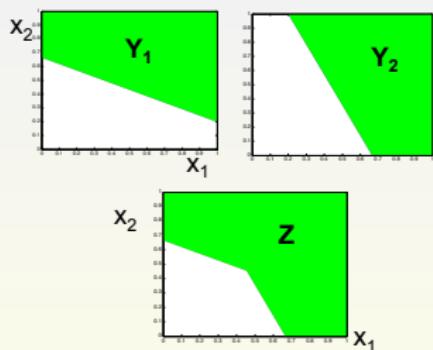
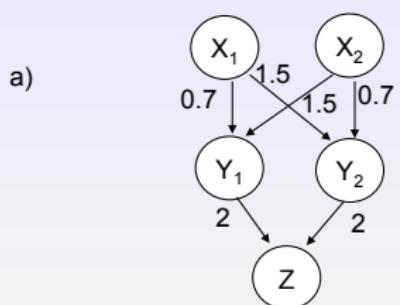


Example

- ▶ 3 layer networks can express unions of intersection of halfspaces

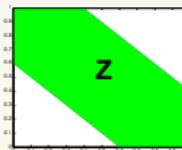
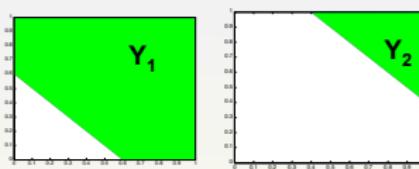
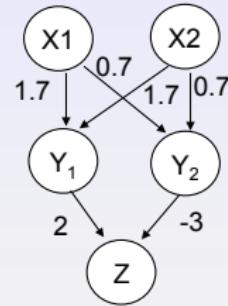
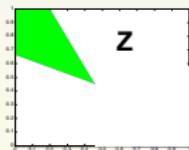
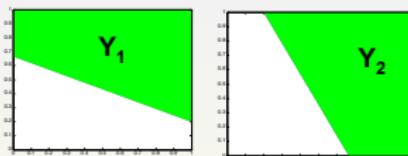
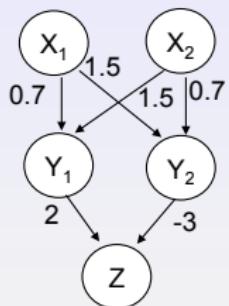


Examples: Step function activation - $f(x) = \mathbf{1}(x)$



Green - $Y = 1, Z = 1$

Examples: Step function activation - $f(x) = \mathbf{1}(x)$



Green - $Y = 1, Z = 1$

How to train neural network?

- ▶ Neural nets: excellent hypothesis class, but difficult to train
- ▶ Main technique: Stochastic Gradient Descent (SGD)
- ▶ Not convex, no guarantees, can take a long time, but:
 - ▶ Often still works fine, finds a good solution
 - ▶ With some luck:)

Stochastic Gradient Descent (SGD) for Neural Networks

Common Training Ideas:

- ▶ Random initialization: rule of thumb, $w[u \rightarrow v] \sim U[-c, c]$ where $c = \sqrt{3/|\{(u', v) \in E\}|}$ (or small Gaussian instead of $U[-c, c]$)
- ▶ Update step with Nesterov's momentum: Initialize $\theta = 0$ and:

$$\begin{aligned}\theta_{t+1} &= \mu_t \theta_t - \eta_t \tilde{\nabla} L(w_t + \mu_t \theta_t) \\ w_{t+1} &= w_t + \theta_{t+1}\end{aligned}$$

where:

μ_t is momentum parameter (e.g. $\mu_t = 0.9$ for all t)

η_t is learning rate (e.g. $\eta_t = 0.01$ for all t)

$\tilde{\nabla} L$ is an estimate of the gradient of L based on a small set of random examples (often called a “minibatch”)

- ▶ Efficient gradient calculation: **Backpropagation**

Backpropagation: Efficient Implementation of Chain Rule

Chain rule

$$(f(g(x)))' = f'(g(x))g'(x)$$

$$(f(f(x)))' = f'(f(x))f'(x)$$

$$(f(f(f(x))))' = f'(f(f(x)))f'(f(x))f'(x)$$

Chain rule in vector form

$$\nabla_{\mathbf{x}} \mathbf{z} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^\top \nabla_{\mathbf{y}} \mathbf{z}$$

Back-Propagation

- ▶ The **back-propagation** algorithm is an efficient way to calculate $\nabla \ell(h_w(x), y)$ using the **chain rule**
- ▶ Let $\mathbf{f}(\mathbf{w}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $\mathbf{f} = (f_1, \dots, f_m)$, $f_i(\mathbf{w}) : \mathbb{R}^n \rightarrow \mathbb{R}$
- ▶ $J_{\mathbf{w}}(\mathbf{f})$ - **Jacobian** of $\mathbf{f}(\mathbf{w})$ is the $m \times n$ matrix whose i, j element is the partial derivative of $\partial f_i(\mathbf{w}) / \partial w_j$
i.e., i th row of $J_{\mathbf{w}}(\mathbf{f})$ is equal to $\nabla f_i(\mathbf{w})$

Examples

- ▶ If $\mathbf{f}(\mathbf{w}) = A\mathbf{w}$ then $J_{\mathbf{w}}(\mathbf{f}) = A$.
- ▶ If $\sigma : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is element-wise application of $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ then $J_{\theta}(\sigma) = \text{diag}((\sigma'(\theta_1), \dots, \sigma'(\theta_n)))$.

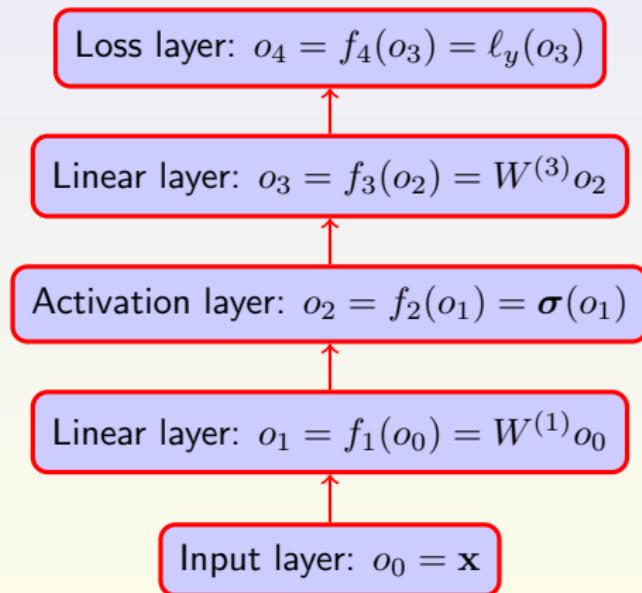
- ▶ **Chain rule:**

$$J_{\mathbf{w}}(\mathbf{f} \circ \mathbf{g}) = J_{g(\mathbf{w})}(\mathbf{f}) J_{\mathbf{w}}(\mathbf{g})$$

Back-Propagation

Let $\ell_y : \mathbb{R}^k \rightarrow \mathbb{R}$ be the loss function at the output layer.

It's convenient to describe the network as a sequence of simple layer functions:



Back-Propagation

- ▶ Can write $\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = (f_{T+1} \circ \dots \circ f_3 \circ f_2 \circ f_1)(\mathbf{x})$
- ▶ Denote $F_t = f_{T+1} \circ \dots \circ f_{t+1}$ and $\delta_t = J_{o_t}(F_t)$, then

$$\begin{aligned}\delta_t &= J_{o_t}(F_t) = J_{o_t}(F_{t-1} \circ f_{t+1}) \\ &= J_{f_{t+1}(o_t)}(F_{t-1})J_{o_t}(f_{t+1}) = J_{o_{t+1}}(F_{t-1})J_{o_t}(f_{t+1}) \\ &= \delta_{t+1}J_{o_t}(f_{t+1})\end{aligned}$$

- ▶ Note that

$$J_{o_t}(f_{t+1}) = \begin{cases} W^{(t+1)} & \text{for linear layer} \\ \text{diag}(\boldsymbol{\sigma}'(o_t)) & \text{for activation layer} \end{cases}$$

- ▶ Using the chain rule again we obtain

$$J_{W^{(t)}}(\ell(h_{\mathbf{w}}, (\mathbf{x}, y))) = \delta_t o_{t-1}^\top$$

Back-Propagation: Pseudo-code

Forward:

- ▶ set $o_0 = \mathbf{x}$ and for $t = 1, 2, \dots, T$ set

$$o_t = f_t(o_{t-1}) = \begin{cases} W^{(t)} o_{t-1} & \text{for linear layer} \\ \sigma(o_{t-1}) & \text{for activation layer} \end{cases}$$

Backward:

- ▶ set $\delta_{T+1} = \nabla \ell_y(o_T)$ and for $t = T, T-1, \dots, 1$ set

$$\delta_t = \delta_{t+1} J_{o_t}(f_{t+1}) = \delta_{t+1} \cdot \begin{cases} W^{(t+1)} & \text{for linear layer} \\ \text{diag}(\sigma'(o_t)) & \text{for activation layer} \end{cases}$$

- ▶ For linear layers, set the gradient w.r.t. the weights in $W^{(t)}$ to be the elements of the matrix $\delta_t o_{t-1}^\top$

Interesting Questions in Deep Learning

Why does it work well?

