

DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 1

Introduction and Review



Definition

- What is machine learning?
 - A set of tools for understanding data by building models *from data*
 - Methods for *automatically* learning and recognizing complex patterns from data
 - Arthur Samuel (1959): “[a] field of study that gives computers the ability to learn without being explicitly programmed”

Definition

- These uncovered patterns are then used to **predict** future data, or to perform other kinds of decision-making under uncertainty.
- The key premise is *learning* from data!!
- **predict** = guess the value(s) of unknown variable(s)
 - (not necessarily prediction of future... c.f. *forecasting*)
- **future data** = data you haven't seen before

Big Data is Everywhere



We are in the era of **big data!**

- 40 billion indexed web pages
- 100 hours of video are uploaded to YouTube every minute

The deluge of data calls for automated methods of data analysis, which is what **machine learning** provides!

Driving Forces

- Explosive growth of data in a great variety of fields
 - Cheaper storage devices with higher capacity
 - Faster communication
 - Better database management systems
- Rapidly increasing computing power
- We want to make the data work for us!!

Examples of Applications of Machine Learning

- In automated decision making
 - e.g. email spam filtering



Examples of Applications of Machine Learning

- In automated decision making
 - forensics / fraud detection



Examples of Applications of Machine Learning

- For automating complex programming tasks
 - e.g. driverless cars



Examples of Applications of Machine Learning

- For automating complex programming tasks
 - e.g. optical character recognition (OCR)

Optical character Recognition
is designed to convert your
handwriting into text.

Optical Character Recognition
is designed to convert your
handwriting into text.

Examples of Applications of Machine Learning

- In self-customizing algorithms
 - e.g. recommendation systems

Recommended for You

These recommendations are based on [items you own and more.](#)

[All](#) | [New Releases](#) | [Coming Soon](#)

 **Cybertext: Perspectives on Ergodic Literature**
by Espen J. Aarseth (Aug 6, 1997)
Average Customer Review: ★★★★☆ (3)
In Stock
List Price: \$22.95
Price: \$19.55 [Add to cart](#) [Add to](#)
29 used & new from \$10.82

I own it Not interested Rate it

Recommended because you added *Hamlet on the Holodeck* to your Shopping Cart and more ([Fix this](#))

 **Narrative as Virtual Reality: Immersion and Interactivity in Literary Media (Parallax: Re-visions of Culture and Society)**
by Michael Renée (Aug 28, 2001)

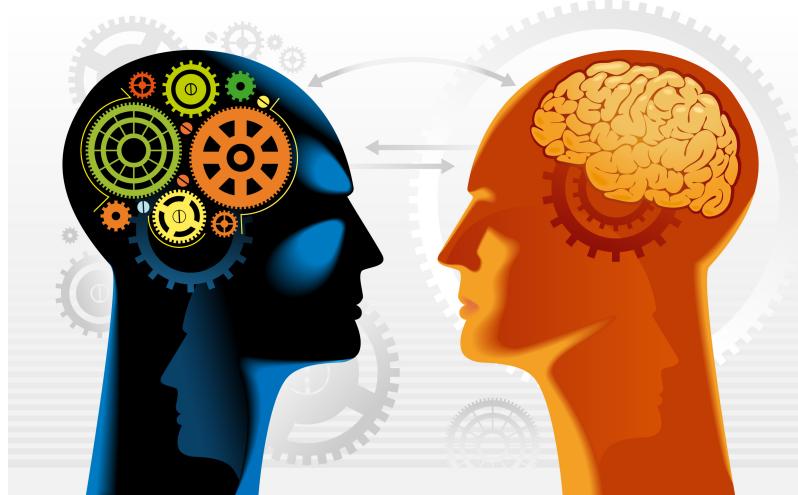
Examples of Applications of Machine Learning

- In finance
 - To predict movements in markets
 - for risk management



Examples of Applications of Machine Learning

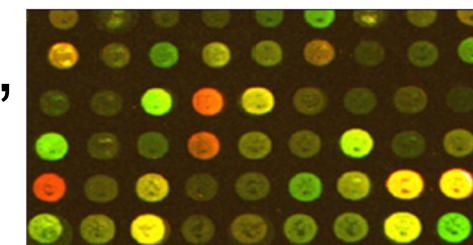
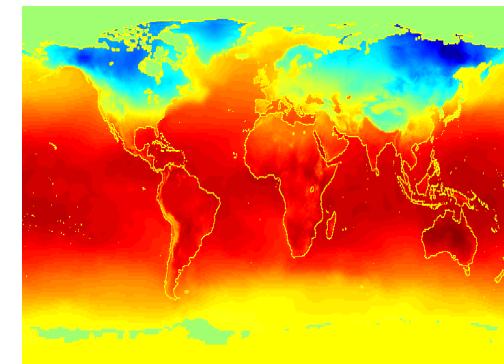
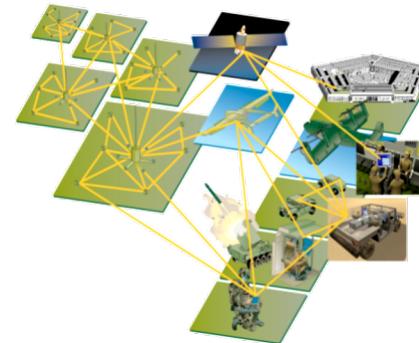
- Natural language processing
 - Speech recognition
 - Machine translation
 - Question Answering
 - Document classification and retrieval (books, email, web pages)
 - Sentiment analysis



Examples of Applications of Machine Learning

- Scientific

- Remote sensing networks:
atmosphere, ocean, fresh-water,
land-based, satellite
 - weather and climate modeling
 - environmental management
 - resource management
- Biomedical: gene sequencing,
gene expression, epidemiology,
disease prediction



Types of Learning

- Supervised learning
 - Goal: Prediction
 - **Semi-Supervised Learning**
- Unsupervised learning
 - Goal: Discovery of Similarity or dissimilarity
 - Goal: Dimensionality Reduction
- Reinforcement learning

Supervised Learning

Starting point:

- Outcome measurement Y (also called dependent variable, response, target).
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables).

Supervised Learning

- In the *regression problem*, Y is quantitative (e.g price, blood pressure).
- In the *classification problem*, Y takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample).

Supervised Learning

- We have input-output data $(x_1, y_1), \dots, (x_N, y_N)$ that are called **training data**.
- These are observations (examples, instances) of these measurements.

Objectives

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

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

Classification

- Document classification
 - Is this email spam?
 - Is this tweet positive toward this product?
 - Is this review/article real?
- Image classification
 - Is this a photo of a cat?
 - Which letter or number is written here?
- Object recognition
 - Identify the faces in this image
 - Identify pedestrians in this video

Classification

A classification algorithm is called a **classifier**

Classifiers require examples of inputs paired with outputs

- Called **training data**

Classifiers learn from training examples to map input to output

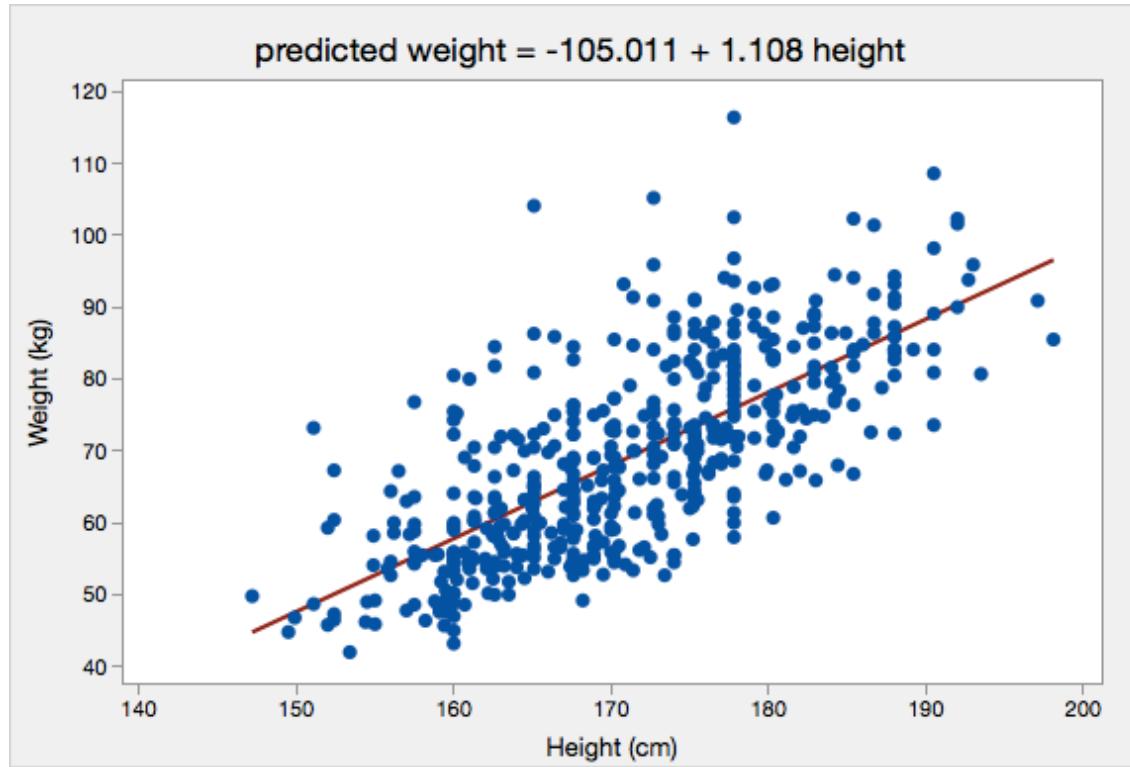
- Then when a classifier encounters new data where the output is unknown, it can make a prediction

Supervised Learning

Two types of prediction:

- Classification
 - Discrete outputs (typically categorical)
- **Regression**
 - **Continuous outputs (usually)**

Regression



Linear regression of weight based on one input variable (height)

Regression

Examples:

- Predict one's weight based on one's height
- Predicting how much money a product will make
- Forecasting a stock price tomorrow
- Estimate someone's age based on their face
- Predict a student's GPA based on their socio-economic conditions

Types of Learning

- Supervised learning
 - Goal: Prediction
 - **Semi-Supervised Learning**
- **Unsupervised learning**
 - **Goal: Discovery of similarity or dissimilarity**
 - **Goal: Dimensionality Reduction**
- Reinforcement learning

Unsupervised Learning

- No outcome variable, just a set of predictors (features) measured on a set of samples.
- Objective is vague— find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation.

Unsupervised Learning

- Difficult to know how well you are doing.
- Different from supervised learning, but can be useful as a pre-processing step for supervised learning.

Unsupervised Learning

Example: **anomaly detection**

- Trying to identify something unusual (e.g., fraud) but you don't know what it looks like

Unsupervised: Clustering & Dimensionality Reduction

- *Cluster analysis*: partition data into subsets that share common characteristics
 - e.g. group similar patients in a medical database
- *Dimensionality reduction*: create new features from original inputs that retain important information
 - e.g. represent a document as a small set of topics instead of as a large collection of words
- Methods: K--means, PCA, ICA

Unsupervised Learning

Example: movie recommendation (the Netflix problem)

- Clustering can be used to put people into different groups based on the kinds of movies they like.

Interest Group 3:

Trainspotting
Fargo
Pulp Fiction
Clerks

Interest Group 18:

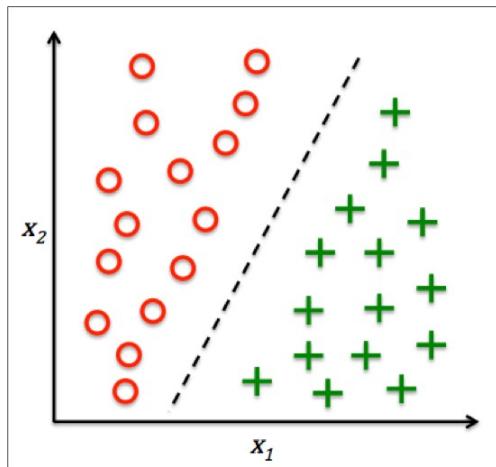
Mary Poppins
Cinderella
The Sound of Music
Dumbo

Interest Group 8:

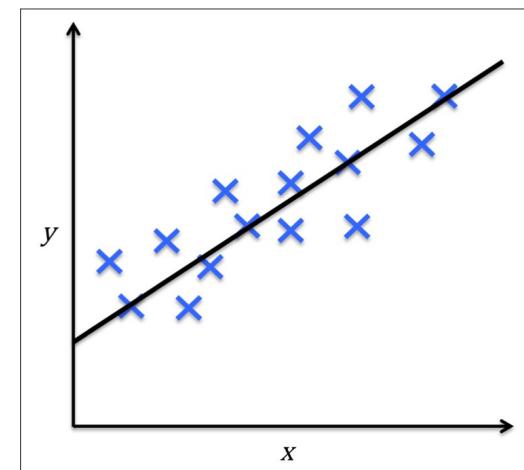
Pretty Woman
Mrs. Doubtfire
Ghost
Sleepless in Seattle

From Hoffman (2004) “Latent Semantic Models for Collaborative Filtering.”

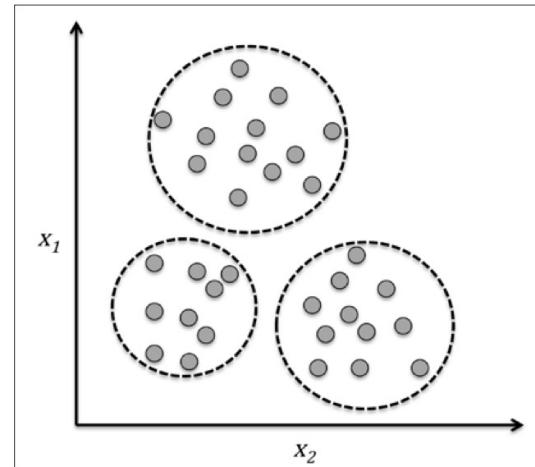
Classification



Regression

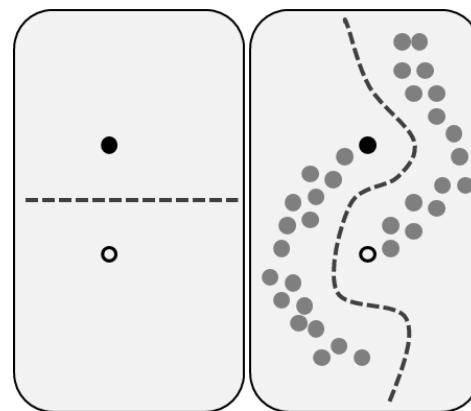


Clustering



Semi-supervised Learning

- A class of **supervised learning** tasks and techniques that also make use of unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data.
- Combines both kinds of learning



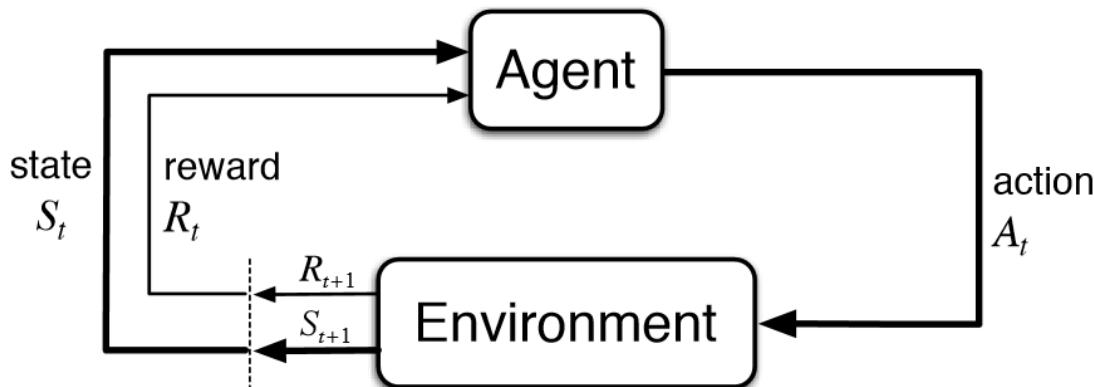
Types of Learning

- Supervised learning
 - Goal: Prediction
 - **Semi-Supervised Learning**
- Unsupervised learning
 - Goal: Discovery of Similarity or dissimilarity
 - Goal: Dimensionality Reduction
- **Reinforcement learning**
 - categorized as supervised learning as well

Reinforcement Learning

Setting:

- an **agent** interacts with an **environment**
- **actions** by the agent lead to different **states** of the environment
- some states will provide **rewards**

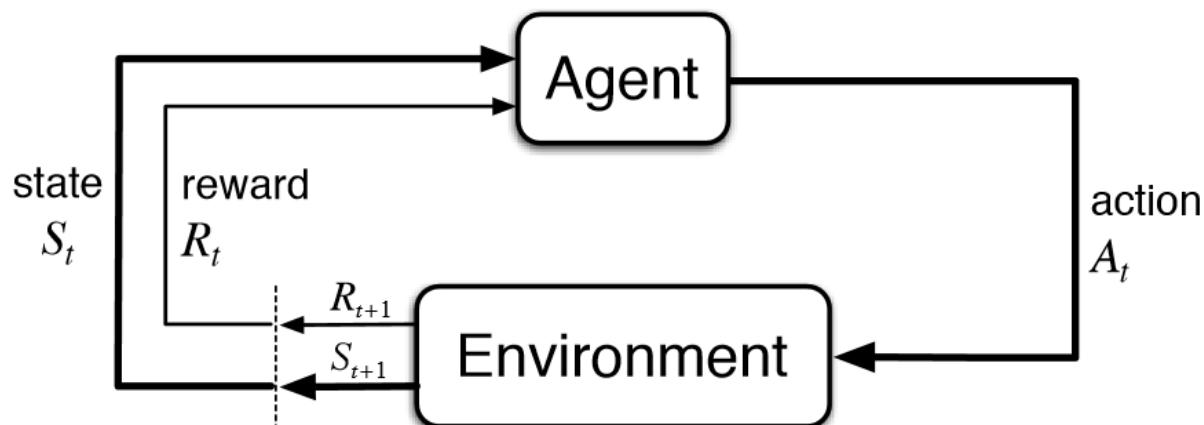


Reinforcement Learning

Setting:

Learning goal is to maximize rewards.

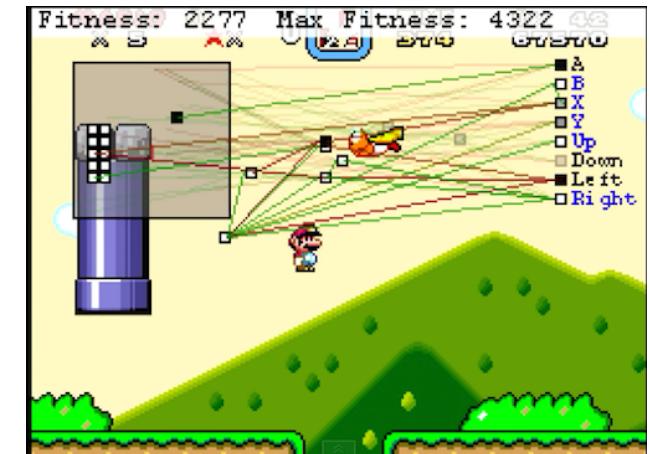
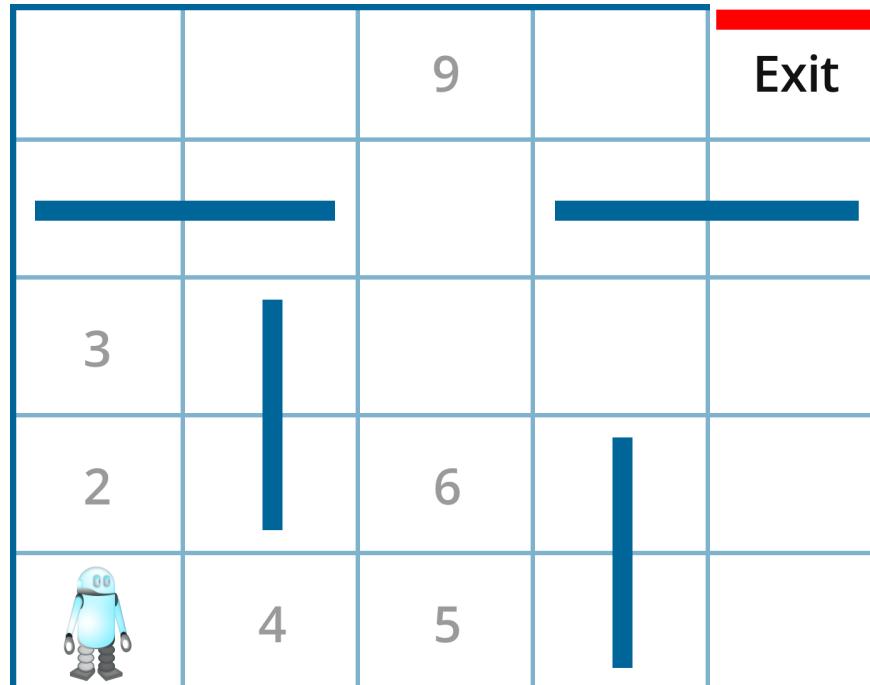
Used to learn models of how to behave,
more complex than just input→output



Reinforcement Learning

Most commonly used for designing autonomous robots and automated vehicles

Can also be trained to play games



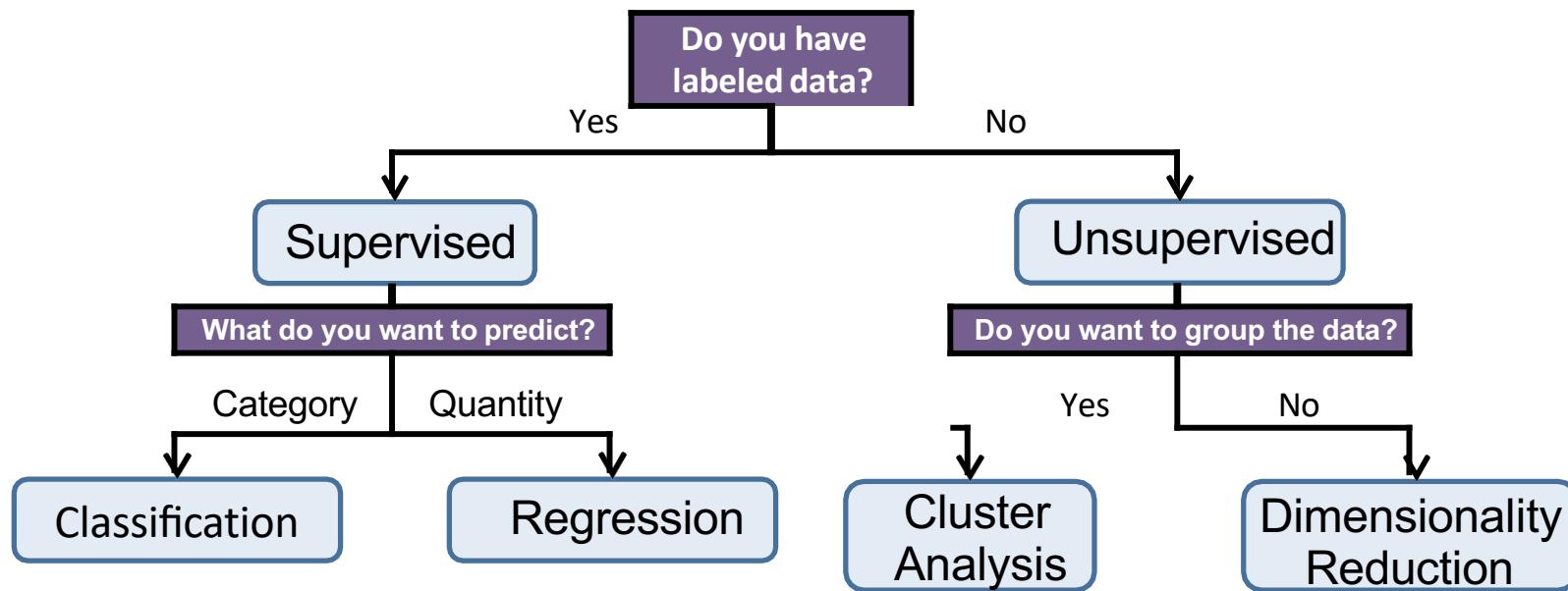
Reinforcement Learning

Some uses in more traditional machine learning tasks by creatively defining what the agent and environment are.

Example: Formulating classification as a reinforcement learning problem!

<https://users.cs.duke.edu/~parr/icml03.pdf>

Types of Learning Algorithms



Terminology

Each data point (i.e., each “thing” you are classifying/regressing/clustering) is called an **instance**

- Alternative name: **observation**
- Also called **examples** or **samples** when used as training data in supervised learning

In a data set, usually each row corresponds to an instance.

Terminology

The “input” variables are called **features**

- Alternative names: **attributes**, **covariates**, **predictors**
- Also referred to as the **independent** variables

In a data set, each column corresponds to a feature. (Except for the last column, which is the output.)

The list of feature values for an instance is called the instance’s **feature vector**

Terminology

The value of the “output” variable (the “thing” you are trying to predict) is the **label**

- Also called the **dependent** variable

In a data set, this is the final column. (Unless there is more than one label, which is a setting we will consider later in the course.)

In classification, the possible values the labels can have are called **classes**

Terminology

In supervised learning:

- a **training instance** is a feature vector paired with a label
- the **training data** (sometimes **labeled data**) is the table of all training instances

In unsupervised learning, the data set contains feature vectors but no labels (sometimes called **unlabeled data**)

Problem set-up: Formalization

- X : *input variables* (predictors, independent variables, or features)
- Y : *output variable* (response or dependent variable)
- *Statistical learning*: a set of approaches for estimating function f that describes the relationship between predictors and response:

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

Prediction and Inference

- *Prediction*: predict the output Y given inputs X using model \hat{f} an estimator for f
 - Prediction accuracy depends on
 - *Reducible error*: imperfect estimate for f
 - *Irreducible error*: ϵ , error not predicted by the inputs
- *Inference*: understand relationship between Y and individual predictors, X_i
 - In inference, we do not want our model to be a black box

Advertising Example

- A company can not directly control sales of product Z , but can control advertising strategy
- Data: sales and advertising budgets for three media (TV, radio, newspaper)
- Shown are **Sales** vs **TV**, **Radio** and **Newspaper**, with a blue linear-regression line fit separately to each.

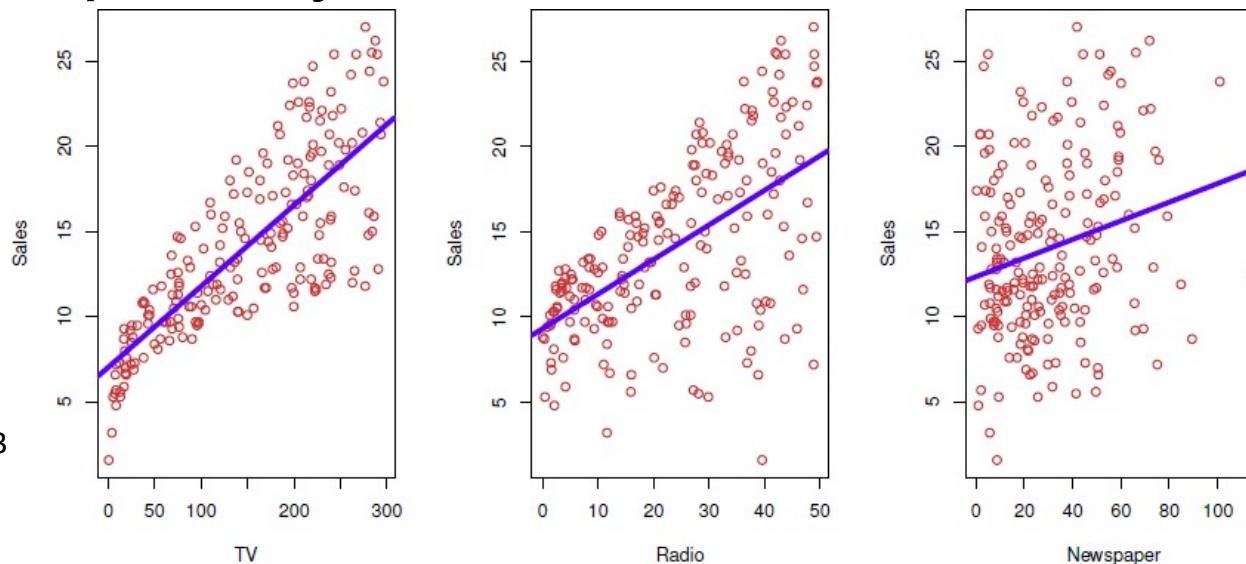


Figure 2.1 , ISL 2013

- In the advertising example, what are the input and output variables?
- Give an example of a prediction question and an inference question that we could attempt to answer with this data.

- In the advertising example, what are the input and output variables?
 - *Output variable*: number of sales
 - *Input variables*: TV advertising budget, Radio advertising budget, Newspaper advertising budget

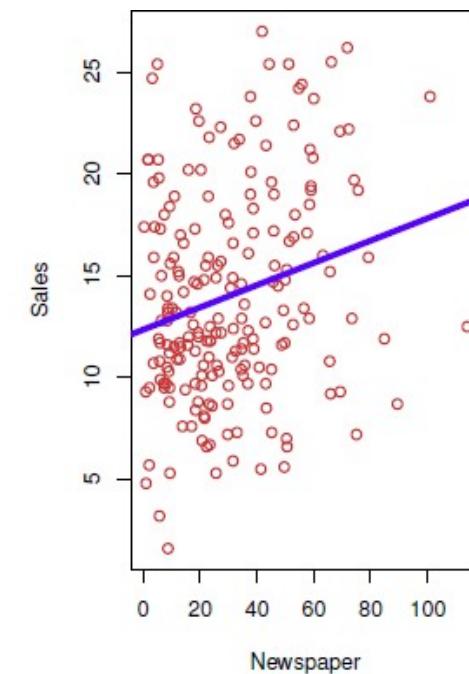
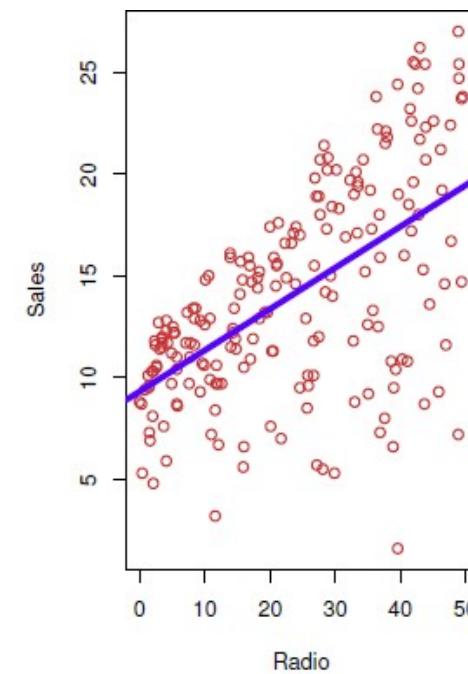
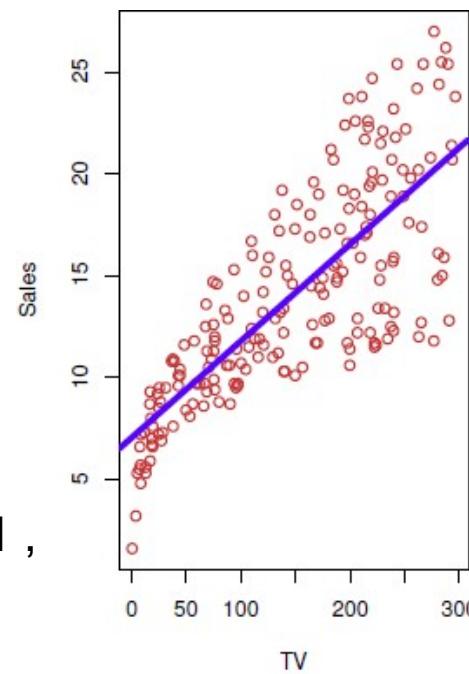
- Give an example of a prediction question and an inference question that we could attempt to answer with this data.
 - Prediction:
 - What are the expected sales in market Z, given its TV, radio and newspaper budgets?
 - Inference:
 - How much is the increase in sales associated with a 10% increase in the TV advertising budget?
 - Which media (TV, radio, newspaper) generates the largest boost in sales?

Advertising Example

Can we predict **Sales** using these three? Perhaps we can do better using a model

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

Figure 2.1 ,
ISL 2013



Notation

Here **Sales** is a *response* or *target* that we wish to predict. We generically refer to the response as Y . **TV** is a *feature*, or *input*, or *predictor*; we name it X_1 . Likewise name **Radio** as X_2 , and so on. We can refer to the *input vector* collectively as

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

Now we write our model as

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

where ε captures measurement errors and other discrepancies.

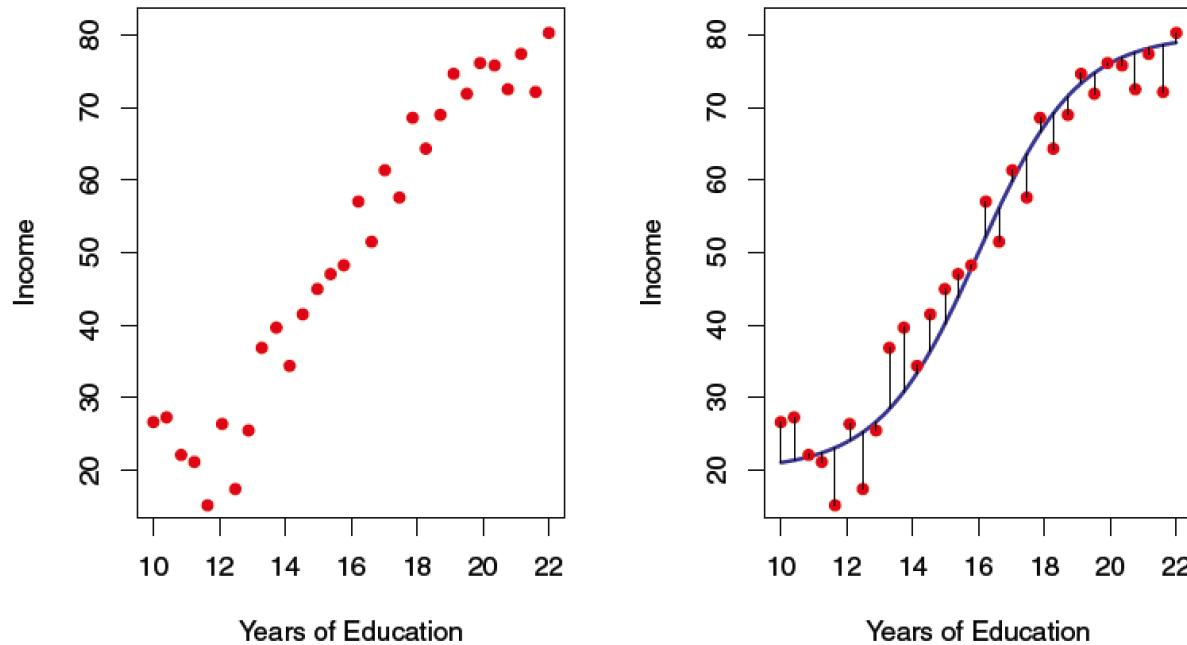
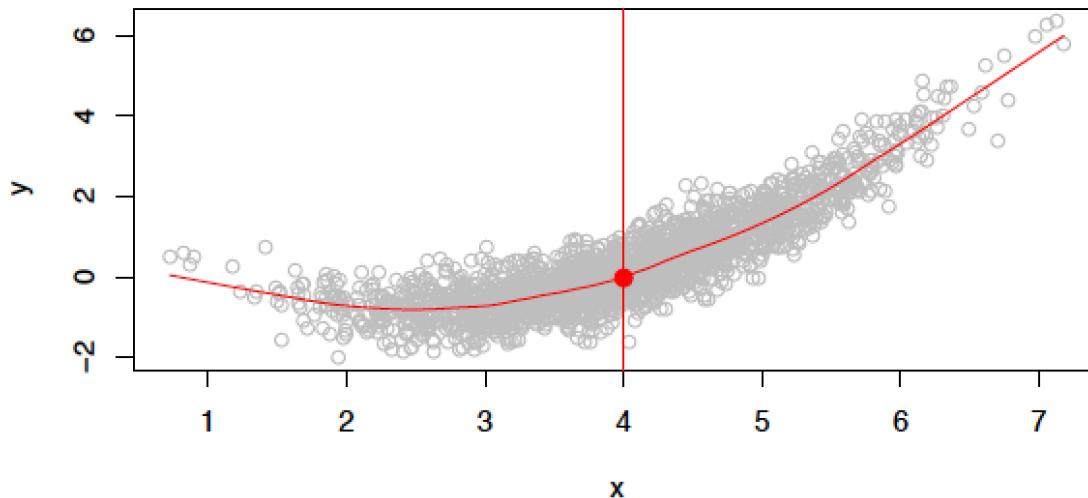


FIGURE 2.2. The `Income` data set. Left: The red dots are the observed values of `income` (in tens of thousands of dollars) and `years of education` for 30 individuals. Right: The blue curve represents the true underlying relationship between `income` and `years of education`, which is generally unknown (but is known in this case because the data were simulated). The black lines represent the error associated with each observation. Note that some errors are positive (if an observation lies above the blue curve) and some are negative (if an observation lies below the curve). Overall, these errors have approximately mean zero.

What is $f(X)$ good for?

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



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

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

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

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

The regression function $f(x)$

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

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

- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y | X = x)$ is the function that minimizes $E[(Y - g(X))^2 | X = x]$ over all functions g at all points $X = x$.

The regression function $f(x)$

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

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

- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y | X = x)$ is the function that minimizes $E[(Y - g(X))^2 | X = x]$ over all functions g at all points $X = x$.

The regression function $f(x)$

- $\varepsilon = Y - f(x)$ is the *irreducible* error — i.e. even if we knew $f(x)$, we would still make errors in prediction, since at each $X = x$ there is typically a distribution of possible Y values.
- For any estimate $\hat{f}(x)$ of $f(x)$, we have

$$E[(Y - \hat{Y})^2 | X = x] = E[(f(X) + \varepsilon - \hat{f}(X))^2 | X = x]$$

=

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

The regression function $f(x)$

- In this course, we study methods to decrease the **reducible** error $f(x) - \hat{f}(x)$.

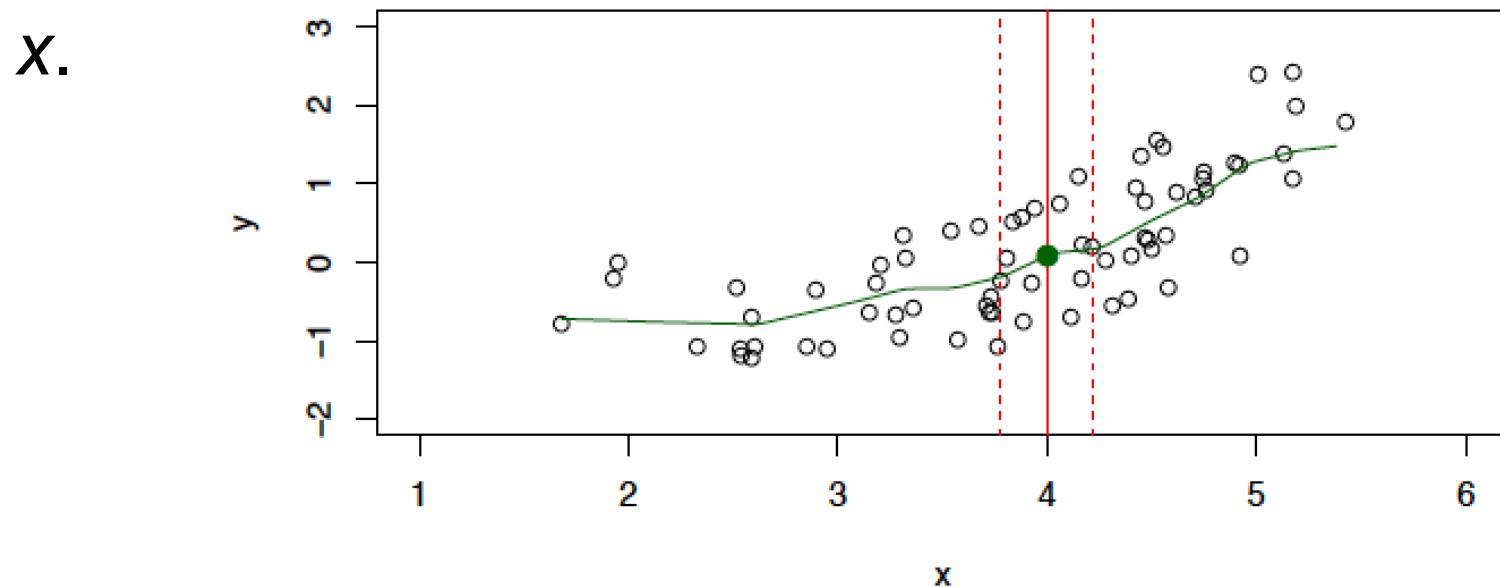
$$E[(Y - \hat{f}(X))^2 | X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible}}$$

How to estimate f

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

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

where $N(x)$ is some *neighborhood* of



Nearest Neighborhood

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

Nearest Neighborhood

- Nearest neighbor methods can be lousy when p is large. Reason: **the curse of dimensionality.**
- Nearest neighbors tend to be far away in high dimensions.
- NN is a *non-parametric* Method

Nearest Neighborhood

- We need to get a reasonable fraction of the N values of y_i to average to bring the variance down—e.g. 10%.
- A 10% neighborhood in high dimensions need no longer be local, so we lose the spirit of estimating $E(Y | X = x)$ by local averaging.

Parametric and structured models

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

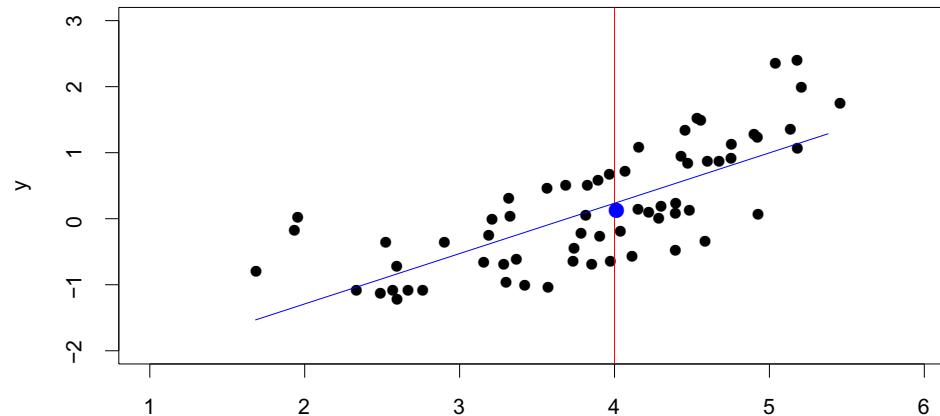
$$f_L(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p.$$

- A linear model is specified in terms of $p+1$ parameters $\beta_0, \beta_1, \dots, \beta_p$.

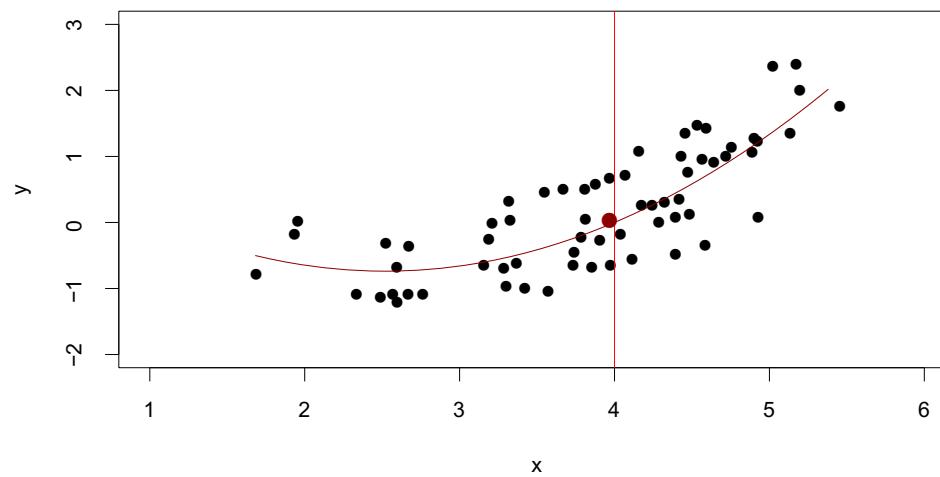
Parametric and structured models

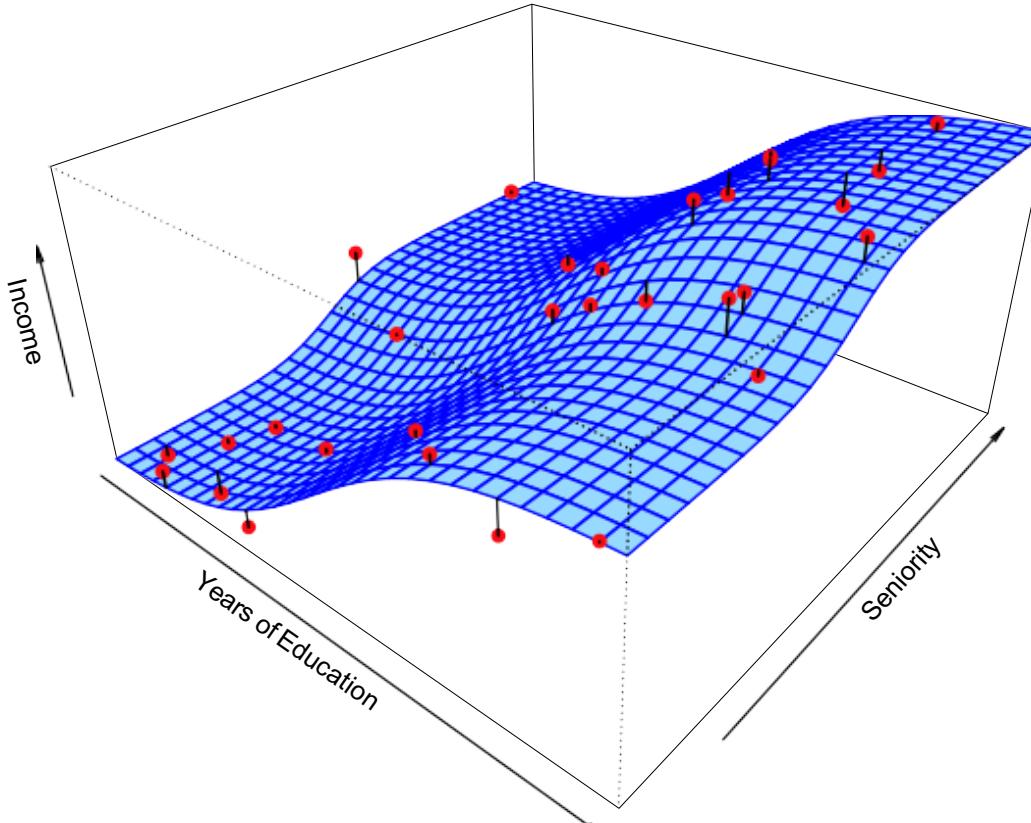
- We estimate the parameters by fitting the model to training data.
- Although it is *almost never correct*, a linear model often serves as a good and interpretable approximation to the unknown true function $f(X)$.

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



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

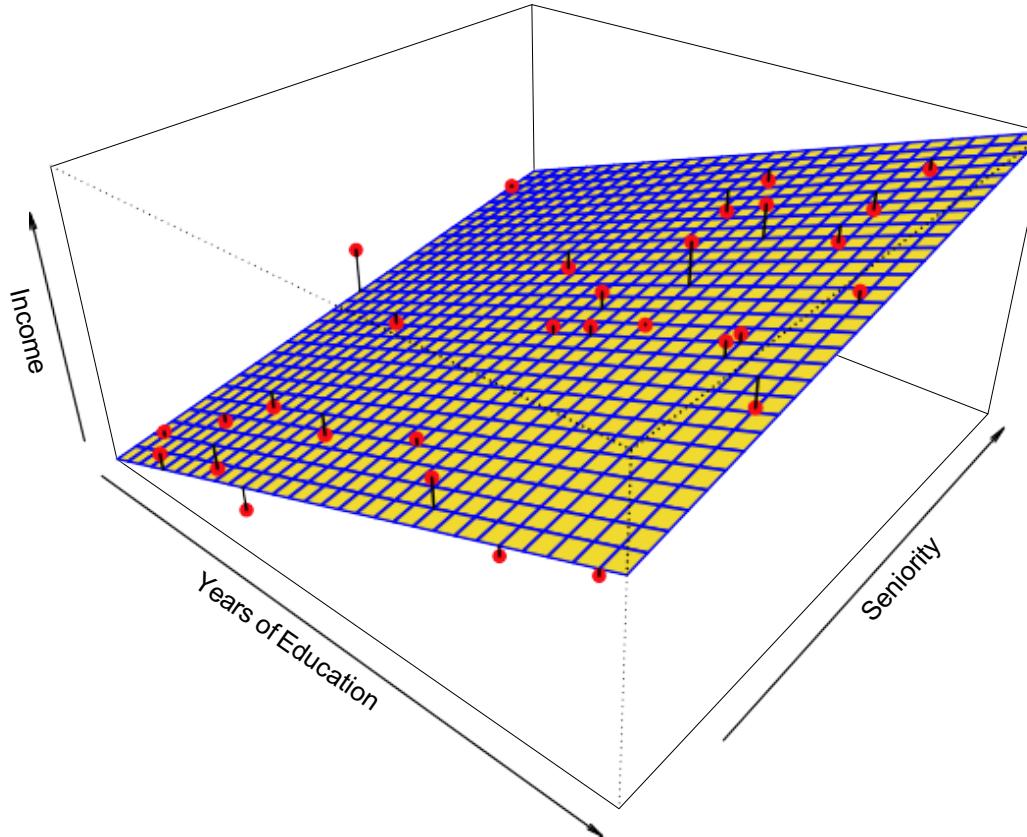




Simulated example. Red points are simulated values for income from the model

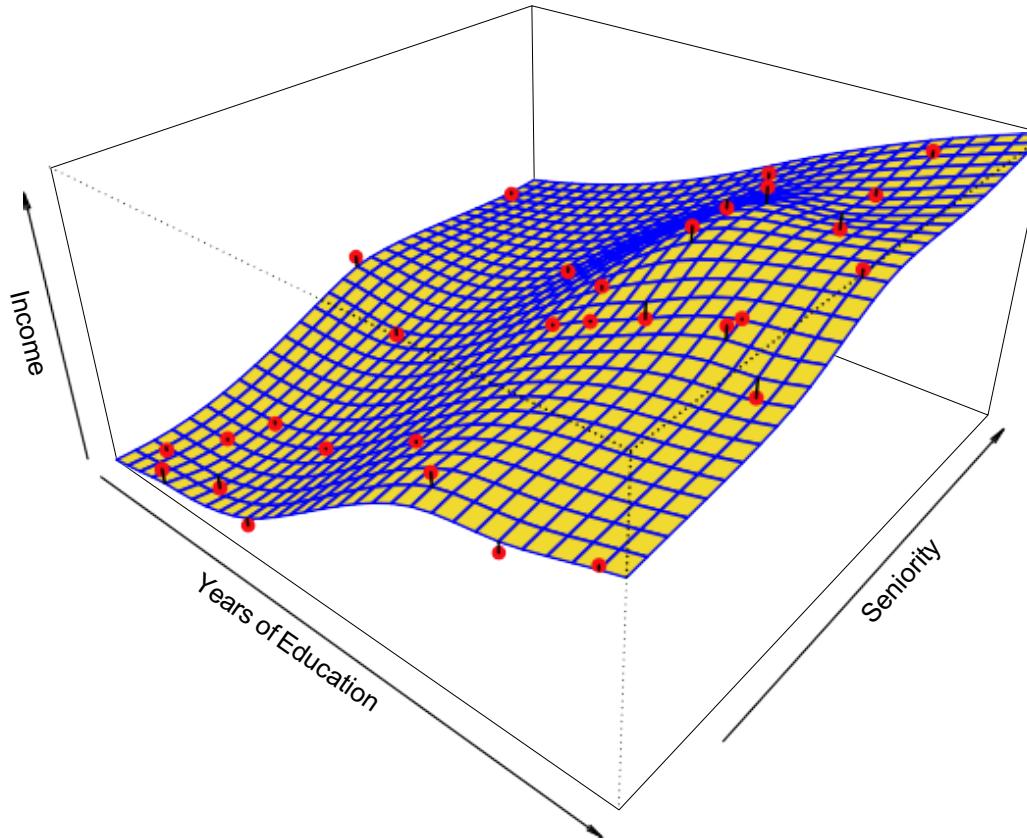
$$\text{income} = f(\text{education}, \text{seniority}) + \epsilon$$

f is the blue surface.

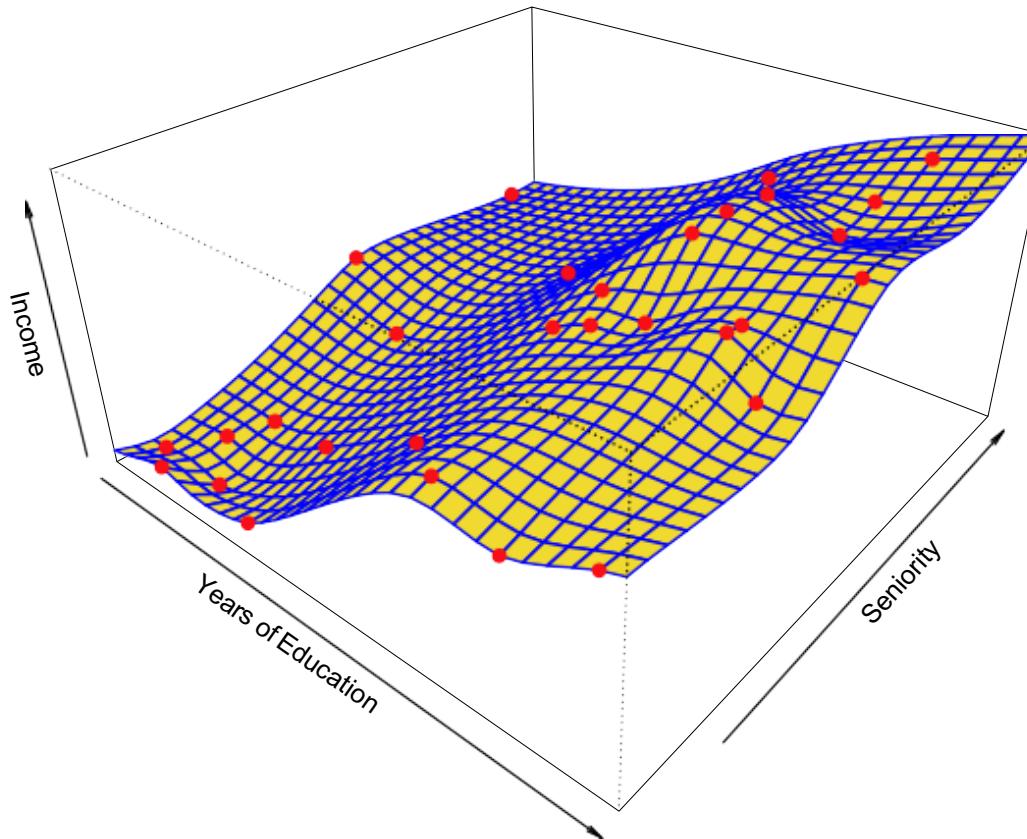


Linear regression model fit to the simulated data.

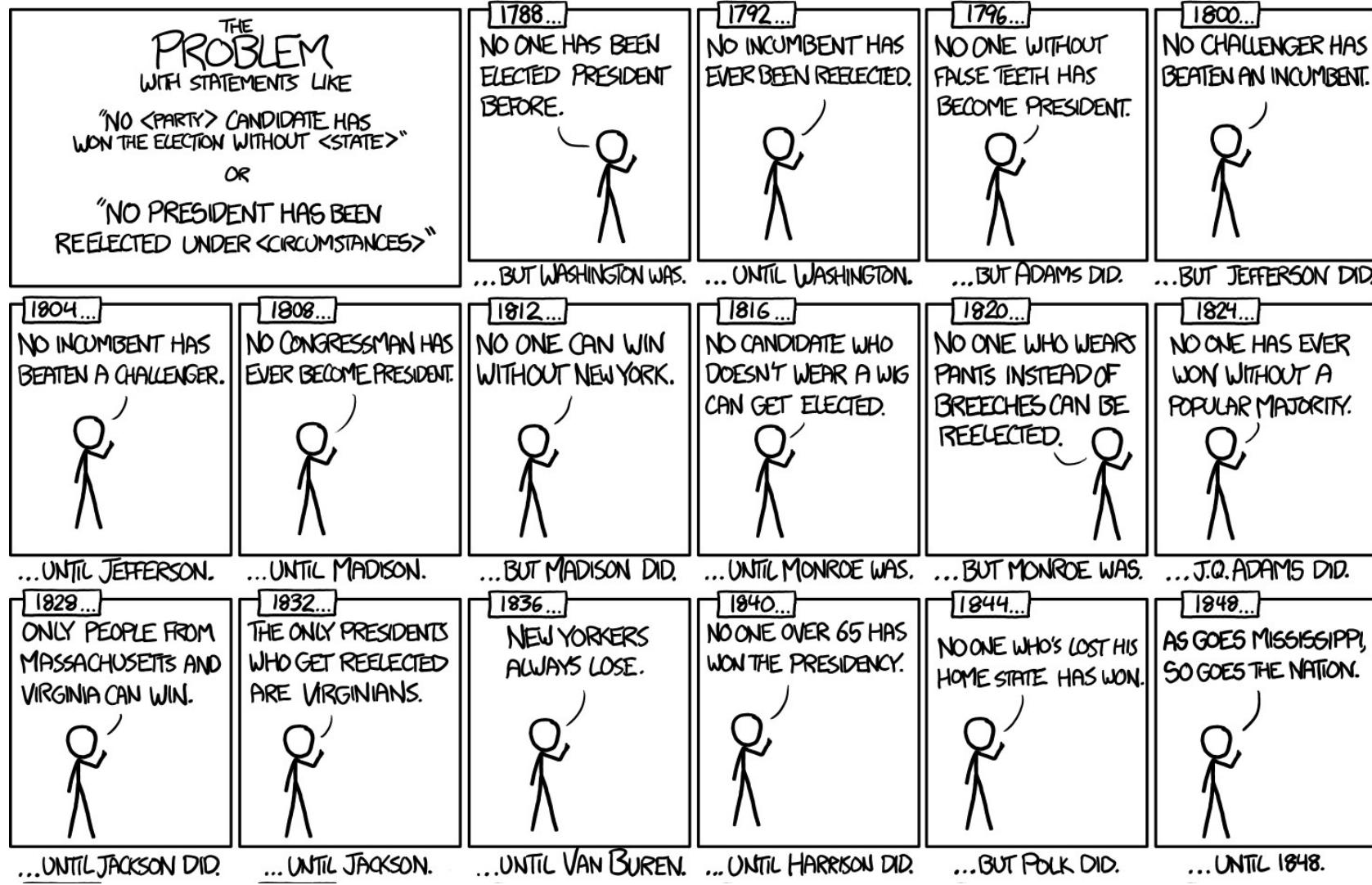
$$\hat{f}_L(\text{education, seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \text{education} + \hat{\beta}_2 \times \text{seniority}$$



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



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



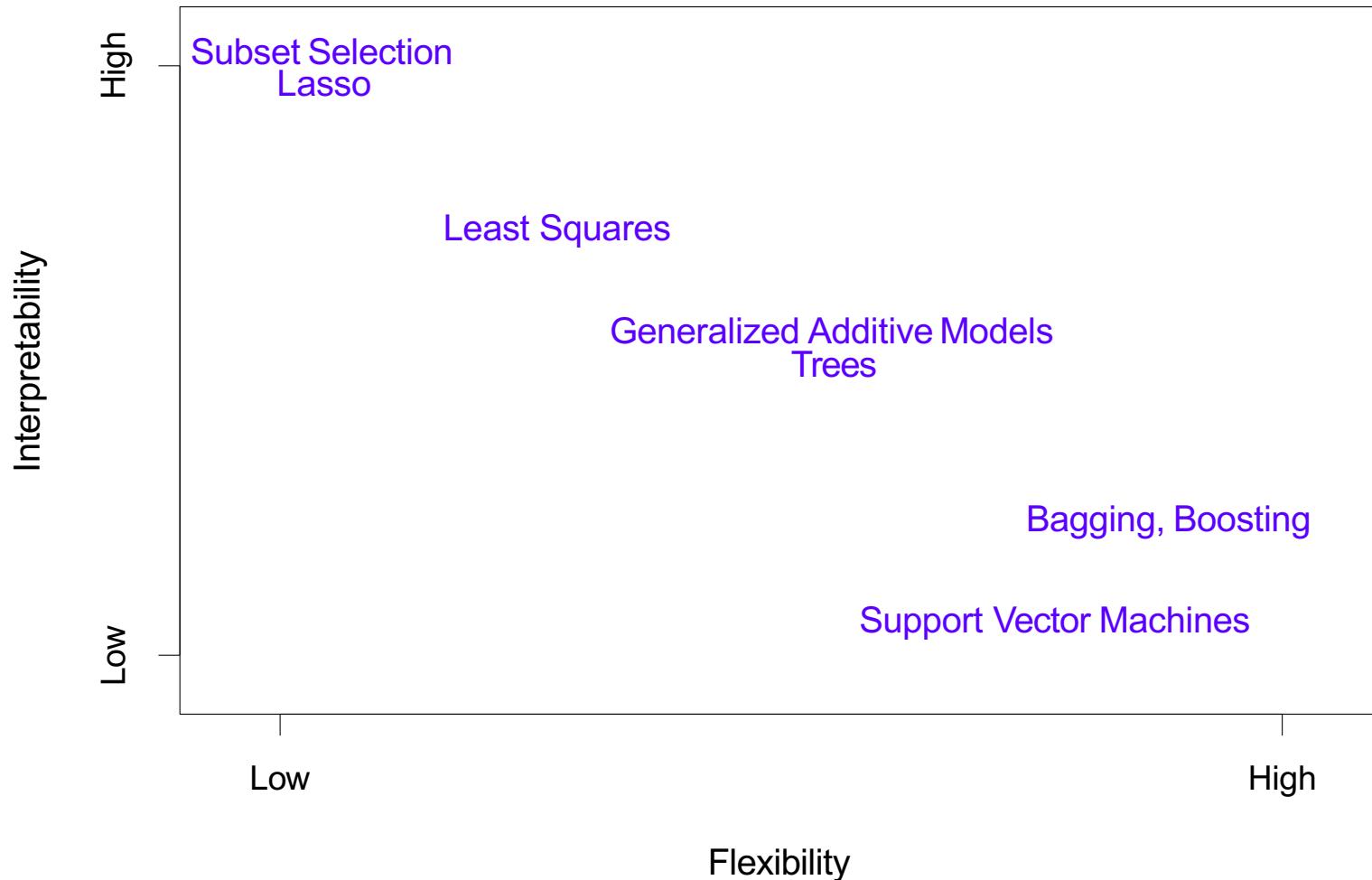
From: <https://xkcd.com/1122/>

Some trade-offs

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

Some trade-offs

- Parsimony versus black-box.
 - We often prefer a simpler model involving fewer variables over a black-box predictor involving them all.



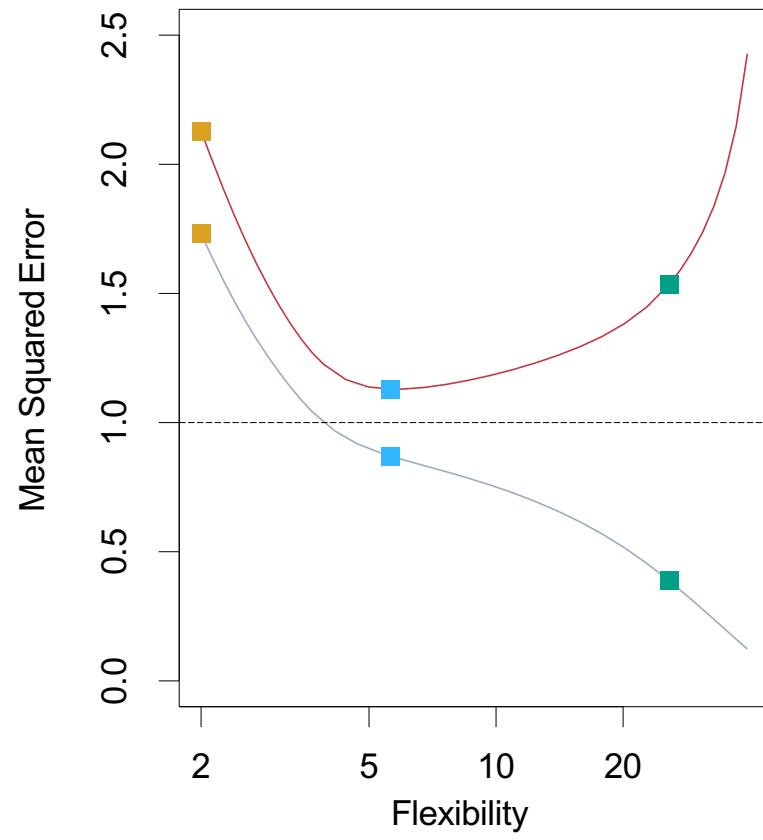
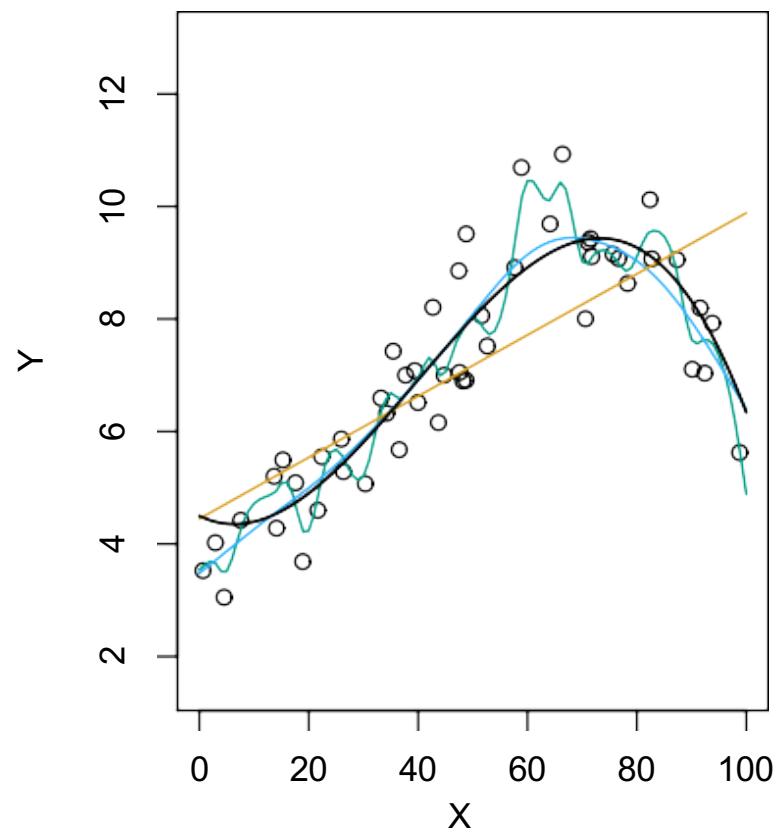
Assessing Model Accuracy

- Suppose we fit a model $\hat{f}(x)$ to some training data $T_r = \{x_i, y_i\}_{i=1}^N$, and we wish to see how well it performs.
- We could compute the average squared prediction error over T_r :

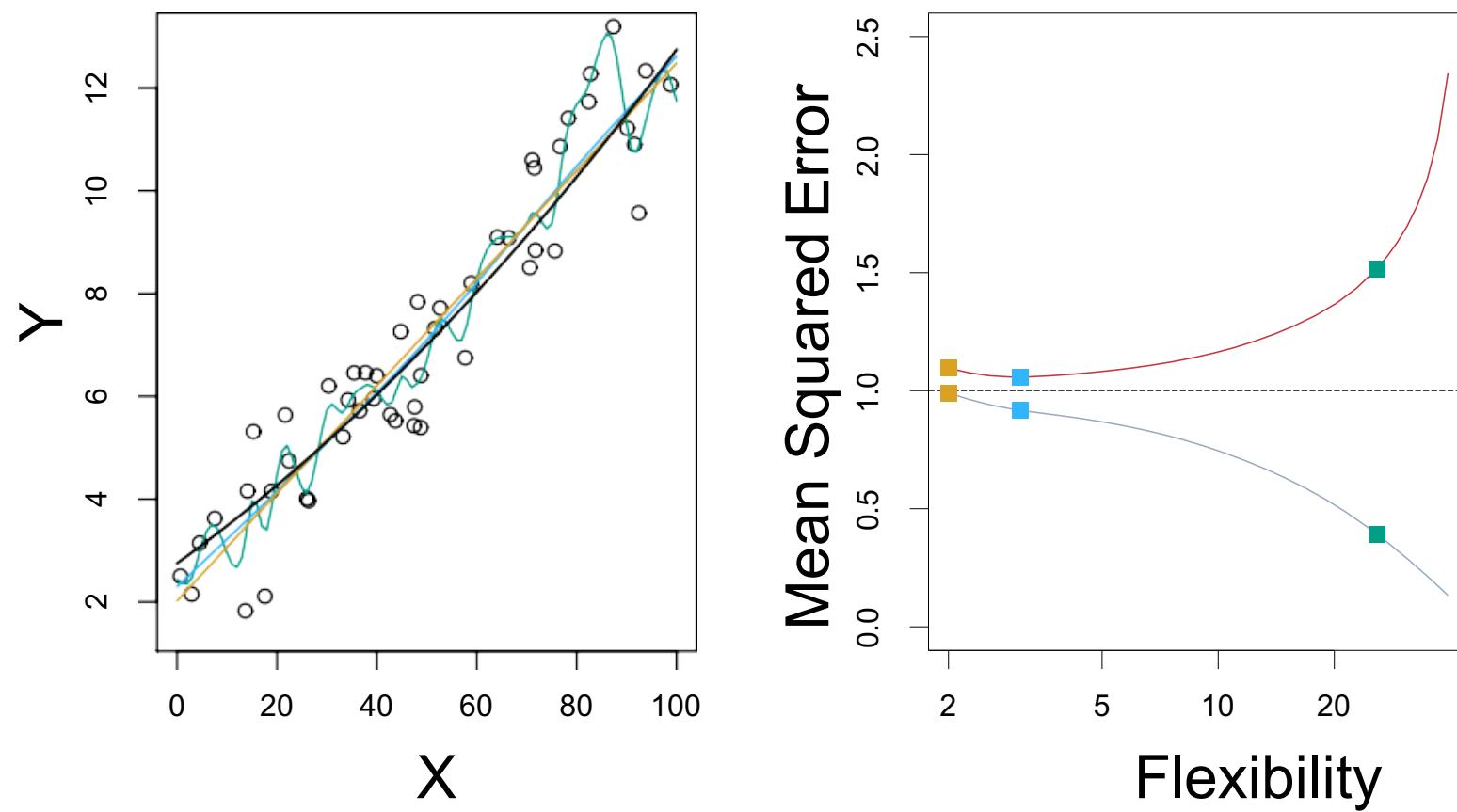
$$MSE_{Tr} = \text{Ave}_{i \in Tr} [y_i - \hat{f}(x_i)]^2$$

- This may be biased toward more overfit models.
- Instead we should, if possible, compute it using fresh *test* data $T_e = \{x_i, y_i\}_{i=1}^M$:

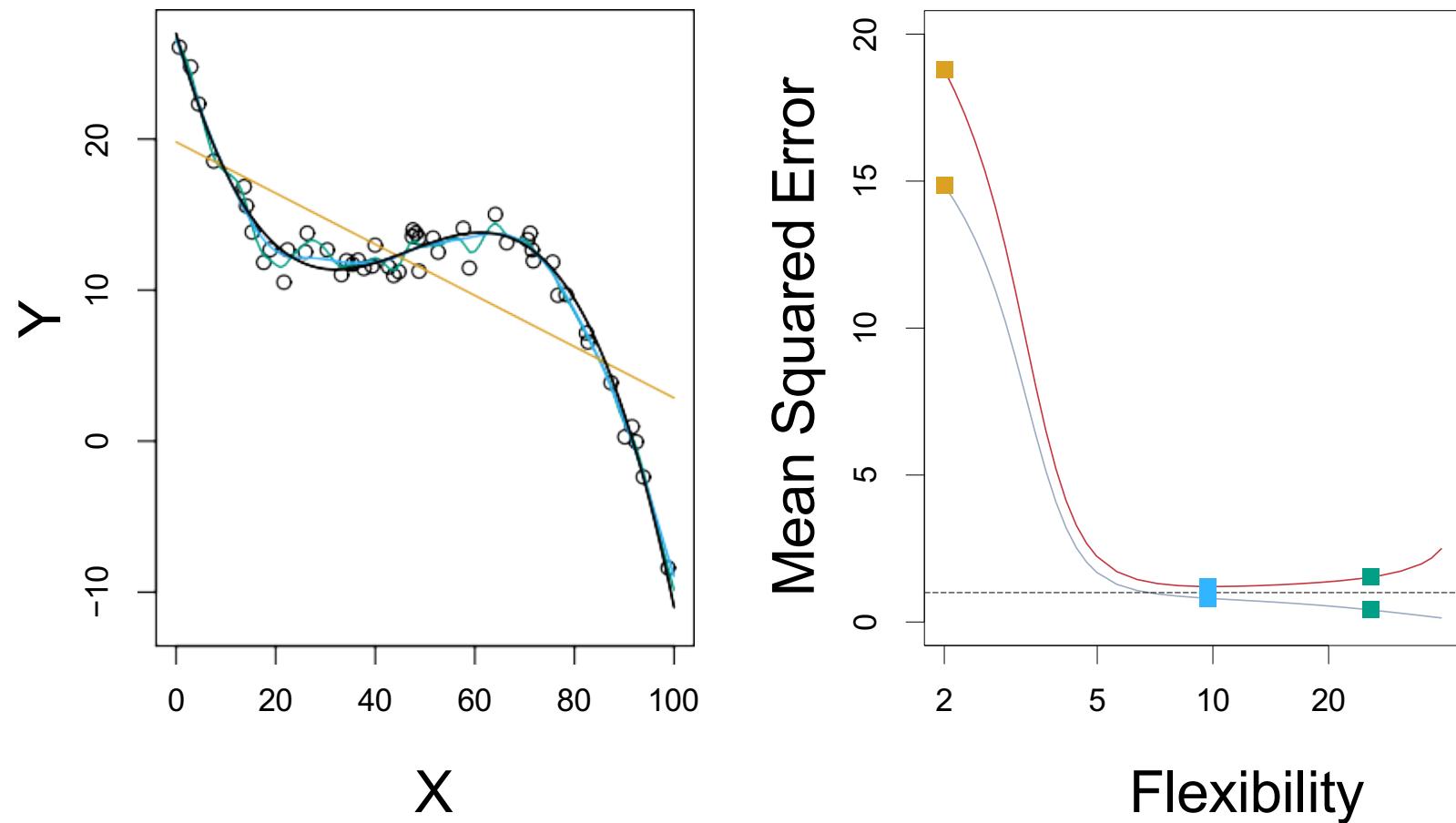
$$MSE_{Te} = \text{Ave}_{i \in Te} [y_i - \hat{f}(x_i)]^2$$



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



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



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

Bias-Variance Trade-off

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

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

Bias-Variance Trade-off

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

The expectation averages over the variability of y_0 as well as the variability in ϵ . Note that

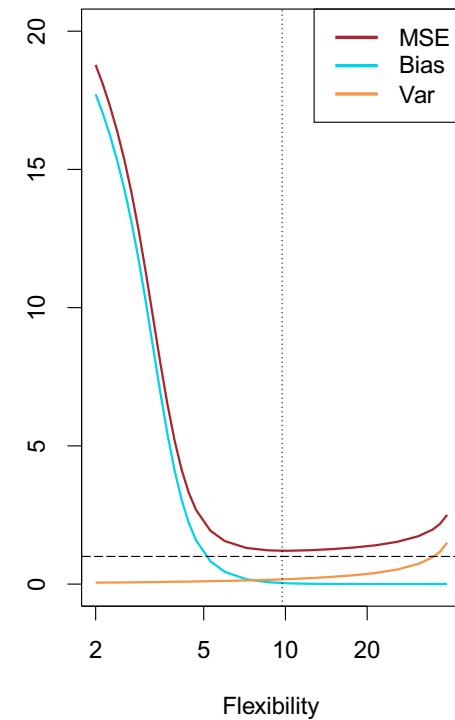
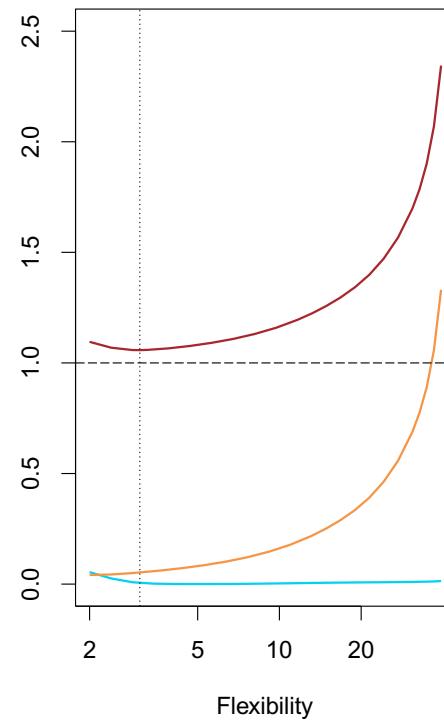
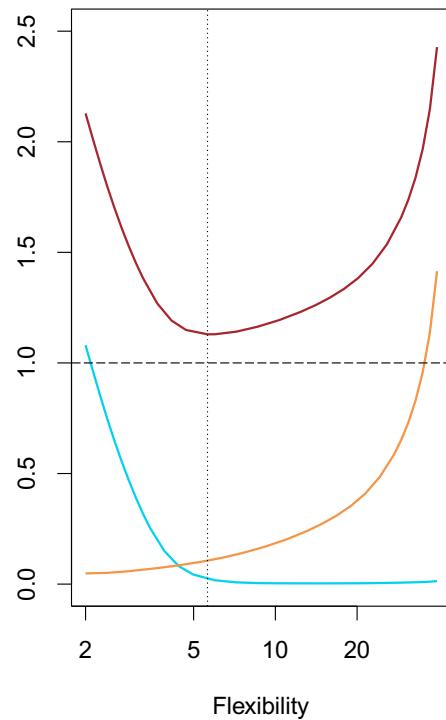
$$\text{Bias}[\hat{f}(x_0)] = E[\hat{f}(x_0)] - f(x_0).$$

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

Bias-Variance Trade-off

Bias-Variance Trade-off

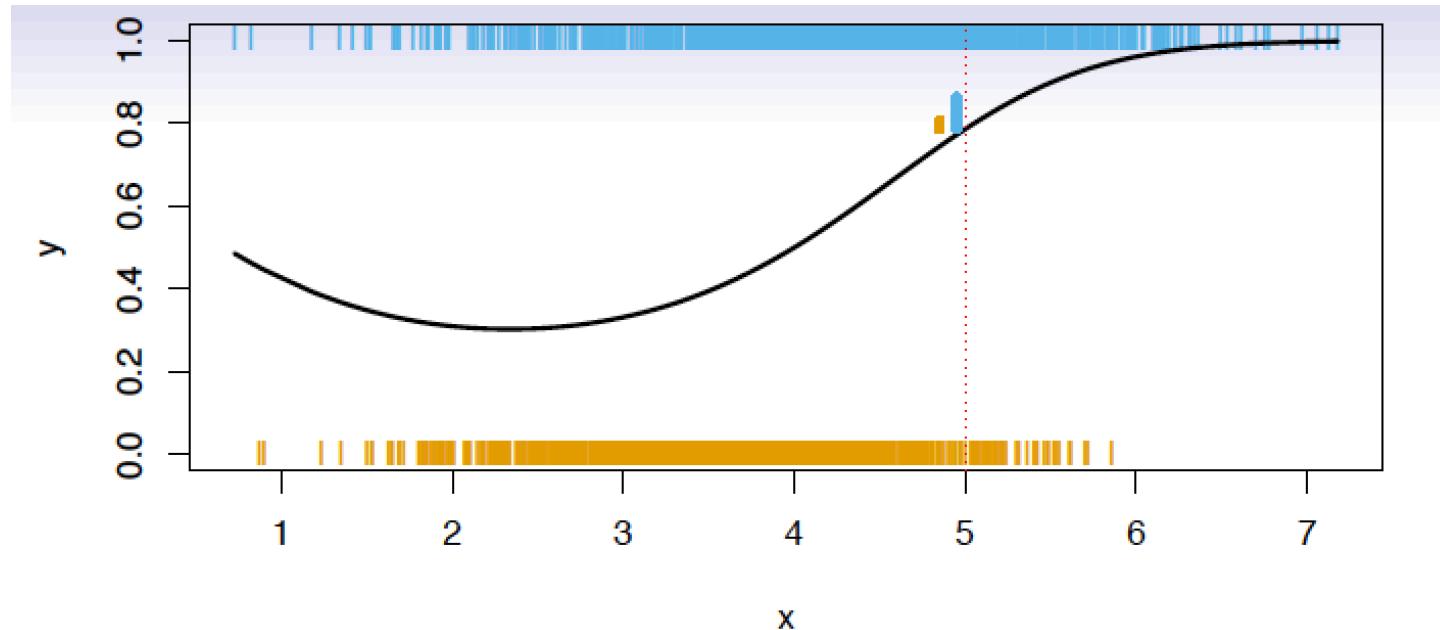
Bias-variance trade-off for the three examples



Classification Problems

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

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

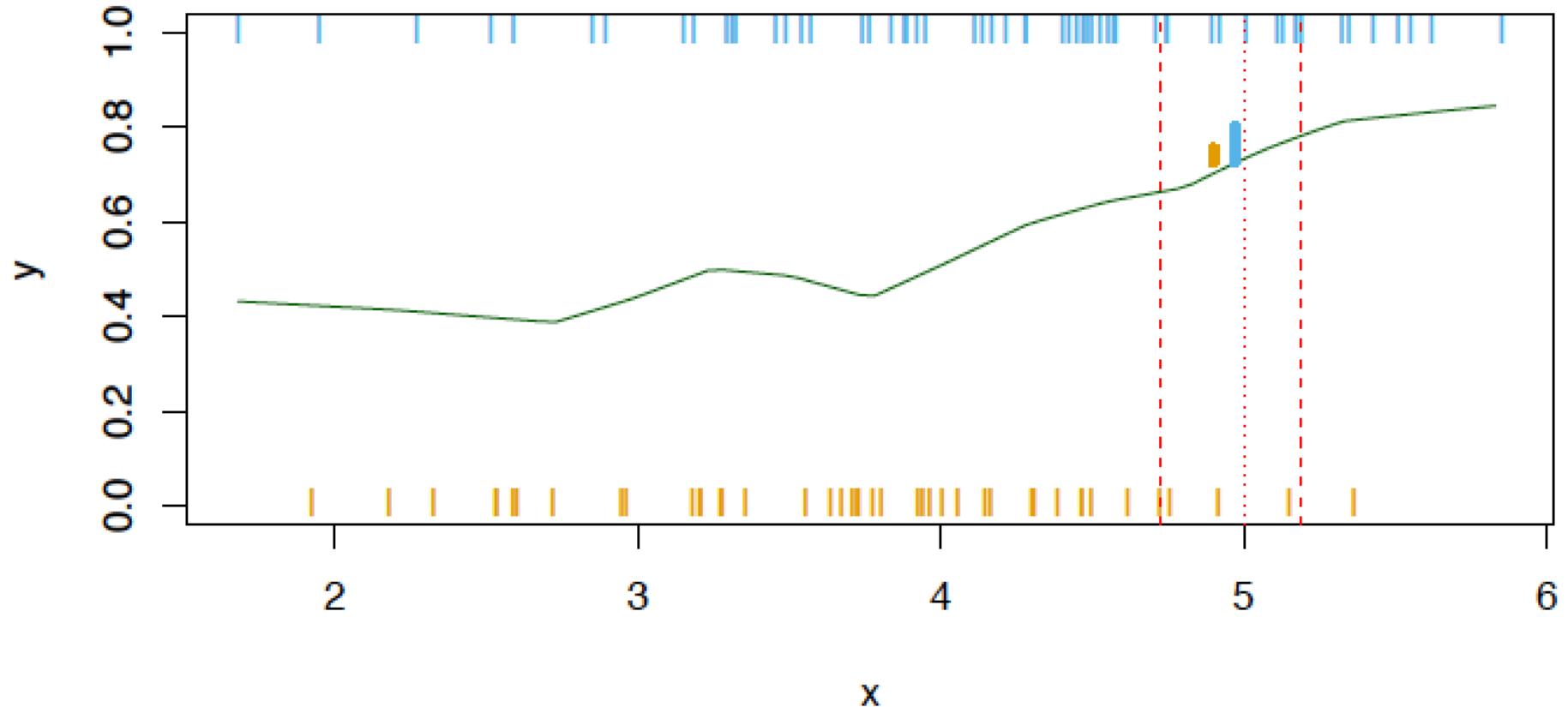


Is there an ideal $C(X)$? Suppose the K elements in C are numbered $1, 2, \dots, K$. Let

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

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

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



Nearest-neighbor averaging can be used as before.

Also breaks down as dimension grows.

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

Classification: some details

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

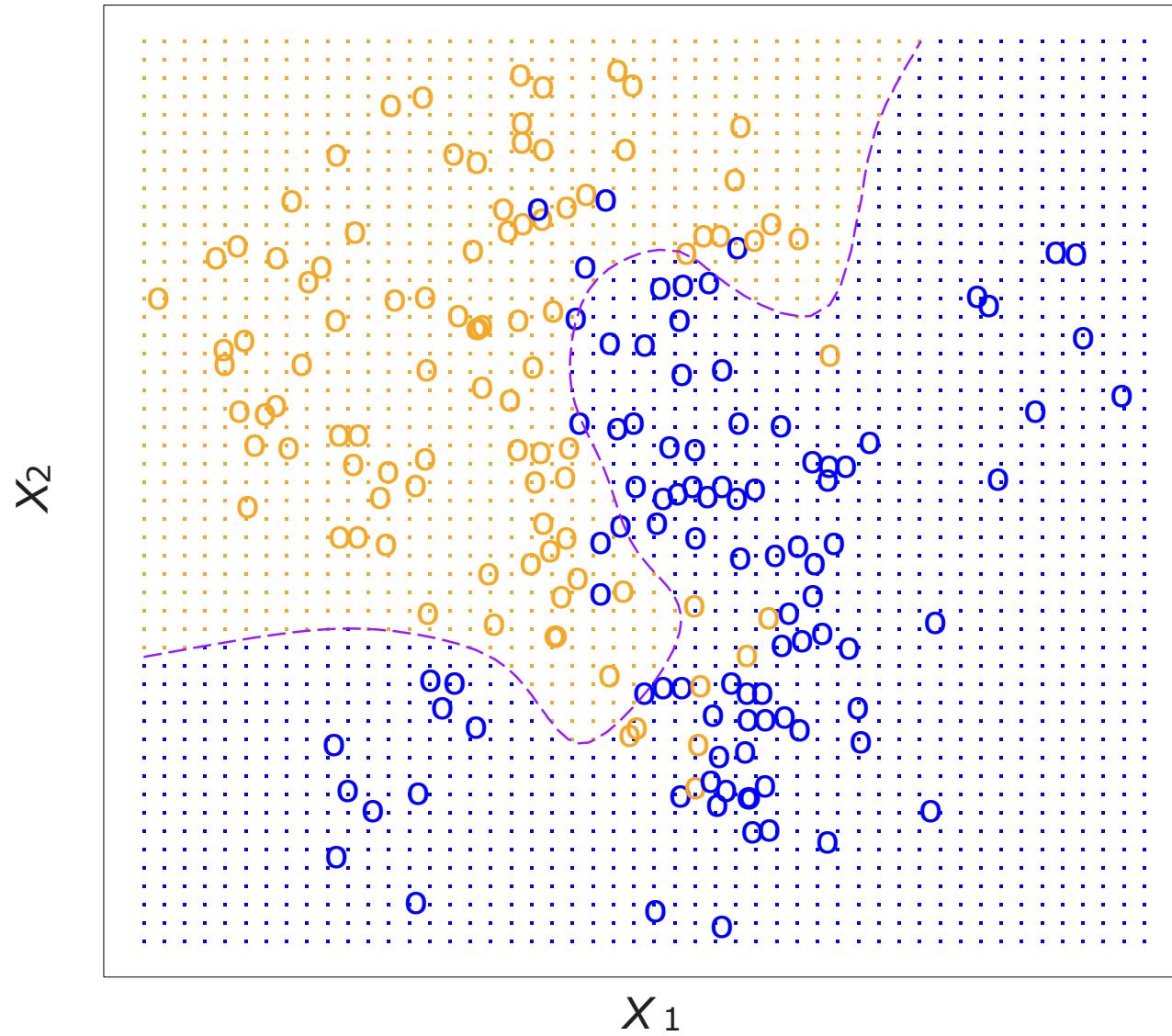
$$\text{Err}_{\text{Te}} = \text{Ave}_{i \in \text{Te}} \mathbf{I}[y_i \neq \hat{C}(x_i)]$$

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

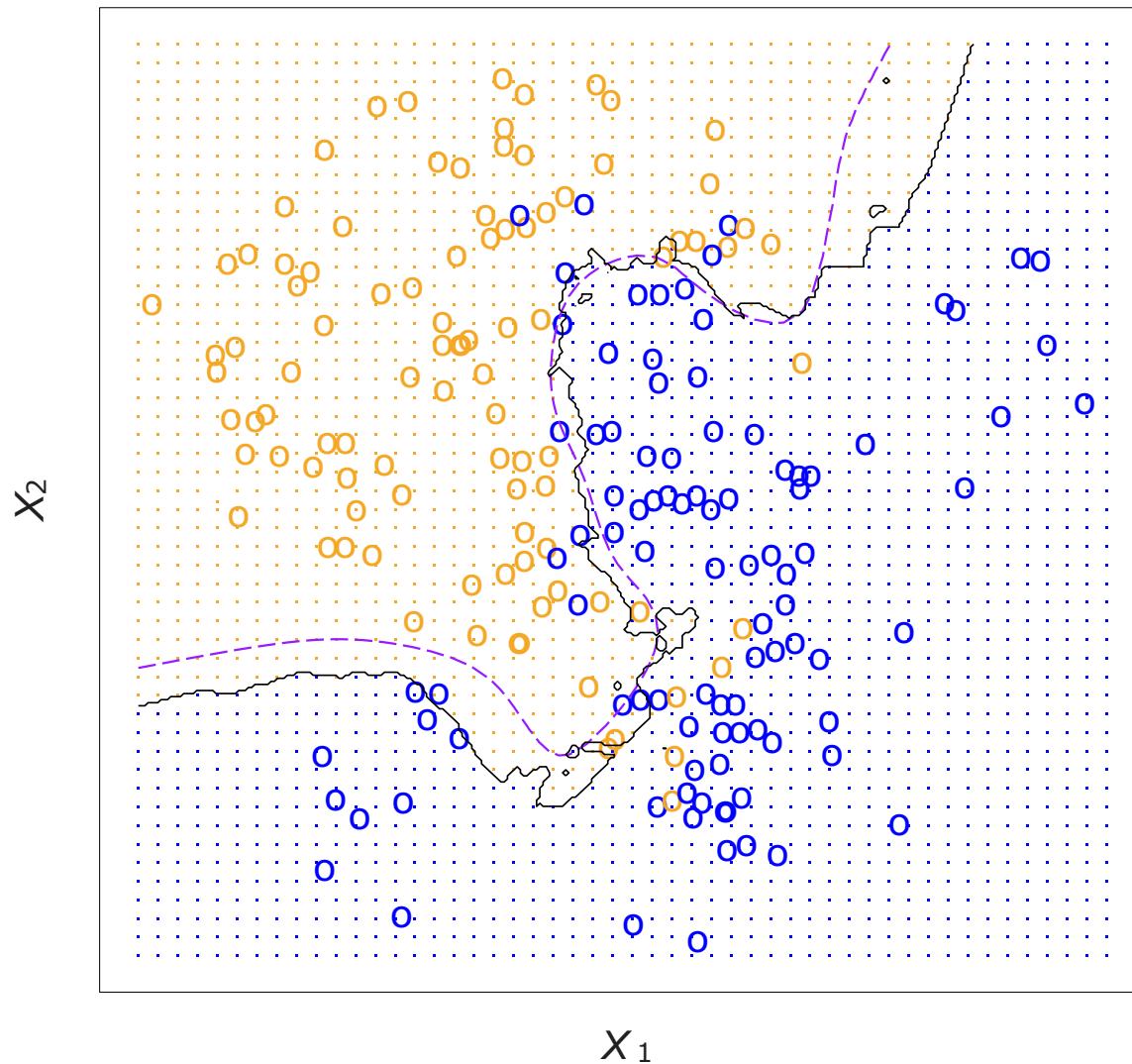
Classification: some details

- Support-vector machines build structured models for $C(x)$.
- We will also build structured models for representing the $p_k(x)$.
e.g. Logistic regression,
generalized additive models.