- ▶ Mathematically, it is not well understood.

This motivated the development of a new class, called

- ▶ E6699: Mathematics of Deep Learning

I thought it in Spring'19, Spring'21, and will teach again in Spring'22.

Here, some interesting topics/questions that the course addressed:
(Maybe I'll discuss further some of the topics next week.)

- ▶ Expressiveness of neural nets
- ▶ Expressive power of deep learning: why is depth good, depth separation results
- ▶ Global versus local optimality
- ▶ Generalization error: Why DL models generalize well?
- ▶ Random initialization
- ▶ Wide nets and connection to Kernels
- ▶ Etc.

Recall the Final Project Outline

- ▶ Done in groups of 4 students - assemble the groups
- ▶ Deliverables: 15+ page **paper** & **presentation** with slides
- ▶ **Due:** during the finals week: Dec 16 - 23, very likely **Dec 17**.
One slot for presentations on **Tue, Dec 14, 4:10-6:40pm.**
- ▶ **Data Repositories:** First, select a paper(s) from either:
 - ▶ UC Irvine Machine Learning Repository
<https://archive.ics.uci.edu/ml/datasets.php>
 - ▶ GEO Data Repository <https://www.ncbi.nlm.nih.gov/geo/>,
or Bioconductor Datasets: <http://www.bioconductor.org>
- ▶ **Final Paper Outline:** 5 sections
 1. **Introduction:** e.g., describe the application area, problems considered, etc
 2. **Data set(s) and paper(s):** e.g., describe data in detail, what was done in the paper(s), common stat/machine learning tools, etc
 3. **Reproduce the results from the paper(s)**
 4. **Try different techniques learned in class, or propose new ones**
 5. **Discussion and conclusion:** e.g., compare different techniques, pros and cons, future work, etc

Bioconductor and Additional Datasets

- ▶ Bioconductor provides tools in R for the analysis genomic data:
<https://www.bioconductor.org/>

- ▶ Installing Bioconductor:

<https://www.bioconductor.org/install/>

Run the following code:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.12")
```

- ▶ Then, install Bioconductor packages:

[https://www.bioconductor.org/install/
#install-bioconductor-packages](https://www.bioconductor.org/install/#install-bioconductor-packages)

- ▶ Datasets supported by Bioconductor: <http://www.bioconductor.org/packages/release/data/experiment/>

Reading

ESL: Chapter 11 on Neural Networks

Homework: Work the final project.

Reading on Deep Learning:

- ▶ Chapters 14& 20 in: Shai Shalev-Shwartz and Shai Ben-David, [Understanding Machine Learning: From Theory to Algorithms](#), Cambridge University Press, 2014.
- ▶ Chapter 6 in: Deep Learning, I. Goodfellow and Y. Bengio and A. Courville, MIT Press, 2016. <http://www.deeplearningbook.org>
- ▶ Software - Tensor Flow in R: <https://tensorflow.rstudio.com>

Optional reading on Kernels (RKHS):

(These books are available online through CU Library)

1. Chapters 1, 2, 13-16 in
B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
2. Chapters 1, 5.3, 6, 7 in (this book is mathematically advanced)
A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer, 2004.

EECS E6690: SL for Bio & Info

Lecture 10: Kernels, NN Nets, Unsupervised Learning: Clustering and Principal Components Analysis (PCA)

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: <http://www.ee.columbia.edu/~predrag>

Last lecture: Unified Supervised Learning Theory

In general, all the supervised learning problems can be written as

$$\hat{f}^* = \arg \min_{\hat{f} \in \mathcal{H}} \sum_{i=1}^n L(y_i, \hat{f}(x_i)) + \lambda \Omega(\hat{f}) \quad (1)$$

where L is a general loss function, and $\lambda \Omega(\hat{f})$ is the penalty, and \mathcal{H} is the space of approximation functions that we are considering.

- ▶ Example: linear functions $\hat{f}(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$

- ▶ \mathcal{H} : set off all p -dimensional linear functions
- ▶ Ridge penalty - square of the ℓ_2 norm

$$\Omega(\hat{f}) = \langle \hat{f}, \hat{f} \rangle = \|f\|^2 = \sum_{j=1}^p \beta_j^2$$

- ▶ Lasso penalty - ℓ_1 norm

$$\Omega(\hat{f}) = \|\hat{f}\|_1 = \sum_{j=1}^p |\beta_j|$$

- ▶ Square loss: $L(y_i, \hat{f}(x_i)) = (y_i - \hat{f}(x_i))^2$

Last lecture: \hat{f} in Kernel Hilbert Spaces

Already used it with SVM: here some more details

- ▶ Hilbert space $\{\mathcal{H}\}$: natural generalization of Euclidian spaces.
- ▶ It has an **inner product**: $\langle f, g \rangle$, for any $f, g \in \mathcal{H}$, which allows computing angles between the elements of $\{\mathcal{H}\}$.
In particular, $\langle f, g \rangle = 0$, then f, g are **orthogonal**.
- ▶ **Norm/distance**: Norm $\|f\| = \sqrt{\langle f, f \rangle}$, and
distance $d(f, g) = \|f - g\| = \sqrt{\langle f - g, f - g \rangle}, \quad f, g \in \mathcal{H}$
- ▶ Euclidian example: if $x, y \in \mathbb{R}^d$, then $\langle x, y \rangle = x_1y_1 + \cdots + x_py_p$

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}, \quad d(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2}$$

- ▶ Example - matrices: $\langle A, B \rangle = \text{trace}(A^\top B)$
- ▶ Example - random variable: $\langle X, Y \rangle = \text{cov}(X, Y)$

Reproducing Kernel Hilbert Spaces (RKHS)

RKHS - \mathcal{H}_k : Subset of Hilbert spaces with really nice properties

- ▶ Kernel is a positive definite function $k(x, y)$
(i.e., $\sum \alpha_i \alpha_j k(x_i, x_j) \geq 0$, for any $\alpha_i, \alpha_j, x_i, x_j$)
- ▶ **Reproducing property:** if $f \in \mathcal{H}_k$, then

$$\langle f, k(\cdot, x) \rangle = f(x)$$

- ▶ Each kernel, $k(x, y)$, defines uniquely the Hilbert space, \mathcal{H}_k . Hence, in this space, \mathcal{H}_k , **all we need to know is the kernel!**
- ▶ Approximation **functions in this space** $f(x) \in \mathcal{H}_k$ can be written as

$$f(\textcolor{blue}{x}) = \sum_i \beta_i k(\textcolor{blue}{x}, x_i)$$

- ▶ Inner product of $f(x) = \sum_i \beta_i k(x, x_i)$ and $g(x) = \sum_j \alpha_j k(x, x_j)$

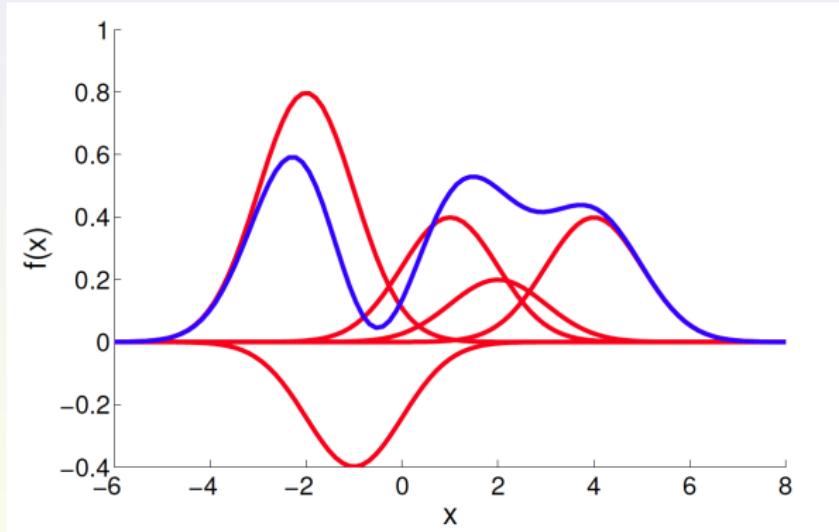
$$\langle f, g \rangle = \sum_{i,j} \alpha_i \beta_j k(x_i, x_j)$$

Example: Gaussian (Radial) Kernel

Gaussian kernel in 1D: $k(x, y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$

Typical function: sum of weighted Gaussian "blobs" - blue line below

$$f(x) = \sum_i \beta_i k(x, x_i) = \sum_i \beta_i e^{\frac{-(x-x_i)^2}{2\sigma^2}}$$



\mathcal{H}_k is a linear space, but $f(x)$ is very nonlinear!

Generalized Learning in RKHS

Here we optimize over all approximation function in \mathcal{H}_k

$$\hat{f}^* = \arg \min_{\hat{f} \in \mathcal{H}_k} \sum_{i=1}^n L(y_i, \hat{f}(x_i)) + \lambda \Omega(\|\hat{f}\|_{\mathcal{H}_k}^2) \quad (2)$$

Representer Theorem For any loss function L and any strictly increasing function $\Omega : \mathbb{R} \rightarrow \mathbb{R}$, an optima solution, \hat{f}^* , of Equation (2) has a form

$$\hat{f}^*(x) = \sum_{i=1}^n \beta_i^* k(x, x_i)$$

\hat{f}^* has a finite representation (!) even though \mathcal{H}_k can be (is) infinite.
(Puts much of the supervised learning under the same umbrella.)

Drawback: works only with ℓ_2 norm, $\|\hat{f}\|_{\mathcal{H}_k}$ - doesn't work for Lasso.)

Generalized Ridge

For quadratic loss function

$$\hat{f}^* = \arg \min_{\hat{f} \in \mathcal{H}_k} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2 + \lambda \|\hat{f}\|_{\mathcal{H}_k}^2 \quad (3)$$

The coefficients, β_i^* , of the optimizer

$$\hat{f}^*(x) = \sum_{i=1}^n \beta_i^* k(x, x_i)$$

are explicitly given by

$$\boldsymbol{\beta}^* = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$$

where \mathbf{K} is a square matrix with elements $K_{ij} = k(x_i, x_j)$, \mathbf{I} is an identity matrix, and \mathbf{y} is the column vector $(y_1, \dots, y_n)^\top$.

Neural Networks Verus Kernel Methods

- ▶ Similarity: both parametric methods
- ▶ Difference: fixed basis for kernels, while NN is discovering basis/features

Neural networks:

- ▶ Rich \mathcal{H} : Provides a versatile parametric class of functions
 - ▶ Universal function approximation
 - ▶ Difficult to train: non-convex optimization
 - ▶ Often accurate predictions, but difficult to interpret
- ▶ Automatic feature/basis extraction
 - ▶ Traditional Feature Engineering approach: expert constructs feature mapping $\phi : \mathcal{X} \rightarrow \Phi$. Then, apply machine learning to find a linear predictor on $\phi(\mathbf{x})$.
 - ▶ “Deep learning” approach: neurons in hidden layers can be thought of as features that are being learned automatically from data
 - ▶ Shallow neurons corresponds to low level features, while deep neurons correspond to high level features

General Parametric Supervised Learning

- ▶ \mathcal{H} is a parametric class of functions
 $f(w, x), w \in \mathcal{W}, w = (w_1, \dots, w_k)$
 - ▶ Examples: Generalized Ridge, or neural networks
- ▶ Finding $f \in \mathcal{H}$ is equivalent to finding $w \in \mathcal{W}$