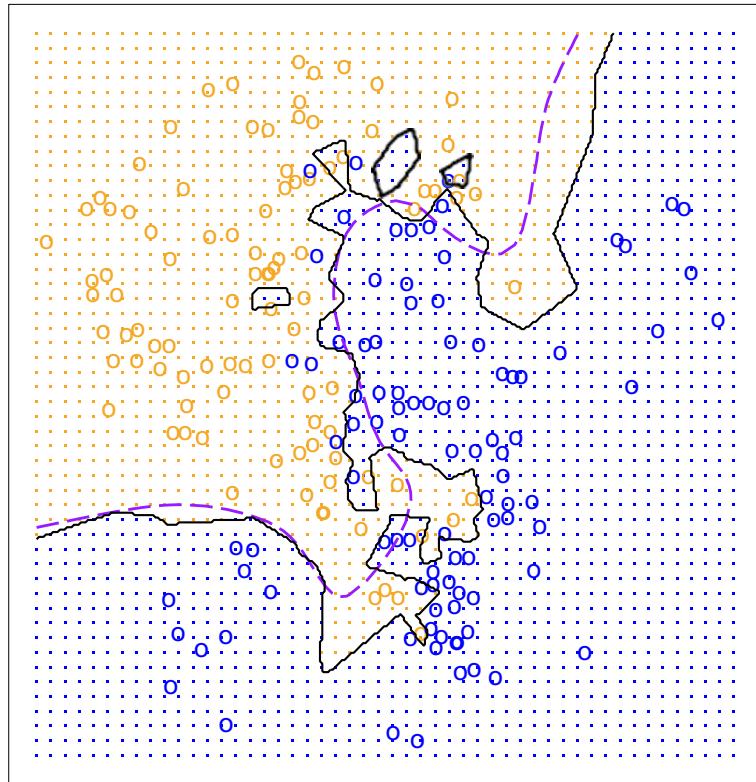
Example: K-nearest neighbors in two dimensions



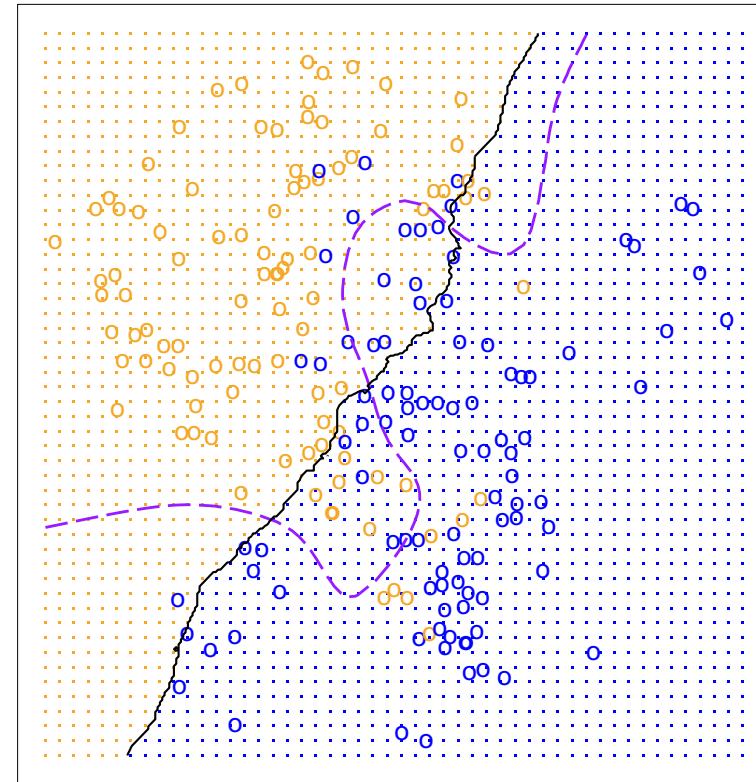
KNN: K=10



KNN: K=1

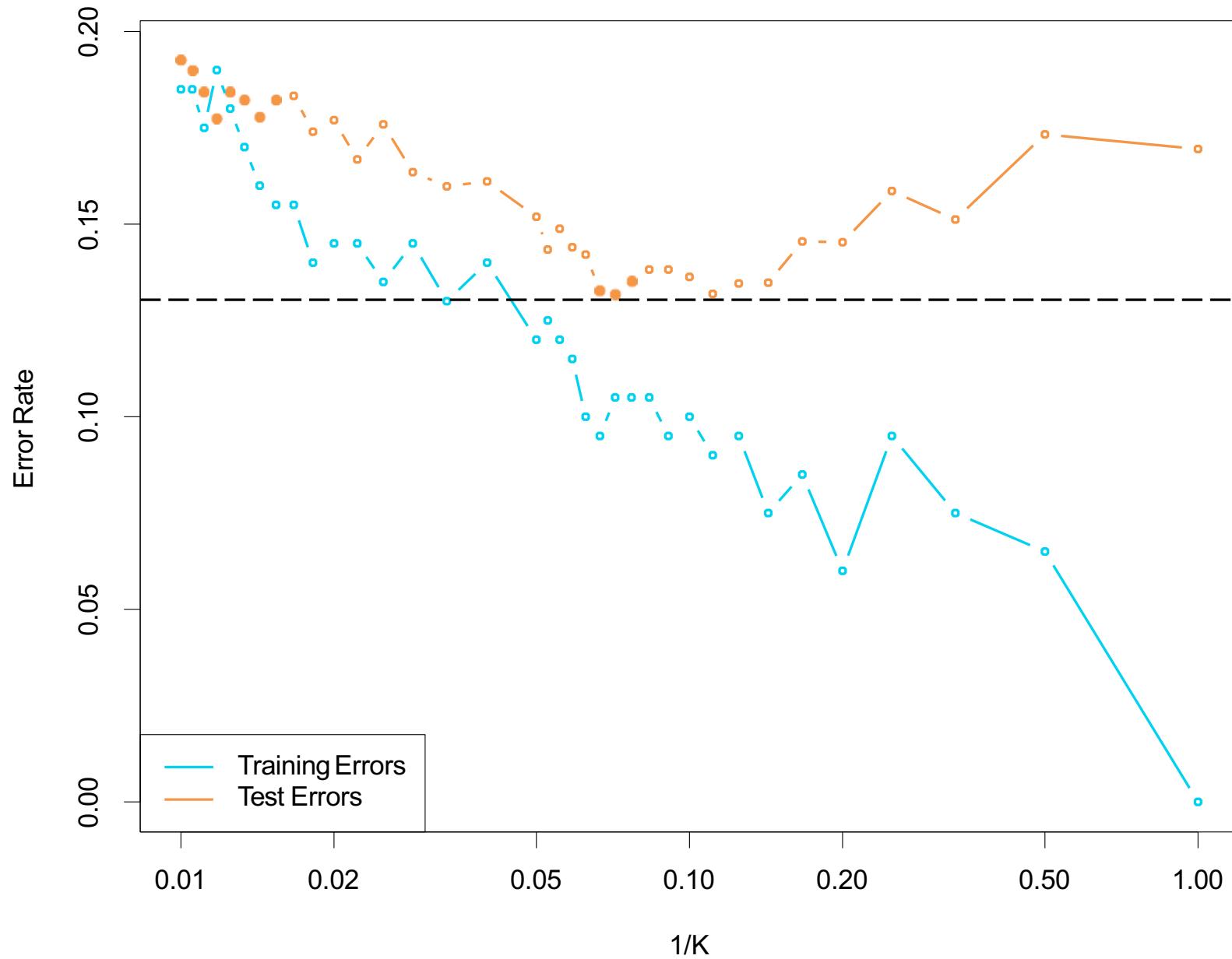


KNN: K=100



K-Nearest Neighbor classifier (KNN)

- Choice of K
 - K small → high variance and low bias
 - Large K → low variance and high bias



K-Nearest Neighbor classifier (KNN)

- Advantages:
 - Simple to implement
 - Few tuning parameters (K , distance metric)
 - Flexible, classes do not have to be linearly separable

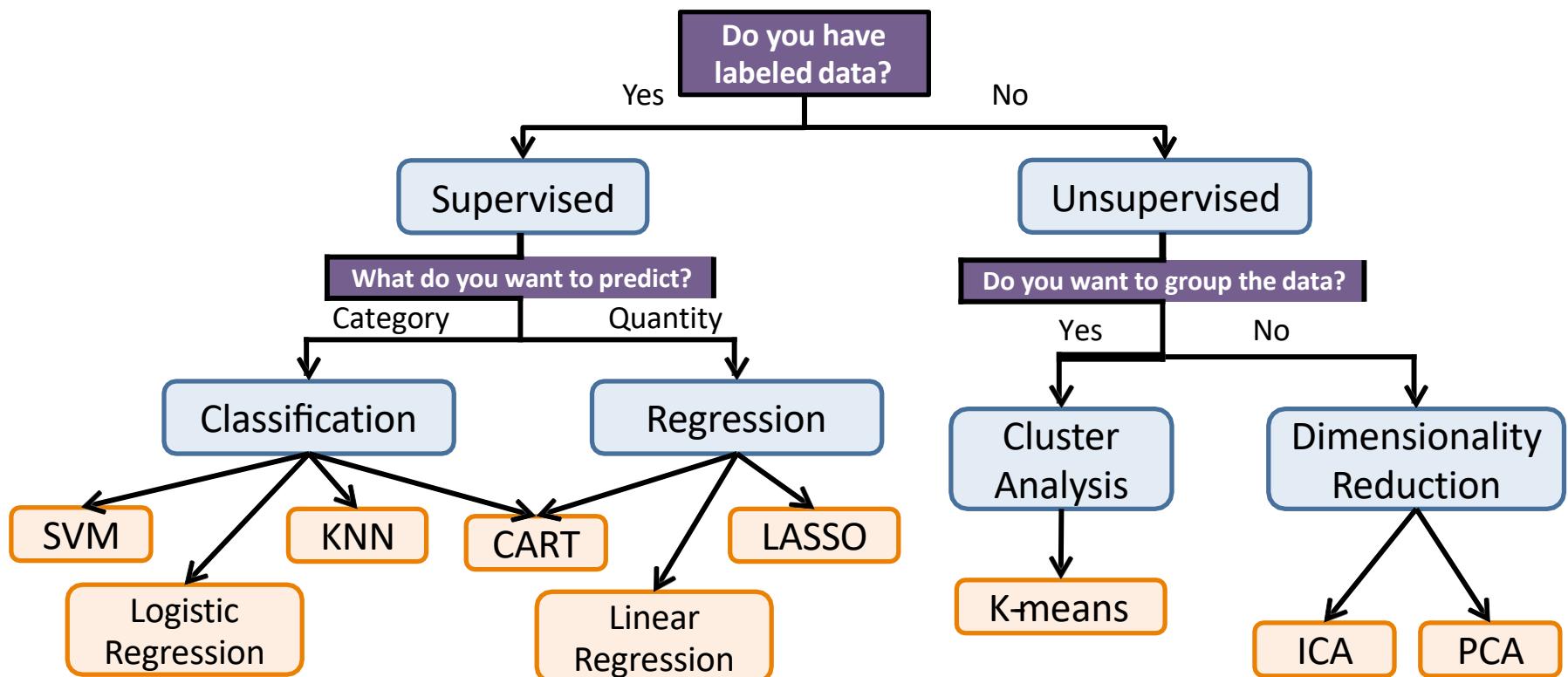
K-Nearest Neighbor classifier (KNN)

- Disadvantages:
 - Computationally expensive – must compute distance from new observation to all known samples
 - Sensitive to imbalanced datasets – may get poor results for infrequent classes
 - Sensitive to irrelevant inputs – these make distances less meaningful for identifying similar neighbors

K-Nearest Neighbor classifier (KNN)

- It is a very well-known example of **“Lazy Learning Algorithms”** in which generalization beyond the training data is delayed until a query is made to the system, as opposed to in **Eager Learning**, where the system tries to generalize the training data before receiving queries.

Types of Algorithms



“Best” Machine Learning Algorithm

- Bad news: no algorithm is the best
 - No machine learning algorithm will perform well on every task / data set
- Good news: all of them are the best
 - Each machine learning algorithm will perform well on some task / dataset

“Best” Machine Learning Algorithm

- “No free lunch” theorem
 - Wolpert (1996): all algorithms perform equally when averaged over all possible problems
 - Alternatively: Generalization can only be obtained by **assumptions**

“Best” Machine Learning Algorithm

- “No free lunch” theorem



Trade--offs in Machine Learning

- Accuracy vs. interpretability
- Bias vs. variance
- Complexity vs. scalability
 - Some models / algorithms for computing them may not scale to large data sets
- Domain-knowledge vs. data-driven
- More data vs. better algorithm

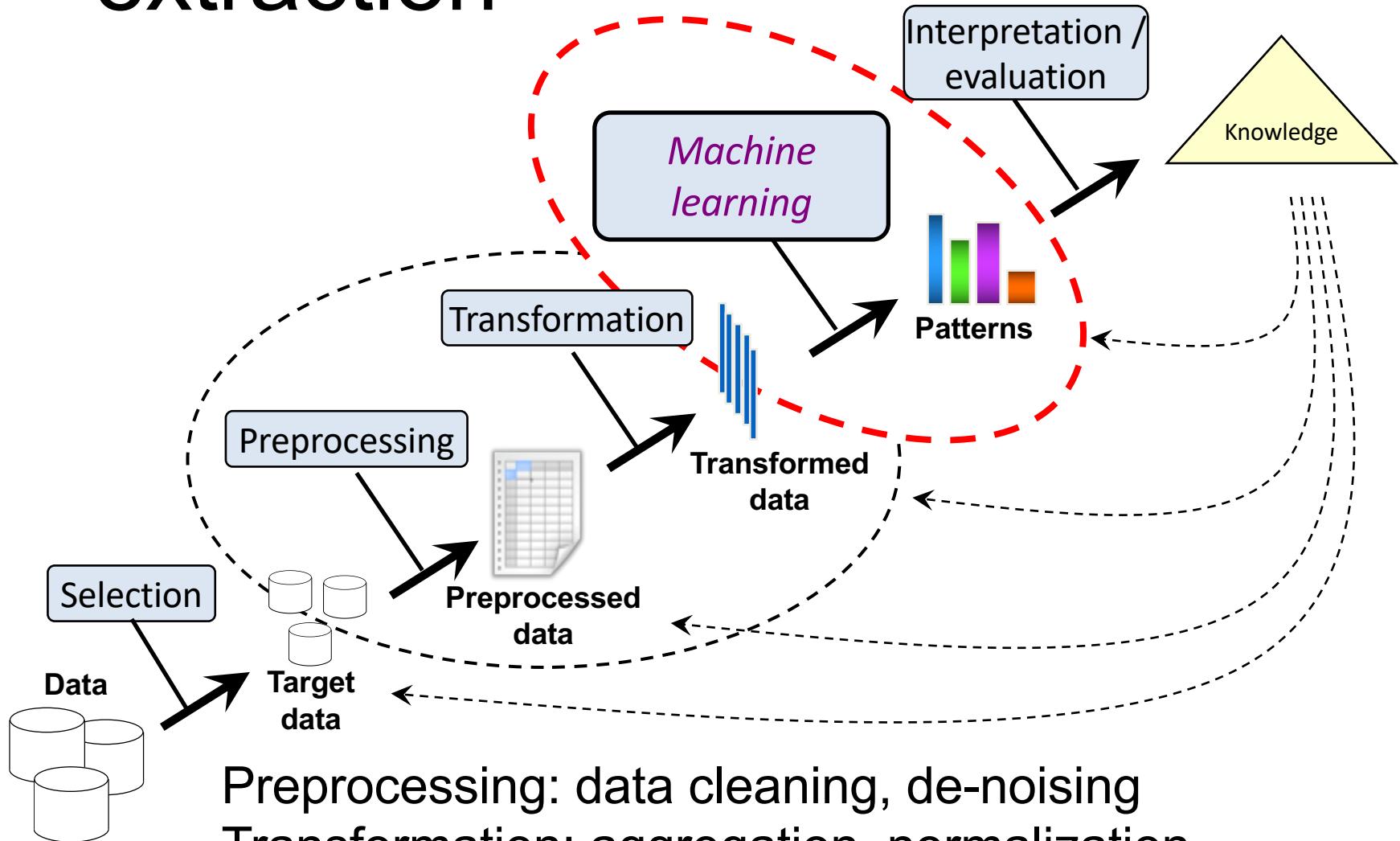
Preparing Data

- Machine learning algorithms require data!
- Preprocessing is often necessary to transform the data prior to applying a learning algorithm
 - Sampling: selecting a subset of observations
 - *Feature extraction*: selecting input variables
 - *Normalization* (standardization, scaling, binarization)
 - Handling missing data and outliers

Preparing Data

- Decision trees can handle missing data / outliers
- PCA requires data to be standardized (zero--mean)

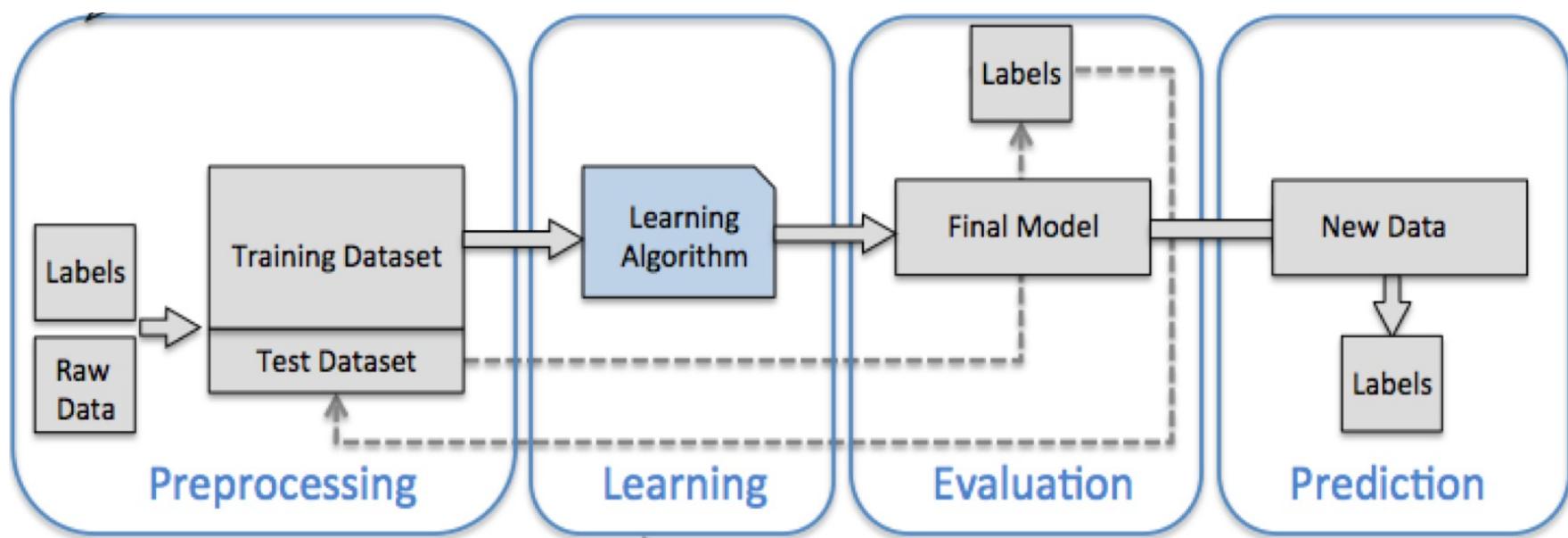
Stages of knowledge extraction



Preprocessing: data cleaning, de-noising

Transformation: aggregation, normalization,
feature creation, feature selection

Supervised Machine Learning



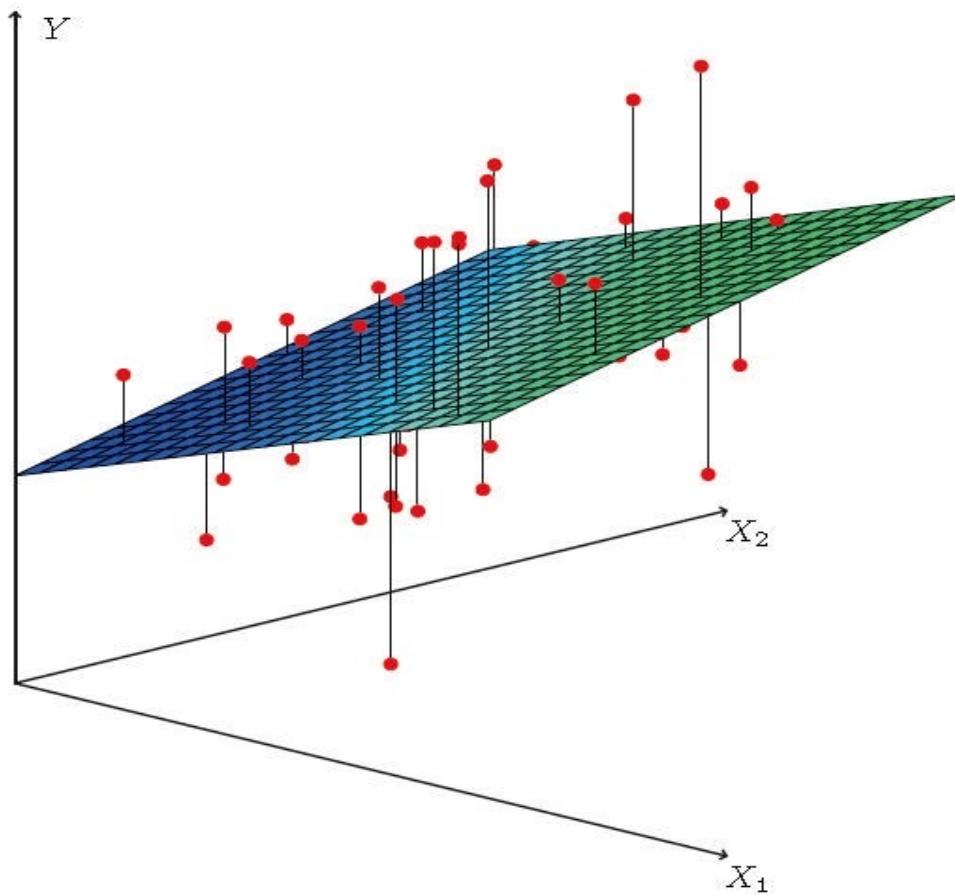
DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 2

Linear Regression

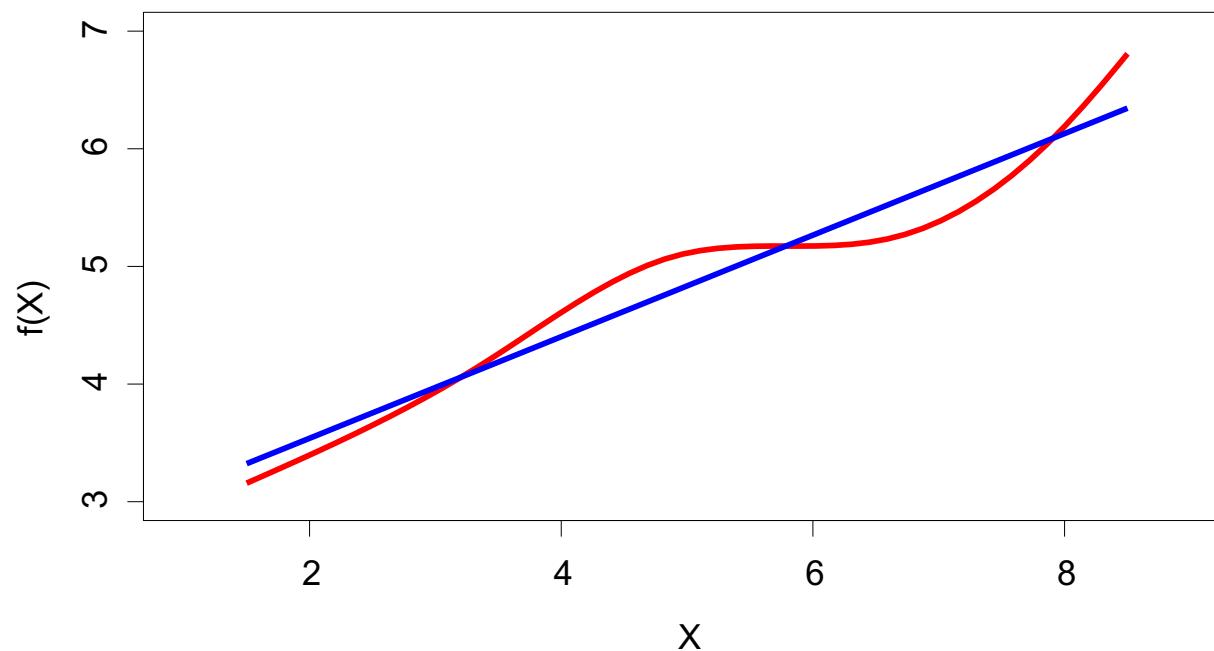


Linear Regression

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

Linear regression

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



Linear regression for the advertising data

Consider the advertising data shown on the next slide. Questions we might ask:

- Is there a relationship between advertising budget and sales?
- How strong is the relationship between advertising budget and sales?

Linear regression for the advertising data

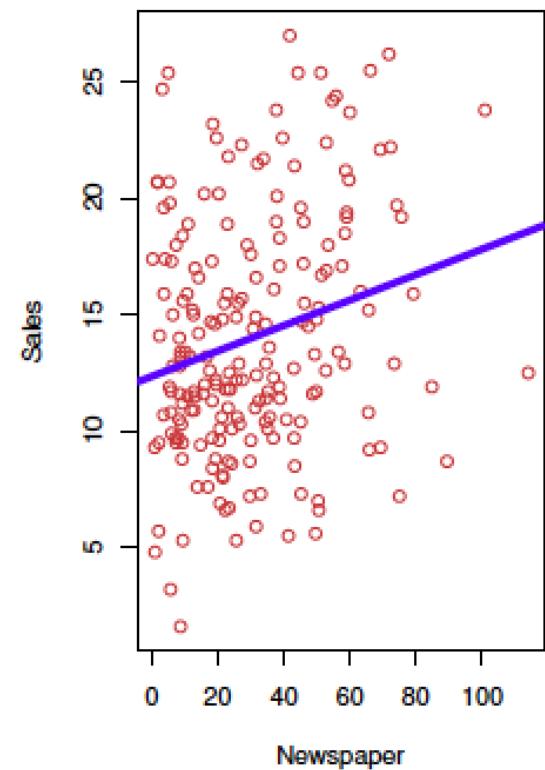
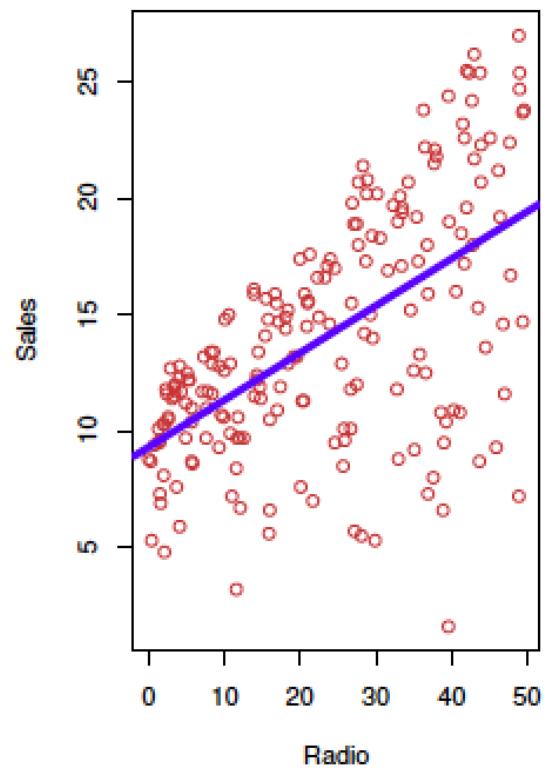
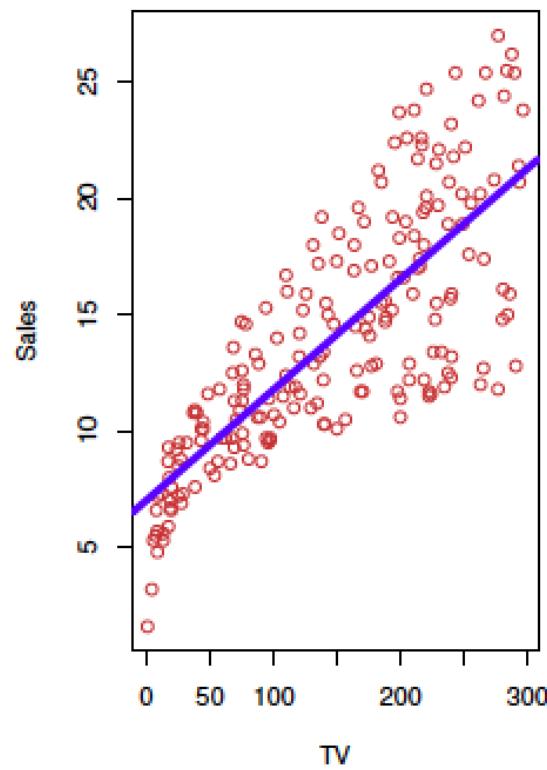
Consider the advertising data shown

on the next slide. Questions we

might ask:

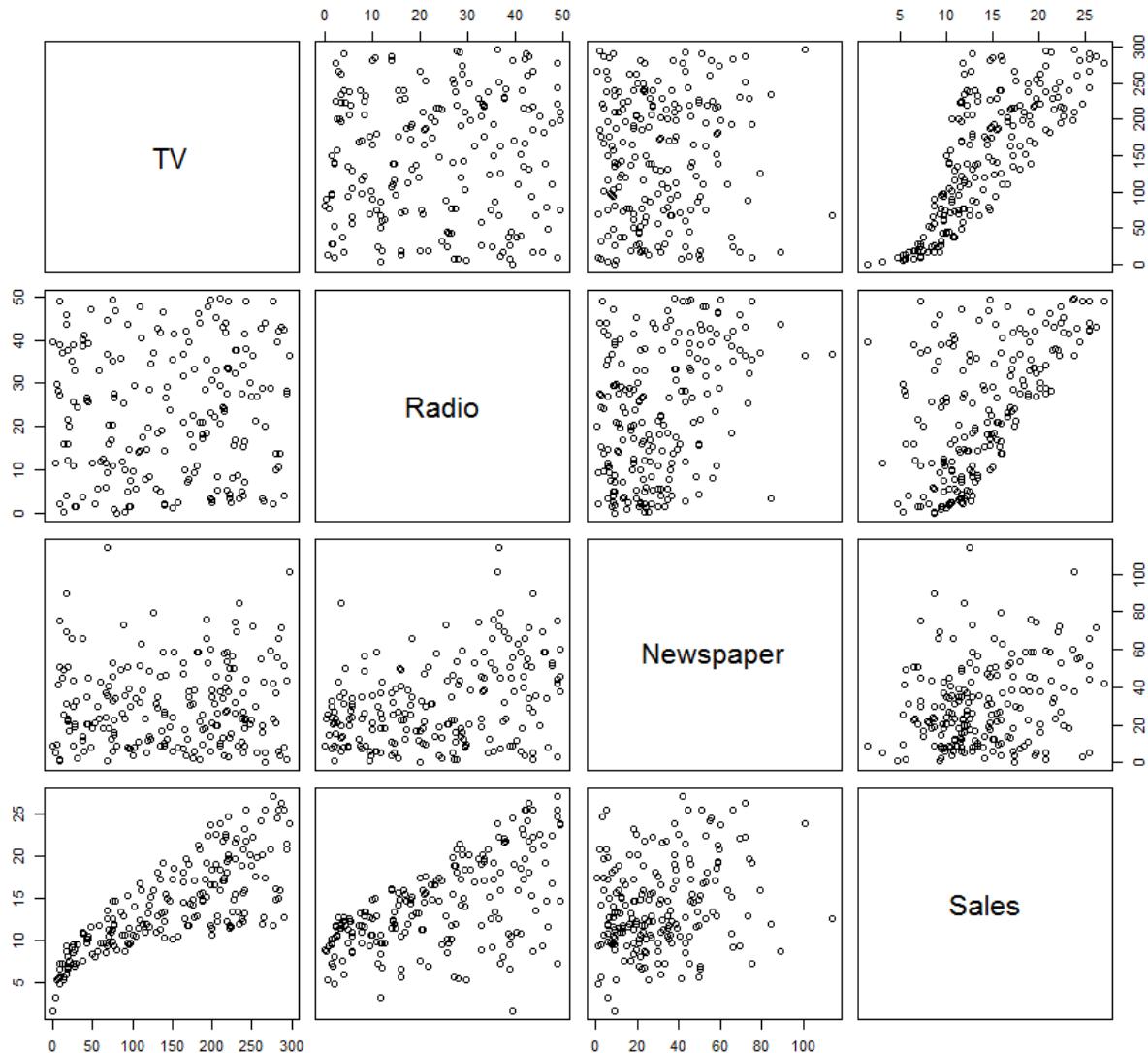
- Which media contribute to sales?
- How accurately can we predict future sales?
- Is the relationship linear?
- Is there synergy among the advertising media?

Advertising data



Case 1: Advertisement Data

```
Advertising=read.csv("http://www-
bcf.usc.edu/~gareth/ISL/Advertising.csv", header=TRUE);
newdata=Advertising[,-1]
fix(newdata)
View(newdata)
names(newdata)
pairs(newdata)
```



Simple linear regression using a single predictor X .

- We assume a model

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

where β_0 and β_1 are two unknown constants that represent the *intercept* and *slope*, also known as *coefficients* or *parameters*, and ε is the error term.

Simple linear regression using a single predictor X .

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

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

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

Estimation of the parameters by least squares

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

Estimation of the parameters by least squares

- Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the i^{th} value of X . Then $e_i = y_i - \hat{y}_i$ represents the i^{th} residual
- We define the *residual sum of squares* (RSS) as $\text{RSS} = e_1^2 + e_2^2 + \dots + e_n^2$ or equivalently as

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

Normality of ε

- Note that in the following, the statistical results including confidence intervals, hypothesis testing assume that ε is normally distributed with mean zero and standard deviation σ .

Estimation of the parameters by least squares

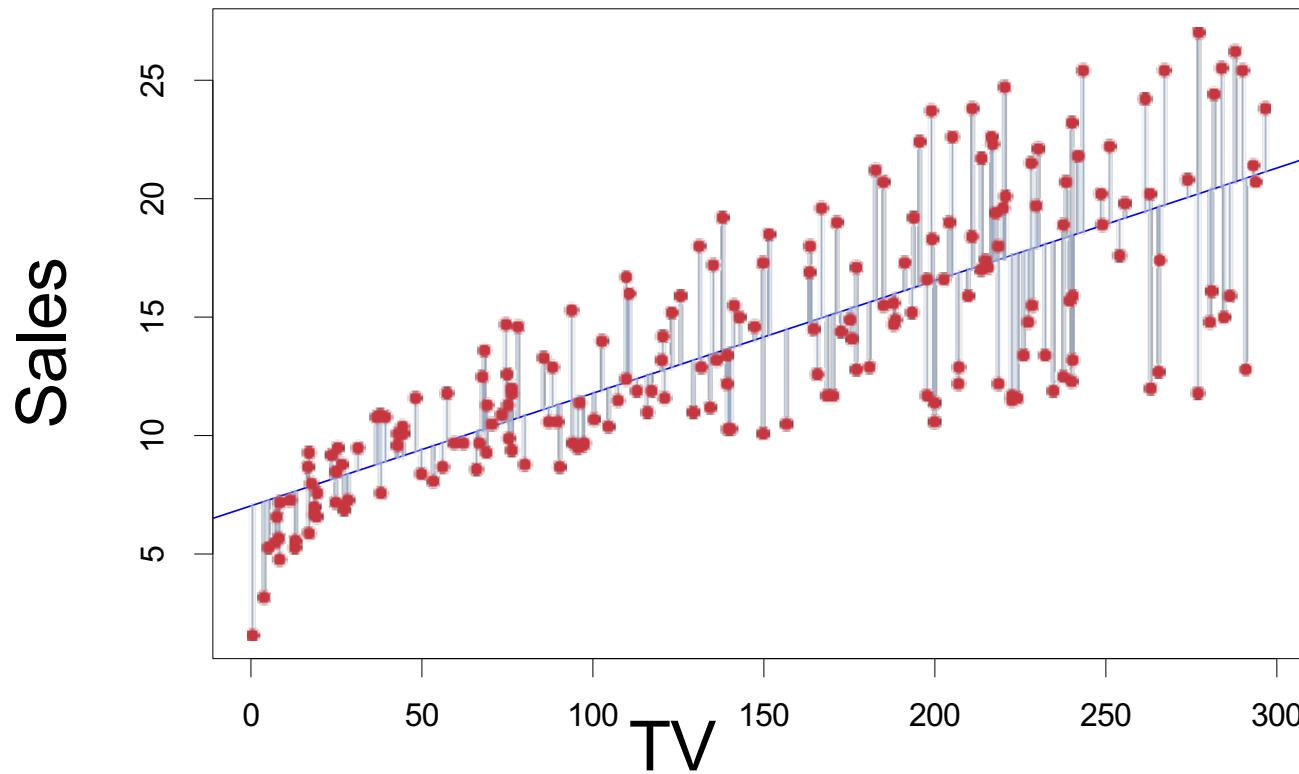
The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. The minimizing values can be shown to be

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

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

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

Example: advertising data



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

Assessing the Accuracy of the Coefficient Estimates

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

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

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

Assessing the Accuracy of the Coefficient Estimates

- These standard errors can be used to compute **confidence intervals**. A 95% confidence interval is defined as a range of values such that 95% of times, the range will contain the true unknown value of the parameter. It has the form $\hat{\beta}_1 \pm 2 \cdot SE(\hat{\beta}_1)$.

Confidence intervals — continued

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

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

will contain the true value of β_1 (under a scenario where we got repeated samples like the present sample)

Confidence intervals — continued

In fact, an interval that will contain the true unknown value of the parameter β_1 in $1-\alpha$ percent of times is

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

Approximate CI for
 $1-\alpha=0.95$ (by the
textbook)

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

More accurate CI

Student's t distribution

Student's t-distribution (or simply the t-distribution) is any member of a family of continuous probability distributions that arises when estimating the mean of a normally distributed population in situations where the sample size is small and the population standard deviation is unknown.

Student's t distribution

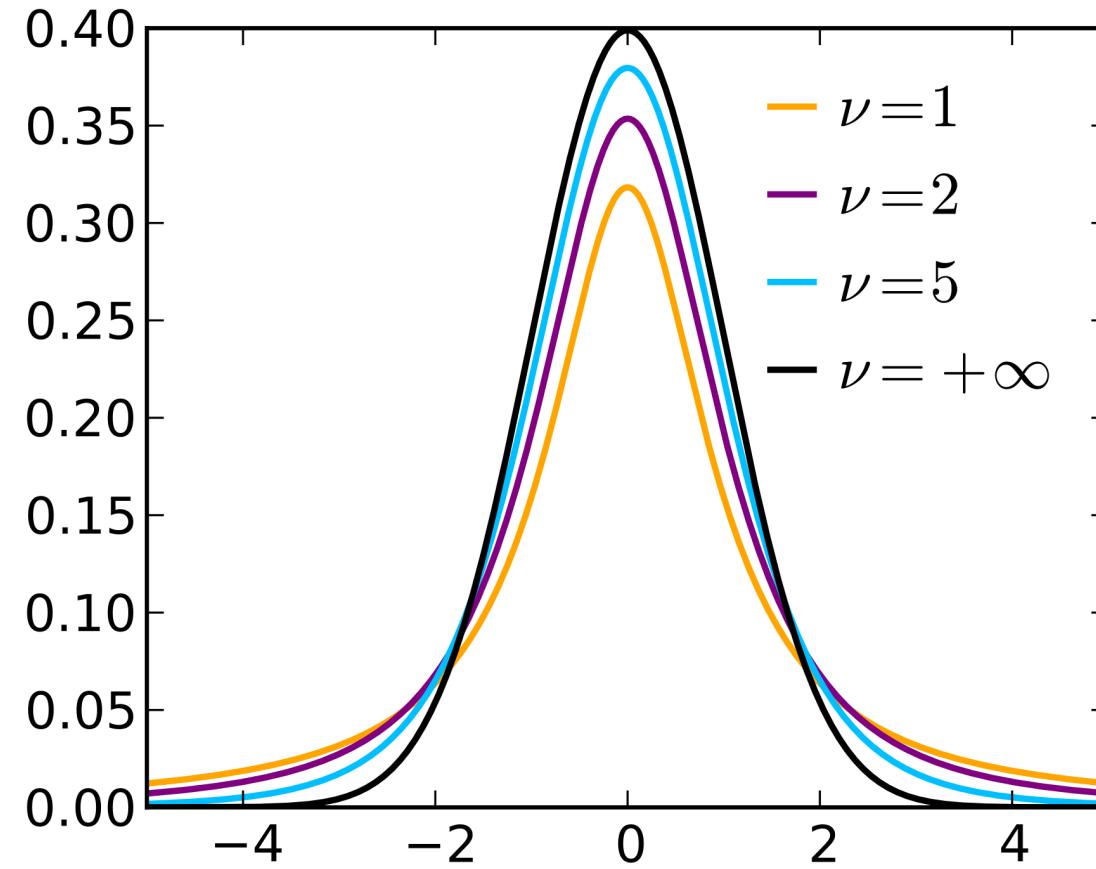
It was developed by William Sealy Gosset under the pseudonym Student.

The family is parameterized by a parameter ν , which is called the degrees of freedom.

The distribution is bell-shaped and has a zero mean, but its tails are heavier than the standard normal distribution.

Student's t distribution

t



Student's t distribution

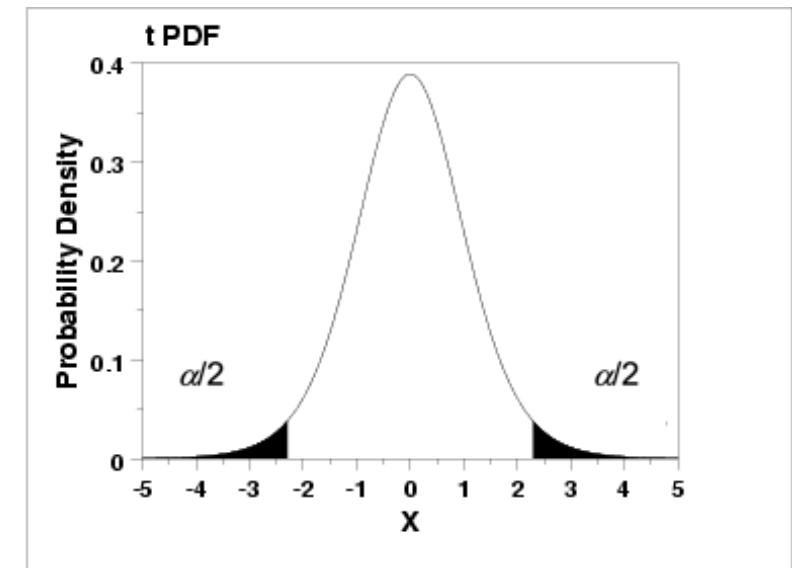
When $\nu \rightarrow \infty$, $t_{\nu} \rightarrow Z$, where Z is a standard normal distribution, i.e., a normal distribution with mean zero and standard deviation 1.

Student's t distribution-cut off points

By $t_{n-2, \alpha/2}$, we mean:

$$\Pr(t_{n-2} > t_{n-2, \alpha/2}) = \alpha/2$$

In other words, the area under the pdf of the t distribution with $n-2$ degrees of freedom is $\alpha/2$ to the right of $t_{n-2, \alpha/2}$.



Advertisement Data for simple linear regression

```
lm.fit=lm(Sales~TV,data=Advertising) ## to get Table 3.1
summary(lm.fit)
names(lm.fit)    call:
              lm(formula = sales ~ TV, data = Advertising)
coef(lm.fit)      Residuals:
                   Min       1Q   Median       3Q       Max
confint(lm.fit) -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 ***
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
```

Results for the advertising data

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

Confidence intervals — continued

For the advertising data, the 95% confidence interval for β_1 is approximately [0.042, 0.053]

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

Approximate CI for
 $1-\alpha=0.95$ (by the
textbook)

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

More accurate CI

Hypothesis testing

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

H_0 :There is no relationship between X and Y

versus the *alternative hypothesis*

H_A :There is some relationship between X and Y .

Hypothesis testing

- Mathematically, this corresponds to testing

$$H_0: \beta_1 = 0$$

versus

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

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

Hypothesis testing — continued

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

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

- This will have a *t*-distribution with $n - 2$ degrees of freedom, assuming $\beta_1 = 0$.

Hypothesis testing — continued

- If the null hypothesis is true, the probability of observing $t > t_{n-2,\alpha/2}$ or $t < t_{n-2,\alpha/2}$ would be α . α is the probability of rejecting a true null hypothesis, i.e. a *Type-I error*, and should be set **ahead of time** (metaphorically, by your boss). **Why?** Usually, α is selected to be 5%.