Hence, our general supervised learning problem can be formulated in terms of w as (say, $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}^m, \ell : \mathbb{R}^{p+m} \rightarrow \mathbb{R}$ - loss function)

$$\hat{w} = \arg \min_{w \in \mathcal{W}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(w, x_i)) + \lambda R(w) \quad (4)$$

How do we solve the preceding problem?

- ▶ When the problem is convex and we are lucky, we can find an explicit \hat{w} by solving (e.g., generalized (Kernel) Ridge regression)

$$\frac{\partial}{\partial w_i} \left(\frac{1}{n} \sum_{i=1}^n \ell(y_i, f(w, x_i)) + \lambda R(w) \right) = 0, \quad i = 1, \dots, k.$$

Parametric Supervised Learning: Numerical Optimization

Let us denote the objective function

$$F(w) := \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(w, x_i)) + \lambda R(w)$$

In general, we use the following numerical algorithm:

1. Initialization: Pick an initial value w_0 , possibly random
2. Iteration: Keep updating w in small steps Δw

$$w_{n+1} = w_n + \Delta w,$$

such that $F(w_{n+1}) < F(w_n)$.

Stoping criteria: $F(w_n) - F(w_{n+1}) < \epsilon$.

For the preceding procedure to find a local minimum

- We need to **find a direction where F has a maximum decrease/steepest descent**
- Avoid getting stuck on a flat surfaces, say flat saddle point

If F is convex, we can find a global minimum.

Recall Multi Calc: Gradient and Directional Derivatives

- ▶ Let $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}, x = (x_1, \dots, x_n)$
- ▶ **Gradient** is a vector of partial derivatives (assuming they exist)

$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

- ▶ Let $u = (u_1, \dots, u_n)$ be a unit vector ($\|u\|_2^2 = \sum u_i^2 = 1$)
- ▶ **Directional derivative** of $f(x)$ in direction u is

$$D_u f(x) := \frac{d}{dt} f(x+ut) = \sum_{i=1}^n \frac{\partial f}{\partial x_i} u_i = \nabla f(x) \cdot u = \|\nabla f(x)\|_2 \cos \theta,$$

where $y \cdot z = \langle y, z \rangle$ is the dot product and θ is the angle between $\nabla f(x)$ and u .

- ▶ Hence, $-\|\nabla f(x)\|_2 \leq D_u f(x) \leq \|\nabla f(x)\|_2$, i.e.,
 - $\nabla f(x)$: direction of steepest ascent/max increase of $f(x)$
 - $-\nabla f(x)$: direction of steepest descent/max decrease of $f(x)$
 - $\nabla f(x)$ is perpendicular to level curves $f(x) = c$ (prove this)

Gradient Descent Algorithm

GD Algorithm is a discrete linear approximation to Equation (??)

$$\frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} \approx \frac{d\mathbf{x}(t)}{dt} = -\eta \nabla f(\mathbf{x}(t))$$

or equivalently (with a small abuse of notation), discrete $t = 0, 1, 2, \dots$,

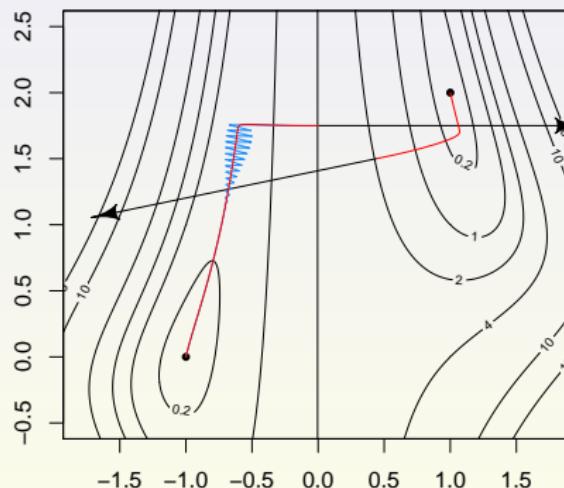
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)})$$

- ▶ Hence, after initialization at $\mathbf{x}^{(0)}$, the GD Algorithm follows the preceding iteration to a local minimum
- ▶ Stopping criterion (could be): $|f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})| < \epsilon$

Adaptive GD (AdaGrad): modifies the learning rate η_t in each iteration t

More Interesting Landscape

- ▶ $f(x) = (x_1^2 - 1)^2 + (x_1^2 x_2 - x_1 - 1)^2$
- ▶ Gradient
$$\nabla f(x) = \begin{bmatrix} 4x_1(x_1^2 - 1) + 2(2x_1 x_2 - 1)(x_1^2 x_2 - x_1 - 1) \\ 2x_1^2(x_1^2 x_2 - x_1 - 1) \end{bmatrix}$$
- ▶ Oscillations in "narrow valleys"

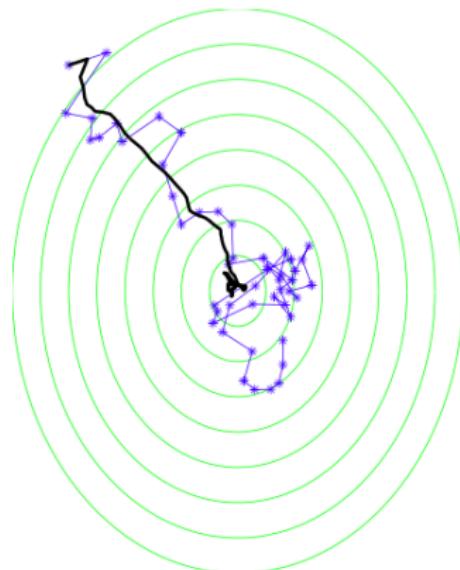
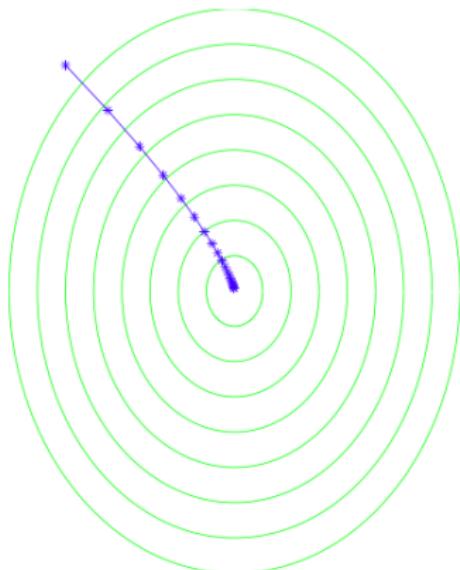


Motivation for momentum: remembers/averages previous Δx

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)}) + \mu_t (\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)})$$

Stochastic Gradient Descent

- ▶ Dates back to Robbins and Monroe (1951).
- ▶ Stochastic gradient is an **unbiased estimator** of the gradient
- ▶ Stochastic versus regular gradient descent



Stochastic Gradient Descent

- ▶ Stochastic approximation of gradient descent
- ▶ Function (typically encountered in learning)

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- ▶ Computationally expensive gradient for large n
- ▶ Approximation: pick a random subset $\mathcal{S} \in [1, n]$

$$\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla f_i(x)$$

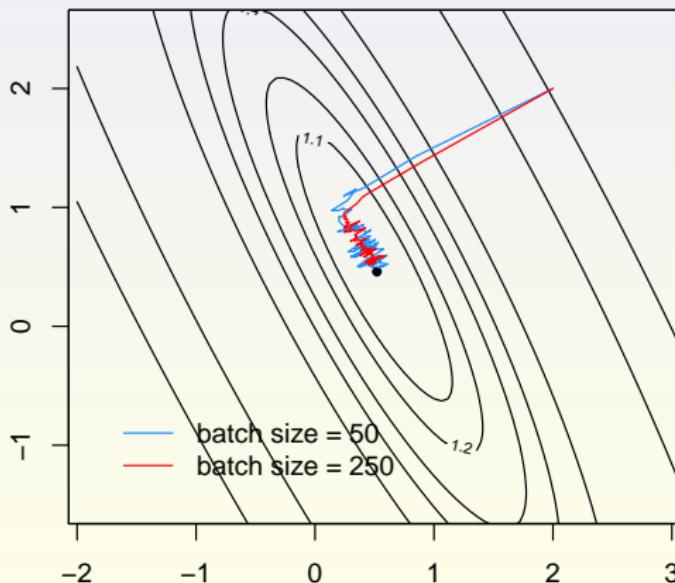
- ▶ \mathcal{S} - called batch/mini-batch
- ▶ Example
 - ▶ n scalar data points x_1, x_2, \dots, x_n
 - ▶ objective

$$\min_c \frac{1}{n} \sum_{i=1}^n (x_i - c)^2$$

SGD Example: Linear Regression

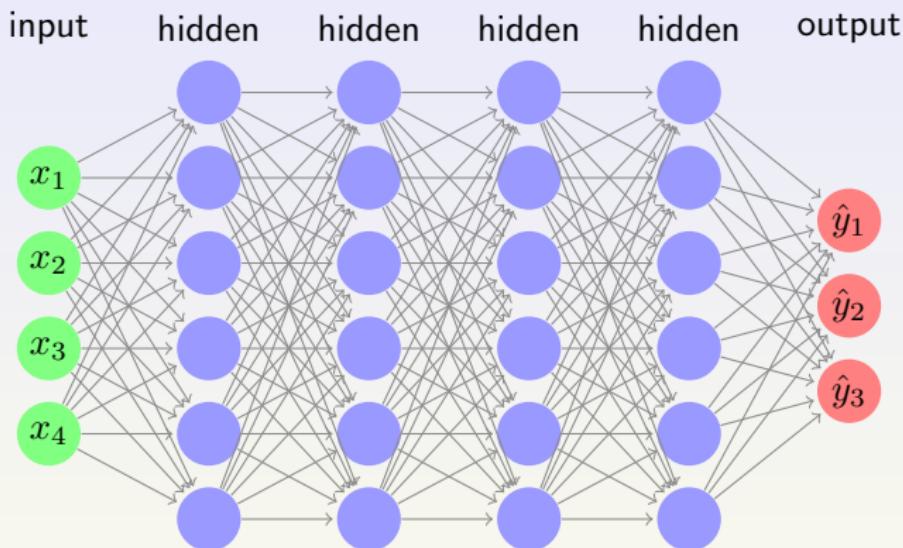
- ▶ Data: $(x_i, y_i)_{i=1}^n, n = 10^5$
- ▶ Loss function: $L(\beta, x, y) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$
- ▶ Stochastic gradient

$$\nabla_{\beta} \hat{L}(\beta, x, y) = \frac{2}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left[\begin{array}{l} (\beta_0 + \beta_1 x_i - y_i) \\ x_i(\beta_0 + \beta_1 x_i - y_i) \end{array} \right]$$



Deep Neural Networks, a.k.a. Multilayer Perceptrons

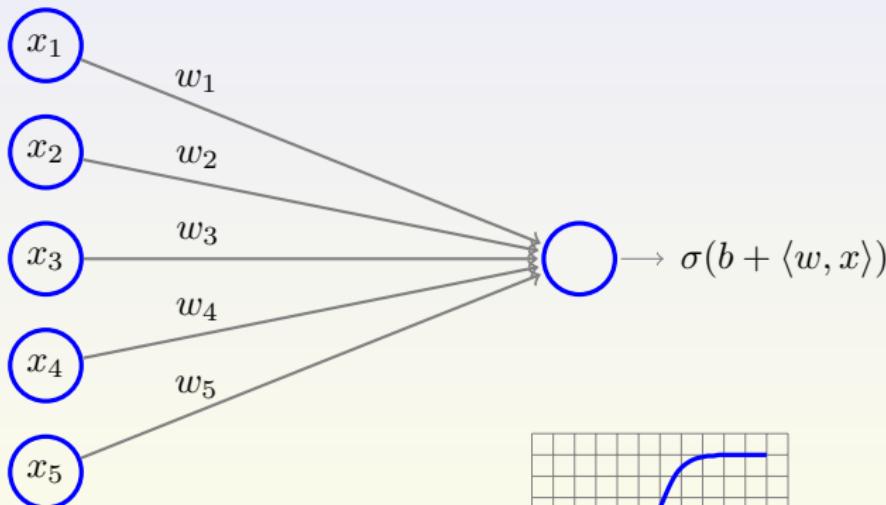
- ▶ Feed-forward network



- ▶ Designed to mimic the function of neurons
- ▶ Blue nodes: activation functions/neurons
- ▶ Depth = lengths of a longest path
- ▶ Deep network: depth ≥ 3
- ▶ Very successful in solving practical problems

A Single Artificial Neuron

- ▶ A single neuron function: $\mathbf{x} \mapsto \sigma(b + \langle \mathbf{w}, \mathbf{x} \rangle)$, where $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is called the activation function of the neuron. Inner/dot product:
$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum x_i w_i.$$
- ▶ More compact notation $\langle \tilde{\mathbf{w}}, \tilde{\mathbf{x}} \rangle$, where $\tilde{\mathbf{x}} = (1, \mathbf{x})$, $\tilde{\mathbf{w}} = (b, \mathbf{w})$



- ▶ E.g., σ is a sigmoidal function

Common Activation Functions/Perceptrons

- ▶ step function

$$\sigma(x) = 1_{\{x>0\}} \quad \sigma'(x) = 0, x \neq 0$$

- ▶ logistic

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \sigma'(x) = \sigma(x)(1 - \sigma(x))$$

- ▶ rectified linear unit (ReLU)

$$\sigma(x) = \max\{x, 0\} \quad \sigma'(x) = 1_{\{x>0\}}, x \neq 0$$

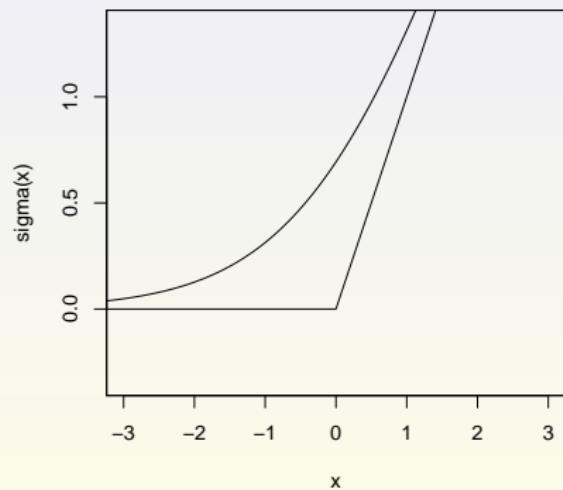
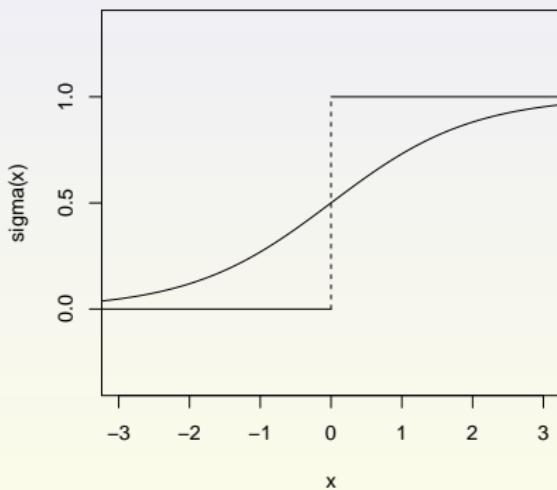
- ▶ soft-plus

$$\sigma(x) = \log(1 + e^x) \quad \sigma'(x) = \frac{1}{1 + e^{-x}}$$

Comparing Activation Functions

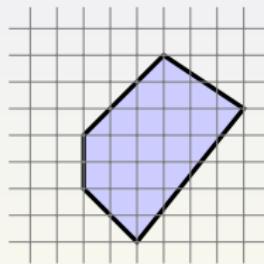
Logistic and soft-plus have continuous derivatives in comparison to step function and ReLU

- ▶ Positive derivative avoids vanishing gradient



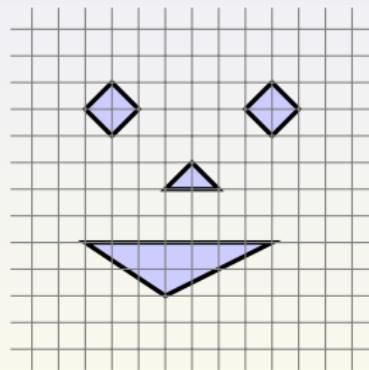
Example

- ▶ Single neuron is a binary half-space classifier: $\text{sign}(w \cdot x + b)$
- ▶ 2 layer networks can express intersection of halfspaces



Example

- ▶ 3 layer networks can express unions of intersection of halfspaces



How to train neural network?

- ▶ Neural nets: excellent hypothesis class, but difficult to train
- ▶ Main technique: Stochastic Gradient Descent (SGD)
- ▶ Not convex, no guarantees, can take a long time, but:
 - ▶ Often still works fine, finds a good solution
 - ▶ With some luck:)

Stochastic Gradient Descent (SGD) for Neural Networks

Common Training Ideas:

- ▶ Random initialization: rule of thumb, $w[u \rightarrow v] \sim U[-c, c]$ where $c = \sqrt{3/|\{(u', v) \in E\}|}$ (or small Gaussian instead of $U[-c, c]$)
- ▶ Update step with Nesterov's momentum: Initialize $\theta = 0$ and:

$$\begin{aligned}\theta_{t+1} &= \mu_t \theta_t - \eta_t \tilde{\nabla} L(w_t + \mu_t \theta_t) \\ w_{t+1} &= w_t + \theta_{t+1}\end{aligned}$$

where:

μ_t is momentum parameter (e.g. $\mu_t = 0.9$ for all t)

η_t is learning rate (e.g. $\eta_t = 0.01$ for all t)

$\tilde{\nabla} L$ is an estimate of the gradient of L based on a small set of random examples (often called a “minibatch”)

- ▶ Efficient gradient calculation: **Backpropagation**

Interesting Questions in Deep Learning

Why does it work well?

- ▶ Mathematically, it is not well understood.

This motivated the development of a new class, called

- ▶ E6699: Mathematics of Deep Learning

I thought it in Spring'19, Spring'21, Spring'22, and will teach again in Spring'23.

Here, some interesting topics/questions that the course addressed:
(Maybe I'll discuss further some of the topics next week.)

- ▶ Expressiveness of neural nets
- ▶ Expressive power of deep learning: why is depth good, depth separation results
- ▶ Global versus local optimality
- ▶ Generalization error: Why DL models generalize well?
- ▶ Random initialization
- ▶ Wide nets and connection to Kernels
- ▶ Etc.

Unsupervised Learning

There is no input-output relationship, $y = f(x)$, i.e. **there is no y .**

- ▶ We only have a bunch of points (x_1, x_2, \dots, x_n)
- ▶ The problem has less structure
- ▶ We are trying to discover a structure - maybe more interesting

Typical questions and approaches

- ▶ Clustering - grouping data points
- ▶ Principal Component Analysis (PCA): used of preprocessing and visualization
- ▶ Association Rules - discovering relationships between data points
- ▶ Community Detection or Graph Clustering: e.g., discovering communities on Face book
- ▶ Ranking: e.g., Google's PageRank algorithm

Clustering

Clustering looks to find homogeneous subgroups among the observations

Approaches

- ▶ Flat: we set in advance how many clusters are there
 - ▶ K-means
- ▶ Hierarchical: bottom up (agglomerative) merging or top down splitting
 - ▶ Bottom up: tree-based representation - dendrogram

K-Means Clustering

K-Means - simple and intuitive

Notation

- ▶ n - number of data points
- ▶ K - fixed and predetermined number of clusters
(weakness of this procedure)
- ▶ C_1, \dots, C_K : sets containing the indices of the observations in each cluster.
- ▶ These sets satisfy two properties:
 1. $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$ - each observation in at least one clusters.
 2. $C_1 \cap C_2 \cap \dots \cap C_K = \emptyset$ - clusters are nonoverlapping

K-Means Clustering

Must have a **measure of closeness/distance** between points.

Examples of distances (we have seen them before)

- ▶ Square of Euclidian distance - ℓ_2 norm

$$d(x_i, x_j) = \sum_{l=1}^p (x_{il} - x_{jl})^2$$

- ▶ ℓ_1 norm - also known Manhattan street norm

$$d(x_i, x_j) = \sum_{l=1}^p |x_{il} - x_{jl}|$$

- ▶ Correlation

$$\rho(x_i, x_j) = \frac{\sum_{l=1}^p (x_{il} - \bar{x}_i)(x_{jl} - \bar{x}_j)}{\sqrt{\sum_{l=1}^p (x_{il} - \bar{x}_i)^2 \sum_{l=1}^p (x_{jl} - \bar{x}_j)^2}}$$

- ▶ Many other distance measures could work, e.g. mutual information, or those we have seen in Hilbert spaces
- ▶ Carefully choosing the distance measure can make a difference

K-Means Clustering

Next, define a weight of a cluster: the closeness of points in a cluster

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,j \in C_k} d(x_i, x_j)$$

where $|C_k|$ denotes the number of points in C_k .

$W(C_k)$ is the average distance in the cluster.

Now, the clustering problem is to find C_1, \dots, C_K that minimize

$$\min_{C_1, \dots, C_K} \sum_{k=1}^K W(C_k)$$

K-Means Clustering

Euclidian example

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,j \in C_k} \sum_{l=1}^p (x_{il} - x_{jl})^2$$

where $|C_k|$ denotes the number of points in C_k .

$W(C_k)$ is the average Euclidean distance

Now, the clustering problem is explicitly given as

$$\min_{C_1, \dots, C_K} \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i,j \in C_k} \sum_{l=1}^p (x_{il} - x_{jl})^2$$

K-Means Algorithm

Algorithm for Euclidean distance

1. Initialization: Randomly assign a number, from 1 to K, to each of the observations.
2. Iterate until the cluster assignments stop changing:
 - (a) 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.
 - (b) Assign each observation to the cluster whose centroid is closest in Euclidean distance.

Properties

- ▶ It converges due to this identity

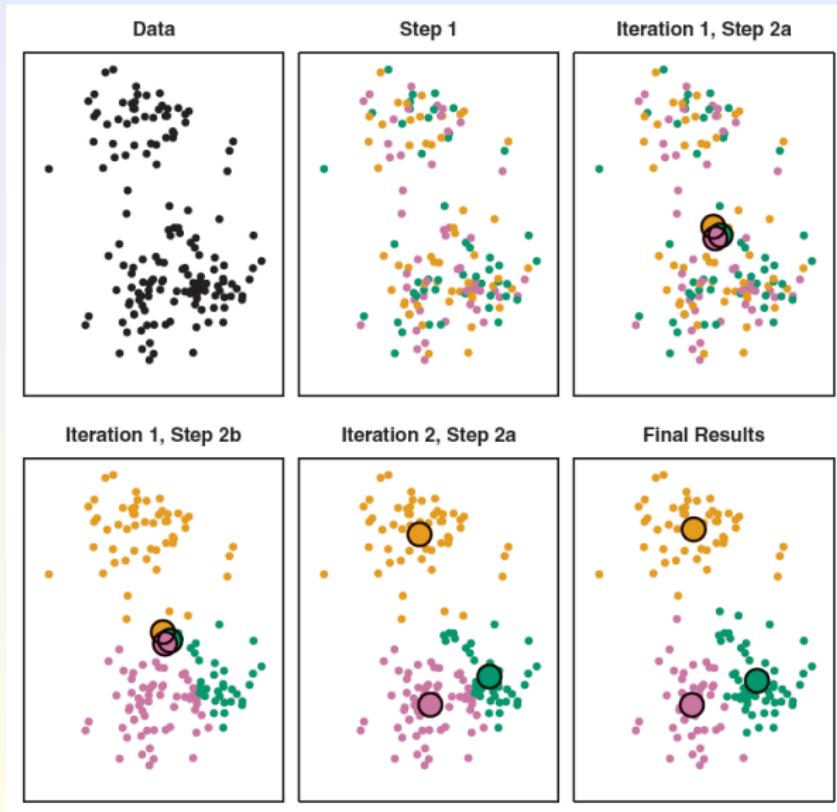
$$\frac{1}{|C_k|} \sum_{i,j \in C_k} \sum_{l=1}^p (x_{il} - x_{jl})^2 = 2 \sum_{i \in C_k} \sum_{l=1}^p (x_{il} - \bar{x}_{kl})^2$$

$\bar{x}_{kl} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{il}$ is the l -coordinate of the cluster centroid

- ▶ Problem: finds only a local minimum
- ▶ Find good starting points and repeat a few times

K-Means: Example

$K = 3$



K-Means: Example

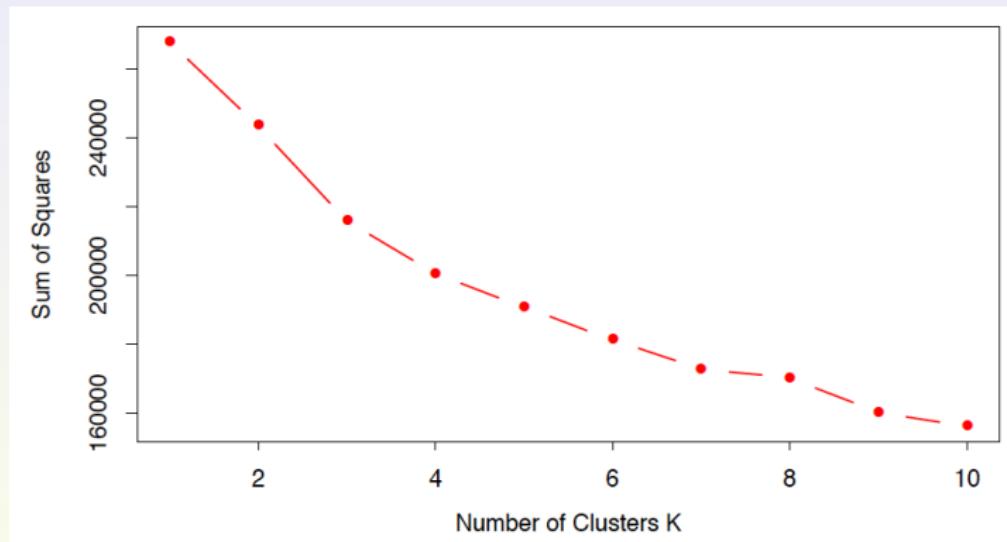
$K = 3$ - Different, possibly bad, local minima



K-Means: What is the best K?

No good analytical procedure

Heuristics: Look for a **kink** on the plot of the total within-cluster sum of squares. **May not work** in general.



Total within-cluster sum of squares for human tumor microarray data

Hierarchical Clustering

- ▶ Bottom up: Start with each point being a cluster
- ▶ Decide on a distance of dissimilarity
- ▶ Compute the distance between all pairs of points, and merge the two closest into one class.
- ▶ **Q:** How do we measure the distance between sets?
 - ▶ This is called [linkage](#)

Types of linkage

- ▶ **Complete:** Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between 2 clusters and pick the largest
- ▶ **Single:** Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between 2 clusters and pick the largest
- ▶ **Average:** Mean intercluster dissimilarity. Compute all pairwise dissimilarities between record the average
- ▶ **Centroid:** Distance between the centroids of two clusters

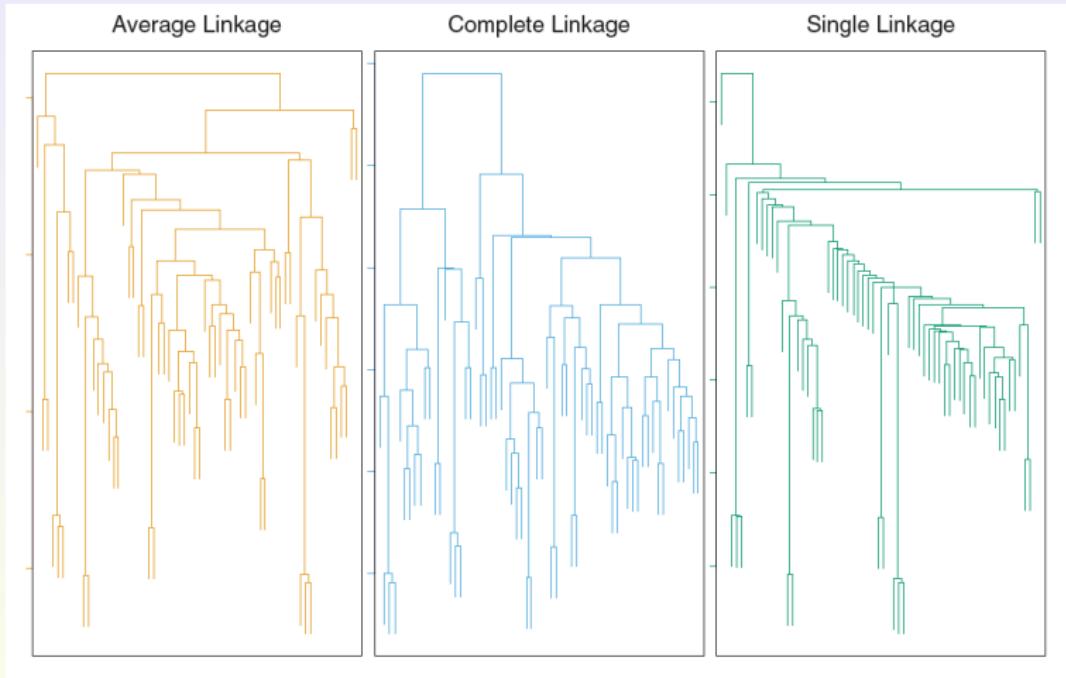
Hierarchical Clustering Algorithm

Algorithm

1. Begin with n observations and a measure (such as Euclidean distance) of all the $n(n - 1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For $i = n, n - 1, \dots, 2$:
 - (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters.
 - (b) Compute the new pairwise inter-cluster dissimilarities among the $i - 1$ remaining clusters.

Hierarchical Clustering: Impact of Linkage

Average and complete linkage tend to yield more balanced clusters.



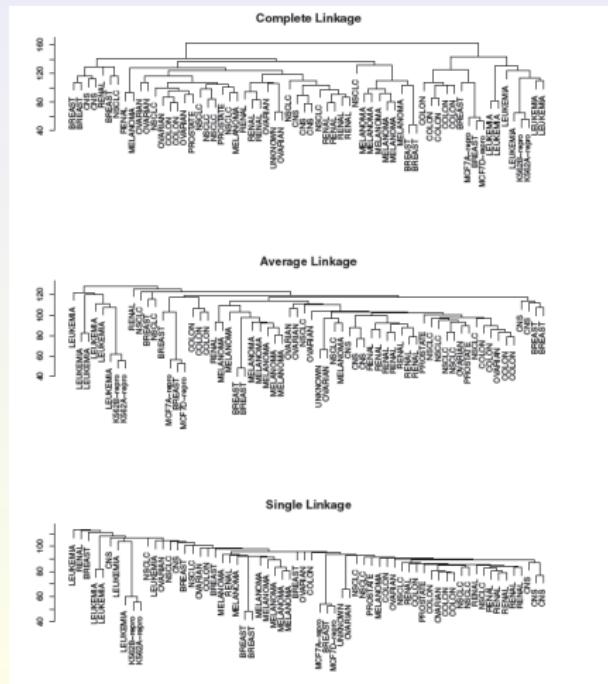
NCI60 Cancer Data

- ▶ 64 cancers: 6,830 gene
- ▶ >library(ISLR)
>nci.labs=NCI60\$labs
>nci.data=NCI60\$data