Rejection Region Approach

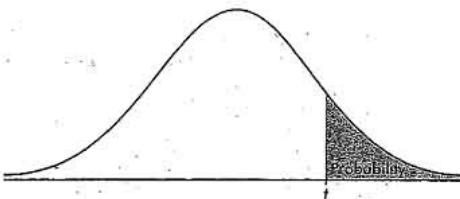


TABLE B: t -DISTRIBUTION CRITICAL VALUES

Rejection Region Approach

Rejection Region Approach

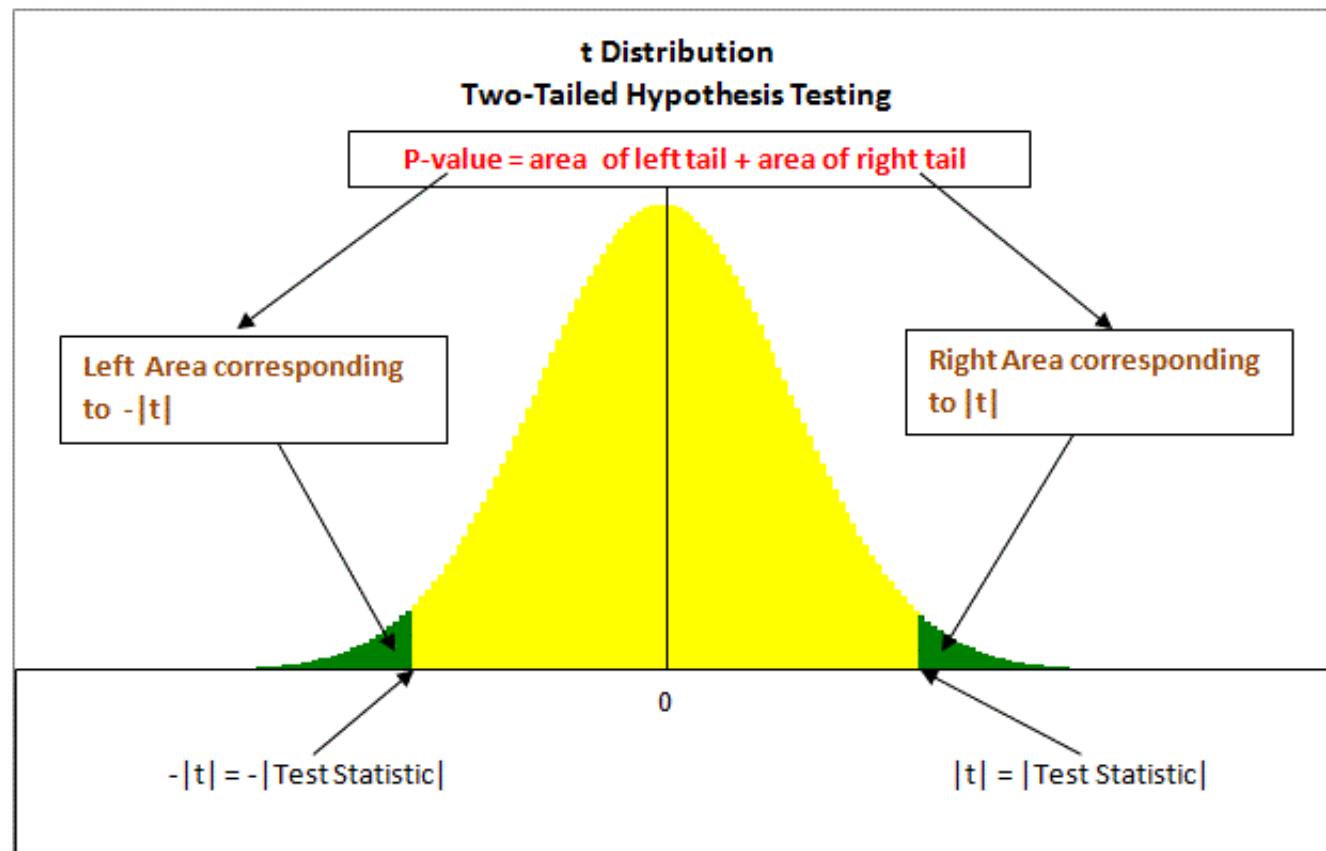
Hypothesis testing — continued

- Using statistical software, it is easy to compute the probability of observing any value equal to $|t|$ or larger. We call this probability the *p-value*.

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

Hypothesis testing — continued

- We call this probability the *p-value*.



Hypothesis testing — continued

- If the p-value is very small, it means that the probability of seeing a t statistic extremer than what was observed assuming that $\beta_1 = 0$ is very small. So we reject the null.

Advertisement Data for simple linear regression

```
lm.fit=lm(Sales~TV,data=Advertising) ## to get Table 3.1
summary(lm.fit)
names(lm.fit)      call:
              lm(formula = sales ~ TV, data = Advertising)
coef(lm.fit)      Residuals:
                   Min       1Q   Median       3Q       Max
-8.3860 -1.9545 -0.1913  2.0671  7.2124

confint(lm.fit)   Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.032594    0.457843   15.36 <2e-16 ***
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
```

Results for the advertising data

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

Inferences about the Slope: t Test Example

Test Statistic: $t = 17.76$

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

From Software output:

	Coefficients	Standard Error	t Stat	P-value
Intercept	7.0325	0.4578	15.36	<0.0001
TV	.0475	0.0027	17.67	<0.0001

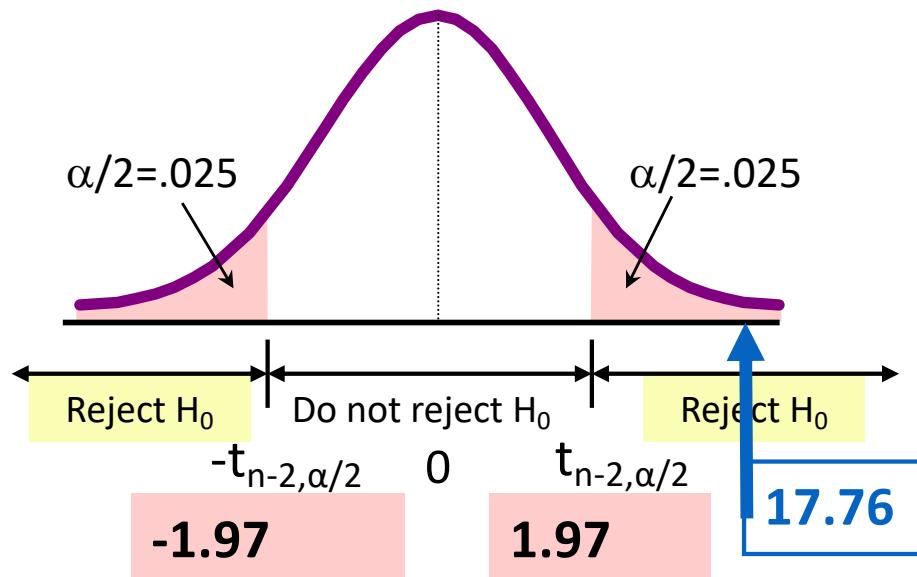
$$\begin{aligned} H_0: \beta_1 &= 0 \\ H_1: \beta_1 &\neq 0 \end{aligned}$$

Inferences about the Slope: t Test Example

$$\begin{aligned} H_0: \beta_1 &= 0 \\ H_1: \beta_1 &\neq 0 \end{aligned}$$

$$d.f. = n-2 = 198$$

$$t_{198,.025} = 1.97$$



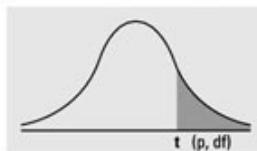
Test Statistic: $t = 17.76$

Decision:
Reject H_0

Conclusion:

There is sufficient evidence that TV affects sales

Numbers in each row of the table are values on a *t*-distribution with
(*df*) degrees of freedom for selected right-tail (greater-than) probabilities (*p*).



df/p	0.40	0.25	0.10	0.05	0.025	0.01	0.005	0.0005
1	0.324920	1.000000	3.077684	6.313752	12.70620	31.82052	63.65674	636.6192
2	0.288675	0.816497	1.885618	2.919986	4.30265	6.96456	9.92484	31.5991
3	0.276671	0.764892	1.637744	2.353363	3.18245	4.54070	5.84091	12.9240
4	0.270722	0.740697	1.533206	2.131847	2.77645	3.74695	4.60409	8.6103
5	0.267181	0.726687	1.475884	2.015048	2.57058	3.36493	4.03214	6.8688
6	0.264835	0.717558	1.439756	1.943180	2.44691	3.14267	3.70743	5.9588
7	0.263167	0.711142	1.414924	1.894579	2.36462	2.99795	3.49948	5.4079
8	0.261921	0.706387	1.396815	1.859548	2.30600	2.89646	3.35539	5.0413
9	0.260955	0.702722	1.383029	1.833113	2.26216	2.82144	3.24984	4.7809
10	0.260185	0.699812	1.372184	1.812461	2.22814	2.76377	3.16927	4.5869
11	0.259556	0.697445	1.363430	1.795885	2.20099	2.71808	3.10581	4.4370
12	0.259033	0.695483	1.356217	1.782288	2.17881	2.68100	3.05454	4.3178
13	0.258591	0.693829	1.350171	1.770933	2.16037	2.65031	3.01228	4.2208
14	0.258213	0.692417	1.345030	1.761310	2.14479	2.62449	2.97684	4.1405
15	0.257885	0.691197	1.340606	1.753050	2.13145	2.60248	2.94671	4.0728
16	0.257599	0.690132	1.336757	1.745884	2.11991	2.58349	2.92078	4.0150
17	0.257347	0.689195	1.333379	1.739607	2.10982	2.56693	2.89823	3.9651
18	0.257123	0.688364	1.330391	1.734064	2.10092	2.55238	2.87844	3.9216
19	0.256923	0.687621	1.327728	1.729133	2.09302	2.53948	2.86093	3.8834
20	0.256743	0.686954	1.325341	1.724718	2.08596	2.52798	2.84534	3.8495
21	0.256580	0.686352	1.323188	1.720743	2.07961	2.51765	2.83136	3.8193
22	0.256432	0.685805	1.321237	1.717144	2.07387	2.50832	2.81876	3.7921
23	0.256297	0.685306	1.319460	1.713872	2.06866	2.49987	2.80734	3.7676
24	0.256173	0.684850	1.317836	1.710882	2.06390	2.49216	2.79694	3.7454
25	0.256060	0.684430	1.316345	1.708141	2.05954	2.48511	2.78744	3.7251
26	0.255955	0.684043	1.314972	1.705618	2.05553	2.47863	2.77871	3.7066
27	0.255858	0.683685	1.313703	1.703288	2.05183	2.47266	2.77068	3.6896
28	0.255768	0.683353	1.312527	1.701131	2.04841	2.46714	2.76326	3.6739
29	0.255684	0.683044	1.311434	1.699127	2.04523	2.46202	2.75639	3.6594
30	0.255605	0.682756	1.310415	1.697261	2.04227	2.45726	2.75000	3.6460
z	0.253347	0.674490	1.281552	1.644854	1.95996	2.32635	2.57583	3.2905
CI	——	——	80%	90%	95%	98%	99%	99.9%

Assessing the Overall Accuracy of the Model

- We compute the *Residual Standard Error*

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

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

Assessing the Overall Accuracy of the Model

- The *Residual Standard Error* is used to estimate the variance of the noise ϵ , i.e. to measure how much on average the response deviated from the regression line.

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

Explanatory Power of a Linear Regression Equation

Total variation is made up of two parts:

$$\text{TSS} = \text{Regression SS} + \text{RSS}$$

Total Sum of Squares

Regression Sum of Squares

Error (residual) Sum of Squares

$$= \sum (y_i - \bar{y})^2$$

$$= \sum (\hat{y}_i - \bar{y})^2$$

$$= \sum (y_i - \hat{y}_i)^2$$

where:

\bar{y} = Average value of the dependent variable

y_i = Observed values of the dependent variable

\hat{y}_i = Predicted value of y for the given x_i value

Explanatory Power of a Linear Regression Equation

TSS = total sum of squares

Measures the variation of the y_i values around their mean, \bar{y}

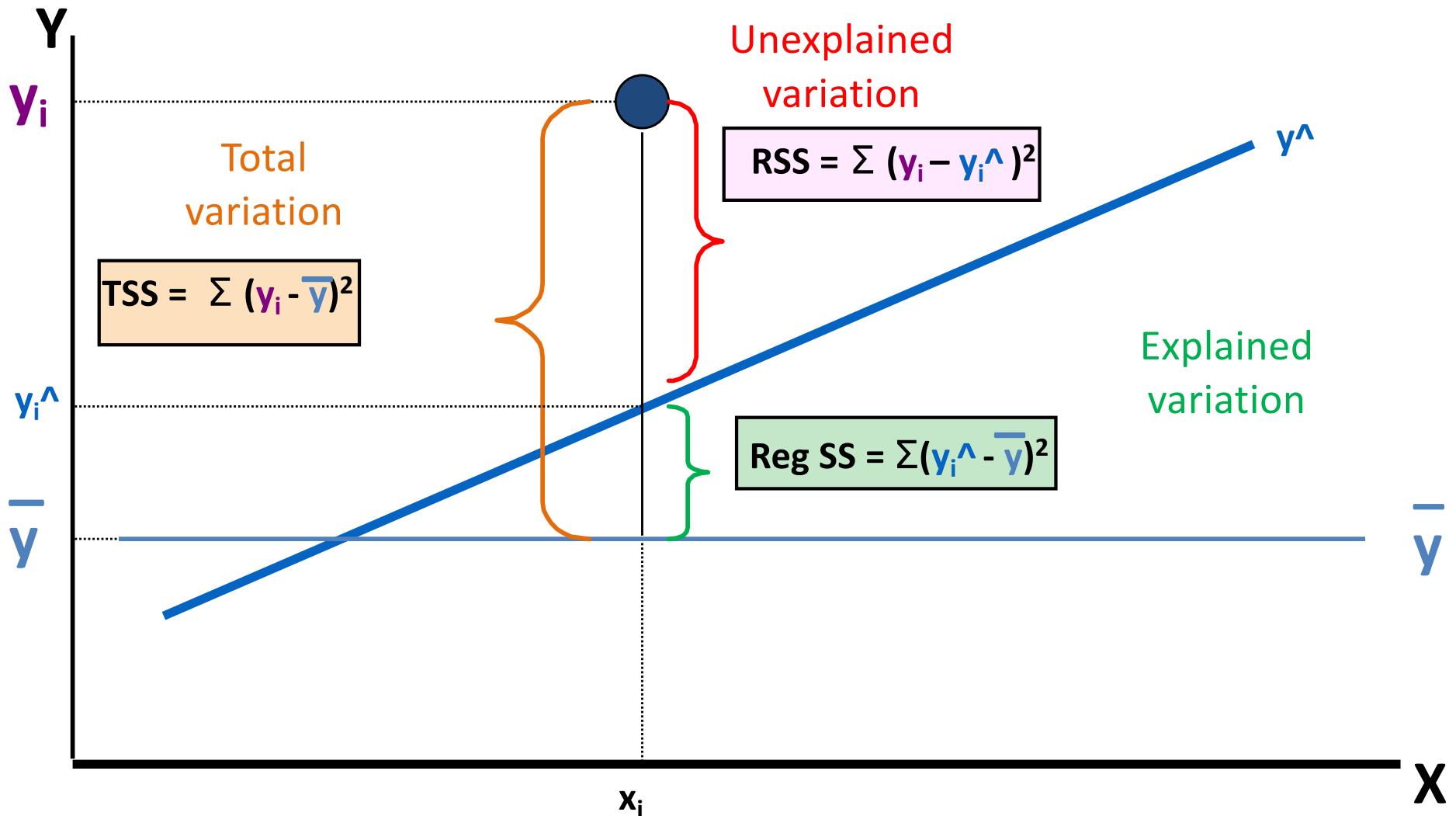
Regression SS = regression sum of squares

Explained variation attributable to the linear relationship between X and Y

RSS = Residual (error) sum of squares

Variation attributable to factors other than the linear relationship between X and Y

Explanatory Power of a Linear Regression Equation



Assessing the Overall Accuracy of the Model

- We are interested in the ratio of variation explained to total variation, i.e.
-

$$\frac{\text{RegSS}}{\text{TSS}} = -$$

Assessing the Overall Accuracy of the Model

- *R-squared* or fraction of total variation explained is

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

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

Assessing the Overall Accuracy of the Model

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

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

$$= \frac{S_{XY}}{S_X S_Y}$$

Advertising data results

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

Multiple Linear Regression

- Here our model is

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

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

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

Interpreting regression coefficients

- The ideal scenario is when the predictors are uncorrelated
 - a *balanced design*:
 - Each coefficient can be estimated and tested separately.
 - Interpretations such as “*a unit change in X_j is associated with a β_j change in Y on average, while all the other variables stay fixed*”, are possible.

Interpreting regression coefficients

- Correlations amongst predictors cause problems:
 - The variance of all coefficients tends to increase, sometimes dramatically
 - Interpretations become hazardous — when X_j changes, everything else changes.

Interpreting regression coefficients

- *Claims of causality* should be avoided for observational data.

The woes of (interpreting) regression coefficients

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

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

The woes of (interpreting) regression coefficients

- Example: Y total amount of change in your pocket;
 X_1 = # of coins; X_2 = # of pennies, nickels and dimes.
By itself, regression coefficient of Y on X_2 will be > 0 . But how about with X_1 in model?

The woes of (interpreting) regression coefficients

- Y = number of tackles by a football player in a season; W and H are his weight and height.
- Fitted regression model is $\hat{Y} = b_0 + 0.50W - 0.10H$. How do we interpret $\hat{\beta}_2 < 0$?

Two quotes by famous Statisticians

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

George Box

Two quotes by famous Statisticians

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

Fred Mosteller and John Tukey,
paraphrasing George Box

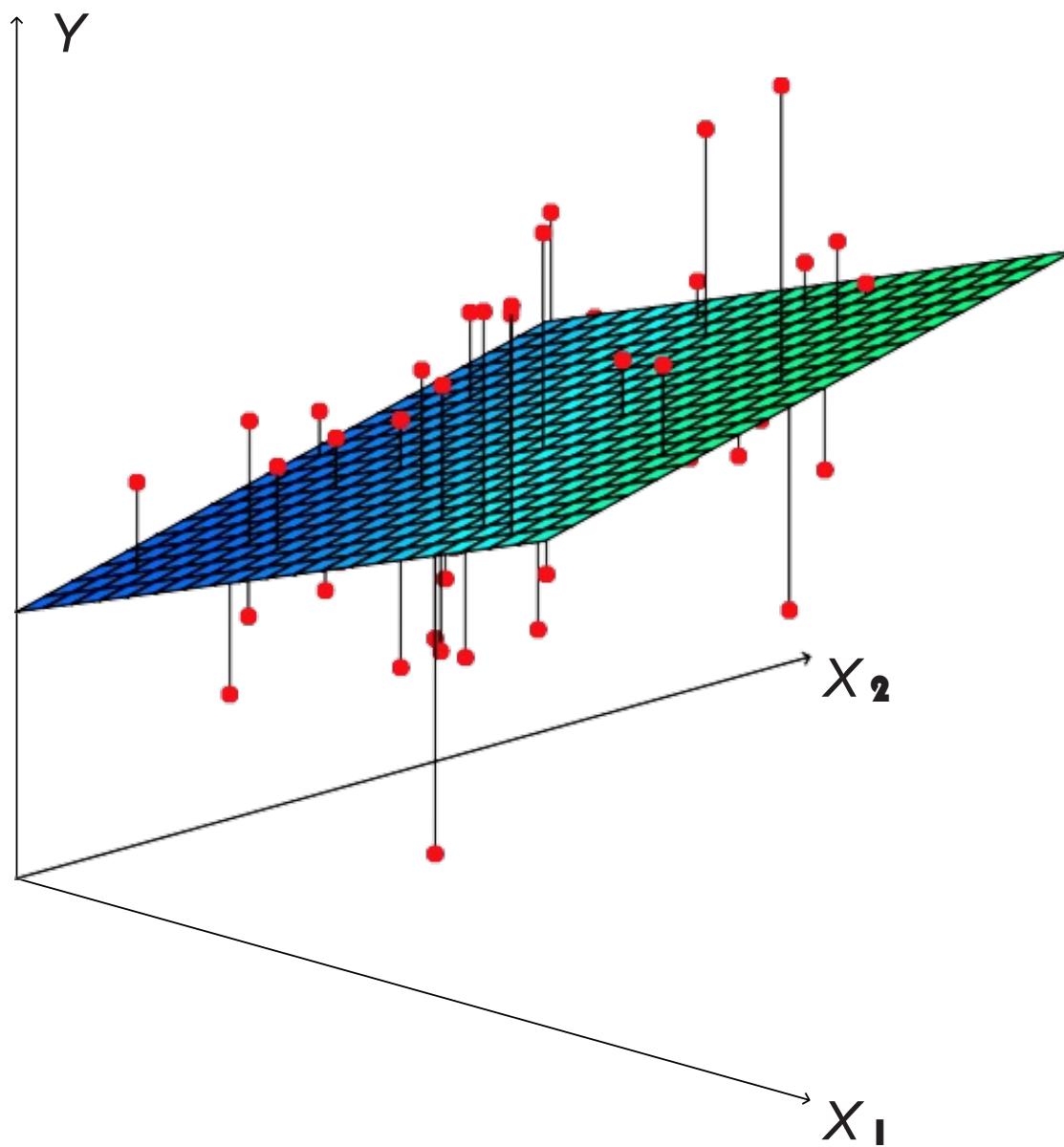
Estimation and Prediction for Multiple Regression

- Given estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$, we can make predictions using the formula
- $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_p x_p$.
- We estimate $\beta_0, \beta_1, \dots, \beta_p$ as the values that minimize the sum of squared residuals RSS

Estimation and Prediction for Multiple Regression

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

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



Confidence intervals for Multiple Regression

An interval that will contain the true unknown value of the parameter β_i in $1-\alpha$ percent of times is

$$[\hat{\beta}_i - t_{n-p-1, \alpha/2} \cdot \text{SE}(\hat{\beta}_i), \hat{\beta}_i + t_{n-p-1, \alpha/2} \cdot \text{SE}(\hat{\beta}_i)]$$

Hypothesis testing

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

H_0 :There is no relationship between X_i ,
and Y

versus the *alternative hypothesis*

H_A :There is some relationship between X_i ,
and Y .

Hypothesis testing

- Mathematically, this corresponds to testing

$$H_0: \beta_i = 0$$

versus

$$H_A: \beta_i \neq 0,$$

since if $\beta_i = 0$ then X_i is not associated with Y .

Hypothesis testing

- In general, to test the following hypothesis

$$H_0: \beta_i = \beta$$

versus

$$H_A: \beta_i \neq \beta,$$

we use a t-statistic:

Hypothesis testing — continued

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

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

Usually zero

- This will have a *t*-distribution with $n - p - 1$ degrees of freedom, assuming $\beta_i = \beta$.

Hypothesis testing — continued

- Using statistical software, it is easy to compute the probability of observing any value equal to $|t|$ or larger. We call this probability the *p-value*.

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

Usually zero

Hypothesis testing — continued

- If the p-value is very small, it means that the probability of seeing a t statistic extremer than what was observed (assuming that $\beta_i = \beta$) is very small. So we reject the null.

Rejection Region Approach

- Similar to simple regression

Results for advertising data

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

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

Some important questions

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

Some important questions

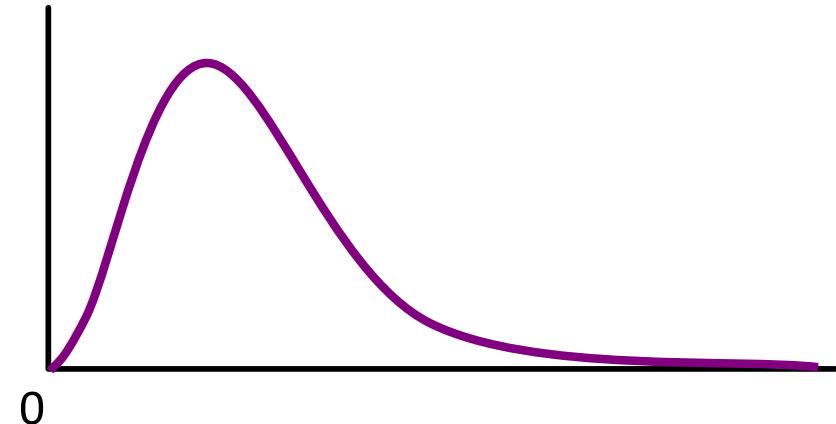
3. *How well does the model fit the data?*
4. *Given a set of predictor values, what response value should we predict, and how accurate is our prediction?*

Is at least one predictor useful?

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

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

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



Tests on Regression Coefficients

Tests on All Coefficients

F-Test for Overall Significance of the Model

Shows if there is a linear relationship between **all** of the X variables considered together and Y

Use F test statistic

Hypotheses:

$$H_0: \beta_1 = \beta_2 = \dots = \beta_p = 0 \text{ (no linear relationship)}$$
$$H_1: \text{at least one } \beta_i \neq 0 \text{ (at least one independent variable affects } Y)$$

F-Test for Overall Significance

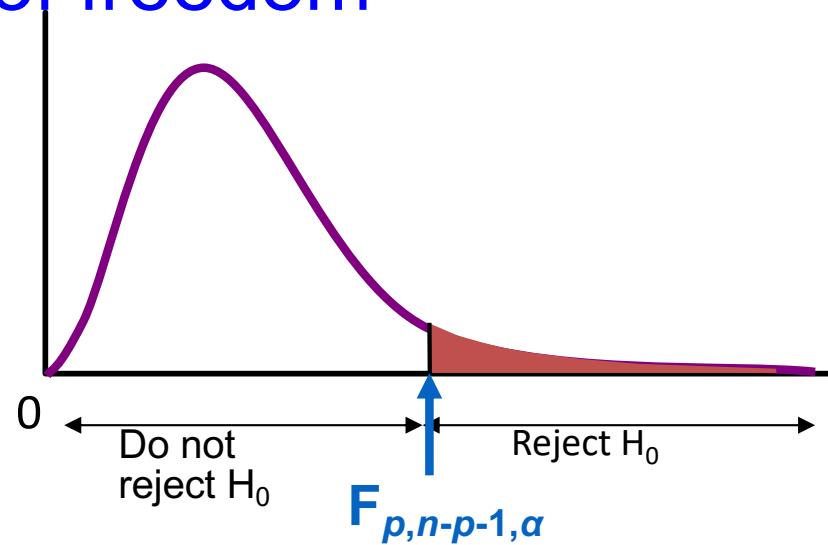
Test statistic:

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

where F has p (numerator) and $(n - p - 1)$ (denominator) degrees of freedom

The decision rule is

Reject H_0 if $F > F_{p,n-p-1,\alpha}$



F-Test for Overall Significance

F - Distribution ($\alpha = 0.05$ in the Right Tail)

Denominator Degrees of Freedom df_2	Numerator Degrees of Freedom df_1	Numerator Degrees of Freedom								
		1	2	3	4	5	6	7	8	9
1	161.45	199.50	215.71	224.58	230.16	233.99	236.77	238.88	240.54	
2	18.513	19.000	19.164	19.247	19.296	19.330	19.353	19.371	19.385	
3	10.128	9.5521	9.2766	9.1172	9.0135	8.9406	8.8867	8.8452	8.8123	
4	7.7086	9.9443	6.5914	6.3882	6.2561	6.1631	6.0942	6.0410	6.9988	
5	6.6079	5.7861	5.4095	5.1922	5.0503	4.9503	4.8759	4.8183	4.7725	
6	5.9874	5.1433	4.7571	4.5337	4.3874	4.2839	4.2067	4.1468	4.0990	
7	5.5914	4.7374	4.3468	4.1203	3.9715	3.8660	3.7870	3.7257	3.6767	
8	5.3177	4.4590	4.0662	3.8379	3.6875	3.5806	3.5005	3.4381	3.3881	
9	5.1174	4.2565	3.8625	3.6331	3.4817	3.3738	3.2927	3.2296	3.1789	
10	4.9646	4.1028	3.7083	3.4780	3.3258	3.2172	3.1355	3.0717	3.0204	
11	4.8443	3.9823	3.5874	3.3567	3.2039	3.0946	3.0123	2.9480	2.8962	
12	4.7472	3.8853	3.4903	3.2592	3.1059	2.9961	2.9134	2.8486	2.7964	
13	4.6672	3.8056	3.4105	3.1791	3.0254	2.9153	2.8321	2.7669	2.7144	
14	4.6001	3.7389	3.3439	3.1122	2.9582	2.8477	2.7642	2.6987	2.6458	
15	4.5431	3.6823	3.2874	3.0556	2.9013	2.7905	2.7066	2.6408	2.5876	
16	4.4940	3.6337	3.2389	3.0069	2.8524	2.7413	2.6572	2.5911	2.5377	
17	4.4513	3.5915	3.1968	2.9647	2.8100	2.6987	2.6143	2.5480	2.4943	
18	4.4139	3.5546	3.1599	2.9277	2.7729	2.6613	2.5767	2.5102	2.4563	
19	4.3807	3.5219	3.1274	2.8951	2.7401	2.6283	2.5435	2.4768	2.4227	
20	4.3512	3.4928	3.0984	2.8661	2.7109	2.5990	2.5140	2.4471	2.3928	
21	4.3248	3.4668	3.0725	2.8401	2.6848	2.5727	2.4876	2.4205	2.3660	
22	4.3009	3.4434	3.0491	2.8167	2.6613	2.5491	2.4638	2.3965	2.3419	
23	4.2793	3.4221	3.0280	2.7955	2.6400	2.5277	2.4422	2.3748	2.3201	
24	4.2597	3.4028	3.0088	2.7763	2.6207	2.5082	2.4226	2.3551	2.3002	
25	4.2417	3.3852	2.9912	2.7587	2.6030	2.4904	2.4047	2.3371	2.2821	
26	4.2252	3.3690	2.9752	2.7426	2.5868	2.4741	2.3883	2.3205	2.2655	
27	4.2100	3.3541	2.9604	2.7278	2.5719	2.4591	2.3732	2.3053	2.2501	
28	4.1960	3.3404	2.9467	2.7141	2.5581	2.4453	2.3593	2.2913	2.2360	
29	4.1830	3.3277	2.9340	2.7014	2.5454	2.4324	2.3463	2.2783	2.2229	
30	4.1709	3.3158	2.9223	2.6896	2.5336	2.4205	2.3343	2.2662	2.2107	
40	4.0847	3.2317	2.8387	2.6060	2.4495	2.3359	2.2490	2.1802	2.1240	
60	4.0012	3.1504	2.7581	2.5252	2.3683	2.2541	2.1665	2.0970	2.0401	
120	3.9201	3.0718	2.6802	2.4472	2.2899	2.1750	2.0868	2.0164	1.9588	
∞	3.8415	2.9957	2.6049	2.3719	2.2141	2.0986	2.0096	1.9384	1.8799	

F-Test for Overall Significance

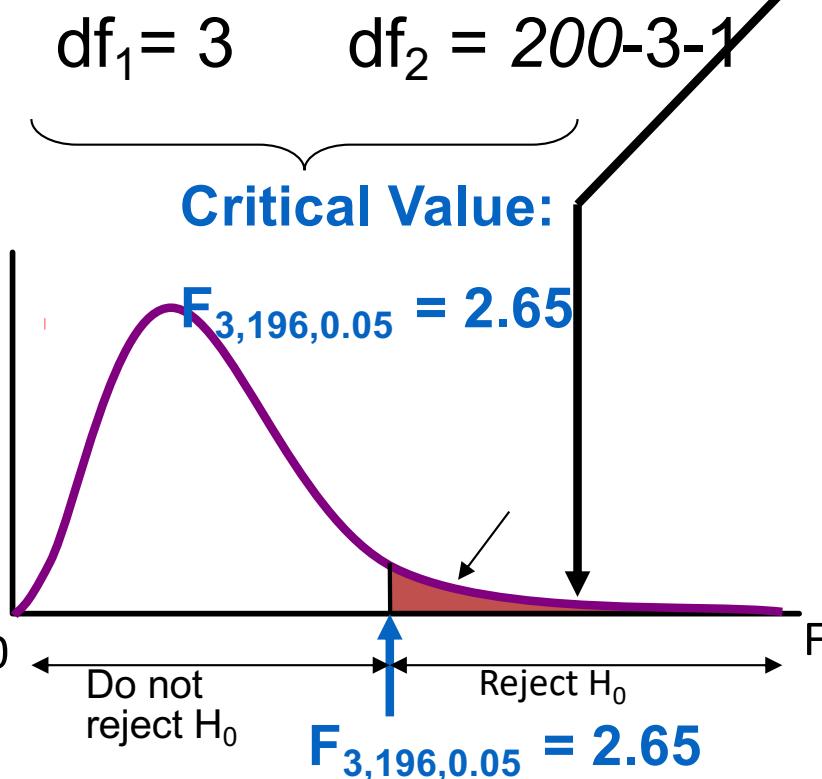
F - Distribution ($\alpha = 0.01$ in the Right Tail)

Denominator Degrees of Freedom df ₂	df ₁	Numerator Degrees of Freedom								
		1	2	3	4	5	6	7	8	9
1	4052.2	4999.5	5403.4	5624.6	5763.6	5859.0	5928.4	5981.1	6022.5	
2	98.503	99.000	99.166	99.249	99.299	99.333	99.356	99.374	99.388	
3	34.116	30.817	29.457	28.710	28.237	27.911	27.672	27.489	27.345	
4	21.198	18.000	16.694	15.977	15.522	15.207	14.976	14.799	14.659	
5	16.258	13.274	12.060	11.392	10.967	10.672	10.456	10.289	10.158	
6	13.745	10.925	9.7795	9.1483	8.7459	8.4661	8.2600	8.1017	7.9761	
7	12.246	9.5466	8.4513	7.8466	7.4604	7.1914	6.9928	6.8400	6.7188	
8	11.259	8.6491	7.5910	7.0061	6.6318	6.3707	6.1776	6.0289	5.9106	
9	10.561	8.0215	6.9919	6.4221	6.0569	5.8018	5.6129	5.4671	5.3511	
10	10.044	7.5594	6.5523	5.9943	5.6363	5.3858	5.2001	5.0567	4.9424	
11	9.6460	7.2057	6.2167	5.6683	5.3160	5.0692	4.8861	4.7445	4.6315	
12	9.3302	6.9266	5.9525	5.4120	5.0643	4.8206	4.6395	4.4994	4.3875	
13	9.0738	6.7010	5.7394	5.2053	4.8616	4.6204	4.4410	4.3021	4.1911	
14	8.8616	6.5149	5.5639	5.0354	4.6950	4.4558	4.2779	4.1399	4.0297	
15	8.6831	6.3589	5.4170	4.8932	4.5556	4.3183	4.1415	4.0045	3.8948	
16	8.5310	6.2262	5.2922	4.7726	4.4374	4.2016	4.0259	3.8896	3.7804	
17	8.3997	6.1121	5.1850	4.6690	4.3359	4.1015	3.9267	3.7910	3.6822	
18	8.2854	6.0129	5.0919	4.5790	4.2479	4.0146	3.8406	3.7054	3.5971	
19	8.1849	5.9259	5.0103	4.5003	4.1708	3.9386	3.7653	3.6305	3.5225	
20	8.0960	5.8489	4.9382	4.4307	4.1027	3.8714	3.6987	3.5644	3.4567	
21	8.0166	5.7804	4.8740	4.3688	4.0421	3.8117	3.6396	3.5056	3.3981	
22	7.9454	5.7190	4.8166	4.3134	3.9880	3.7583	3.5867	3.4530	3.3458	
23	7.8811	5.6637	4.7649	4.2636	3.9392	3.7102	3.5390	3.4057	3.2986	
24	7.8229	5.6136	4.7181	4.2184	3.8951	3.6667	3.4959	3.3629	3.2560	
25	7.7698	5.5680	4.6755	4.1774	3.8550	3.6272	3.4568	3.3239	3.2172	
26	7.7213	5.5263	4.6366	4.1400	3.8183	3.5911	3.4210	3.2884	3.1818	
27	7.6767	5.4881	4.6009	4.1056	3.7848	3.5580	3.3882	3.2558	3.1494	
28	7.6356	5.4529	4.5681	4.0740	3.7539	3.5276	3.3581	3.2259	3.1195	
29	7.5977	5.4204	4.5378	4.0449	3.7254	3.4995	3.3303	3.1982	3.0920	
30	7.5625	5.3903	4.5097	4.0179	3.6990	3.4735	3.3045	3.1726	3.0665	
40	7.3141	5.1785	4.3126	3.8283	3.5138	3.2910	3.1238	2.9930	2.8876	
60	7.0771	4.9774	4.1259	3.6490	3.3389	3.1187	2.9530	2.8233	2.7185	
120	6.8509	4.7865	3.9491	3.4795	3.1735	2.9559	2.7918	2.6629	2.5586	
∞	6.6349	4.6052	3.7816	3.3192	3.0173	2.8020	2.6393	2.5113	2.4073	

F-Test for Overall Significance

$H_0: \beta_1 = \beta_2 = \beta_3 = 0$

$H_1:$ Not all three of $\beta_1, \beta_2, \beta_3$ are zero



Test Statistic: $F=570$

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

Decision:

Since F test statistic is in the rejection region ($p\text{-value} < .05$), reject H_0

Conclusion:

There is evidence that at least one independent variable affects Y

Deciding on the important variables

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

Deciding on the important variables

- However we often can't examine all possible models, since they are 2^p of them; for example when $p = 40$ there are over a trillion models!
- Instead we need an automated approach that searches through a subset of them. We discuss two commonly used approaches next.

Forward selection

- Begin with the *null model* — a model that contains an intercept but no predictors.
- Fit p simple linear regressions and add to the null model the variable that results in the lowest RSS.

Forward selection

- Add to that model the variable that results in the lowest RSS amongst all two-variable models.
- Continue until some stopping rule is satisfied, for example when all **remaining** variables have a p-value above some threshold.

Backward selection

- Start with all variables in the model.
- Remove the variable with the largest p-value — that is, the variable that is the least statistically significant.
- The new $(p - 1)$ -variable model is fit, and the variable with the largest p-value is removed.

Backward selection

- Continue until a stopping rule is reached. For instance, we may stop when all remaining variables have a significant p-value defined by some significance threshold.

Model selection — continued

- Later we discuss more systematic criteria for choosing an “optimal” member in the path of models produced by forward or backward stepwise selection.

Model selection — continued

- These include *Mallow's C_p* ,
Akaike information criterion (AIC), *Bayesian information criterion (BIC)*, *adjusted R^2* and
Cross-validation (CV).

Other Considerations in the Regression Model

Qualitative Predictors

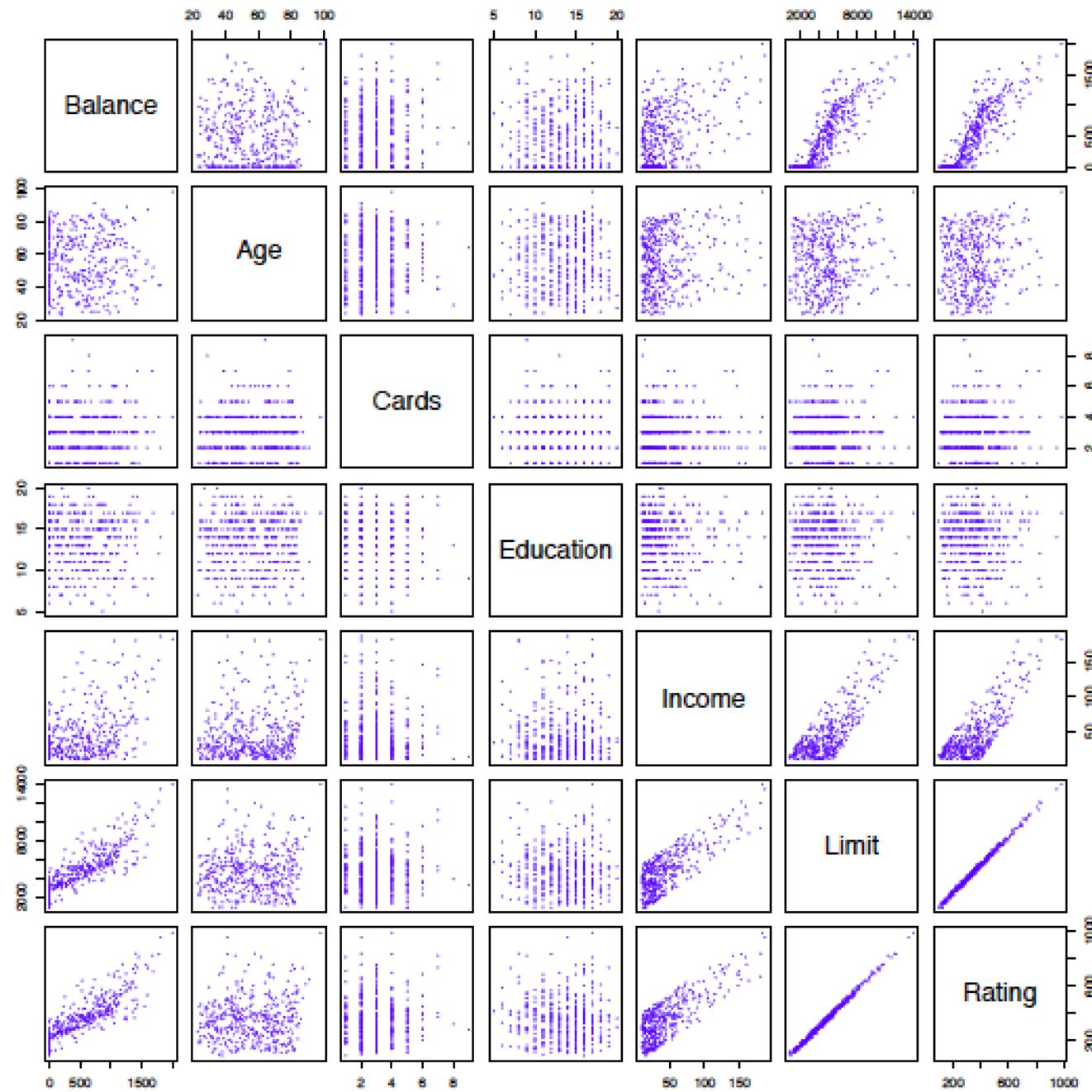
- Some predictors are not *quantitative* but are *qualitative*, taking a discrete set of values.
- These are also called *categorical* predictors or *factor variables*.

Other Considerations in the Regression Model

See for example the scatterplot matrix of the credit card data in the next slide.

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

Credit Card Data



Qualitative Predictors — cont'd

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

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

Resulting model:

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

Intrepretation?

Credit card data — continued

Results for gender model:

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

Qualitative predictors with more than two levels

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

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

and the second could be

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

Qualitative predictors with more than two levels

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

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

Qualitative predictors with more than two levels

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

Results for ethnicity

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

Extensions of the Linear Model

Removing the additive assumption:

interactions and *nonlinearity*

Interactions:

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

Extensions of the Linear Model

- For example, the linear model

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

states that the average effect on **sales** of a one-unit increase in **TV** is always β_1 , regardless of the amount spent on **radio**.

Interactions — continued

- But suppose that spending money on radio advertising actually increases the effectiveness of TV advertising, so that the slope term for **TV** should increase as **radio** increases.

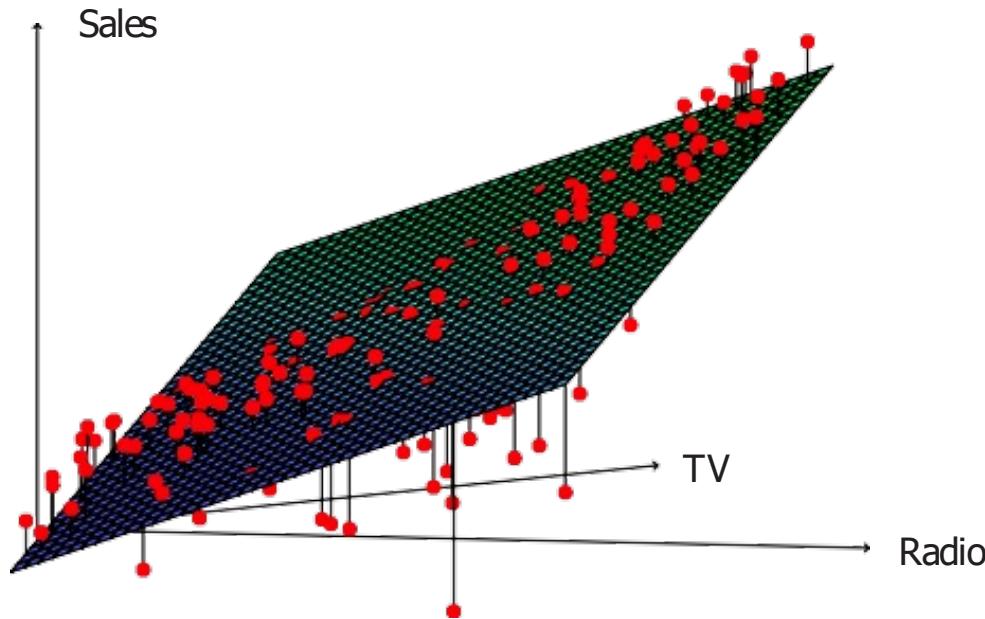
Interactions — continued

- In this situation, given a fixed budget of \$100, 000, spending half on **radio** and half on **TV** may increase **sales** more than allocating the entire amount to either **TV** or to **radio**.

Interactions — continued

- In marketing, this is known as a *synergy* effect, and in statistics it is referred to as an *interaction* effect.

Interaction in the Advertising data?



When levels of either **TV** or **radio** are low, then the true **sales** are lower than predicted by the linear model.

But when advertising is split between the two media, then the model tends to underestimate **sales**.

Modelling interactions — Advertising data

Model takes the form

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

Results:

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

Interpretation

- The results in this table suggest that interactions are important.
- The p-value for the interaction term **TV×radio** is extremely low, indicating that there is strong evidence for $H_A: \beta_3 \neq 0$.

Interpretation

- The R^2 for the interaction model is 96.8%, compared to only 89.7% for the model that predicts **sales** using **TV** and **radio** without an interaction term.

Interpretation — continued

- This means that

$(96.8 - 89.7)/(100 - 89.7) = 69\%$ of the variability in **sales** that remains after fitting the additive model has been explained by the interaction term.

Interpretation — continued

- The coefficient estimates in the table suggest that an increase in TV advertising of \$1, 000 is associated with increased sales of $(\hat{\beta}_1 + \hat{\beta}_3 \times \text{radio}) \times 1000 = 19 + 1.1 \times \text{radio}$ units.

Interpretation — continued

- An increase in radio advertising of \$1, 000 will be associated with an increase in sales of $(\hat{\beta}_2 + \hat{\beta}_3 \times \text{TV}) \times 1000 = 29 + 1.1 \times \text{TV}$ units.