```
> nci.labs[1:4]
[1] "CNS"    "CNS"    "CNS"    "RENAL"
> table(nci.labs)
nci.labs
      BREAST          CNS          COLON K562A-repro K562B-repro
      7             5             7           1           1
      LEUKEMIA  MCF7A-repro  MCF7D-repro      MELANOMA      NSCLC
      6             1             1           8           9
      OVARIAN      PROSTATE        RENAL      UNKNOWN
      6             2             9           1
```

NCI60 Dendrogram with Euclidean Distance

Complete and average linkage tend to yield evenly sized clusters whereas single linkage tends to yield extended clusters to which single leaves are fused one by one.



Principal Components Analysis (PCA)

Principal Components Analysis - unsupervised approach (no use of Y)

- ▶ Finding directions along which data is located - direction of largest variance
- ▶ These directions define lines/subspaces that are close to the "data cloud"
- ▶ Can be used for visualization/understanding data
- ▶ Can be used as preprocessing for supervised learning

First Principal Components

- ▶ Look for the linear combination of the sample feature of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$

- ▶ *First loading vector*: $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$
- ▶ Assume x_i -s are **centered** ($\sum x_i = 0$)
- ▶ Look for ϕ_1 that has the largest sample variance, i.e.

$$\max_{\phi_1} \frac{1}{n} \sum_{i=1}^n z_{i1}^2 = \max_{\phi_1} \frac{1}{n} \sum_{i=1}^n (\phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip})^2$$

subject to $\sum_{j=1}^p \phi_{j1}^2 = 1$ (i.e., ϕ_1 is a unit vector)

- ▶ z_{i1} are the **scores**
- ▶ This optimization problem can be solved via eigen-decomposition

First Principal Components: Geometric interpretation

- ▶ Loading vector ϕ_1 represents the direction along which the data varies the most
- ▶ If we project x_1, \dots, x_n onto ϕ_1 , the projected values are the PC scores z_{i1} since

$$z_{i1} = \langle x_i, \phi_1 \rangle$$

Second and higher principal components

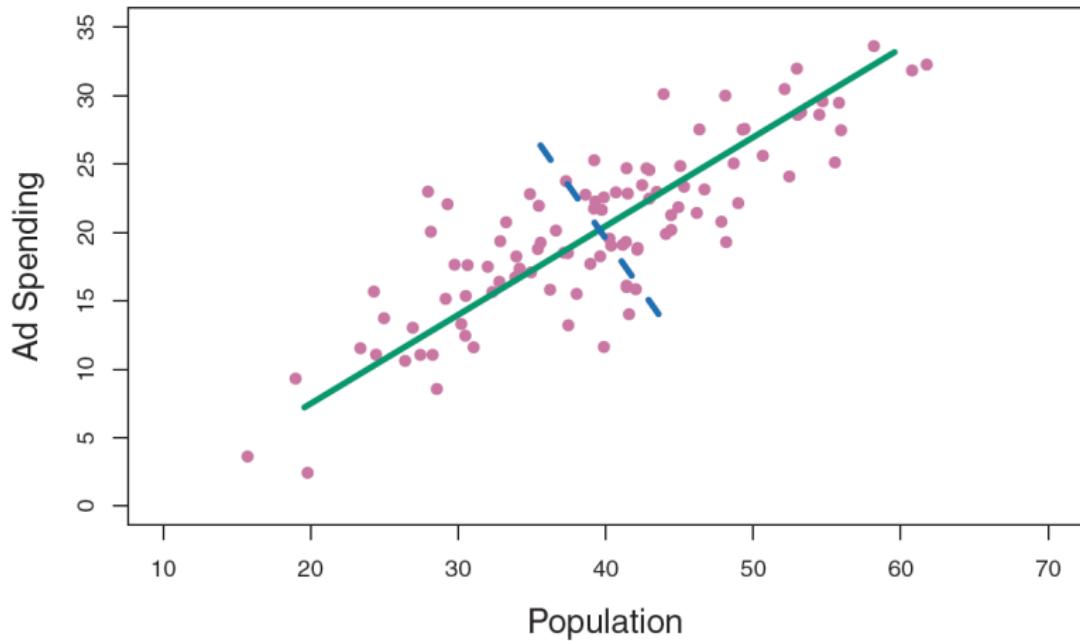
- ▶ After ϕ_1 has been determined, we look for ϕ_2 in a similar way, but with the additional constraint that ϕ_1, ϕ_2 are uncorrelated

$$\langle \phi_1, \phi_2 \rangle = 0$$

- ▶ We continue this procedure until we find as many PC as we want

PCA: Example

Two PC-s: Solid line: First PC; Dashed line: Second PC



PCA: Good for High Dimensional Data - Large p

- ▶ Dimensionality reduction - e.g. gene expression data
- ▶ Assume we compute M principal components
- ▶ The best M -dimensional approximation to x_{ij}

$$x_{ij} = \sum_{m=1}^M z_{im} \phi_{jm}$$

How good is PC approximation?

- ▶ Proportion of Variance Explained (PVE)
- ▶ For each component, m , PVE is equal to

$$\frac{\sum_{i=1}^n \left(\sum_{j=1}^p \phi_{jm} x_{ij} \right)^2}{\sum_{ij} x_{ij}^2}$$

PCA is an Eigen-Decomposition Problem

- In general finding k -principal components is equivalent to finding a $k \times p$ matrix such that $\mathbf{z}_i = W\mathbf{x}_i$

- We also want linear recovery, i.e., to find a matrix U , such that

$$\hat{\mathbf{x}}_i = U\mathbf{z}_i = UW\mathbf{x}_i \approx \mathbf{x}_i$$

- Hence, this reduces to an optimization problem

$$\arg \min_{W,U} \sum_{i=1}^n \|\mathbf{x}_i - UW\mathbf{x}_i\|^2 \quad (5)$$

Theorem Let $A = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$ and let $\mathbf{u}_1, \dots, \mathbf{u}_k$ be the k leading eigenvectors of A . Then, the solution to the PCA problem is to set the columns of U to be $\mathbf{u}_1, \dots, \mathbf{u}_k$ and to set $W = U^\top$.

Other types of linear dimensionality reduction:

- Compressed sensing:** Also finds a matrix W with special properties and compresses \mathbf{x} into $W\mathbf{x}$. Reconstruction solves a linear program.

PCA is an Eigen-Decomposition Problem: Proof

Lemma Let (U, W) be a solution to Equation (5). Then, the columns of U are orthonormal: namely, $U^\top U = I$ and $W = U^\top$.

Proof:

- ▶ Consider space $R = \{UW\mathbf{x} : \mathbf{x} \in \mathbb{R}^p\} \subset \mathbb{R}^p$
- ▶ Let V be orthonormal matrix ($V^\top V = I$) whose columns are the basis of R
- ▶ Hence, each vector in R can be written as $V\mathbf{z}$, $\mathbf{z} \in \mathbb{R}^k$ and

$$\|\mathbf{x} - V\mathbf{z}\|_2^2 = \|\mathbf{x}\|_2^2 + \mathbf{z}^\top V^\top V\mathbf{z} - 2\mathbf{z}^\top (V^\top \mathbf{x})$$

which, by taking derivative w.r.t. \mathbf{z} is minimized for $\mathbf{z} = V^\top \mathbf{x}$, implying

$$VV^\top \mathbf{x} = \arg \min_{\tilde{\mathbf{x}} \in R} \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2$$

and furthermore

$$\arg \min_{U,W} \sum_{i=1}^n \|\mathbf{x}_i - UW\mathbf{x}_i\|_2^2 \geq \arg \min_V \sum_{i=1}^n \|\mathbf{x}_i - VV^\top \mathbf{x}_i\|_2^2$$

PCA: Proof of Theorem

- Based on the preceding lemma, optimization in Equation (5) can be written as

$$\arg \min_{U \in \mathbb{R}^{p,k}: U^\top U = I} \sum_{i=1}^n \|\mathbf{x}_i - UU^\top \mathbf{x}_i\|_2^2$$

- Next, since $U^\top U = I$, it follows that

$$\|\mathbf{x} - UU^\top \mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2 - \text{trace}(U^\top \mathbf{x} \mathbf{x}^\top U), \quad (y^\top y = \text{trace}(yy^\top))$$

which means that one can equivalently solve

$$\arg \max_{U: U^\top U = I} \text{trace}(U^\top \sum_{i=1}^p \mathbf{x}_i \mathbf{x}_i^\top U)$$

- Let $V\Lambda V^\top = A := \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$, be the spectral decomposition of A with $VV^\top = V^\top V = I$, and Λ a diagonal matrix whose diagonal elements are the eigenvalues of A ; assume that $\lambda_1 \geq \dots \geq \lambda_p$. Then,

$$U^\top AU = U^\top V\Lambda V^\top U =: B^\top \Lambda B, \quad B := V^\top U$$

PCA: Proof of Theorem

- ▶ Hence,

$$\text{trace}(U^\top A U) = \sum_{j=1}^p \lambda_j \sum_{i=1}^k B_{j,i}^2$$

- ▶ Also, $B^\top B = I$, which implies

$$\sum_{j=1}^p \sum_{i=1}^k B_{j,i}^2 = k.$$

- ▶ Next, let $\tilde{B} \in \mathbb{R}^{p,p}$ be a matrix whose first k columns are the columns of B . Then,

$$\beta_j := \sum_{i=1}^k B_{j,i}^2 \leq \sum_{i=1}^p \tilde{B}_{j,i}^2 = 1$$

PCA: Proof of Theorem

- ▶ Therefore

$$\text{trace}(U^\top AU) \leq \max_{\beta \in [0,1]^p : \|\beta\|_1 \leq k} \sum_{j=1}^p \lambda_j \beta_j \leq \sum_{j=1}^k \lambda_j$$

- ▶ Furthermore, this upper bound is achieved when U is chosen to be the matrix whose columns are equal to the k leading eigenvectors of A , i.e., direct calculation yields

$$\text{trace}(U^\top AU) = \sum_{j=1}^k \lambda_j,$$

which completes the proof.

More details can be found in Chapter 23 of ML book by Shalev-Shwartz & Ben-David, 2014: <https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf>

Kernel PCA

- ▶ PCA is based on orthogonal projections
- ▶ So we can use Kernels and Hilbert spaces
- ▶ Recall that our approximation functions are of the form

$$g_1(x) = \sum_{i=1}^n \phi_{i1} k(x, x_i)$$

- ▶ Hence, the optimization is

$$\max_{g_1 \in \mathcal{H}_k} \text{Sample Var}(g_1(X)) \quad \text{subject to} \quad \|g_1\|_{\mathcal{H}_k} = 1$$

- ▶ Now the approximation "directions" are not lines any more
- ▶ This is also eigen-decomposition problem

PCA on the NCI60 Data

- ▶ First *scale* the data

```
> pr.out=prcomp(nci.data, scale=TRUE)
```

- ▶ Then, **assign a color** to each of the 64 cancer cell lines

```
Cols=function(vec){  
+ cols=rainbow(length(unique(vec)))  
+ return(cols[as.numeric(as.factor(vec))]) }
```

- ▶ We now can plot the principal component score vectors

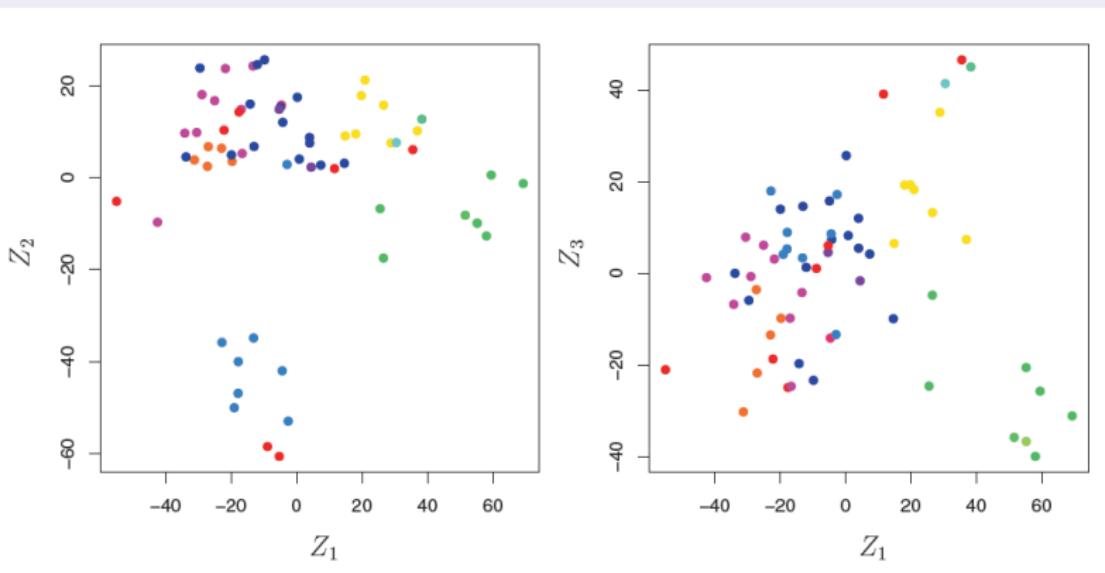
```
> par(mfrow=c(1,2))
```