Hierarchy

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

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

Hierarchy — continued

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

Interaction between Quantitative and Qualitative Variables

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

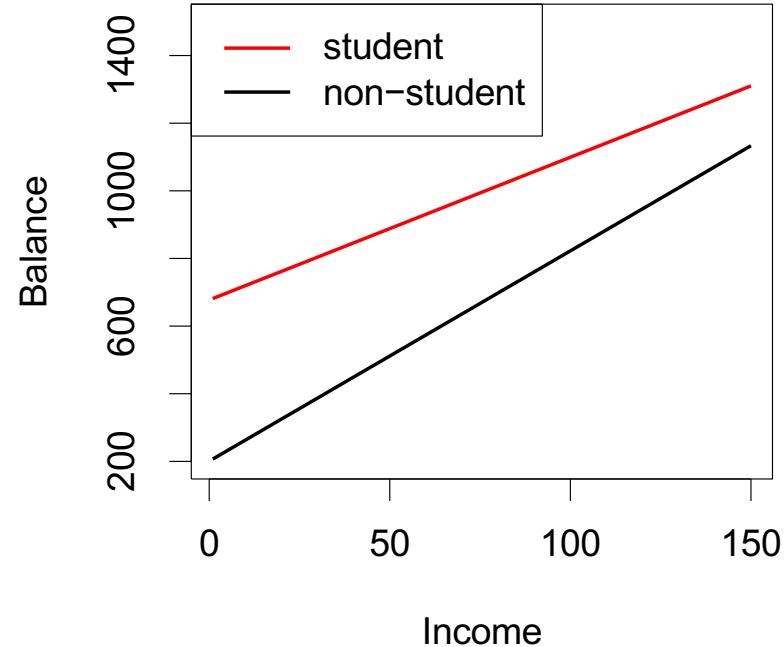
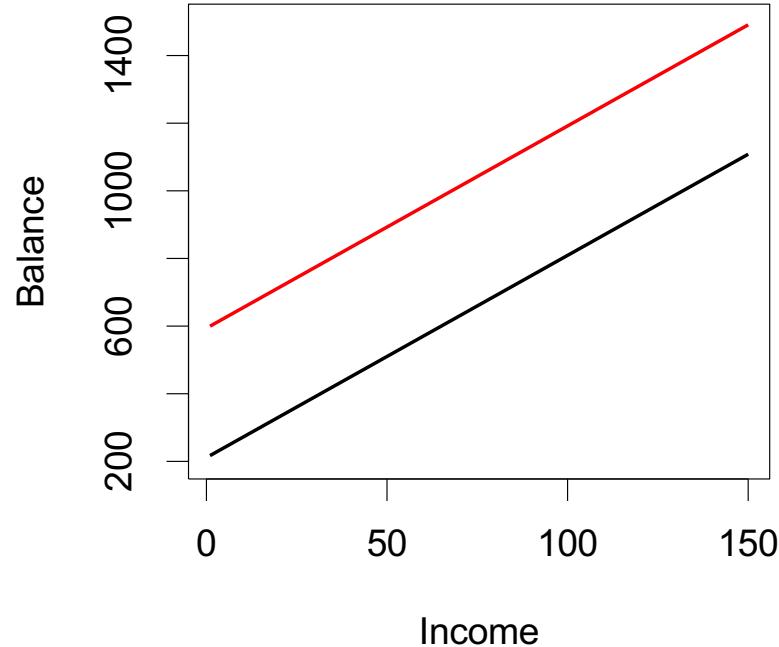
Without an interaction term, the model takes the form

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

Interaction between Quantitative and Qualitative Variables

With interactions, it takes the form

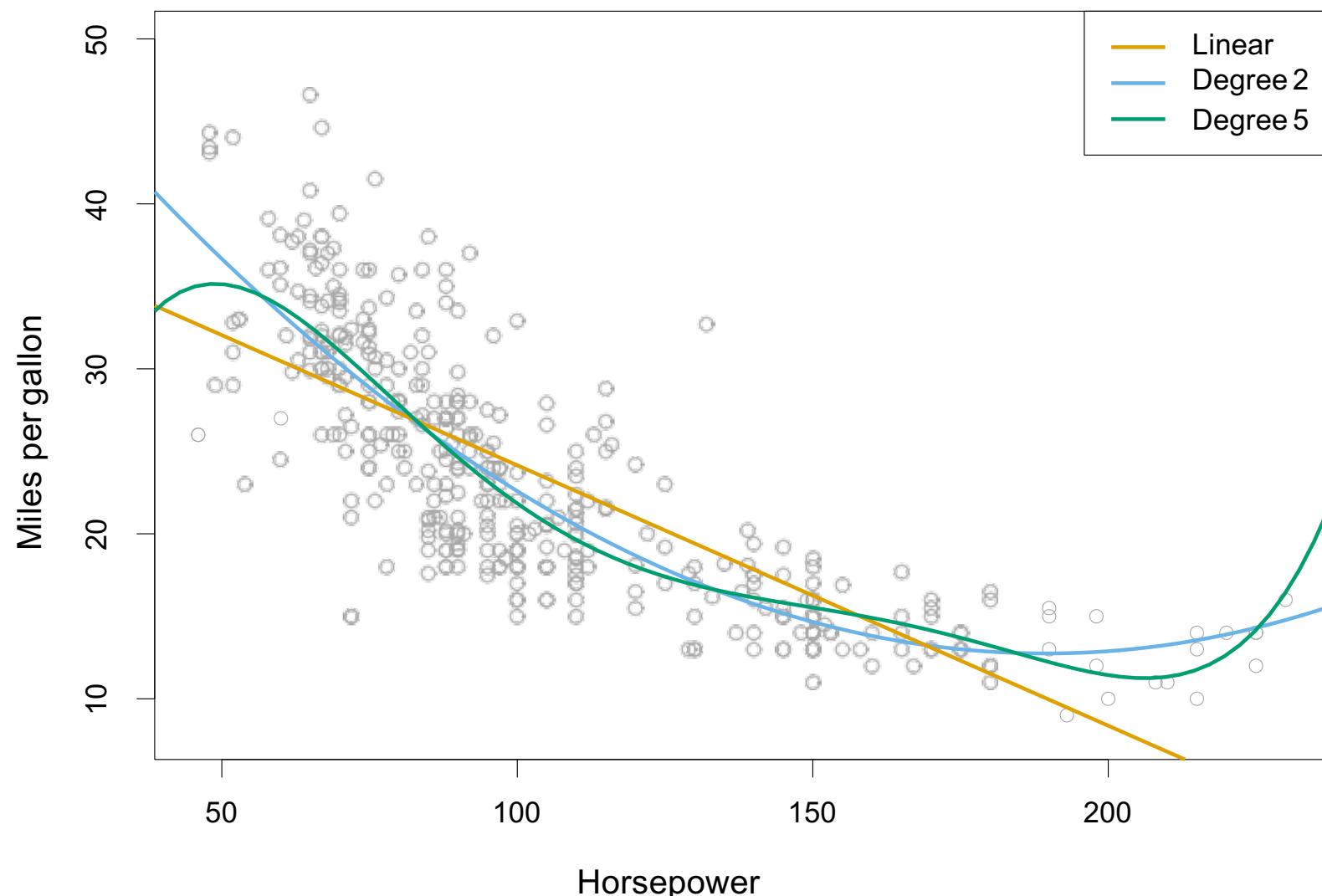
$$\begin{aligned}\text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 + \beta_3 \times \text{income}_i & \text{if student} \\ 0 & \text{if not student} \end{cases} \\ &= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \text{income}_i & \text{if student}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



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

Non-linear effects of predictors

polynomial regression on Auto data



The figure suggests that

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

may provide a better fit.

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

What we did not cover

Outliers

Non-constant variance of error terms

High leverage points

Collinearity

See text Section 3.33

Generalizations of the Linear Model

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

Generalizations of the Linear Model

- *Classification problems:* logistic regression, support vector machines
- *Non-linearity:* kernel smoothing, splines and generalized additive models; nearest neighbor methods.

Generalizations of the Linear Model

- *Interactions*: Tree-based methods, bagging, random forests and boosting (these also capture nonlinearities)
- *Regularized fitting*: Ridge regression and lasso

Appendix: More on Qualitative/ Categorical Variables

Material from:

<https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/>

Qualitative/ Categorical Variables

- Challenges faced while dealing with categorical variables:
 - A categorical variable has too many levels. This pulls down performance level of the model. For example, a cat. variable “zip code” would have numerous levels.

Qualitative/ Categorical Variables

- Challenges faced while dealing with categorical variables:
 - A categorical variable has levels which rarely occur. Many of these levels have minimal chance of making a real impact on model fit. For example, a variable ‘disease’ might have some levels which would rarely occur.

Qualitative/ Categorical Variables

- Challenges faced while dealing with categorical variables:
 - There is one level which always occurs i.e. for most of the observations in data set there is only one level. Variables with such levels fail to make a positive impact on model performance due to very low variation.

Qualitative/ Categorical Variables

- Challenges faced while dealing with categorical variables:
 - If the categorical variable is masked, it becomes a laborious task to decipher its meaning. Such situations are commonly found in data science competitions.

Qualitative/ Categorical Variables

- Challenges faced while dealing with categorical variables:
 - You can't fit categorical variables into a regression equation in their raw form. They must be treated.

Qualitative/ Categorical Variables

- Challenges faced while dealing with categorical variables:
 - Most of the algorithms (or ML libraries) produce better result with numerical variable. In python, library “sklearn” requires features in numerical arrays.

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Label Encoder: It is used to transform non-numerical labels to numerical labels (or nominal categorical variables). Numerical labels are always between 0 and n_classes-1.

Methods to deal with Qualitative/Categorical Variables

Label Encoder:

```
In [53]: train.head(5)
```

```
out[53]:
```

	sex	pclass
0	male	3
1	female	1
2	female	3
3	female	1
4	male	3

```
In [54]: from sklearn.preprocessing import LabelEncoder
```

```
number = LabelEncoder()
train['sex'] = number.fit_transform(train['sex'].astype('str'))
test['sex'] = number.fit_transform(test['sex'].astype('str'))
```

```
train.head(5)
```

```
out[54]:
```

	sex	pclass
0	1	3
1	0	1
2	0	3
3	0	1
4	1	3

Methods to deal with Qualitative/ Categorical Variables

Label Encoder: A common challenge with nominal categorical variable is that, it may decrease performance of a model. For example: We have two features “age” (range: 0-80) and “city” (81 different levels).

Methods to deal with Qualitative/ Categorical Variables

Now, when we'll apply label encoder to 'city' variable, it will represent 'city' with numeric values range from 0 to 80. The 'city' variable is now similar to 'age' variable since both will have similar data points, which is certainly not a right approach.

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Convert numeric bins to number: Let's say, bins of a continuous variable are available in the data set (shown next).

Methods to deal with Qualitative/ Categorical Variables

Convert to Number
Convert numeric bins to number

User_ID	Product_ID	Gender	Age	Occupatio	City_Cate	Stay_In_C	Marital_St	Product_C	Product_C	Product_C	Purchase
1000001	P00069042	F	0-17	10	A	2	0	3			8370
1000001	P00248942	F	0-17	10	A	2	0	1	6	14	15200
1000001	P00087842	F	0-17	10	A	2	0	12			1422
1000001	P00085442	F	0-17	10	A	2	0	12	14		1057
1000002	P00285442	M	55+	16	C	4+	0	8			7969
1000003	P00193542	M	26-35	15	A	3	0	1	2		15227
1000004	P00184942	M	46-50	7	B	2	1	1	8	17	19215
1000004	P00346142	M	46-50	7	B	2	1	1	15		15854
1000004	P0097242	M	46-50	7	B	2	1	1	16		15686
1000005	P00274942	M	26-35	20	A	1	1	8			7871
1000005	P00251242	M	26-35	20	A	1	1	5	11		5254

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Convert numeric bins to number:

Variable “Age” has bins (0-17, 17-25, 26-35 ...). We can convert these bins into definite numbers using the following methods:

Using label encoder for conversion. But, these numerical bins will be treated same as multiple levels of non-numeric feature. Hence, wouldn’t provide any additional information

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Convert numeric bins to number:

Variable “Age” has bins (0-17, 17-25, 26-35 ...). We can convert these bins into definite numbers using the following methods:

Create a new feature using mean or mode (most relevant value) of each age bucket.
It would comprise of additional weight for levels.

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Convert numeric bins to number:

User_ID	Product_ID	Gender	Age	New_Age	Occupatio	City_Cate	Stay_In_C	Marital_St	Product_C	Product_C	Product_C	Purchase	
1000001	P00069042	F	0-17	14	10	A		2	0	3		8370	
1000001	P00248942	F	0-17	14	10	A		2	0	1	6	14	15200
1000001	P00087842	F	0-17	14	10	A		2	0	12			1422
1000001	P00085442	F	0-17	14	10	A		2	0	12	14		1057
1000002	P00285442	M	55+	60	16	C	4+		0	8			7969
1000003	P00193542	M	26-35	30	15	A		3	0	1	2		15227
1000004	P00184942	M	46-50	47	7	B		2	1	1	8	17	19215
1000004	P00346142	M	46-50	47	7	B		2	1	1	15		15854
1000004	P0097242	M	46-50	47	7	B		2	1	1	16		15686
1000005	P00274942	M	26-35	30	20	A		1	1	8			7871
1000005	P00251242	M	26-35	30	20	A		1	1	5	11		5254

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Convert numeric bins to number:

Variable “Age” has bins (0-17, 17-25, 26-35 ...). We can convert these bins into definite numbers using the following methods:

Create two new features, one for lower bound of age and another for upper bound. In this method, we’ll obtain more information about these numerical bins compare to earlier two methods.

Methods to deal with Qualitative/ Categorical Variables

Convert to Number

Convert numeric bins to number:

User_ID	Product_ID	Gender	Age	Lower_Age	Upper_Age	Occupatio	City_Cate	Stay_In_C	Marital_St	Product_C	Product_C	Product_CPurchase	
1000001	P00069042	F	0-17	0	17	10 A		2	0	3		8370	
1000001	P00248942	F	0-17	0	17	10 A		2	0	1	6	14	15200
1000001	P00087842	F	0-17	0	17	10 A		2	0	12			1422
1000001	P00085442	F	0-17	0	17	10 A		2	0	12	14		1057
1000002	P00285442	M	55+	55	80	16 C	4+		0	8			7969
1000003	P00193542	M	26-35	26	35	15 A		3	0	1	2		15227
1000004	P00184942	M	46-50	46	50	7 B		2	1	1	8	17	19215
1000004	P00346142	M	46-50	46	50	7 B		2	1	1	15		15854
1000004	P0097242	M	46-50	46	50	7 B		2	1	1	16		15686
1000005	P00274942	M	26-35	26	35	20 A		1	1	8			7871
1000005	P00251242	M	26-35	26	35	20 A		1	1	5	11	.	5254

Methods to deal with Qualitative/ Categorical Variables

Combine Levels: one can sometimes simply combine the different levels. There are various methods of combining levels.
Here are commonly used ones:
Using Business Logic

Methods to deal with Qualitative/ Categorical Variables

Combine Levels: Using Business Logic

For example, we can combine levels of a variable “zip code” at state or district level. This will reduce the number of levels and improve the model performance also.

Methods to deal with Qualitative/ Categorical Variables

Combine Levels: Using Business Logic

Zip Code	District
110044	South Delhi
110048	South Delhi
110049	South Delhi
110006	North Delhi
110007	North Delhi
110058	West Delhi
110059	West Delhi
110063	West Delhi
110064	West Delhi

Methods to deal with Qualitative/ Categorical Variables

Combine Levels:

Using frequency or response rate:

When we don't have domain knowledge about the levels, we combine levels by considering the frequency distribution or response rate.

Methods to deal with Qualitative/ Categorical Variables

Combine Levels:

Using frequency or response rate:

Consider the frequency distribution of each level and combine levels having frequency less than 5% of total observation (5% is standard but you can change it based on distribution). This is an effective method to deal with rare levels.

Methods to deal with Qualitative/ Categorical Variables

Combine Levels:

Using frequency or response rate:

We can also combine levels by considering the response rate of each level. We can simply combine levels having similar response rate into same group.

Methods to deal with Qualitative/ Categorical Variables

Combine Levels:

Using frequency or response rate:

Finally, you can also look at both frequency and response rate to combine levels. You first combine levels based on response rate then combine rare levels to relevant group.

Methods to deal with Qualitative/Categorical Variables

Combine Levels:

Based on Frequency

Levels	Frequency	New_Level
HA001	9%	HA001
HA002	12%	HA002
HA003	4%	New
HA004	1%	New
HA005	3%	New
HA006	11%	HA006
HA007	1%	New
HA008	4%	New
HA009	10%	HA009
HA010	4%	New
HA011	8%	HA011
HA012	12%	HA012
HA013	3%	New
HA014	11%	HA014
HA015	2%	New
HA016	4%	New
HA017	0%	New

Based on Response Rate

Levels	Response_Rate	New_Level
HA014	98%	1
HA001	97%	1
HA003	93%	1
HA009	81%	2
HA015	75%	3
HA010	73%	3
HA006	66%	4
HA017	60%	4
HA007	49%	5
HA004	36%	6
HA005	31%	6
HA012	28%	7
HA008	25%	7
HA013	23%	7
HA016	22%	7
HA002	21%	8
HA011	5%	9

Based on Frequency and Response Rate

Levels	Frequency	Response_Rate	New_Level1	New_Level2
HA014	11%	98%	1	1
HA001	9%	97%	1	1
HA003	4%	93%	1	1
HA009	10%	81%	2	2
HA015	2%	75%	3	2
HA010	4%	73%	3	2
HA006	11%	66%	4	4
HA017	0%	60%	4	4
HA007	1%	49%	5	4
HA004	1%	36%	6	4
HA005	3%	31%	6	4
HA012	12%	28%	7	7
HA008	4%	25%	7	7
HA013	3%	23%	7	7
HA016	4%	22%	7	7
HA002	12%	21%	8	8
HA011	8%	5%	9	9

Methods to deal with Qualitative/ Categorical Variables

Dummy Coding

Dummy coding is a commonly used method for converting a categorical input variable into continuous variable. ‘Dummy’, as the name suggests is a duplicate variable which represents one level of a categorical variable.

Methods to deal with Qualitative/ Categorical Variables

Dummy Coding

Presence of a level is represented by 1 and absence is represented by 0. For every level present, one dummy variable will be created. Look at the representation below to convert a categorical variable using dummy variable.

Methods to deal with Qualitative/Categorical Variables

Dummy Coding

```
In [46]: train.head(5)
```

```
Out[46]:
```

	sex	pclass
0	male	3
1	female	1
2	female	3
3	female	1
4	male	3

```
In [47]: train=train=pd.get_dummies(train)  
train.head(5)
```

```
Out[47]:
```

	pclass	sex_female	sex_male
0	3	0	1
1	1	1	0
2	3	1	0
3	1	1	0
4	3	0	1

Methods to deal with Qualitative/ Categorical Variables

Dummy Coding

Note: Assume, we have 500 levels in categorical variables. Then, should we create 500 dummy variables? If you can automate it, very well. Or else, I'd suggest you to first, reduce the levels by using combining methods and then use dummy coding. This would save your time. This method is also known as “One_Hot Encoding”.

Methods to deal with Qualitative/ Categorical Variables

Feature Hashing

Read:

<https://blog.myyellowroad.com/using-categorical-data-in-machine-learning-with-python-from-dummy-variables-to-deep-category-66041f734512>

DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 3

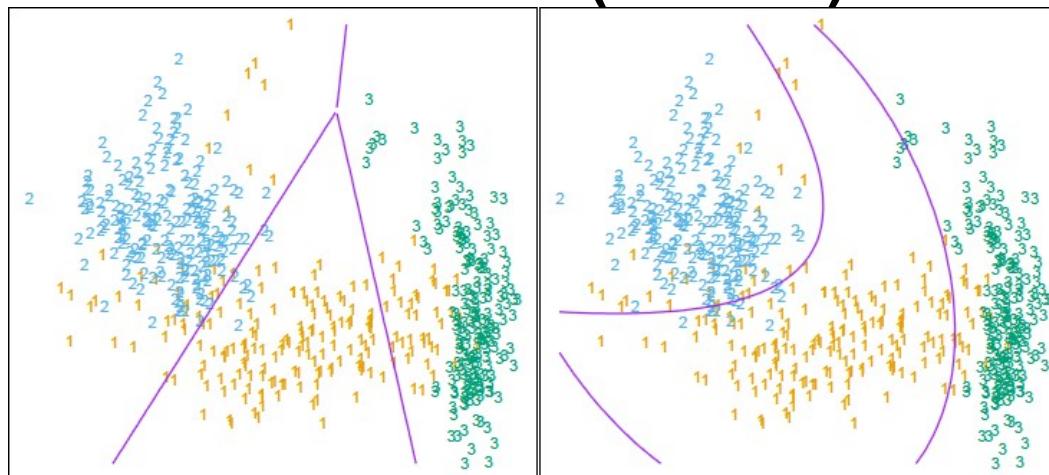
Classification

Linear and Logistic Regression:

LDA, QDA

k-NN (k Nearest Neighbors)

optimal separating hyperplane – will
be discussed later (SVM)



Classification

- Qualitative variables take values in an *unordered* set C , such as:
 $\text{eye color} \in \{\text{brown, blue, green}\}$
 $\text{digit} \in \{0, 1, \dots, 9\}$
 $\text{email} \in \{\text{spam, ham}\}$.
- Given a feature vector X and a qualitative response Y taking values in the set C , the classification task is to build a function $C(X)$ that takes as input the feature vector X and predicts its value for Y ; i.e. $C(X) \in C$.

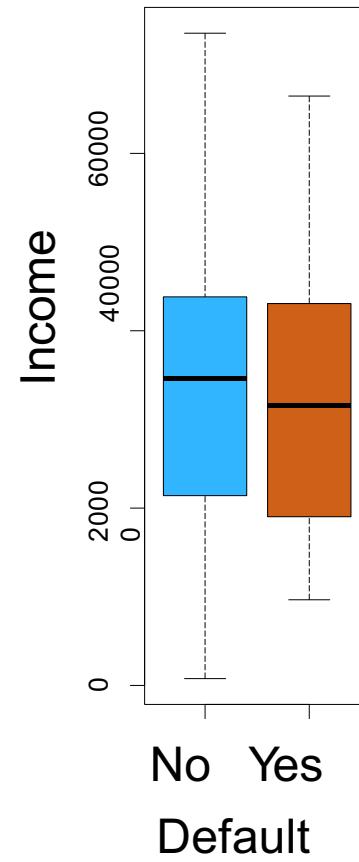
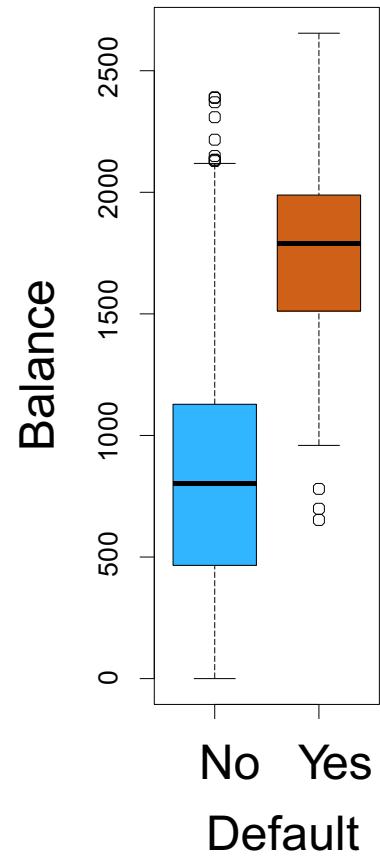
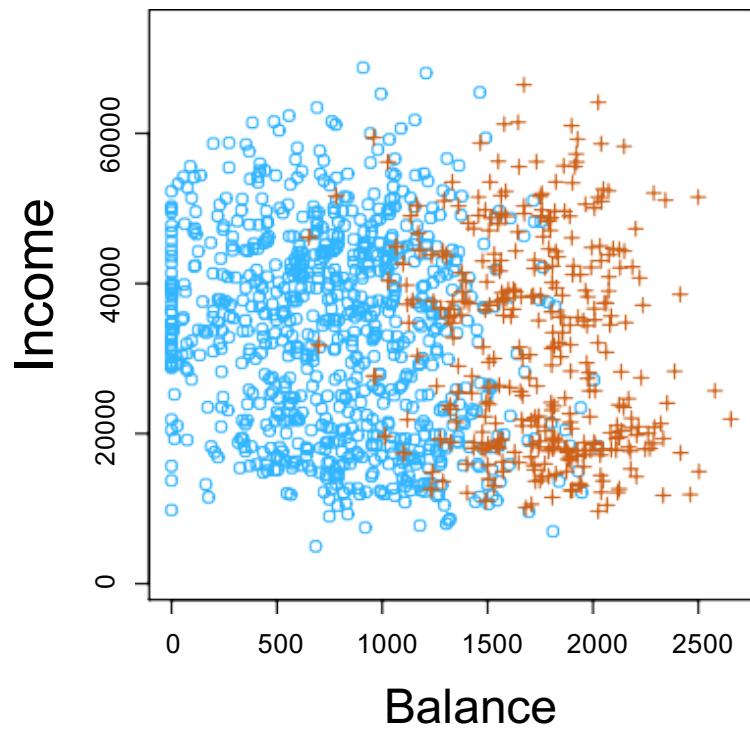
Classification

- Often we are more interested in estimating the *probabilities* that X belongs to each category in C .

Case: Credit Card Default Data

- To predict customers that are likely to default
- **Possible X** variables are:
 - Annual Income
 - Monthly credit card balance
- The Y variable (Default) is categorical: Yes or No
- How do we check the relationship between Y and X ?

Example: Credit Card Defualt



Can we use Linear Regression?

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

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

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

Can we use Linear Regression?

Can we simply perform a linear regression of Y on X and classify as

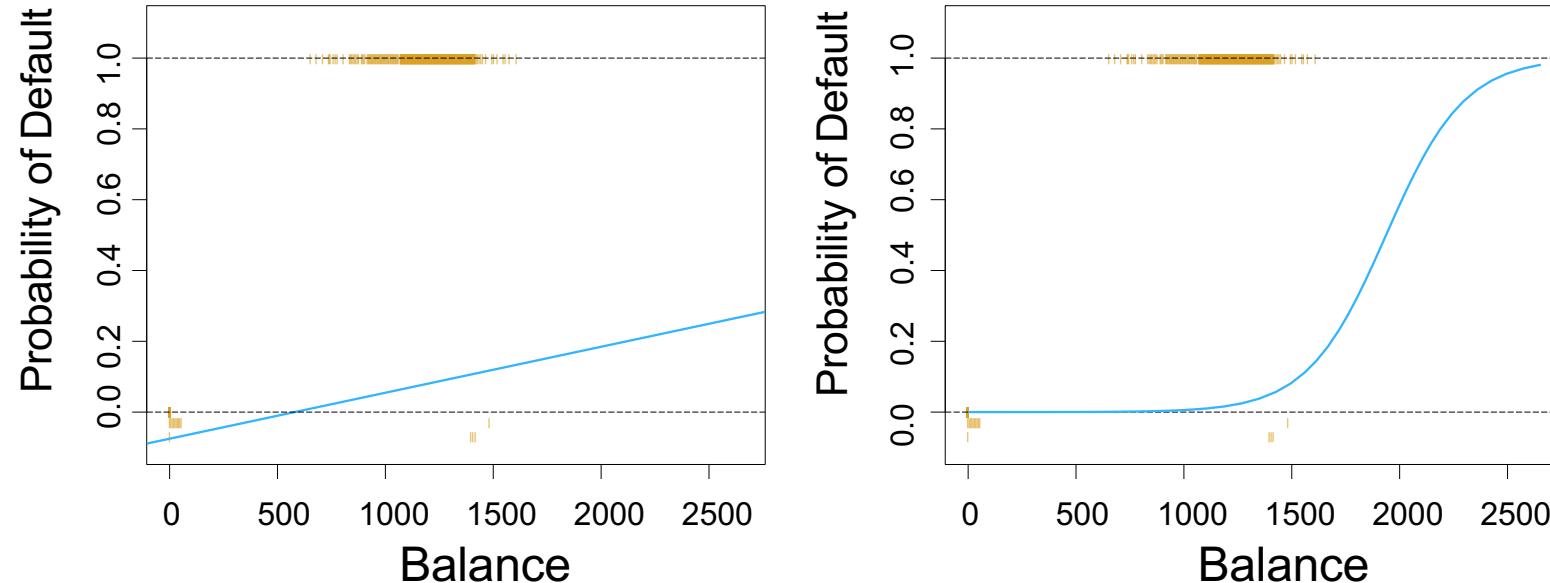
Yes if $\hat{Y} > 0.5$?

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

Can we use Linear Regression?

- However, *linear* regression might produce probabilities less than zero or bigger than one.
Logistic regression is more appropriate.

Linear versus Logistic Regression



The orange marks indicate the response Y , either 0 or 1.

Linear regression does not estimate $\Pr(Y = 1|X)$ well. Logistic regression seems well suited to the task.

Linear Regression continued

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

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

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

Linear Regression continued

- This coding suggests an ordering, and in fact implies that the difference between **stroke** and **drug overdose** is the same as between **drug overdose** and **epileptic seizure**.
- Linear regression is not appropriate here.
- *Multiclass Logistic Regression* or *Discriminant Analysis* are more appropriate.

Multi-Class and Multi-Label Problems

Multiclass classification means a classification task with more than two classes; e.g., classify a set of images of animals which may be horses, birds, or fish.

Multiclass classification makes the assumption that each sample is **assigned to one and only one label**: an animal can be either a horse or a bird but not both at the same time.

Multi-Class and Multi-Label Problems

Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document.

A text might be about any of religion, politics, finance or education at the same time or none of these.

Binary Classification

A binary classification task assigns only one of the two possible classes to each observation.

Because multi-class and multi-label classification tasks can be performed using binary classification techniques, many times we focus on binary classification.

Logistic Regression

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

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

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

It is easy to see that no matter what values β_0 , β_1 or X take, $p(X)$ will have values between 0 and 1.

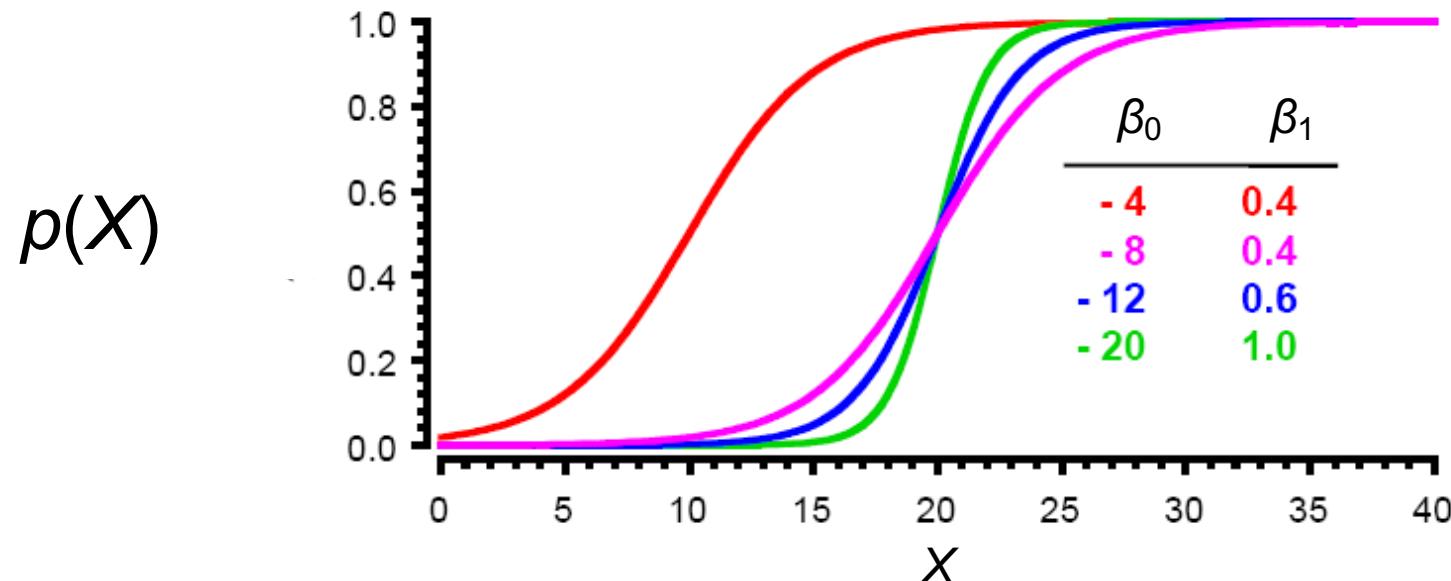
Sigmoid Function

Parameters control shape and location of sigmoid curve

β_0 controls location of midpoint

β_1 controls slope of rise

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



When $x = -\beta_0 / \beta_1$, $\beta_0 + \beta_1 x = 0$; thus $p(X) = 0.5$

Logistic Regression

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

A bit of rearrangement gives

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

This monotone transformation is called the *log odds* or *logit* transformation of $p(X)$.

Definition of Odds

- The probability of an event divided by the probability of its complement is called its odds.
- Example: The probability of winning in a casino is 1%. What is the odds of winning in that casino?

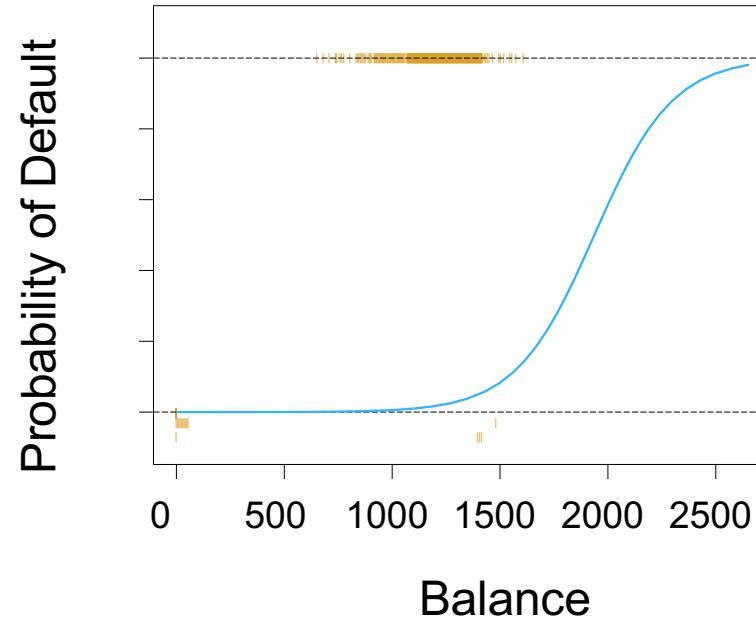
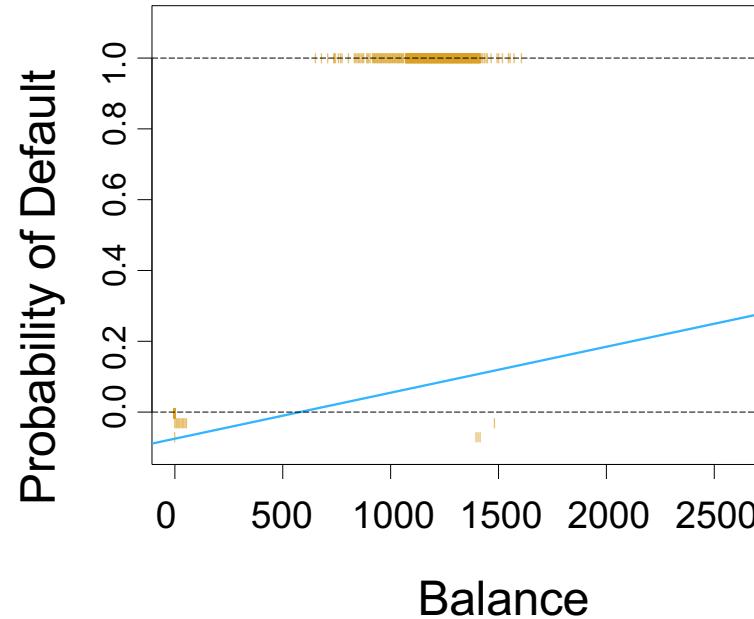
$$\begin{aligned}O(W) &= \Pr(W)/(1-\Pr(W)) \\&= 0.01/0.99 = 1/99\end{aligned}$$

Logit Model

Therefore, the logit model is trying to predict the log of odds of a model as a linear combination of the predictor(s):

$$\log(O(Y=1|X=x)) = \beta_0 + \beta_1 X$$

Linear versus Logistic Regression



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

Class as A Bernoulli Random Variable

One can see class in a binary classification problem as a Bernoulli random variable that can take two values 0 and 1:

$$\Pr(Y=1|X=x) = p(x)$$

$$\Pr(Y=0|X=x) = 1-p(x)$$

This can be rewritten as:

$$p_{Y|X}(y|x) = \Pr(Y=y|X=x) = [p(x)]^y [1-p(x)]^{1-y}, y=0,1$$

Independent Sample of Bernoulli Variables

Assume that we have an *independent* sample whose classes are $Y^{(1)} = y_1, Y^{(2)} = y_2, \dots, Y^{(N)} = y_N$
 y_i is 0 or 1.

The *joint probability mass* function of this independent sample is:

$$\begin{aligned} & p_{Y|X}(y_1, y_2, \dots, y_N | \text{Data}) \\ &= \Pr(Y^{(1)} = y_1, Y^{(2)} = y_2, \dots, Y^{(N)} = y_N | \text{Data}) \\ &= \Pr(Y^{(1)} = y_1 | \text{Data}) \Pr(Y^{(2)} = y_2 | \text{Data}) \dots \Pr(Y^{(N)} = y_N | \text{Data}) \\ & \text{Data: } X^{(1)} = x_1, \dots, X^{(N)} = x_N \end{aligned}$$

Independent Sample of Bernoulli Variables

The assumption is that the probability that $Y^{(i)}=y_i$ is a function of the features $X^{(i)}=x_i$

So

Independent Sample of Bernoulli Variables

The joint probability mass function is a function of β_0 , β_1 and is called the **likelihood function**, given the data samples.

Maximum Likelihood: Simple Example

Your friend gives you a biased coin and tells you that he is sure that the probability of Heads is either 0.1 or 0.5. You flip the coin 100 times and see 90 Heads. What would be your best estimate of the probability of heads, given that you are sure that your friend is right?

Maximum Likelihood: Simple Example

What would be your best estimate of the probability of heads, given that you are sure that your friend is right?

Maximum Likelihood: Simple Example

What would be your best estimate of the probability of heads, given that you are sure that your friend is right?

Maximum Likelihood: Simple Example

What would be your best estimate of the probability of heads, given that you are sure that your friend is right?

This simple example motivates us to estimate parameters from data samples by maximizing their likelihood.

Maximum Likelihood

We use maximum likelihood to estimate the parameters β_0, β_1 .

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

This *likelihood* gives the probability of the observed zeros and ones in the data. We pick β_0 and β_1 to maximize the likelihood of the observed data.

Maximum Likelihood

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

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

Maximum Likelihood

Methods of Maximizing Likelihood:

- 1- Gradient Methods (to be seen in the Lesson on Neural Networks)
- 2- Expectation Maximization

Are the coefficients significant?

- We perform a hypothesis test to see whether β_0 and β_1 are significantly different from zero.
- A Z test is used instead of a T test, but the p-value is interpreted similarly.

$$\begin{aligned} H_0: \beta_i &= 0 \\ H_A: \beta_i &\neq 0 \end{aligned}$$

Are the coefficients significant?

- Because the estimates of β_i 's are Maximum Likelihood Estimates, one can show that assuming that the null hypothesis

$$H_0: \beta_i = \beta$$

is true, the quantity

$$z = \frac{\hat{\beta}_i - \beta}{\text{SE}(\hat{\beta}_i)}$$

Usually zero

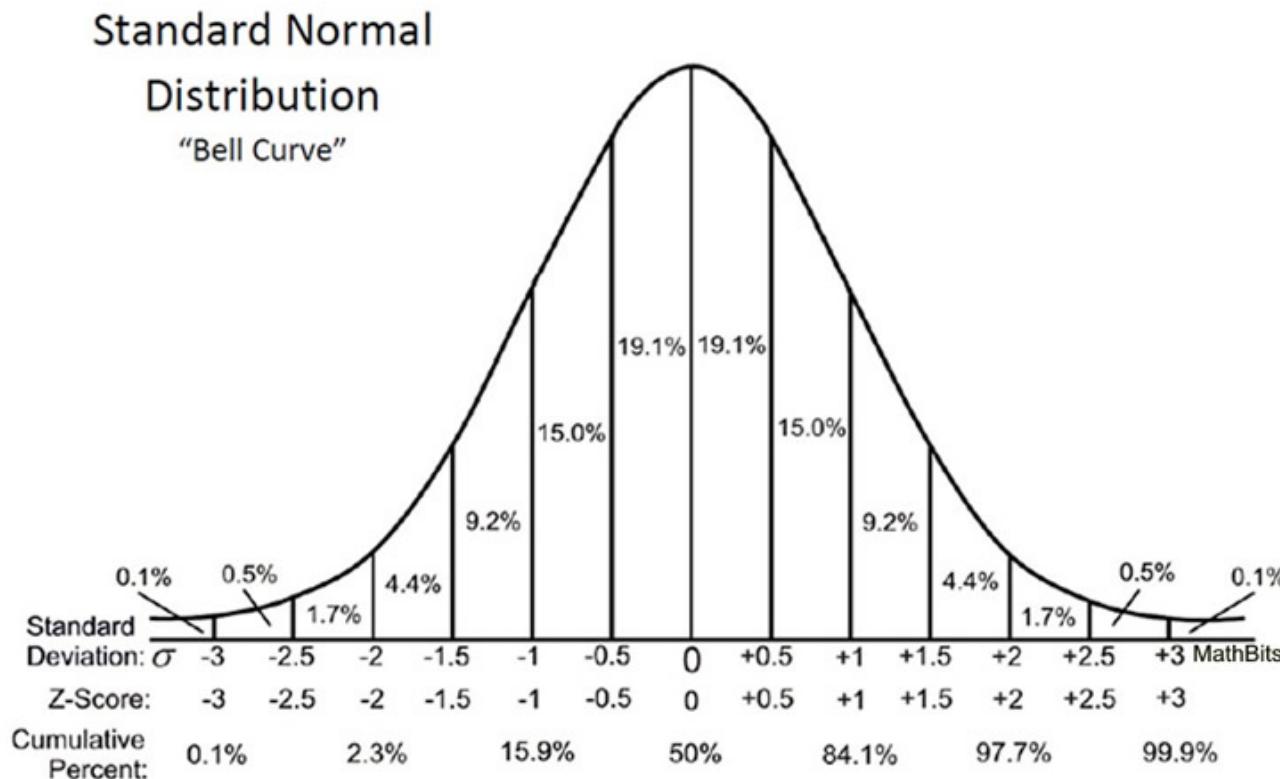
follows a *standard normal distribution* $N(0,1)$.

Are the coefficients significant?

The quantity

$$z = \frac{\hat{\beta}_i - \beta}{\text{SE}(\hat{\beta}_i)}$$

follows a *standard normal distribution* $N(0,1)$.



Are the coefficients significant?

Note: This statement is true for any maximum likelihood estimate, provided that the number of samples is large:

The quantity

$$z = \frac{\hat{\beta}_i - \beta}{\text{SE}(\hat{\beta}_i)}$$

follows a *standard normal distribution* $N(0,1)$ when the null hypothesis is true.

Are the coefficients significant?

The quantity $z = \frac{\hat{\beta}_i - \beta}{\text{SE}(\hat{\beta}_i)}$

follows a *standard normal distribution* $N(0,1)$.

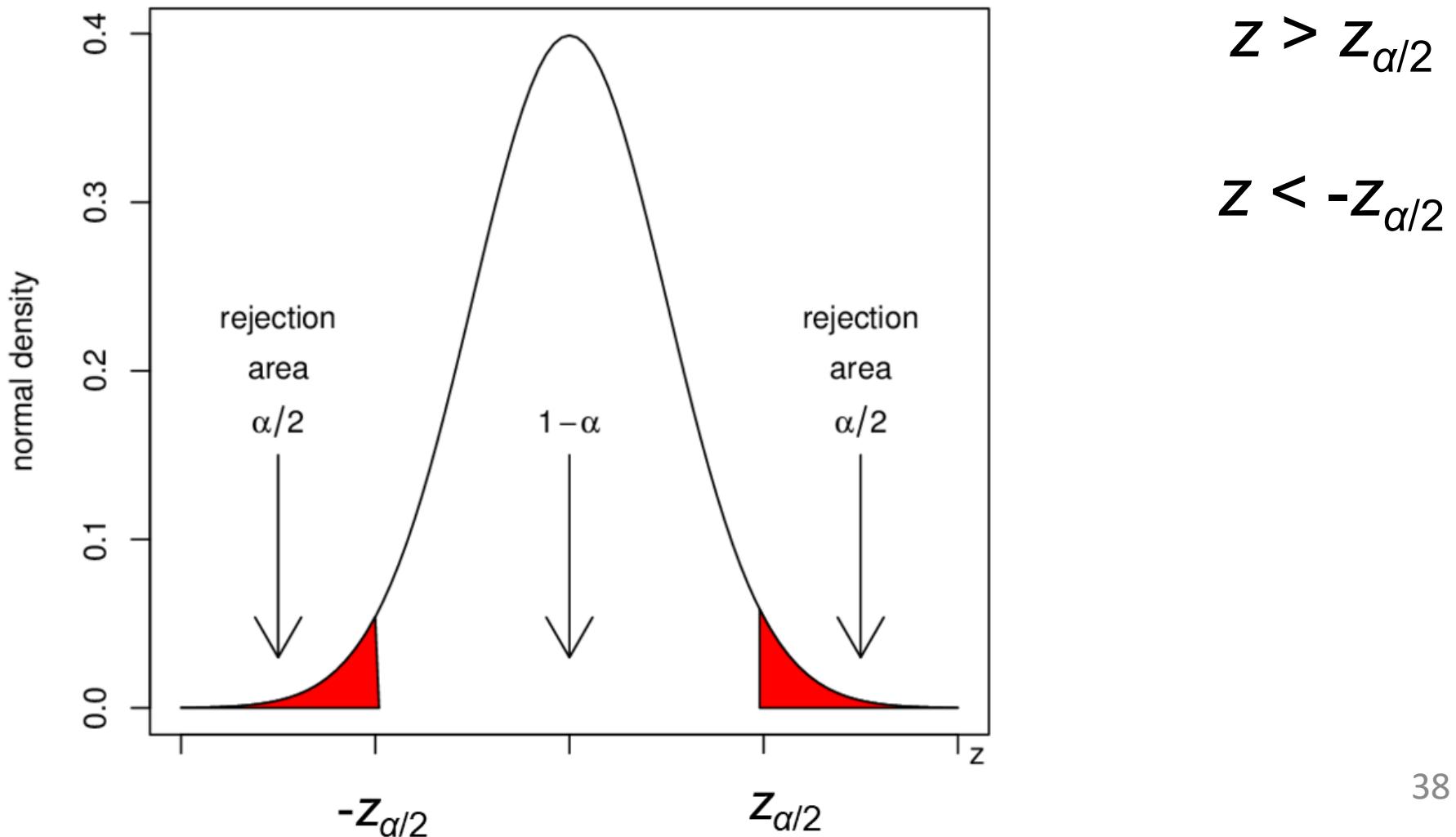
The rejection region is:

$$z > z_{\alpha/2}$$

$$z < -z_{\alpha/2}$$

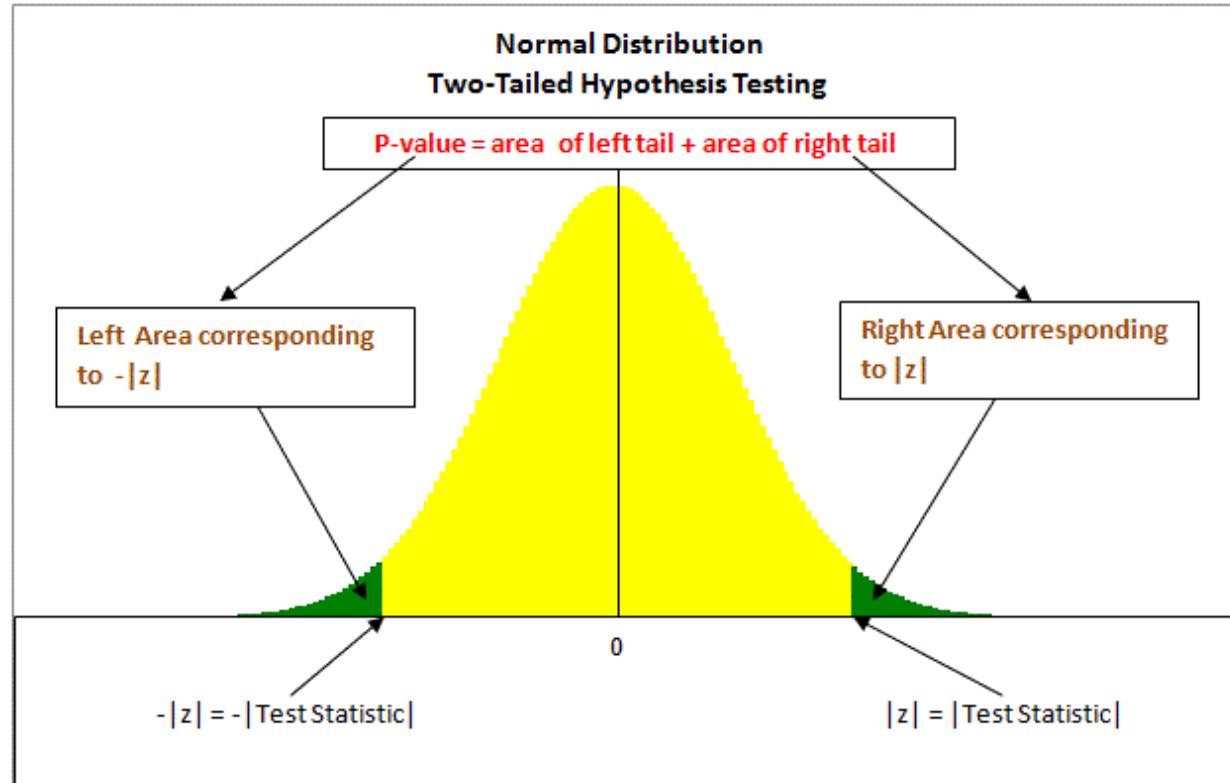
Are the coefficients significant?