```
> plot(pr.out$x[,1:2], col=Cols(nci.labs), pch=19,  
+ xlab="Z1",ylab="Z2")
```

```
> plot(pr.out$x[,c(1,3)], col=Cols(nci.labs), pch=19,  
+ xlab="Z1",ylab="Z3")
```

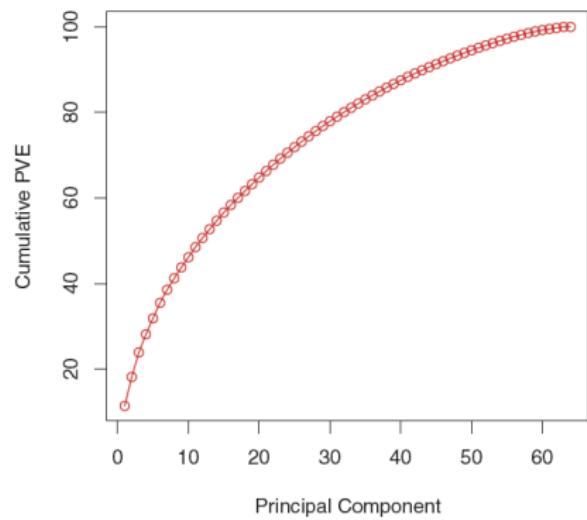
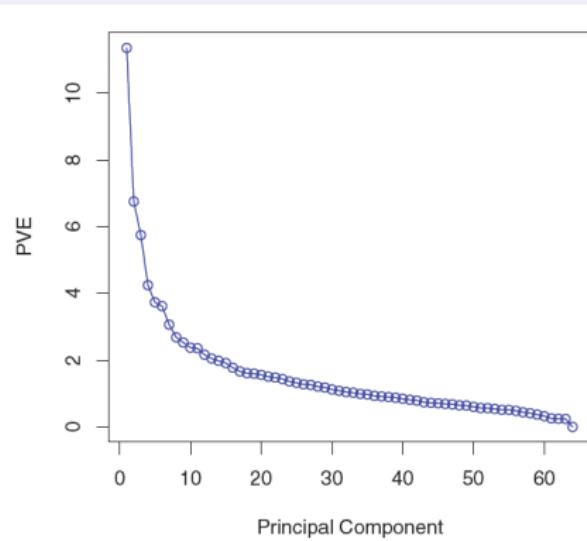
PCA on the NCI60 Data

- ▶ Observations belonging to a single cancer type tend to lie near each other in this low-dimensional space.
- ▶ Not possible to visualize data without PCA, $\binom{6832}{2}$ scatter plots (!)



Principal Value Explained (PVE) on NCI60 Data

- ▶ First 7 PC-s explain 40% of data
- ▶ 70 PC-s explain 100% of data
- ▶ Compare 70 to $p = 6832$ (!)



Reading

- ▶ ESL: Chapter 11, 14; ISL: Chapter 10

Homework: Work on the final project. Today is the deadline for group formation and project selection.

Reading on Deep Learning:

- ▶ Chapters 14& 20 in: Shai Shalev-Shwartz and Shai Ben-David, [Understanding Machine Learning: From Theory to Algorithms](#), Cambridge University Press, 2014.
- ▶ Chapter 6 in: Deep Learning, I. Goodfellow and Y. Bengio and A. Courville, MIT Press, 2016. <http://www.deeplearningbook.org>
- ▶ Software - Tensor Flow in R: <https://tensorflow.rstudio.com>

Optional reading on Kernels (RKHS):

(These books are available online through CU Library)

1. Chapters 1, 2, 13-16 in
B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
2. Chapters 1, 5.3, 6, 7 in (this book is mathematically advanced)
A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer, 2004.