The rejection region is:



Are the coefficients significant?

The p-value is the probability of observing something whose magnitude is bigger than $|z|$:



Are the coefficients significant?

- The p-value for balance is very small, and estimate of β_1 is positive, so we are sure that if the balance increase, then the probability of default will increase as well.

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

Predictions

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

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

Predictions

With a balance of \$2000?

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

Lets do it again, using student as the predictor.

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

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

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

Logistic regression with several variables

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

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

Logistic regression is a linear classifier?!

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

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

Logistic regression with several variables

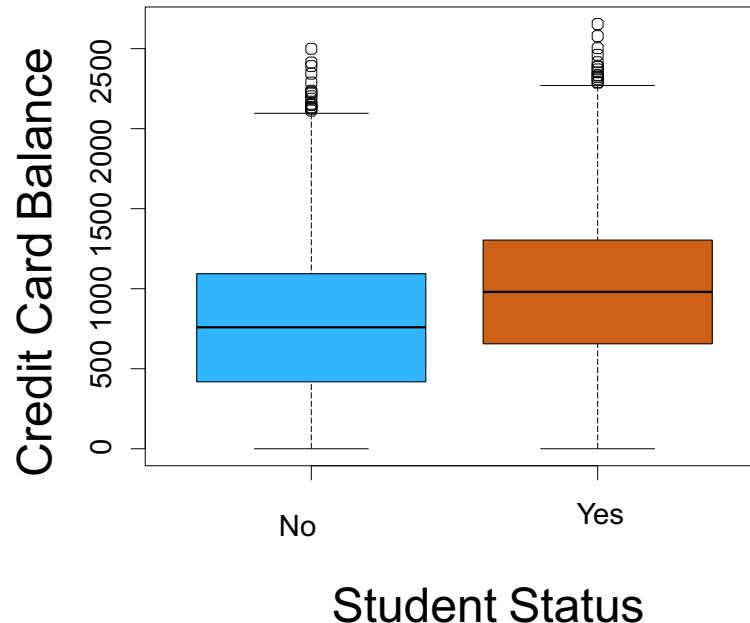
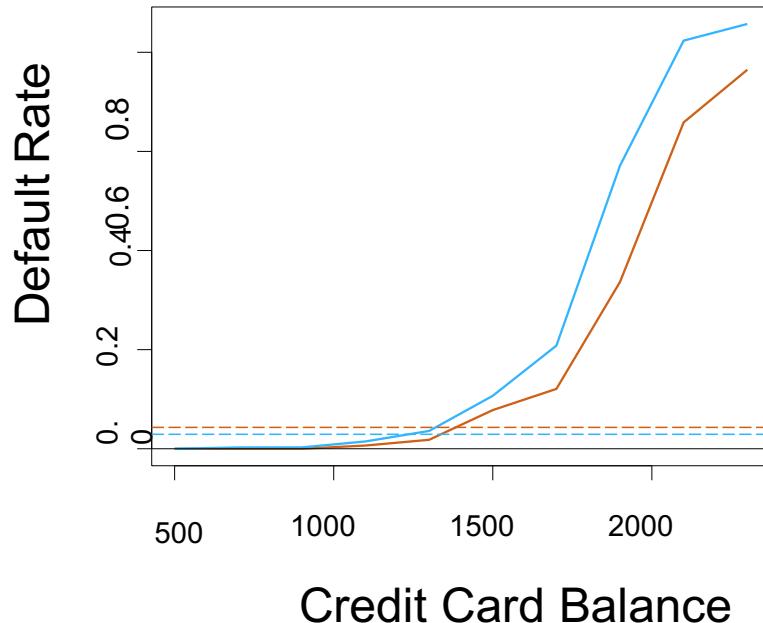
$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

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

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

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

Confounding



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

To whom should credit be offered?

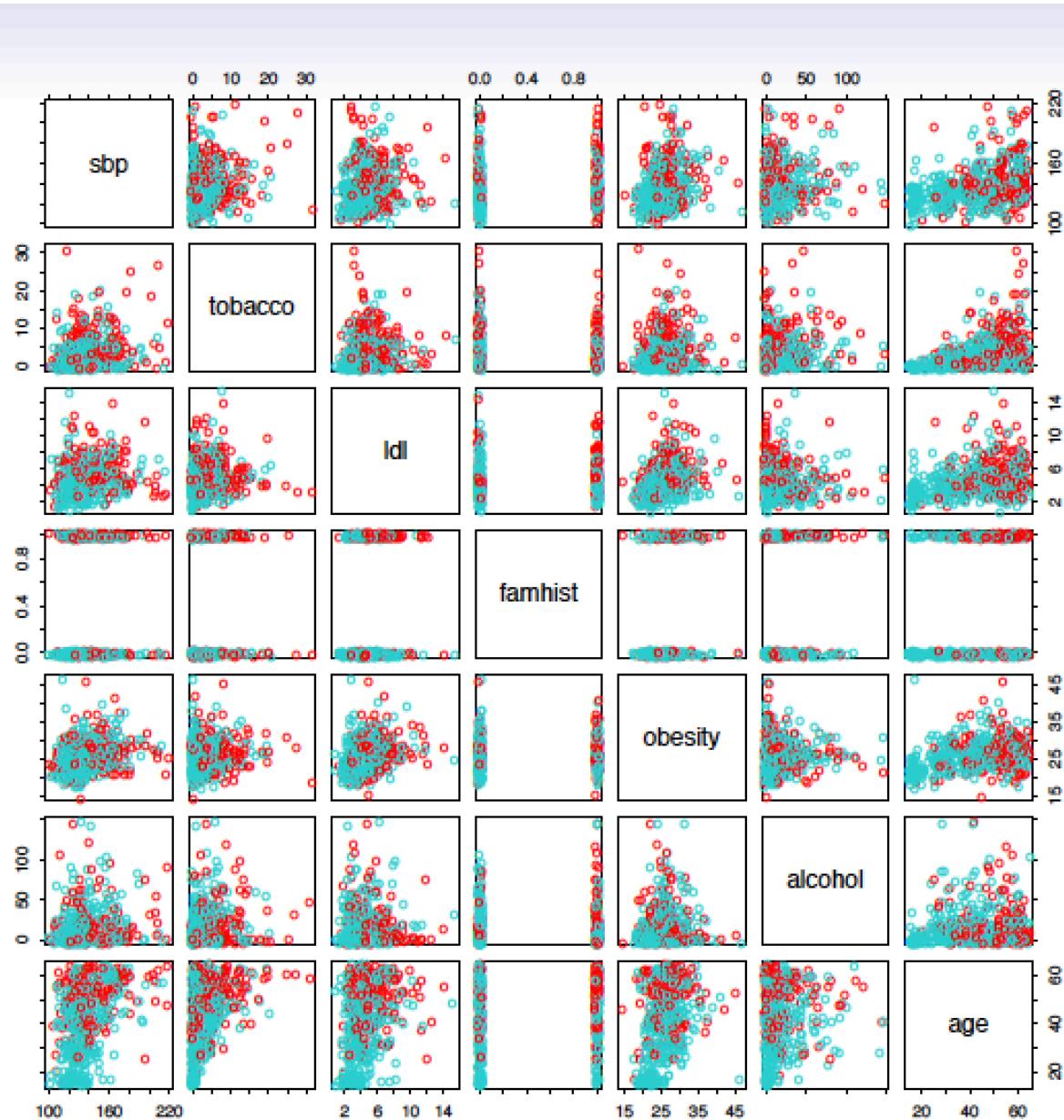
- A student is riskier than non students if no information about the credit card balance is available
- However, that student is less risky than a non student with the same credit card balance!
- Example of **data-driven decision-making**

Example: South African Heart Disease

- 160 cases of MI (myocardial infarction) and 302 controls (all male in age range 15-64), from Western Cape, South Africa in early 80s.
- Overall prevalence very high in this region: 5.1%.

Example: South African Heart Disease

- Measurements on seven predictors (risk factors), shown in scatterplot matrix.
- Goal is to identify relative strengths and directions of risk factors.
- This was part of an intervention study aimed at educating the public on healthier diets.



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

```

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

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

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

(Dispersion parameter for binomial family taken to be 1)

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

```

Class Imbalance

- Intuitively, a dataset is imbalanced when members of certain class(es) are rare.
- The lack of observations of certain classes does not always imply their irrelevance.
- For example, in medical studies of rare diseases, the small number of infected patients (cases) conveys the most valuable information for diagnosis and treatments.

Types of Imbalance

- Formally, an imbalanced dataset exhibits one or more of the following properties:
 - **Marginal Imbalance.** A dataset is marginally imbalanced if one class is rare compared to the other class. In other words, $\Pr(Y=1) \approx 0$.

Types of Imbalance

Marginal Imbalance.

Such imbalance typically occurs in data sets for predicting click-through rates in online advertising, detecting fraud or diagnosing rare diseases.

Types of Imbalance

- Formally, an imbalanced dataset exhibits one or more of the following properties:
 - ***Conditional Imbalance.*** A dataset is conditionally imbalanced when it is easy to predict the correct labels in most cases. For example, if $X \in \{0, 1\}$, the dataset is conditionally imbalanced if $\Pr(Y=1|X=0) \approx 0$ and $\Pr(Y=1|X=1) \approx 1$.

Subsampling (Down-sampling)

- Typically in such problems, the statistical noise is primarily driven by the number of representatives of the rare class
- We might hope to remedy the problem by subsampling the training set in a way that enriches for the rare class.

Subsampling (Down-sampling)

However, if the training set is **randomly sampled** to be balanced, the test set should be sampled to be more consistent with the state of nature and should reflect the imbalance so that honest estimates of future performance can be computed.

Upsampling

- Ling and Li (1998) provide an approach to up-sampling in which cases from the minority classes are sampled with replacement until each class has approximately the same size.
- Some minority class samples may show up in the training set with a fairly high frequency

Ling C, Li C (1998). “Data Mining for Direct Marketing: Problems and solutions.” In “Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining,” pp. 73–79.

SMOTE

- The *synthetic minority over-sampling technique* (SMOTE) is a data sampling procedure that uses both up-sampling and down-sampling, depending on the class
- To up-sample for the minority class, SMOTE synthesizes new cases. To do this, a data point is randomly selected from the minority class and its K -nearest neighbors (KNNs) are determined.

Chawla N, Bowyer K, Hall L, Kegelmeyer W (2002). “SMOTE: Synthetic Minority Over-Sampling Technique.” Journal of Artificial Intelligence Research, 16 (1), 321–357.

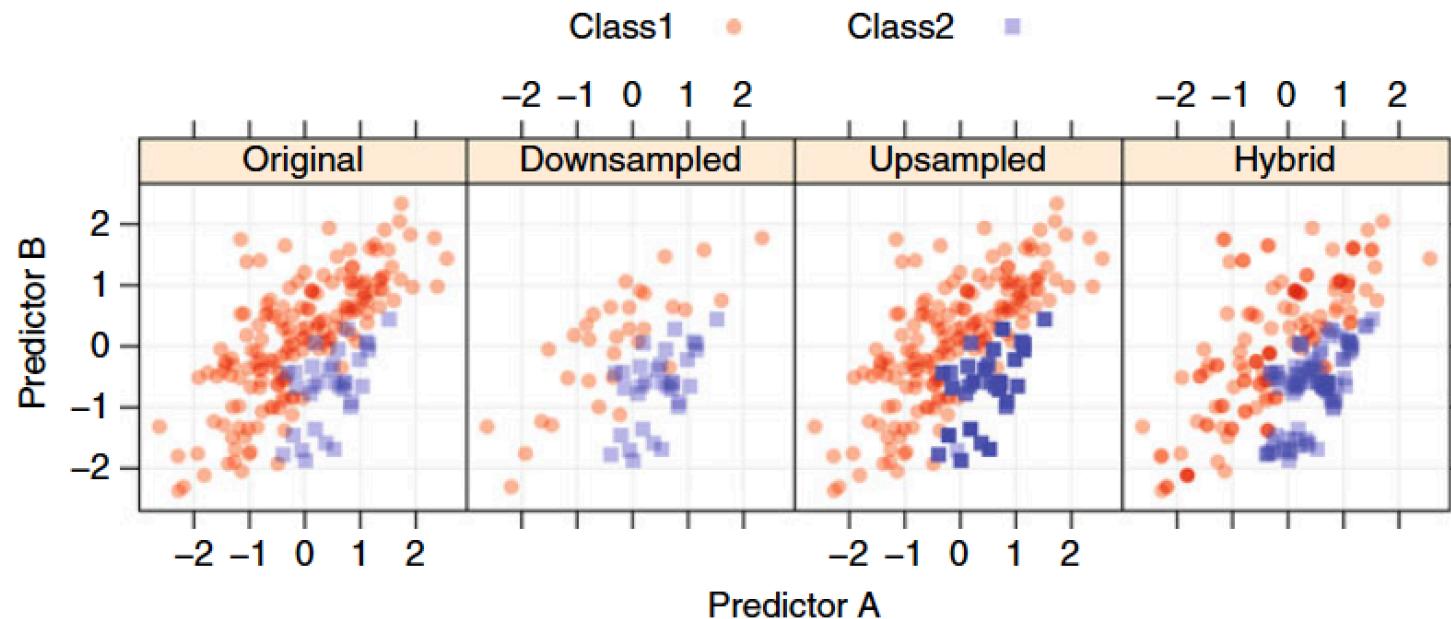
SMOTE

- The new synthetic data point is a random combination of the predictors of the randomly selected data point and its neighbors.
- SMOTE can down-sample cases from the majority class via random sampling.
- Three operational parameters:
 - the amount of up-sampling,
 - the amount of down-sampling,
 - and the number of neighbors

Chawla N, Bowyer K, Hall L, Kegelmeyer W (2002). “SMOTE: Synthetic Minority Over-Sampling Technique.” *Journal of Artificial Intelligence Research*, 16 (1), 321–357.

SMOTE

From left to right: The original simulated data set and realizations of a down-sampled version, an up-sampled version, and sampling using SMOTE



Case-control sampling and logistic regression

- In South African data, there are 160 cases, 302 controls — $\tilde{\pi} = 0.35$ are cases. Yet the prevalence of MI in this region is $\pi = 0.05$.
- With case-control samples, we can estimate the regression parameters β_j accurately (if our model is correct); the constant term β_0 is incorrect.

Case-control sampling and logistic regression

- Can correct the estimated intercept by a simple transformation

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

- Why?

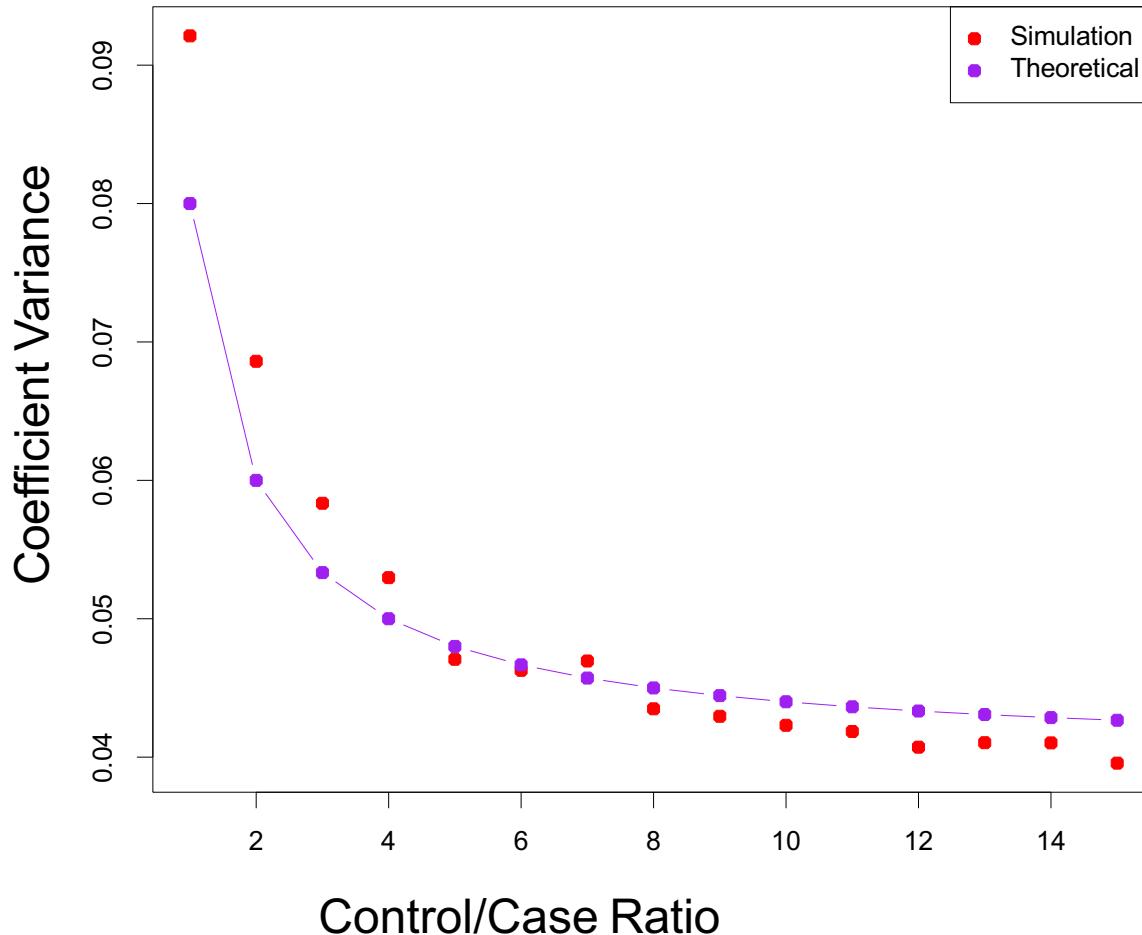
Case-control sampling and logistic regression

- Can correct the estimated intercept by a simple transformation

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

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

Diminishing returns in unbalanced binary data



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

Logistic regression with more than two classes

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

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

Logistic regression with more than two classes

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

Here there is a linear function for *each* class. (Examine the logit model)

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

Logistic regression with more than two classes

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

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

(Bayesian) Discriminant Analysis

- Here the approach is to model the distribution of X in each of the classes separately, and then use *Bayes theorem* to flip things around and obtain $\Pr(Y|X)$.
- When we use normal (Gaussian) distributions for each class, this leads to linear or quadratic discriminant analysis.

Discriminant Analysis

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

Bayes theorem for classification

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

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

Bayes theorem for classification

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

One writes this slightly differently for discriminant analysis:

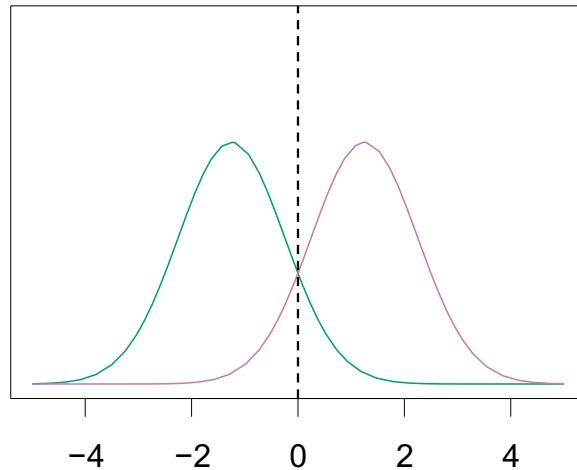
$$\Pr(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

where

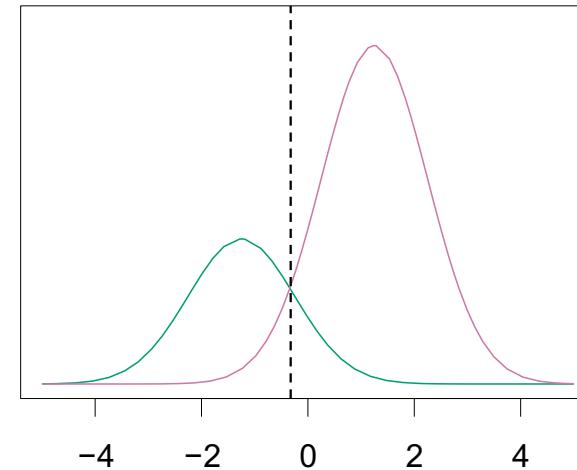
- $f_k(x) = \Pr(X = x | Y = k)$ is the density for X in class k . Here we will use normal densities for these, separately in each class.
- $\pi_k = \Pr(Y = k)$ is the marginal or prior probability for class k .

Classify to the highest density

$$\pi_1 = 0.5, \pi_2 = 0.5$$



$$\pi_1 = 0.3, \pi_2 = 0.7$$



- We classify a new point according to which density is highest.
- When the priors are different, we take them into account as well, and compare $\pi_k f_k(x)$. On the right, we favor the pink class — the decision boundary has shifted to the left.

Why discriminant analysis?

- When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis does not suffer from this problem.

Why discriminant analysis?

- If n is small and the distribution of the predictors X is approximately normal in each of the classes, the linear discriminant model is again more stable than the logistic regression model.

Why discriminant analysis?

- Linear discriminant analysis is popular when we have more than two response classes, because it also provides low-dimensional views of the data.

Linear Discriminant Analysis when $p = 1$

The Gaussian density has the form

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

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

Linear Discriminant Analysis when $p = 1$

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

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

Happily, there are simplifications and cancellations.

Linear Discriminant Analysis when $p = 1$

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

Happily, there are simplifications and cancellations.

Linear Discriminant Analysis when $p = 1$

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

Happily, there are simplifications and cancellations.

Discriminant functions

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

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

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

Discriminant functions

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

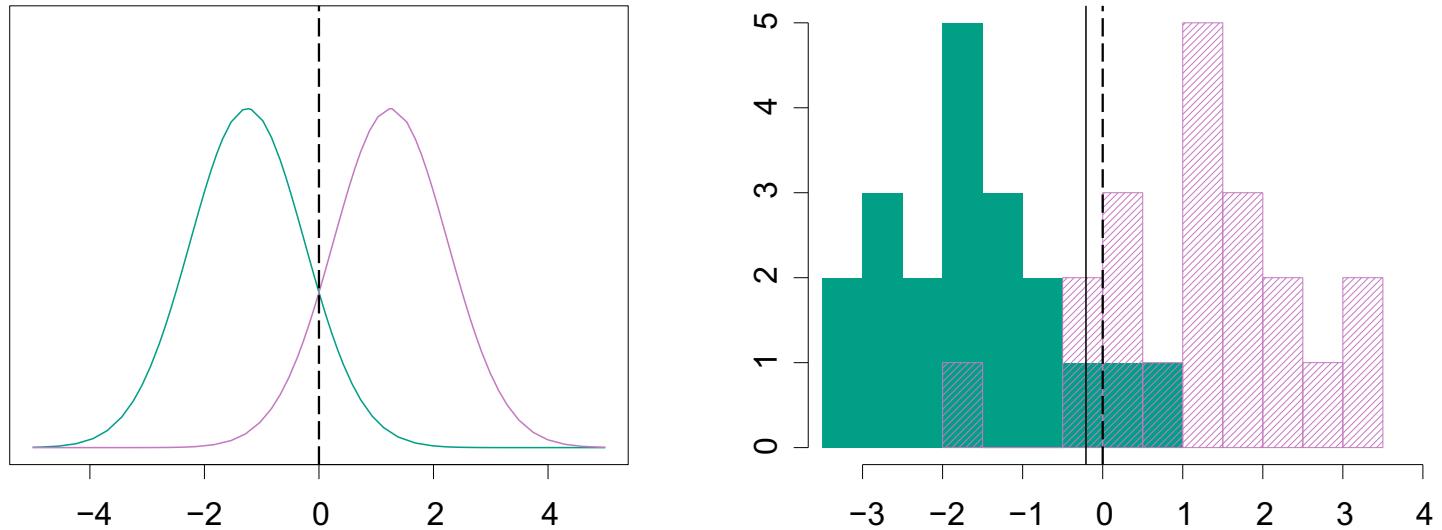
If there are $K = 2$ classes and $\pi_1 = \pi_2 = 0.5$, then one can see that the *decision boundary* is at

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

(show this)

Discriminant functions (show this)

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



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

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

Estimating the parameters

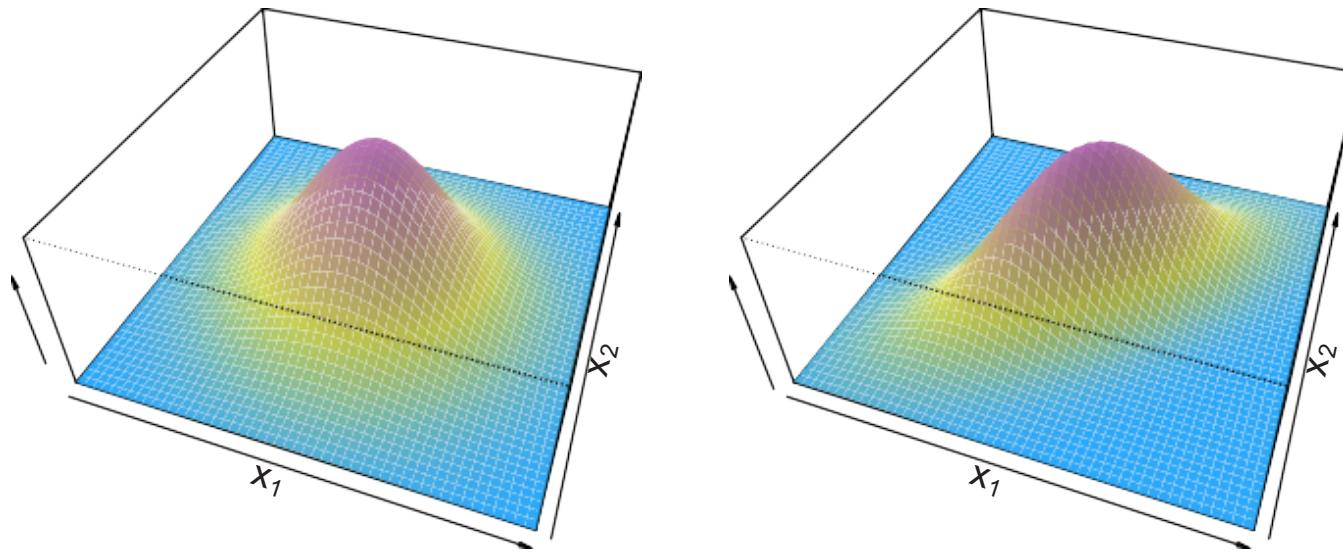
$$\hat{\pi}_k = \frac{n_k}{n}$$

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

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

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

Linear Discriminant Analysis when $p > 1$



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

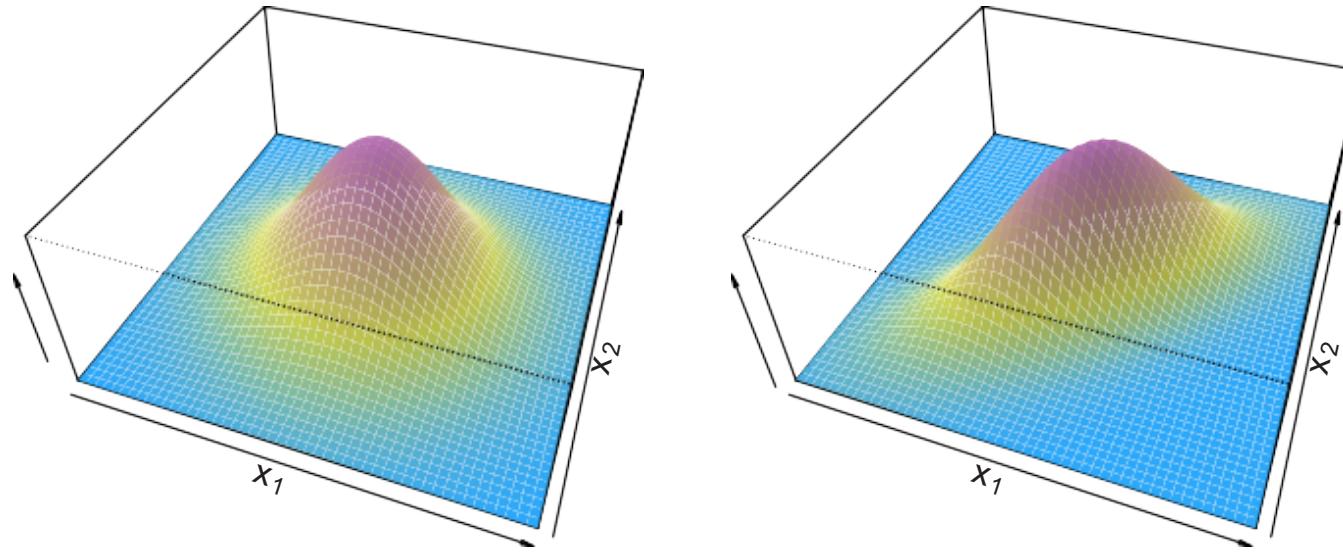
Discriminant function: $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$

Linear Discriminant Analysis when $p > 1$

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

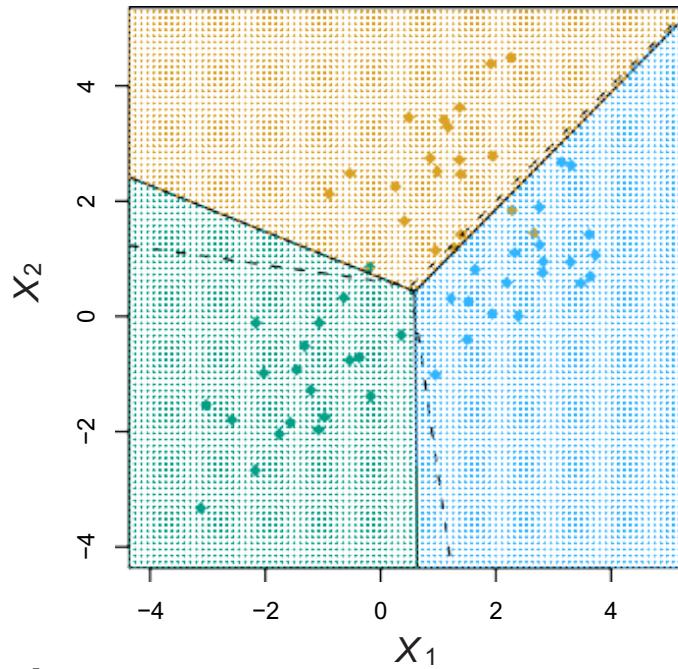
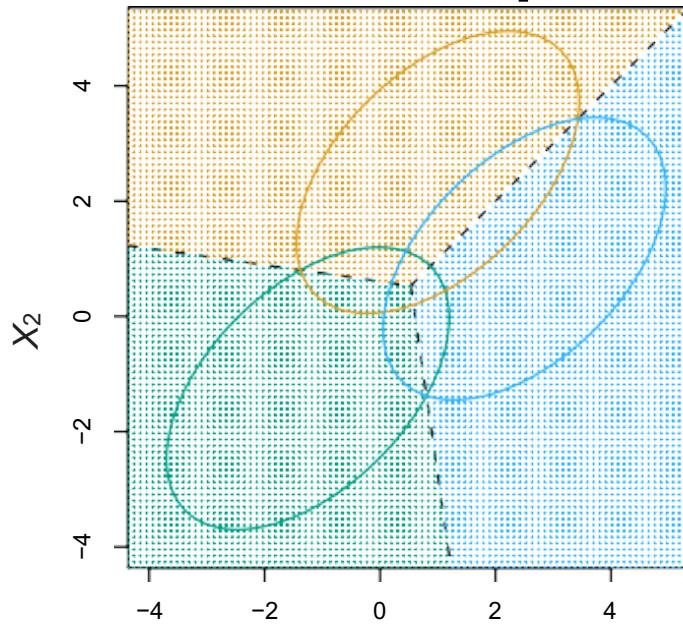
Discriminant function: $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$

Linear Discriminant Analysis when $p > 1$



Despite its complex form,
 $\delta_k(x) = c_{k0} + c_{k1}x_1 + c_{k2}x_2 + \dots + c_{kp}x_p$
is a **linear function**.

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



- Here $\pi_1 = \pi_2 = \pi_3 = 1/3$.
- The dashed lines are known as the *Bayes decision boundaries*. Were they known, they would yield the fewest misclassification errors, among all possible classifiers.

Fisher's Iris Data

4 variables

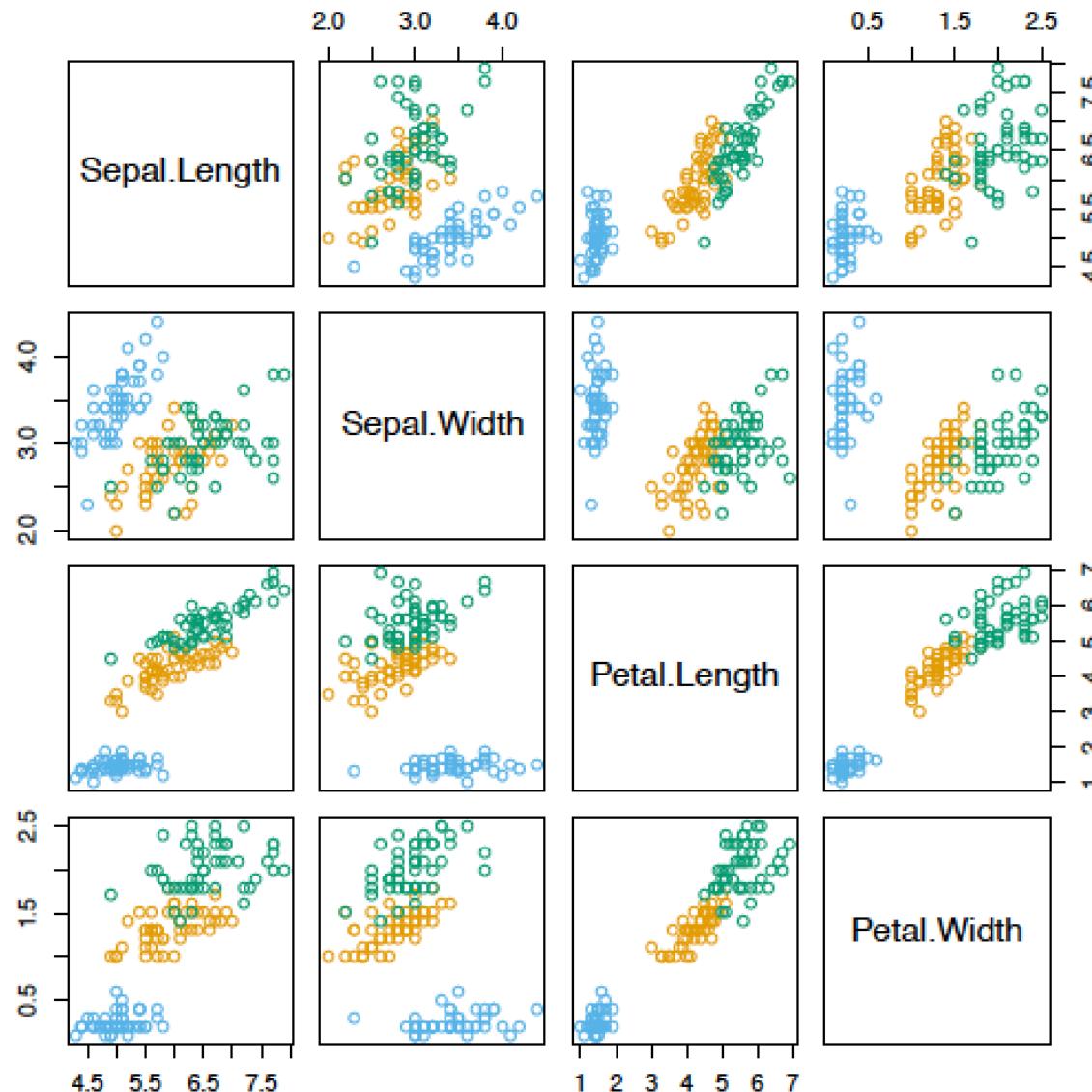
3 species

50

samples/class

- Setosa
- Versicolor
- Virginica

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

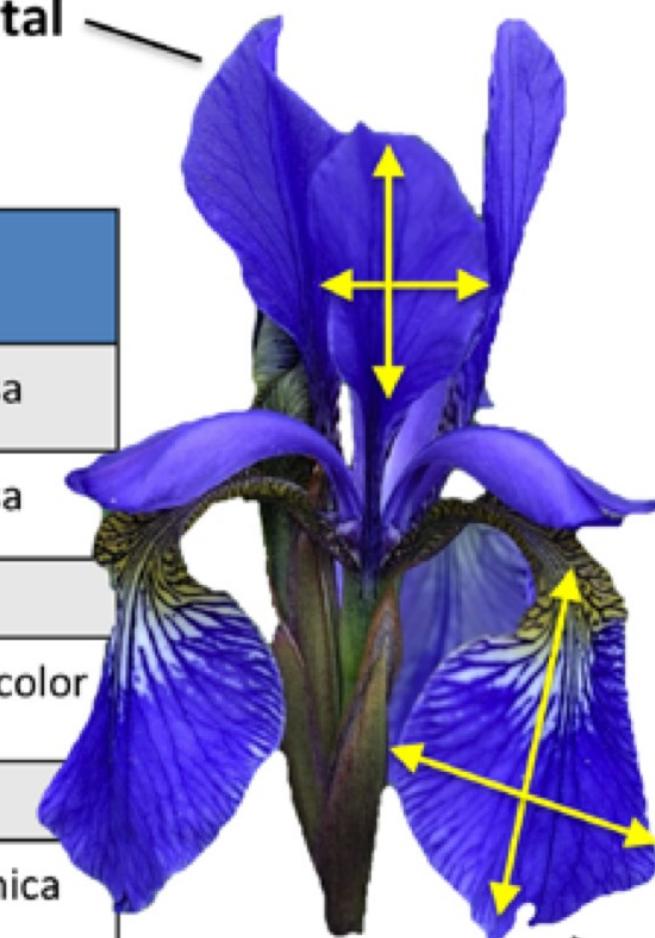


Samples

(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Petal



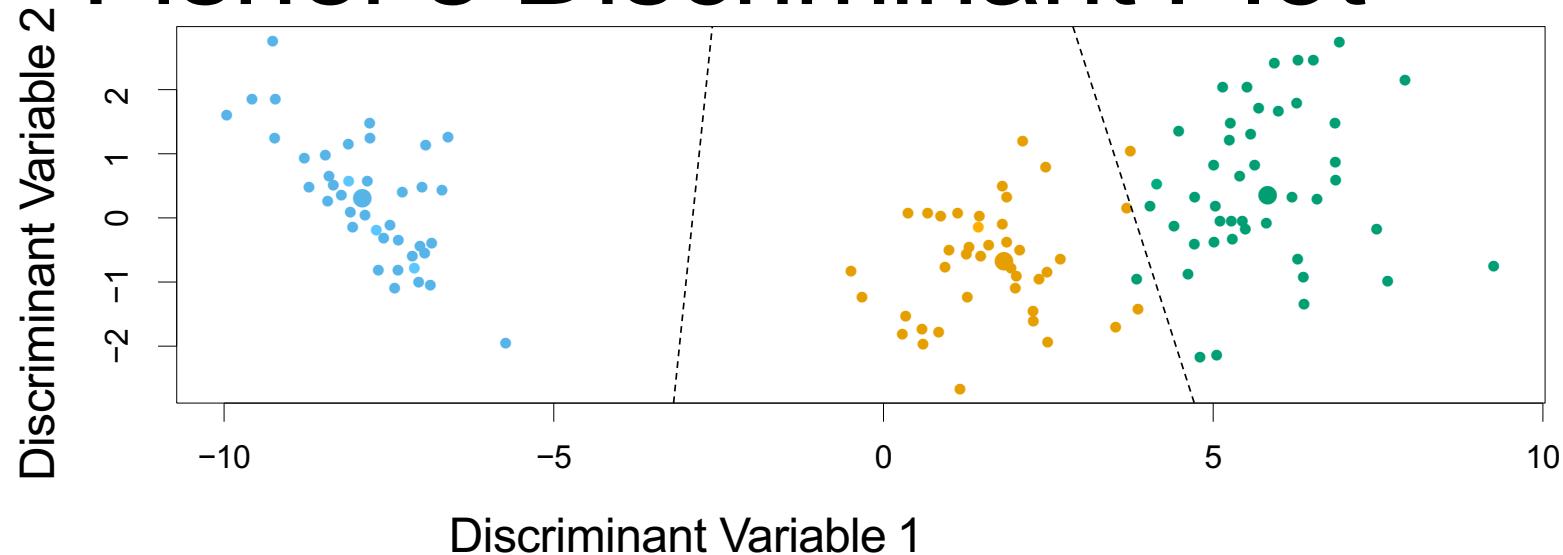
Sepal

Class labels
(targets)

Features

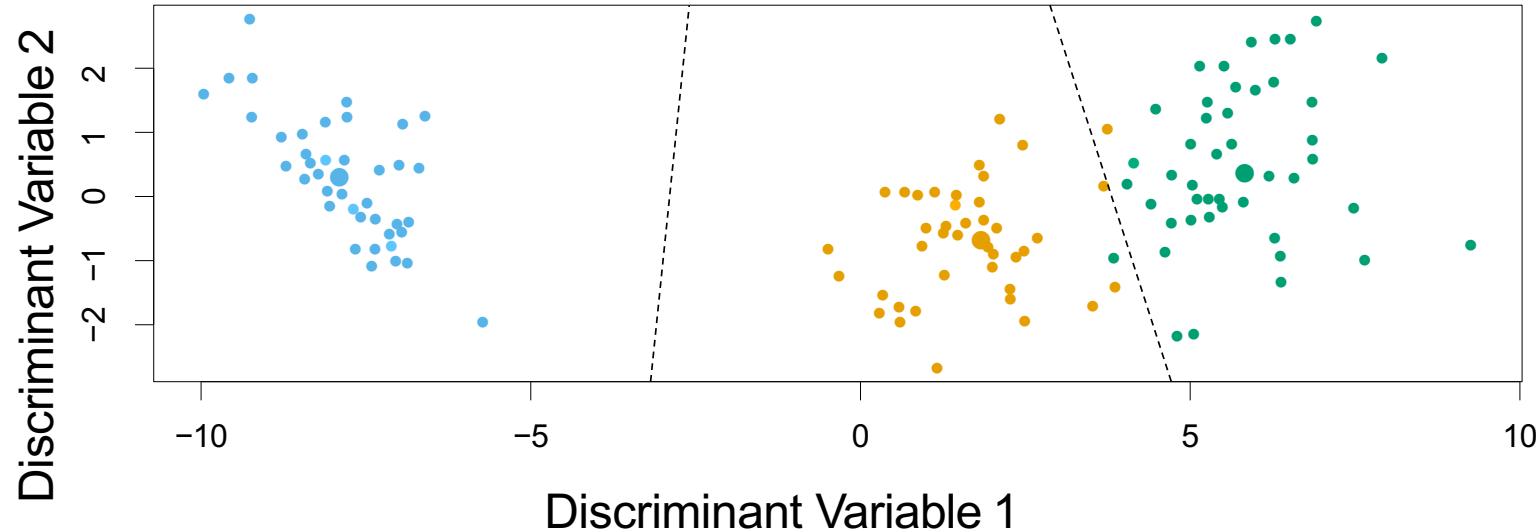
(attributes, measurements, dimensions)

Fisher's Discriminant Plot



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

Fisher's Discriminant Plot



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

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

From $\delta_k(x)$ to probabilities

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

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

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

From $\delta_k(x)$ to probabilities

- So classifying to the largest $\delta_k(x)$ amounts to classifying to the class for which $\Pr^*(Y = k|X = x)$ is largest.
- When $K = 2$, we classify to class 2 if $\Pr^*(Y = 2|X = x) \geq 0.5$, else to class 1.

LDA on Credit Data: Confusion Matrix

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

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

LDA on Credit Data

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since $n = 10000$ and $p = 4$!

LDA on Credit Data

Some caveats:

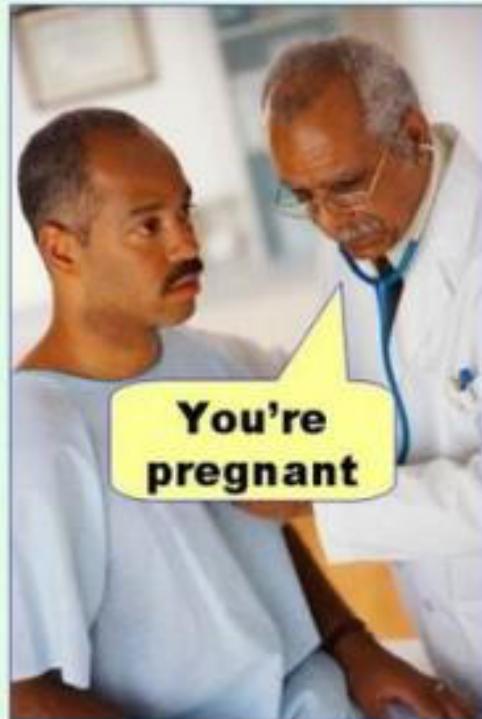
- If we classified to the prior — always to class **No** in this case — we would make 333/10000 errors, or only 3.33%.
- Of the true **No**'s, we make $23/9667 = 0.2\%$ errors; of the true **Yes**'s, we make $252/333 = 75.7\%$ errors!
- This is because of **class imbalance!**

Types of errors

- False positive (type I error)
rate: The fraction of negative examples that are classified as positive — 0.2% in example.
- False negative (type II error)
rate: The fraction of positive examples that are classified as negative — 75.7% in example.

Types of errors

Type I error
(false positive)



Type II error
(false negative)



<https://chemicalstatistician.files.wordpress.com/2014/05/pregnant.jpg?w=500>

Measures for Different Types of Error

The **sensitivity** or **recall** of a binary classifier is the rate that the event of interest is predicted correctly for all samples having the event, or

$$TP/P = TP / (TP + FN)$$

It is also called the True Positive Rate (TPR) or Hit Rate (HR).

- What proportion of credit card defaults did we detect?

Measures for Different Types of Error

The **specificity** of a binary classifier is the rate that non-events are predicted correctly for all non-event samples or

$$TN/N = TN / (TN + FP)$$

It is also called the True Negative Rate (TNR).

- What proportion of credit card non-defaults did we detect?

Types of errors

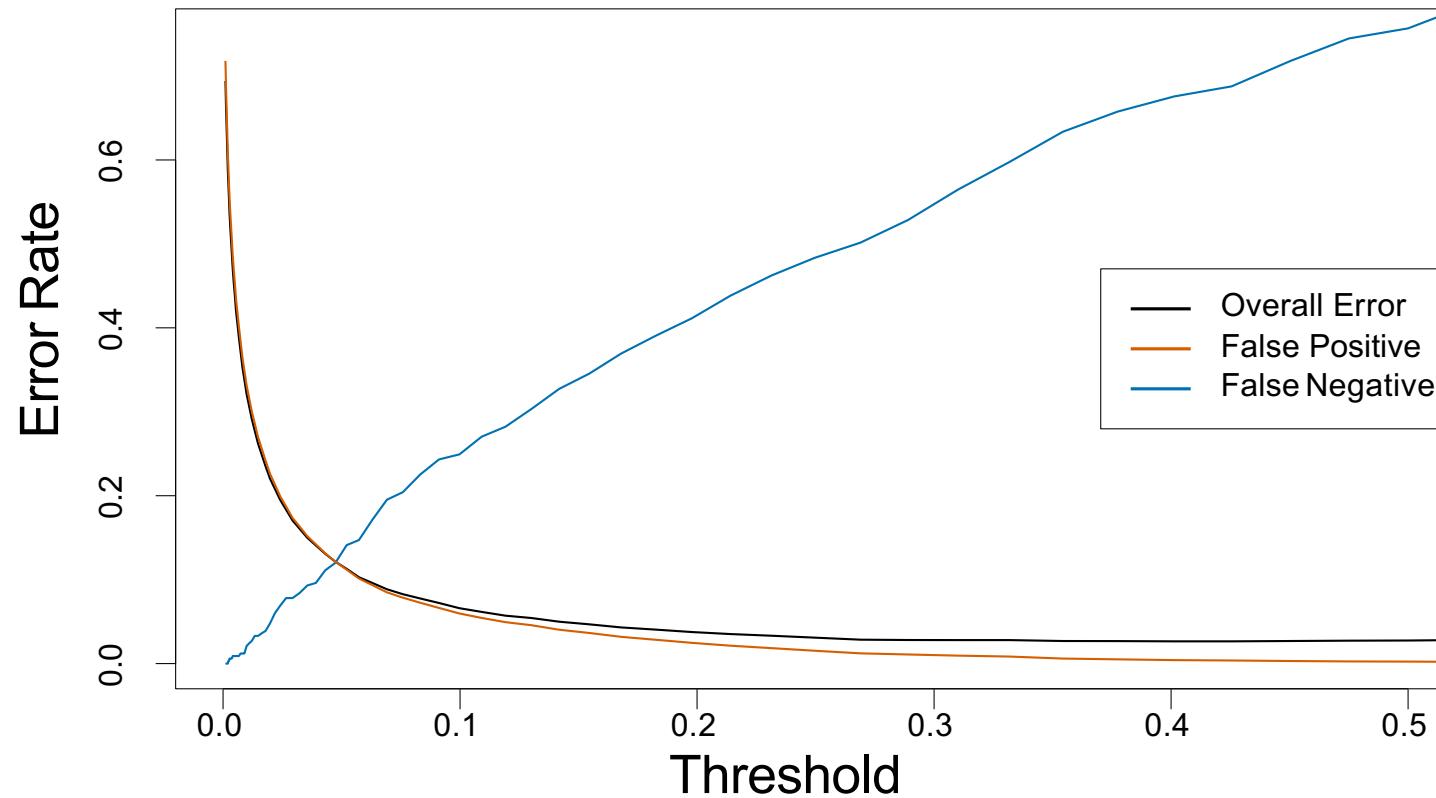
We produced the confusion matrix for credit data by classifying to class **Yes** if

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

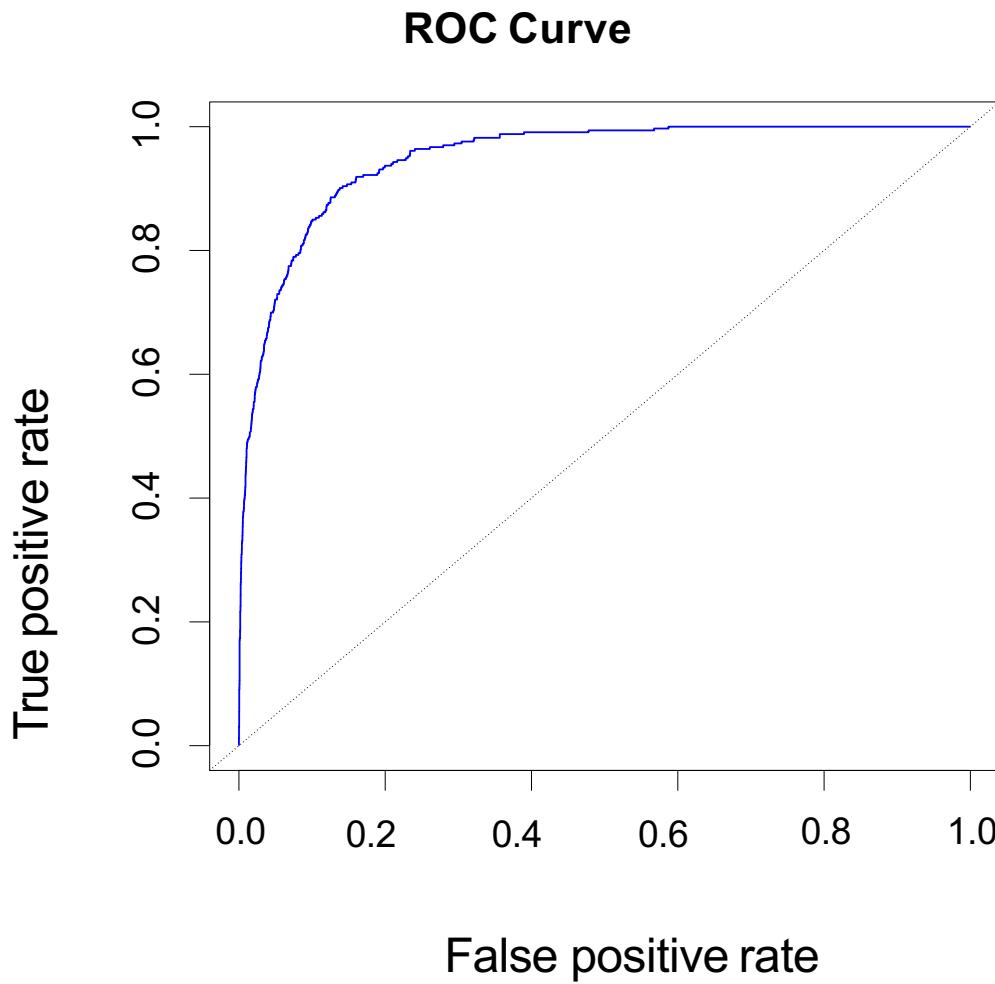
We can change the two error rates by changing the threshold from 0.5 to some other value in [0, 1]:

$\Pr^{\wedge}(\text{Default} = \text{Yes} | \text{Balance}, \text{Student}) \geq \text{threshold}$, and vary *threshold*.

Varying the *threshold*



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



The *ROC plot* displays both simultaneously. Sometimes we use the *AUC* or *area under the curve* to summarize the overall performance. Higher *AUC* is good.

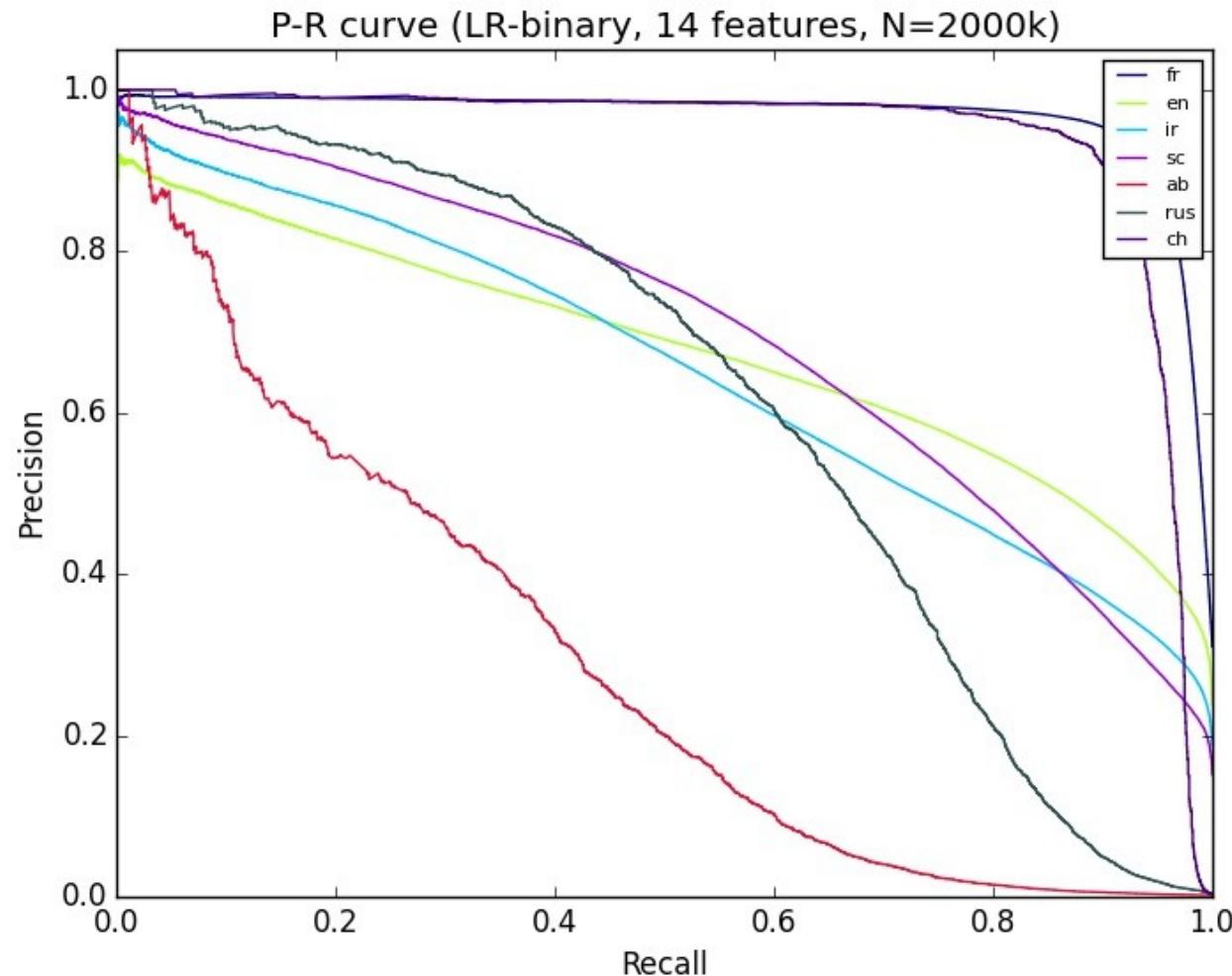
Measures for Different Types of Error

The **precision** or the **positive predictive value** of a binary classifier is the ratio of true positives with respect to all detected positives.

$$\text{Precision} = \text{PPV} = \text{TP}/(\text{TP} + \text{FP})$$

Of those defaults that we detected, what proportion actually defaulted?

Precision-Recall Curve



From: <https://stackoverflow.com/questions/33294574/good--roc--curve--but--poor--precision--recall--curve>

Measures for Different Types of Error

The **negative predictive value** of a binary classifier is the ratio of true negatives with respect to all detected negatives.

$$NPV = TN / (TN + FN)$$

Of those non-defaults that we detected, what proportion actually did not default?

Measures for Different Types of Error

The **F1 score or F measure** of a binary classifier is the harmonic mean of precision and recall:

$$F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Measures for Different Types of Error

F1 Score: seeks a balance between Precision (**ratio of true positives to all detected positives**) and Recall (**True Positive Rate**).

F1 Score might be better than accuracy if we seek a balance between Precision and Recall AND there is a class imbalance (large number of Actual Negatives).

Measures for Different Types of Error

F_β Score: seeks a *skewed* balance between Precision (ratio of true positives to all detected positives) and Recall (True Positive Rate).

$$F_\beta = \frac{\beta^2 + 1}{\frac{\beta^2}{recall} + \frac{1}{precision}}$$

Multiple Classes?

Precision/recall/ F_β are specific to one class

- How to summarize for multiple classes?

Multiple Classes?

Two different ways of averaging:

- A **macro** average just averages the individually calculated scores of each class
 - Weights each *class* equally

$$PRE_{macro} = \frac{PRE_1 + \dots + PRE_k}{k}$$

Multiple Classes?

Two different ways of averaging:

- A **micro** average calculates the metric by first pooling all instances of each class
 - Weights each *instance* equally

$$PRE_{micro} = \frac{TP_1 + \dots + TP_k}{TP_1 + \dots + TP_k + FP_1 + \dots + FP_k}$$

Measures: Training or Testing?

Note that all of the measures used to evaluate types of error can be computed over both training and test sets.

Other forms of Discriminant Analysis

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

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

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

Other forms of Discriminant Analysis

- With Gaussians but different Σ_k in each class, we get *quadratic discriminant analysis*. Let's show it for $p=1$.

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

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

Other forms of Discriminant Analysis

- With Gaussians but different Σ_k in each class, we get *quadratic discriminant analysis*. Let's show it for $p=1$.

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

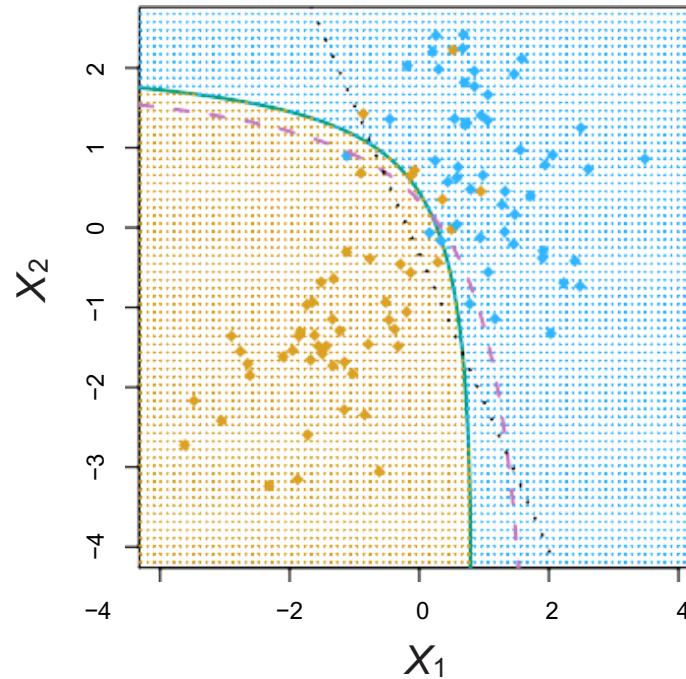
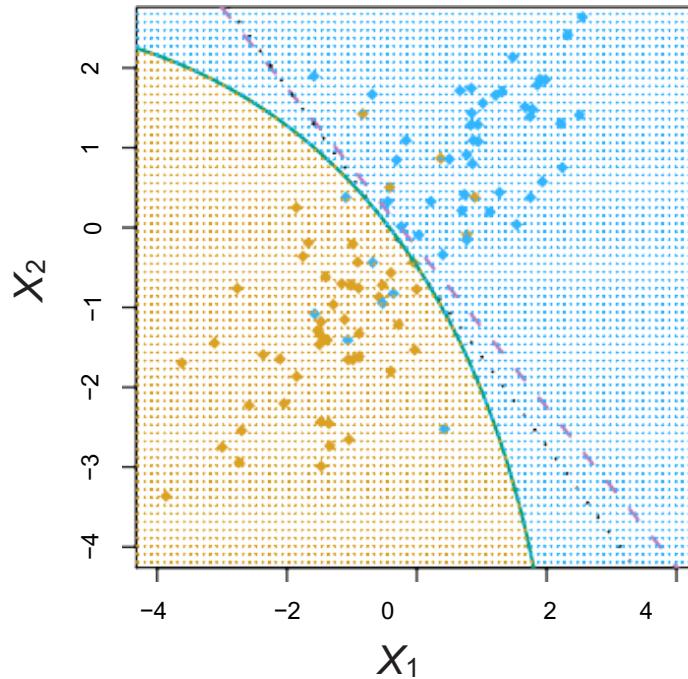
$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma_k}\right)^2}}$$

Other forms of Discriminant Analysis

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

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

Quadratic Discriminant Analysis



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

Because the Σ_k are different, the quadratic terms matter.

Logistic Regression versus LDA

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

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

So it has the same form as logistic regression.

The difference is in how the parameters are estimated.

Logistic Regression versus LDA

The difference is in how the parameters are estimated.

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

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

Summary

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

Naïve Bayes

Assumes features are independent in each class.

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

- Gaussian naïve Bayes assumes each Σ_k is diagonal:

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

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

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

Bayesian: Classify to the highest density

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

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

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

Bayesian: Classify to the highest density

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

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

- We classify a new point according to which density is highest.
- When the priors are different, we take them into account as well, and compare

$$\pi_k f_k(x) = \Pr(Y = k) \Pr(X = x | Y = k).$$

Bayesian classifiers

Easy to estimate priors π_k from data. (*How?*)
The real challenge: how to estimate

$$\begin{aligned}f_k(x) &= \Pr(X = x | Y = k) \\&= \Pr(X = (x_1, x_2, \dots, x_p) | Y = k)\end{aligned}$$

Bayesian classifiers

How to estimate

$$f_k(x_1, x_2, \dots, x_p) = \Pr(X = (x_1, x_2, \dots, x_p) | Y = k)$$

- In the general case, where the attributes x_j have dependencies, this requires estimating the full joint distribution $f_k(x_1, x_2, \dots, x_p)$ for each class k in C .
- There is almost never enough data to confidently make such estimates.

Naïve Bayes classifier

Assume independence among attributes x_j when class is given:

$$f_k(x_1, x_2, \dots, x_p) = f_k(x_1) f_k(x_2) \dots f_k(x_n)$$

Usually straightforward and practical to estimate $f_k(x_i) = \Pr(X_i = x_i | Y = k)$ for all x_j and k .

New sample is classified to $Y=k$ if
 $\pi_k \prod_i f_k(x_i)$ is maximal.

How to estimate $f_k(x_i) = \Pr(X_i = x_i | Y = k)$ from data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class priors:

$$\pi^k = N_k / N$$

$$\pi(\text{No}) = 7/10$$

$$\pi(\text{Yes}) = 3/10$$

For discrete attributes:

$$\Pr^k(X_i = x_i | Y = k) = |x_{ik}| / N_k$$

where $|x_{ik}|$ is number of instances in class k having attribute value x_i
Examples:

$$\Pr^k(\text{Status} = \text{Married} | \text{No}) = 4/7$$

$$\Pr^k(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

How to estimate $f_k(x_i)$ from data?

For continuous attributes:

Discretize the range into bins

replace with an ordinal attribute

Two-way split: $(x_i < v)$ or $(x_i > v)$

replace with a binary attribute

Probability density estimation:

- assume attribute follows some standard parametric probability distribution (usually a Gaussian)
- use data to estimate parameters of distribution (e.g. mean and variance)
- once distribution is known, can use it to estimate the conditional probability $\Pr(X_i = x_i | Y = k)$

How to estimate $f_k(x_i)$ from data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Gaussian distribution:

$$f_k(x_i) = \Pr(X_i = x_i | Y = k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} e^{-\frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2}}$$

one for each (x_i, k) pair

For (Income | Class = No):

sample mean = 110

sample variance = 2975

$$\Pr(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Naïve Bayes classifier

Problem: if one of the conditional probabilities is zero, then the entire expression becomes zero.

This is a significant practical problem, especially when training samples are limited.

Ways to improve probability estimation:

$$\text{Original: } p(x_j | C_i) = \frac{N_{ji}}{N_i}$$

$$\text{Laplace: } p(x_j | C_i) = \frac{N_{ji} + 1}{N_i + c}$$

$$\text{m - estimate: } p(x_j | C_i) = \frac{N_{ji} + mp}{N_i + m}$$

c: number of levels
variable x_j can take.

p: prior probability

m: parameter

Summary of Naïve Bayes

Robust to isolated noise samples.

Handles missing values by ignoring the sample during probability estimate calculations.

Robust to irrelevant attributes.

NOT robust to redundant attributes.

Independence assumption does not hold in this case.

Use other techniques such as Bayesian Belief Networks (BBN).

Appendix:

Text Classification

Is this spam?

Subject: Important notice!

From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

<http://www.123contactform.com/contact-form-StanfordNew1-236335.html>

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

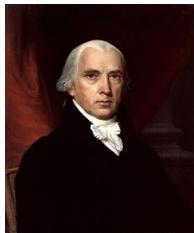
© Stanford University. All Rights Reserved.

Who wrote which Federalist papers?

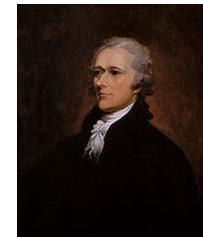
1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.

Authorship of 12 of the letters in dispute

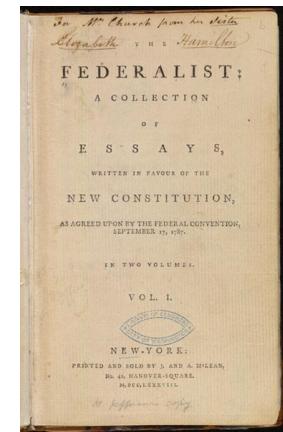
1963: solved by Mosteller and Wallace using Bayesian methods



James Madison



Alexander Hamilton



Male or female author?

1. By 1925 present-day Vietnam was divided into three parts under French colonial rule. The southern region embracing Saigon and the Mekong delta was the colony of Cochinchina; the central area with its imperial capital at Hue was the protectorate of Annam...
2. Clara never failed to be astonished by the extraordinary felicity of her own name. She found it hard to trust herself to the mercy of fate, which had managed over the years to convert her greatest shame into one of her greatest assets...

S. Argamon, M. Koppel, J. Fine, A. R. Shimoni, 2003. “Gender, Genre, and Writing Style in Formal Written Texts,” *Text*, volume 23, number 3, pp. 321–346

Positive or negative movie review?



unbelievably disappointing

Full of zany characters and richly applied satire, and some great plot twists
this is the greatest screwball comedy ever filmed

It was pathetic. The worst part about it was the boxing scenes.



What is the subject of this article?

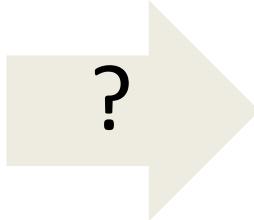
MEDLINE Article



141

MeSH Subject Category Hierarchy

Antagonists and Inhibitors
Blood Supply
Chemistry
Drug Therapy
Embryology
Epidemiology
...



Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- ...

Classification Methods: Hand-coded rules

Rules based on combinations of words or other features

spam: black-list-address OR (“dollars” AND “have been selected”)

Accuracy can be high

If rules carefully refined by expert

But building and maintaining these rules is expensive

Classification Methods: Supervised Machine Learning

Any kind of classifier
 Naïve Bayes
 Logistic regression
 Support-vector machines
 k-Nearest Neighbors

...

$\gamma($

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

) = C



The bag of words representation

The bag of words representation

Y(

I **love** this movie! It's **sweet**, but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun**... It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.

)=C



The bag of words representation: using a subset of words

$y($

```
x love xxxxxxxxxxxxxxxx sweet
xxxxxxxx satirical xxxxxxxxxxxx
xxxxxxxxxxx great xxxxxxxx
xxxxxxxxxxxxxxxx fun xxxx
xxxxxxxxxxxxxx whimsical xxxx
romantic xxxx laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxx recommend xxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxx
xxxxx happy xxxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx
```

)=c



The bag of words representation

Y() = C

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

thumb up icon

thumb down icon

Bag of Words

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR).

In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

Example

- (1) John likes to watch movies. Mary likes movies too.
- (2) John also likes to watch football games.

Parsed Model:

"John", "likes", "to", "watch", "movies", "Mary", "likes", "movies", "too"

"John", "also", "likes", "to", "watch", "football", "games"

Example

- (1) John likes to watch movies. Mary likes movies too.
- (2) John also likes to watch football games.

Bag of Words Model:

BoW1 =

```
{ "John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
```

BoW2 =

```
{ "John":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};
```

Example

(1) John likes to watch movies. Mary likes movies too.

John also likes to watch football games.

Combined Documents:

BoW3 =

```
{ "John":2,"likes":3,"to":2,"watch":2,"movies":2,"Mary":1,"too":1,"also":1,"football":1,"games":1};
```

Application

The Bag-of-words model is mainly used as a tool of **feature generation**.

After transforming the text into a "**bag of words**", we calculate various measures to characterize the text.

The most common type of characteristics, or features calculated from the Bag-of-words model is term frequency, namely, the number of times a term appears in the text.

Application

For the example above, we can construct the following two lists to record the term frequencies of all the distinct words (BoW1 and BoW2 ordered as in BoW3):

- (1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
- (2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]

Term Frequency

The simplest choice to calculate Term Frequency to measure the importance of a word in a document is to use the *raw count* of a term in a document.

$tf(t,d)$ =number of occurrences of term t in document d

Term Frequency

Other choices:

Boolean Frequencies $tf(t,d)=1$ if t occurs in d
otherwise 0

Adjustment for document length:

$tf(t,d)=[\text{number of occurrences of term } t \text{ in document } d]/\text{number of words in document } d$

Inverse Document Frequency

A measure of information each word provides, i.e. if a word is common or rare among documents
 $IDF(t,D) = \log(\text{total number of documents in the corpus } D / \text{number of documents where the term } t \text{ appears})$

TFIDF

A combined measure for each word is the TF-IDF which is a product of TF and IDF:

$$\text{TFIDF}(t,d,D) = \text{TF}(t,d) \cdot \text{IDF}(t,D)$$

Therefore, each document can be represented as a vector representing the bag of words, but instead of simple frequencies, the TFIDF of each word can be given in the vector.

Naïve Bayes and Language Modeling

Naïve bayes classifiers can use any sort of feature

URL, email address, dictionaries, network features

But if, as in the previous slides

We use **only** word features

we use **all** of the words in the text
(not a subset)

Then

Naïve bayes has an important similarity to language modeling.

Each class = a unigram language model

Sec 13.2.1

Assigning each word: $P(\text{word} | c)$

Assigning each sentence: $P(s|c) = \prod P(\text{word}|c)$

Class *pos*

0.1	I		<u>I</u>	<u>love</u>	<u>this</u>	<u>fun</u>	<u>film</u>
0.1	love		0.1	0.1	.05	0.01	0.1
0.01	this						
0.05	fun						
0.1	film						$P(s pos) = 0.0000005$
...							

Naïve Bayes as a Language Model

Which class assigns the higher probability to s?

Model pos		Model neg						
0.1	I	0.2	I	I				
0.1	love	0.001	love		love			
0.01	this	0.01	this	0.1		this		
0.05	fun	0.005	fun	0.2			fun	
0.1	film	0.1	film	0.001				film

$P(s|pos) > P(s|neg)$

Appendix

How to Deal with Missing Values?

From:

<https://machinelearningmastery.com/handle-missing-data-python/>

A Simple Strategy

- The simplest strategy for handling missing data is to remove samples that contain a missing value (feature).
- **Advantage:**
 - Simplicity
- **Disadvantage:**
 - Missing a lot of information
 - Works poorly if the percentage of missing values is high (say 30%), compared to the whole dataset

Data Imputation

- Imputing refers to using a model to replace missing values.
 - A constant value that has meaning within the domain, such as 0, distinct from all other values. This works for categorical features.

Data Imputation

- Imputing refers to using a model to replace missing values.
 - A feature value from another randomly selected observation.

Data Imputation

- Imputing refers to using a model to replace missing values.
 - A statistics such as mean, median or mode value of the feature/column.

Data Imputation

- Imputing refers to using a model to replace missing values.
 - A value estimated by another predictive model:
 - Treat the missing value as the output of your predictive model
 - Predict it based on other data points that do not have missing values
 - Examples: KNN regression and classification and Linear Regression

Data Imputation

- Imputing refers to using a model to replace missing values.
 - A value estimated by another predictive model:
 - What if a lot of data has missing values?
 - There will not be enough data without missing values for prediction of other missing values
 - An iterative method based on Expectation Maximization can be used

Data Imputation

- Imputing refers to using a model to replace missing values.
 - A value estimated by another predictive model:
 - An iterative method based on Expectation Maximization can be used:
 - First, impute all missing values randomly or using mean or median
 - Predict missing values using other data points
 - Update all missing values and iterate

Algorithms That Support Missing Values

- k-Nearest Neighbors can ignore a feature from a distance measure when a value is missing.
 - Calculate distance measure without the missing feature

Algorithms That Support Missing Values

- There are also algorithms that can use the missing value as a unique and different value (say NaN) when building the predictive model, such as classification and regression trees.

Algorithms That Support Missing Values

- The scikit-learn implementations of decision trees and k-NN are not robust to missing values. Although it is being considered.
- This remains as an option if you consider using another algorithm implementation (such as xgboost) or developing your own implementation.

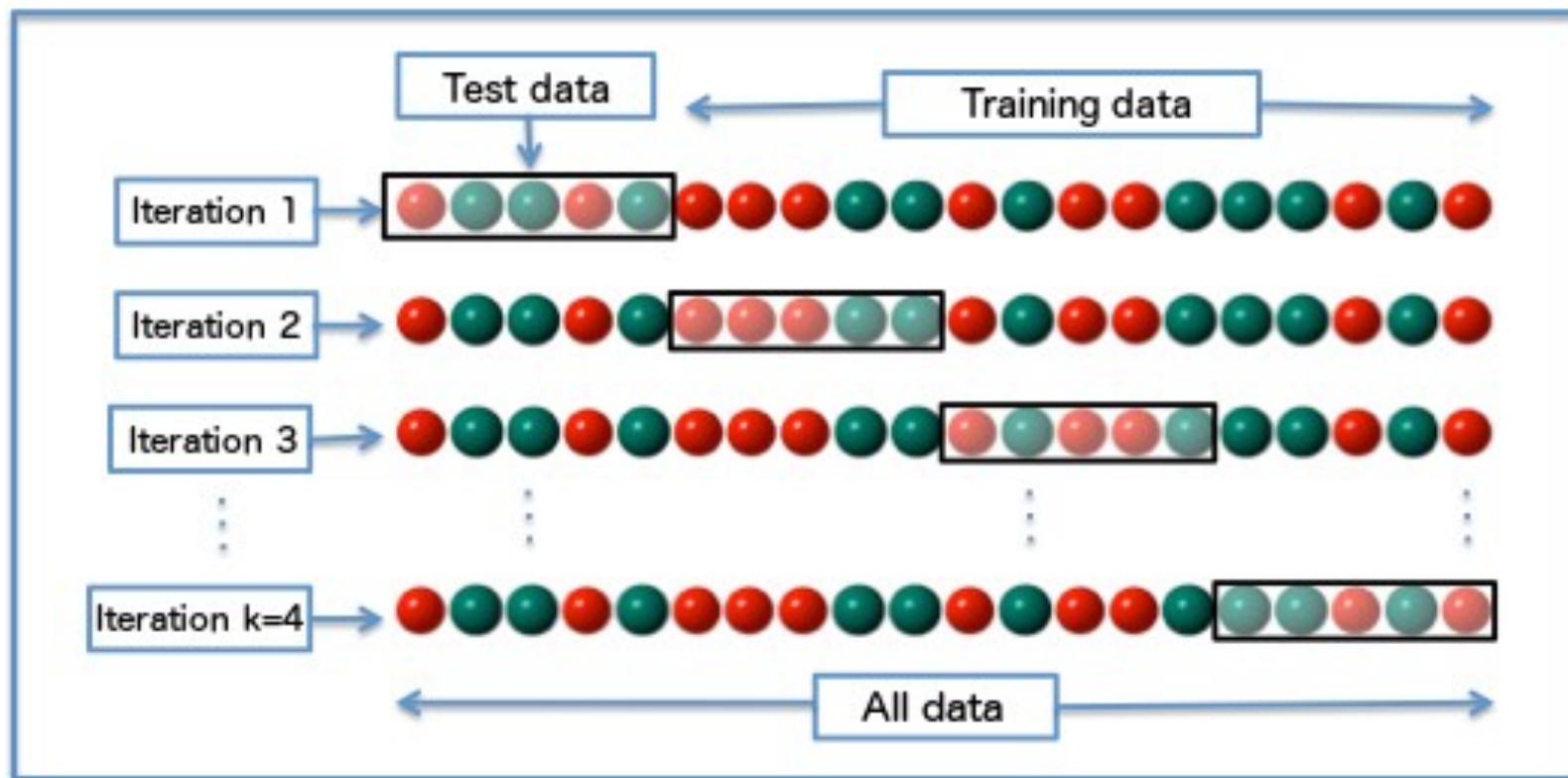
DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 4

Resampling



Resampling Methods

- Repeatedly drawing samples from a training set and refitting a model on each sample to obtain more information about the fitted model.
- Example: Estimate the variability of a linear regression fit by repeatedly drawing different samples from the training data, fitting a regression model to each new sample, and then examining the extent to which the resulting fits differ.

Resampling Methods

- **Model Assessment:** having chosen a final model, estimating its prediction error on new data.
- **Model Selection:** estimating the performance of different models in order to choose the best one.

Resampling Methods (cont.)

Cross-Validation

Used to estimate test set prediction error rates associated with a given machine learning method to evaluate its performance, or to select the appropriate level of model flexibility.

Bootstrap

Used most commonly to provide a measure of accuracy of a parameter estimate or of a given machine learning method.

Model Assessment

The *generalization* performance of a machine learning method relates to its prediction capability on independent test sets.

Assessment of this performance is extremely important in practice, since it guides the choice of the machine learning method or model.

Further, this gives us a measure of the quality of the ultimately chosen model.

Model Assessment (cont.)

Test Error

The average prediction error of a machine learning method on new observations.

The prediction error over an independent test sample.

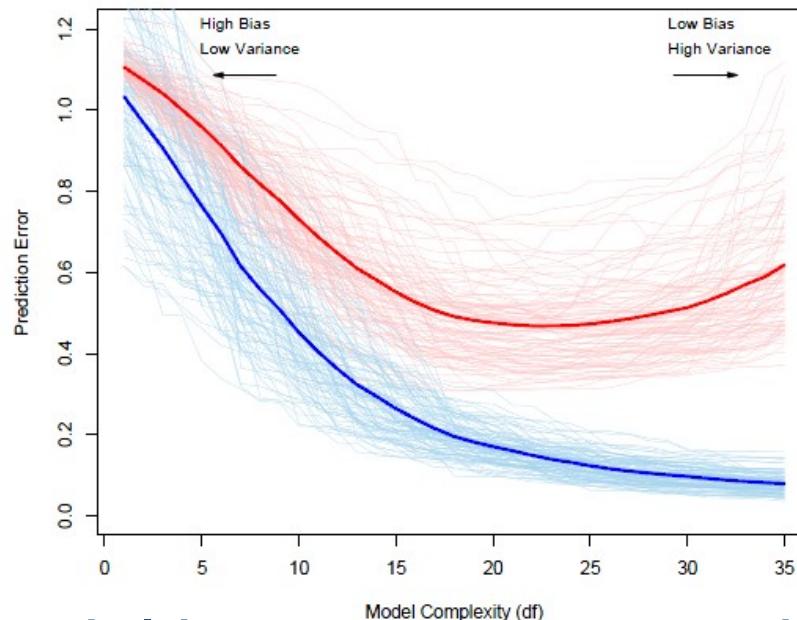
Training Error

The average loss over the training sample:

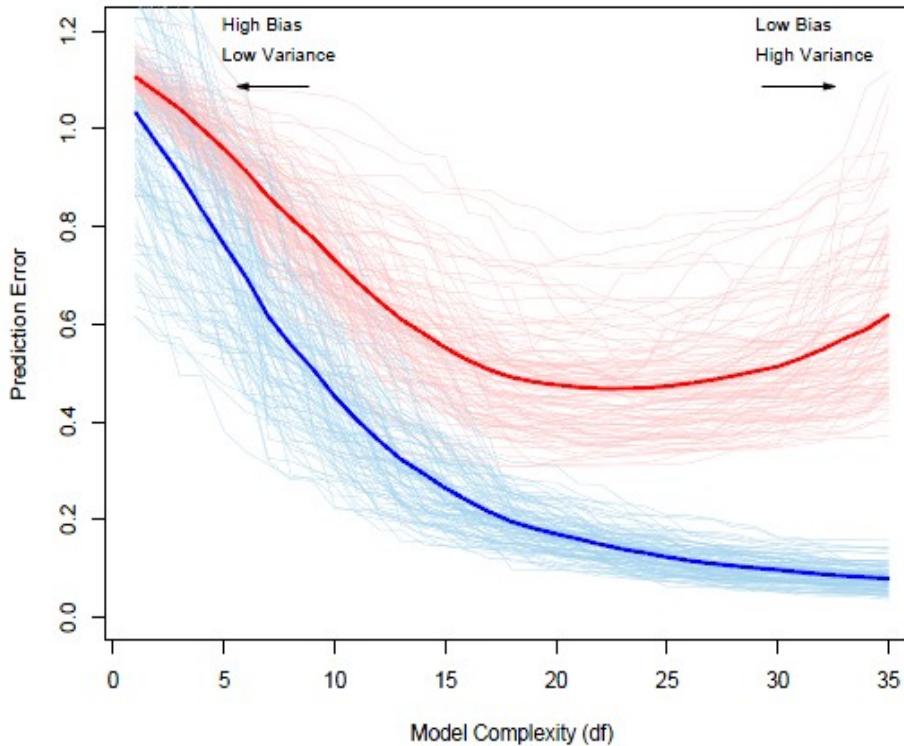
$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}(x_i))$$

Note: The training error rate can dramatically *underestimate* the test error rate

Model Assessment (cont.)



- As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures.
- Hence, there is a decrease in bias but an increase in variance.
- However, training error is not a good estimate of the test error.



- Training error consistently decreases with model complexity.
- A model with zero training error is overfit to the training data and will typically generalize poorly.

Model Assessment (cont.)

- If we are in a data-rich situation, the best approach for both *model selection* and *model assessment* is to randomly divide the dataset into three parts: training set, validation set, and test set.



Model Assessment (cont.)

- The *training set* is used to fit the models. The *validation set* is used to estimate prediction error for model selection. The *test set* is used for assessment of the prediction error of the final chosen model.



- A typical split might by 50% for training, and 25% each for validation and testing.

Model Assessment (cont.)

Best solution: use a large designated test set, which is often not available. For the methods presented here, there is insufficient data to split it into three parts.

There are some methods to make mathematical adjustments to the training error rate in order to estimate the test error rate:

Cp statistic, AIC, BIC (we will discuss these next)

Model Assessment (cont.)

Here, we consider cross-validation (CV) methods that estimate the test error by *holding out* a subset of the training observations from the fitting process, and then applying the machine learning method to those held out observations.

Overview: Model Selection

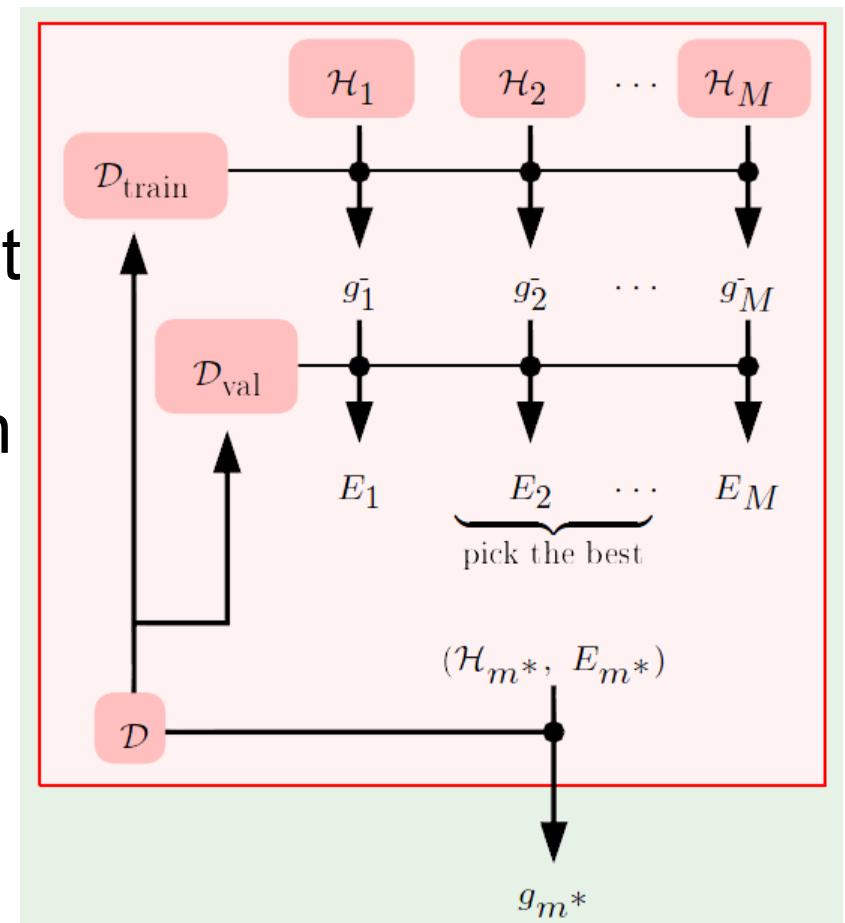
- The most important use of validation is for *model selection*
- In almost every learning situation, there are some choices to be made and we need a principled way of making these choices.
- The leap is to realize that *validation* can be used to estimate the out-of-sample error for more than one model.

Overview: Model Selection (cont.)

Suppose we have M models; validation can be used to select one of these models.

We use the training data to fit the model, and we evaluate each model on the validation set to obtain the validation errors.

It is now a simple matter to select the model with the lowest validation error.

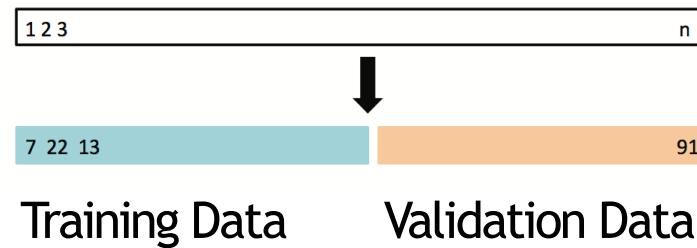


Validation Set Approach

We wish to find a set of variables that give the lowest *validation error rate* (an estimate of the *test error rate*).

If we have a large data set, we can randomly split the data into separate training and validation data sets.

Then, we use the training data set to build each possible model and select the model that gives the lowest error rate when applied to the validation dataset.



Validation Set Approach: Example

Example: we want to predict *mpg* from
horsepower

Two models:

$$\text{mpg} \sim \text{horsepower}$$

$$\text{mpg} \sim \text{horsepower} + \text{horsepower}^2$$

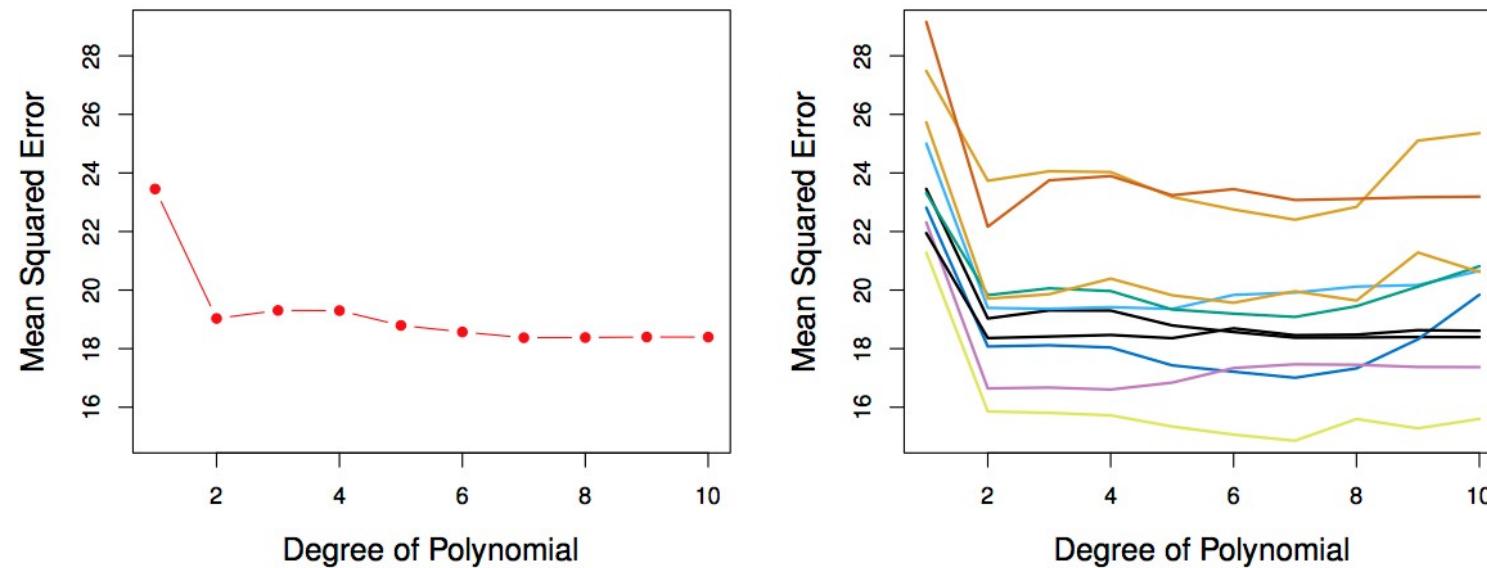
Which model gives a better fit?

We randomly split (50/50) *392 observations* into training and validation data sets, and we fit both models using the training data.

Next, we evaluate both models using the validation data set.

Winner = model with the *lowest* validation (testing)
MSE

Validation Set Approach: Example Results



Left Panel: Validation error estimates for a single split into training and validation data sets.

Right Panel: Validation error estimates for multiple splits; shows the test error rate is highly variable.

Validation Set Approach: Review

Advantages:

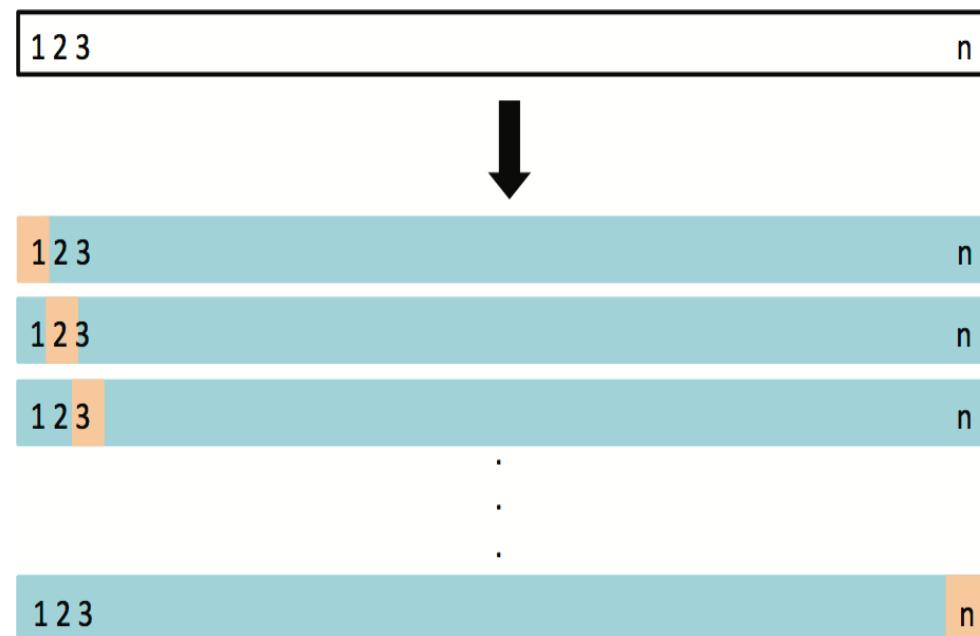
- Conceptually simple and easy implementation.

Drawbacks:

- The validation set error rate (MSE) can be **highly variable**.
- Only a subset of the observations (those in the training set) are used to fit the model.
- Machine learning methods tend to perform worse when trained on fewer observations.
- Thus, the validation set error rate may tend to **overestimate** the test error rate for the model fit on the entire data set.

Leave-One-Out Cross-Validation

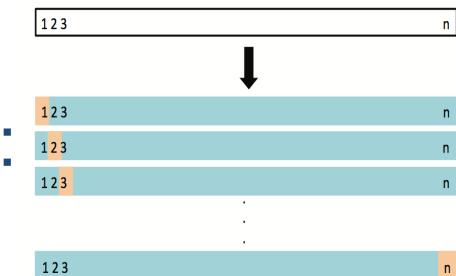
Instead of two subsets of comparable size, a single observation is used for the validation set and the remaining observations ($n - 1$) make up the training set.



Leave-One-Out Cross-Validation

- **LOOCV Algorithm:**

- Split the entire data set of size n into:
 - Blue = training data set
 - Beige = validation data set
- Fit the model using the training data set
- Evaluate the model using validation set and compute the corresponding MSE.
- Repeat this process n times, producing n squared errors. The average of these n squared errors estimates the test MSE.



$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n \text{MSE}_i$$

Validation Set Approach vs. LOOCV

LOOCV has far less bias and, therefore, tends not to overestimate the test error rate.

Performing LOOCV multiple times always yields the same results because there is no randomness in the training/validation set splits.

LOOCV is computationally intensive because the model has to be fit n times.

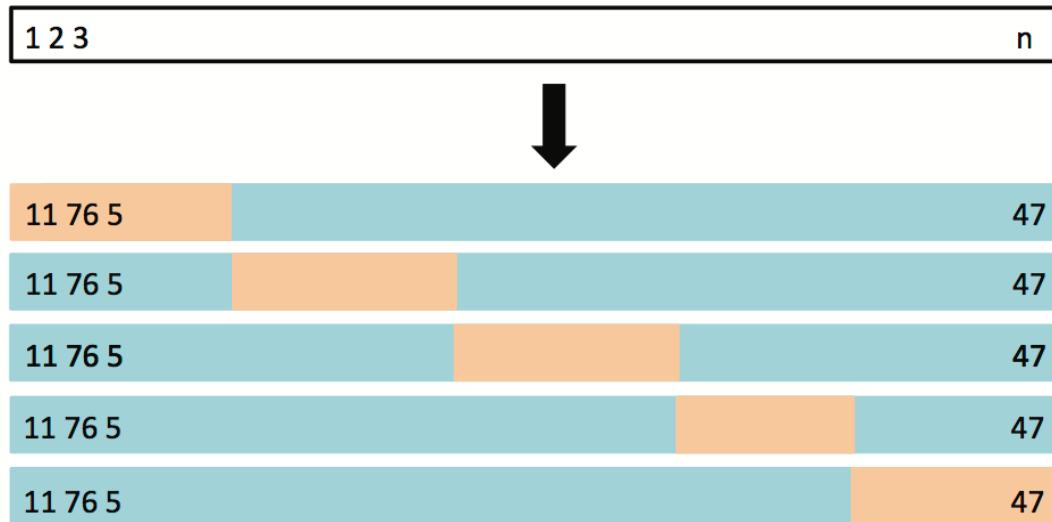
K-Fold Cross-Validation

- The **simplest and most widely used** method for estimating prediction error.
- Directly estimates the average prediction error when the model is applied to an independent test sample.

K-Fold Cross-Validation

- Had we enough data, we would use a validation set to assess the performance of the model.
- To finesse the problem, *K*-fold cross-validation uses part of the available data to fit the model, and a different part to test it.

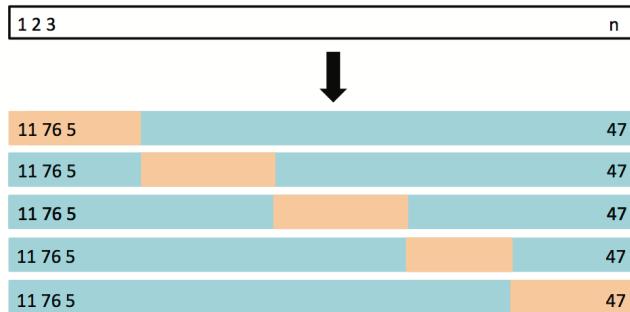
K -Fold Cross-Validation (cont.)



We use this method because LOOCV is computationally intensive.

We randomly divide the data set of into K folds (typically $K = 5$ or 10).

K-Fold Cross-Validation (cont.)



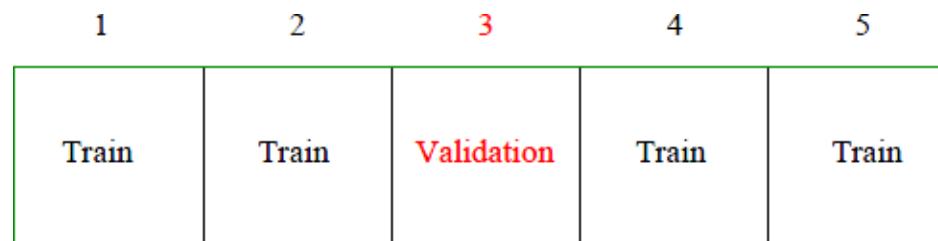
- The first fold is treated as a validation set, and the method is fit on the remaining $K - 1$ folds. The MSE is computed on the observations in the *held-out* fold. The process is repeated K times, taking out a different part each time.
- By averaging the K estimates of the test error, we get an estimated validation (test) error rate for new observations.

K -Fold Cross-Validation (cont.)

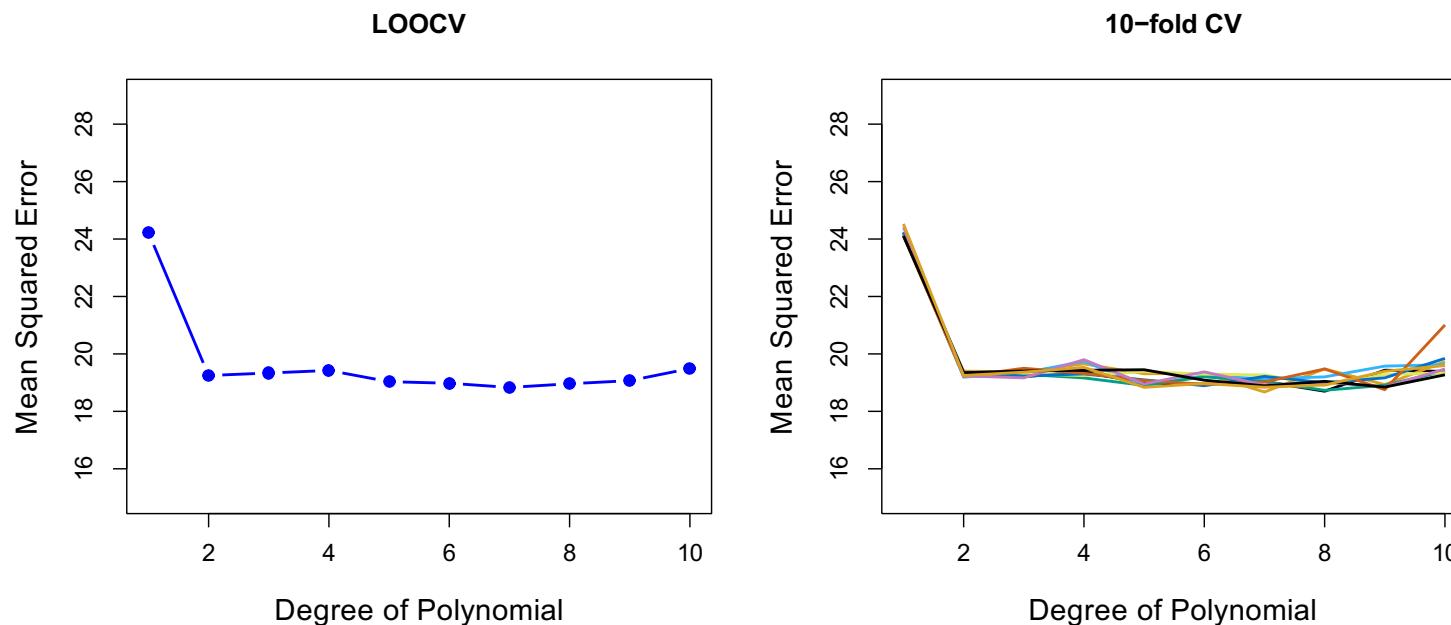
Let the K folds be F_1, \dots, F_K , where F_k denotes the indices of the observations in fold k . There are n_k observations in fold k : if N is a multiple of K , then $n_k = n / K$.

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

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



K-Fold Cross-Validation vs. LOOCV



Left Panel: LOOCV Error Curve

Right Panel: 10-fold CV run nine separate times, each with a different random split of the data into ten parts.

Note: LOOCV is a special case of K -fold, where $K = n$

Bias-Variance Trade-off for K -Fold Cross-Validation

Which is better, LOOCV or K -fold CV?

- LOOCV is more computationally intensive
- LOOCV has **less bias** than K -fold CV (when $K < n$) :
 - Each training set is $(K-1)/K$ as big as the original training set, so the estimates of prediction error will typically be biased upward. *Why?*

Bias-Variance Trade-off for K -Fold Cross-Validation

Which is better, LOOCV or K -fold CV?

- However, LOOCV has **higher variance** than K -fold CV (when $K < n$):
 - It doesn't *shake up* the data enough.
The estimates from each fold are highly correlated and hence their average can have high variance.
- Thus, we see the bias-variance trade-off between the two resampling methods

Bias-Variance Trade-off for K -Fold Cross-Validation

K -fold CV with $K = 5$ or $K = 10$ are commonly used, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance

Cross-Validation on Classification Problems

We will cover classification problems in more detail later in the course, but we briefly show how CV can be used when Y is qualitative (categorical) as opposed to quantitative. Here, rather than use MSE to quantify test error, we instead use the number of misclassified observation.

Cross-Validation on Classification Problems

LOOCV Error Rate:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n Err_i, \text{ where } Err_i = I(Y_i \neq \hat{Y}_i)$$

We use CV as follows:

- Divide data into K folds; hold-out one part and fit using the rest (compute error rate on hold-out data); repeat K times.
- CV Error Rate: average over the K errors we have computed

Cross-Validation on Classification Problems: Class Imbalance

Note1: prediction accuracy is not always a good measure of how well a classifier performs. Hence, for highly imbalanced data sets, one can estimate other measures such as precision, recall, or F scores, using cross validation

Cross-Validation on Classification Problems: Class Imbalance

Note 2: when data is highly imbalanced, some of the folds may contain few points or even no points from a rare class.

Therefore, sometimes *stratified cross validation* is used, which seeks to ensure that each fold has the same ratio of each class as the original data set.

Cross-validation: right and wrong

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

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

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

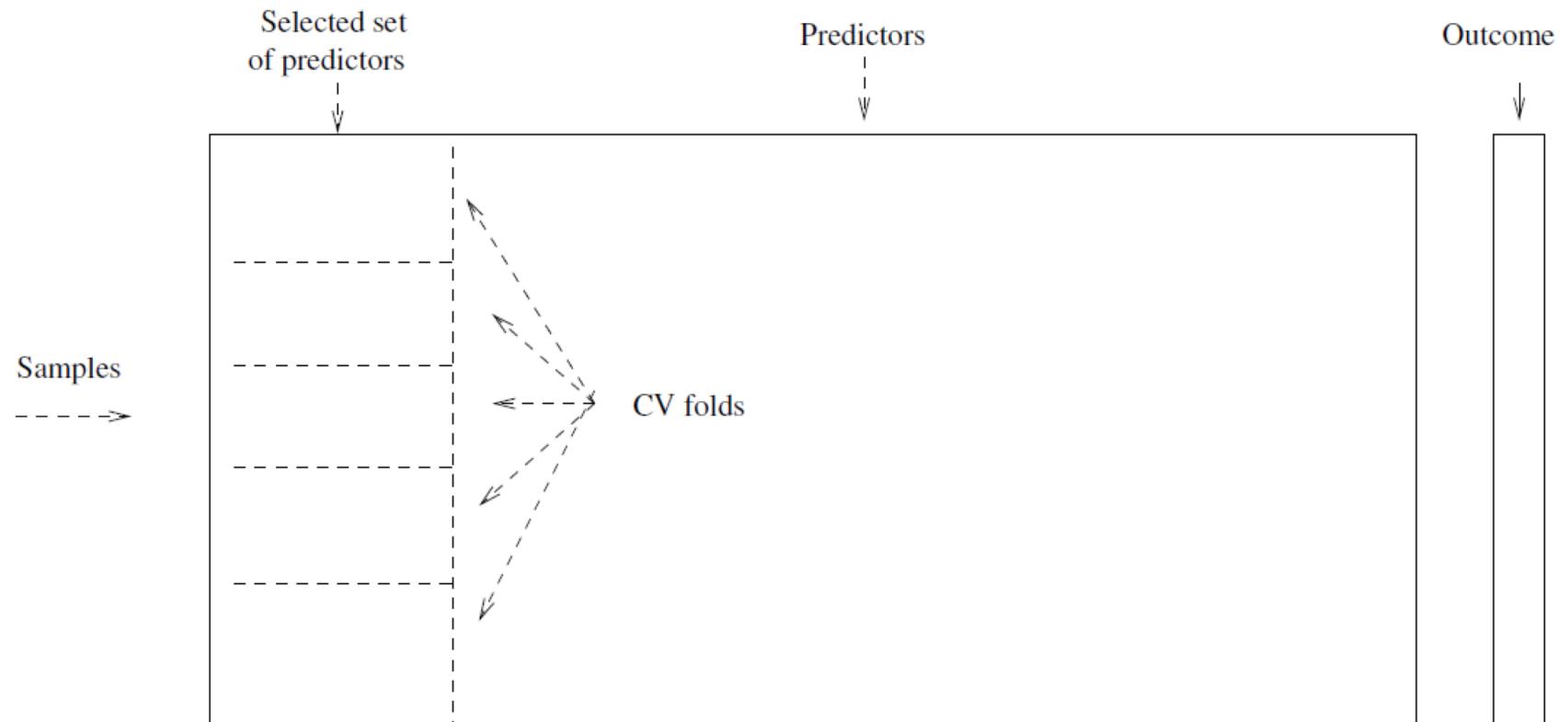
NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and **must be included** in the validation process.

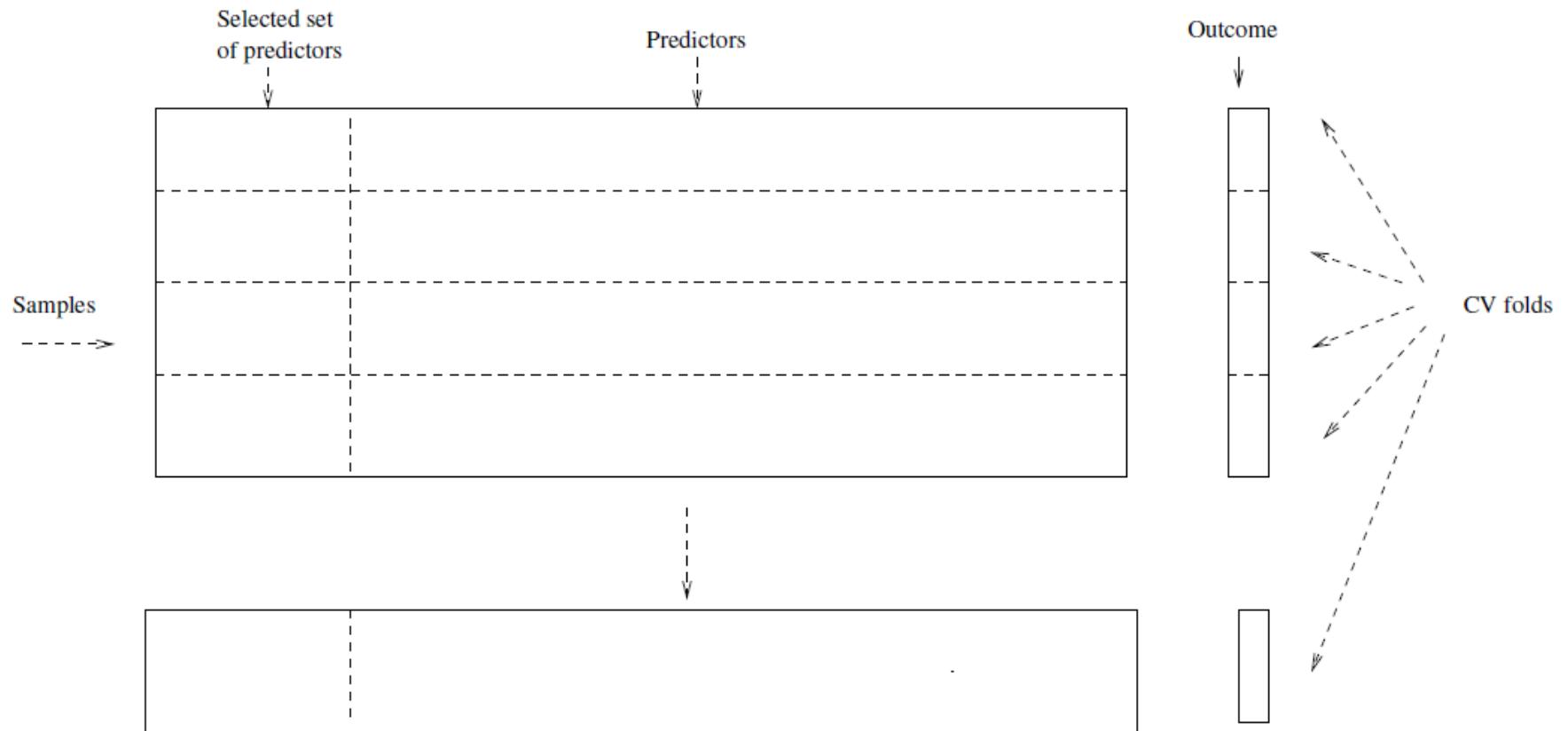
The Wrong and Right Way

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

Cross-Validation: Wrong Way



Cross-Validation: Right Way



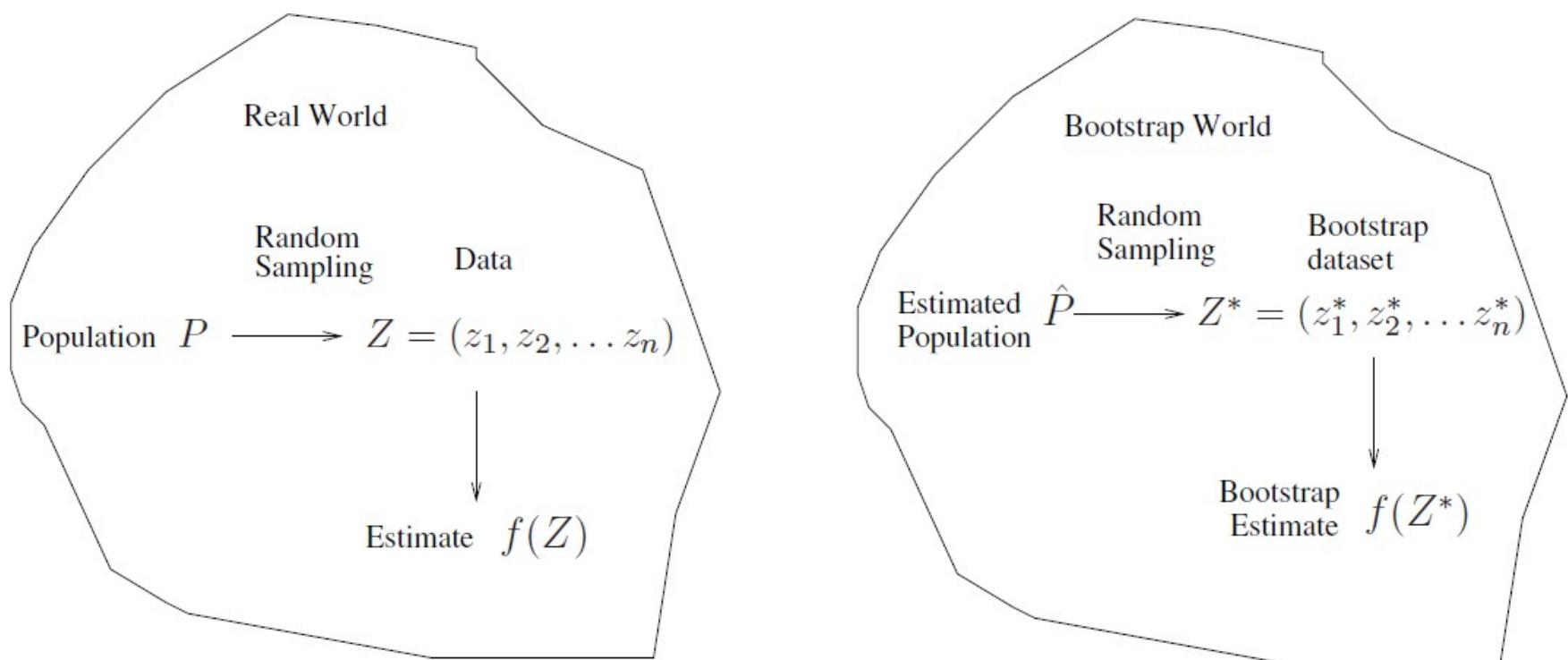
The Bootstrap

- The *bootstrap* is used to quantify uncertainty associated with a given estimator or machine learning method; it is a general tool for assessing statistical accuracy.

The Bootstrap

- As an example, it can be used to estimate the standard errors of the coefficients from a linear regression fit, or a confidence interval for that coefficient.
- The use of the term bootstrap derives from the phrase “to pull oneself up by one’s bootstraps.”

The Bootstrap: Overview



The Bootstrap: Overview (cont.)

Suppose we have a model fit to a set of training data. We denote the training set by $\mathbf{Z} = (z_1, z_2, \dots, z_N)$ where $z_i = (x_i, y_i)$.

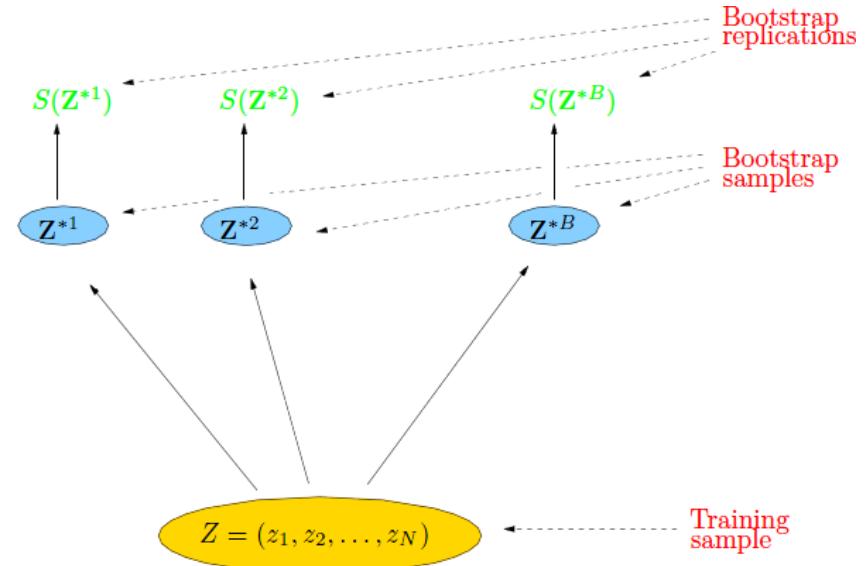
The basic idea is to randomly draw datasets **with replacement** from the training data, each sample the same size as the original training set.

This is done B times, producing B bootstrap datasets. Then we refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the B replications.

The Bootstrap: Overview (cont.)

$S(\mathbf{Z})$ is any quantity computed from the data \mathbf{Z} , for example, the prediction at some input point.

From the bootstrap sampling we can estimate any aspect of the distribution of $S(\mathbf{Z})$, for example, its variance:



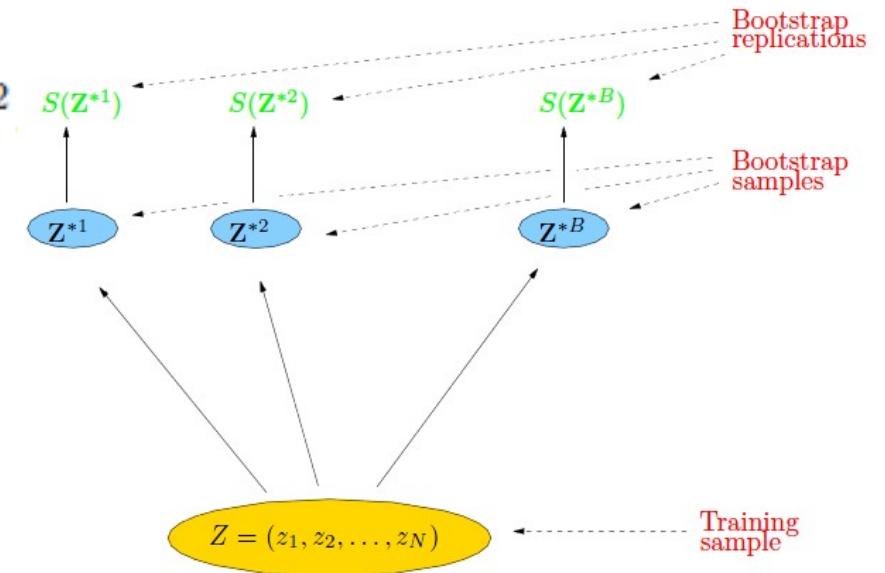
$$\widehat{\text{Var}}[S(\mathbf{Z})] = \frac{1}{B-1} \sum_{b=1}^B (S(\mathbf{Z}^{*b}) - \bar{S}^*)^2$$

$$\bar{S}^* = \sum_b S(\mathbf{Z}^{*b}) / B$$

The Bootstrap: Overview (cont.)

$$\widehat{\text{Var}}[S(\mathbf{Z})] = \frac{1}{B-1} \sum_{b=1}^B (S(\mathbf{Z}^{*b}) - \bar{S}^*)^2$$

$$\bar{S}^* = \sum_b S(\mathbf{Z}^{*b})/B$$



Note that this estimated variance can be thought of as a Monte-Carlo estimate of the variance of $S(\mathbf{Z})$ under sampling from the empirical distribution function.

The Bootstrap: An Example

Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns X and Y ; note that X and Y are random quantities.

We will invest a fraction α of our money in X , and will invest the remaining $1 - \alpha$ in Y .

Goal: Since there is variability associated with the returns on these two assets, we wish to choose α to minimize the total risk, or variance, of our investment.

The Bootstrap: An Example (cont.)

In other words, we wish to minimize

$$\text{Var}(\alpha X + (1 - \alpha) Y) =$$

The value that minimizes the risk, in this examples, is:

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

where

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

The Bootstrap: An Example (cont.)

In reality, the quantities, σ_X^2 , σ_Y^2 and σ_{XY} are unknown, so we can compute estimates for these quantities using a data set that contains past measurements for X and Y . We can then estimate the value of α that minimizes risk using:

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

The Bootstrap: An Example (cont.)

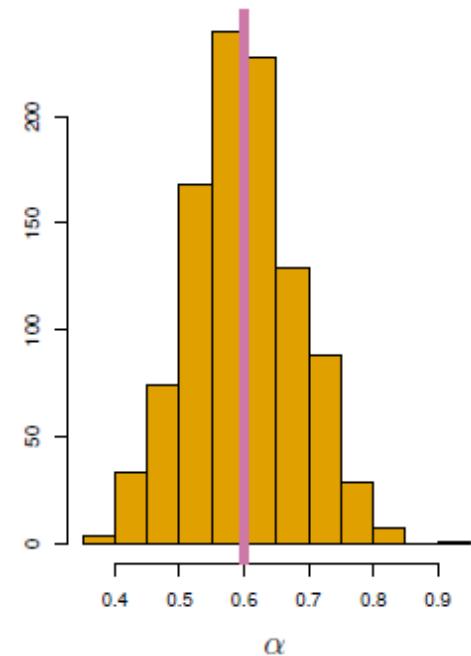
To do so, the process of simulating 100 paired observations of X and Y is repeated 1,000 times; now we have 1,000 estimates for α .

The mean over all 1,000 estimates

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

The standard deviation of the

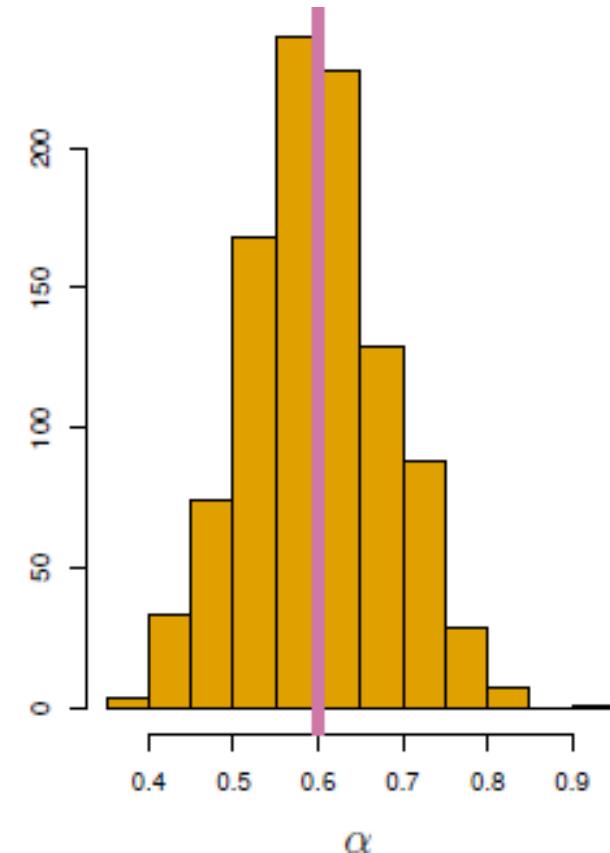
$$\text{estimates} \rightarrow \sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$



The Bootstrap: An Example (cont.)

We can use this approach for estimating α on a simulated data set, where 100 pairs of returns for the investments X and Y are simulated.

In order to quantify the accuracy of our estimate of α , we also estimate the standard deviation of $\hat{\alpha}$



The Bootstrap: An Example (cont.)

In rough terms, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

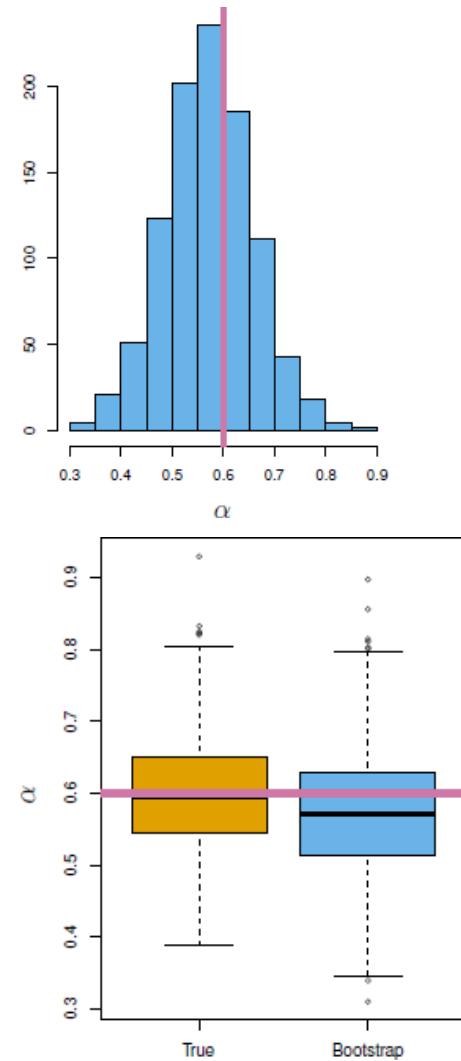
In the real world, this procedure **cannot be applied** because for real data we cannot generate new samples from the original population.

The Bootstrap: An Example (cont.)

The **bootstrap** approach, however, allows us to use a computer to mimic the process of obtaining new data sets; this enables use to estimate the variability of our estimate without generating additional samples.

The Bootstrap: An Example (cont.)

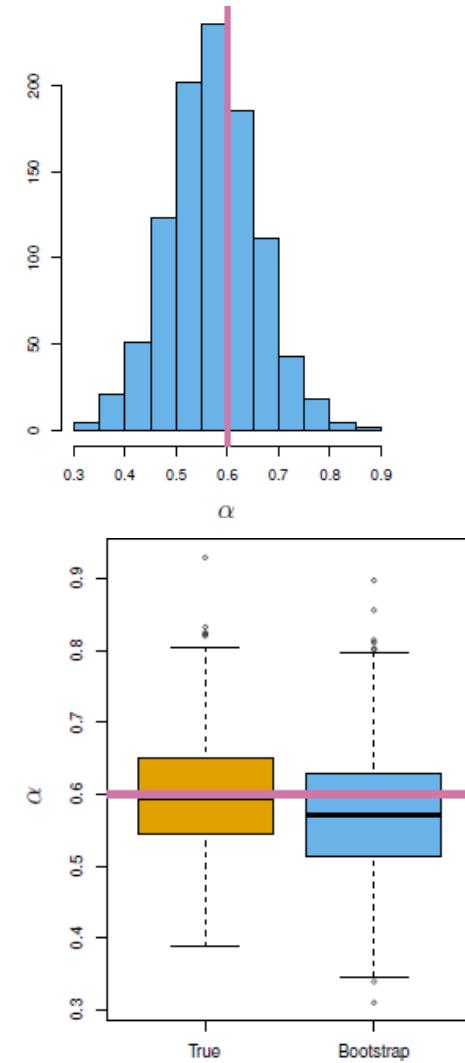
Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set **with replacement**.



The Bootstrap: An Example (cont.)

Each of these *bootstrap data sets* is created by sampling *with replacement*, and is the *same size* as the original data set.

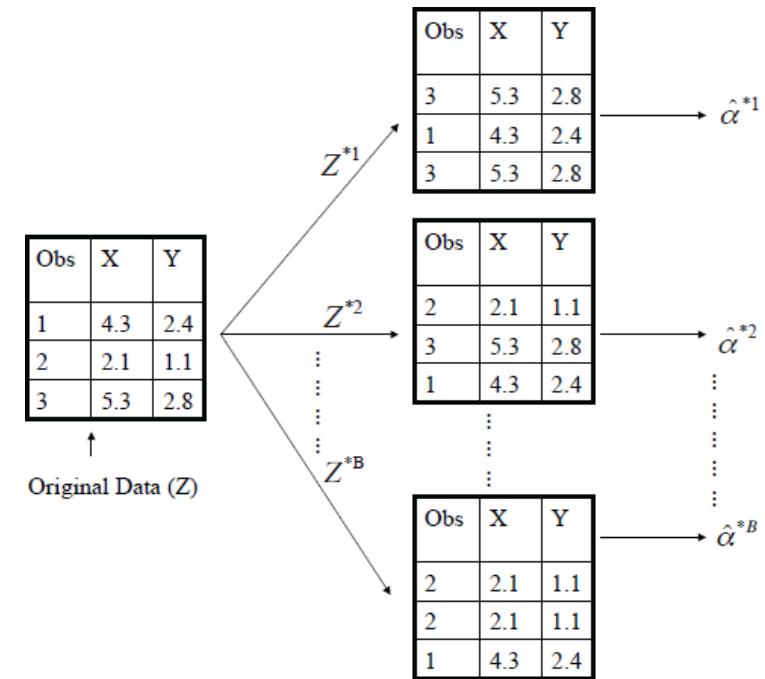
As a result, some observations may appear more than once in a given bootstrap data set and some not at all.



The Bootstrap: An Example (cont.)

A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations.

Each bootstrap data set contains n observations, sampled **with replacement** from the original data set.



Each bootstrap data set is used to obtain an estimate of α .

The Bootstrap: An Example (cont.)

Denoting the first bootstrap data set by Z^{*1} , we can use Z^{*1} to produce a new bootstrap estimate for α , which we call $\hat{\alpha}^{*1}$.

This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \dots, Z^{*B}$, and B corresponding α estimates, $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \dots, \hat{\alpha}^{*B}$.

The Bootstrap: An Example (cont.)

We compute the standard error of these bootstrap estimates using:

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

This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set.

The Bootstrap: More Details

In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thoughts.

For example, if the data is a time series, we cannot simply sample the observations with replacement.

The Bootstrap: More Details

However, we can instead create blocks of consecutive observations, and sample those with replacement. Then, we paste together sampled blocks to obtain a bootstrap dataset.

The Bootstrap: More Details (cont.)

Although the bootstrap is used primarily to obtain standard errors of an estimate, it can also provide approximate confidence intervals for a population parameters.

One approach is known as *Bootstrap Percentile* confidence interval.

Bootstrap Confidence Intervals

Easy Algorithm for Building $(1-\alpha)$ Confidence Interval Using Bootstrap

1. Create B bootstrap datasets from original dataset with the same size N .
2. Calculate the mean of each bootstrap sample m_1, m_2, \dots, m_B . Consider them as a sample of the sample means.
3. Order the sample means and call them $m_{(1)}, m_{(2)}, \dots, m_{(B)}$
4. The middle $(1-\alpha)B$ of the sample yield the $(1-\alpha)$ Confidence interval for the mean

Bootstrap Confidence Intervals

Easy Algorithm for Building $(1-\alpha)$ Confidence Interval for The Mean Using Bootstrap

Note: The algorithm on the previous slide can be applied to any statistic like median, variance, standard deviation, quartiles, etc.

To calculate a Bootstrap C.I. for any statistic, it is adequate to calculate that statistic from each bootstrap sample, instead of the sample mean.

Bootstrap Percentile Confidence Interval: Example

Assume the sample data is

30, 37, 36, 43, 42, 43, 43, 46, 41, 42

Problem: Estimate the mean μ of the underlying distribution and give an 80% bootstrap confidence interval.

Bootstrap Percentile Confidence Interval: Example

The sample mean is $\bar{x} = 40.3$, which is an estimate of the true mean μ of the underlying distribution.

To construct the confidence interval we need to know how much the distribution of \bar{x} varies around μ .

Bootstrap Percentile Confidence Interval: Example

20 bootstrap samples were generated, each of size 10. Each of the 20 columns in the following array is one bootstrap sample.

43	36	46	30	43	43	43	37	42	42	43	37	36	42	43	43	42	43	42	43
43	41	37	37	43	43	46	36	41	43	43	42	41	43	46	36	43	43	43	42
42	43	37	43	46	37	36	41	36	43	41	36	37	30	46	46	42	36	36	43
37	42	43	41	41	42	36	42	42	43	42	43	41	43	36	43	43	41	42	46
42	36	43	43	42	37	42	42	42	46	30	43	36	43	43	42	37	36	42	30
36	36	42	42	36	36	43	41	30	42	37	43	41	41	43	43	42	46	43	37
43	37	41	43	41	42	43	46	46	36	43	42	43	30	41	46	43	46	30	43
41	42	30	42	37	43	43	42	43	43	46	43	30	42	30	42	30	43	43	42
46	42	42	43	41	42	30	37	30	42	43	42	43	37	37	37	42	43	43	46
42	43	43	41	42	36	43	30	37	43	42	43	41	36	37	41	43	42	43	43

Bootstrap Percentile Confidence Interval: Example

Next we compute \bar{x} for each bootstrap sample (i.e. each column) and sort them from smallest to biggest:

43	36	46	30	43	43	43	37	42	42	43	37	36	42	43	43	42	43	42	43
43	41	37	37	43	43	46	36	41	43	43	42	41	43	46	36	43	43	43	42
42	43	37	43	46	37	36	41	36	43	41	36	37	30	46	46	42	36	36	43
37	42	43	41	41	42	36	42	42	43	42	43	41	43	36	43	43	41	42	46
42	36	43	43	42	37	42	42	42	46	30	43	36	43	43	42	37	36	42	30
36	36	42	42	36	36	43	41	30	42	37	43	41	41	43	43	42	46	43	37
43	37	41	43	41	42	43	46	46	36	43	42	43	30	41	46	43	46	30	43
41	42	30	42	37	43	43	42	43	43	46	43	30	42	30	42	30	43	43	42
46	42	42	43	41	42	30	37	30	42	43	42	43	37	37	37	42	43	43	46
42	43	43	41	42	36	43	30	37	43	42	43	41	36	37	41	43	42	43	43

Sorted Sample Means

38.7 38.9 38.9 39.4 39.8 40.1 40.2 40.4 40.5 40.5 40.7 40.7 40.7 41 41.2 41.4 41.5 41.5 41.9 41.9 42.3

Bootstrap Percentile Confidence Interval: Example

The middle 80% of the data gives us the C.I., i.e. $[m_3, m_{18}] = [38.9, 41.9]$

The Bootstrap: More Details (cont.)

In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training (i.e. there is no overlap).

To estimate the prediction error using the bootstrap, one approach would be to fit the model in question on a set of bootstrap samples, and then keep track of how well it predicts the **original training set**.

The Bootstrap: More Details (cont.)

If $\hat{f}^{*b}(x_i)$ is the predicted value at x_i , from the model fitted to the b^{th} bootstrap dataset, our estimate is:

$$\widehat{\text{Err}}_{\text{boot}} = \frac{1}{B} \frac{1}{N} \sum_{b=1}^B \sum_{i=1}^N L(y_i, \hat{f}^{*b}(x_i))$$

The Bootstrap: More Details (cont.)

This estimate does not provide a good estimate in general because the bootstrap datasets are acting as the training samples, while the original training set is acting as the test sample, and these two samples have observations in common.

Each bootstrap sample has significant overlap with the original data.

The Bootstrap: More Details (cont.)

- Note that about 2/3 of the original N data points appear in each bootstrap sample:

$$\begin{aligned}\Pr\{\text{Observation } i \in \text{bootstrap sample } b\} \\ = 1 - \Pr\{\text{Observation } i \notin \text{bootstrap sample } b\}\end{aligned}$$

- On the other hand

$$\begin{aligned}\Pr\{\text{Observation } i \notin \text{bootstrap sample } b\} \\ = \Pr\{\text{Observation } i \text{ is not any of the } N \text{ members of the bootstrap sample } b\}\end{aligned}$$

The Bootstrap: More Details (cont.)

- One the other hand

$$\Pr\{\text{Observation } i \notin \text{bootstrap sample } b\}$$

$$= \Pr\{\text{Observation } i \text{ is not any of the } N \text{ members of the bootstrap sample } b\}$$

$$= \Pr\{\text{Each of the } N \text{ members of the bootstrap sample } b \text{ is one of the } N-1 \text{ observations other than } i\}$$

- Therefore

$$\Pr\{\text{Observation } i \notin \text{bootstrap sample } b\} = (1 - 1/N)^N$$

when N is large, this is equal to e^{-1}

- So

$$\Pr\{\text{Observation } i \in \text{bootstrap sample } b\} \approx 1 - e^{-1} \approx 0.632$$

The Bootstrap: More Details (cont.)

This overlap can make overfit predictions like unrealistically good, and is the reason that cross-validation explicitly uses non-overlapping data for the training and test samples.

In other words, this will cause the bootstrap to seriously *underestimate* the true prediction error.

By mimicking cross-validation, a better bootstrap estimate can be obtained.

The Bootstrap: More Details (cont.)

For each observation, we only keep track of prediction from bootstrap samples not containing that observation.

The leave-one-out bootstrap estimate of prediction error is defined by:

$$\widehat{\text{Err}}^{(1)} = \frac{1}{N} \sum_{i=1}^N \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_i, \hat{f}^{*b}(x_i))$$

Here C^{-i} is the set of indices of the bootstrap samples b that do *not* contain observation i , and $|C^{-i}|$ is the number of such samples.

The Bootstrap: More Details (cont.)

Note that the leave-one-out bootstrap solves the problem of overfitting, but has a training-set-size bias.

The “.632 estimator” is designed to alleviate this bias.

Summary

Resampling methods for machine learning model selection and assessment.

The validation set approach, leave-one-out cross-validation, and K -fold cross-validation.

The bootstrap method for assessing statistical accuracy.

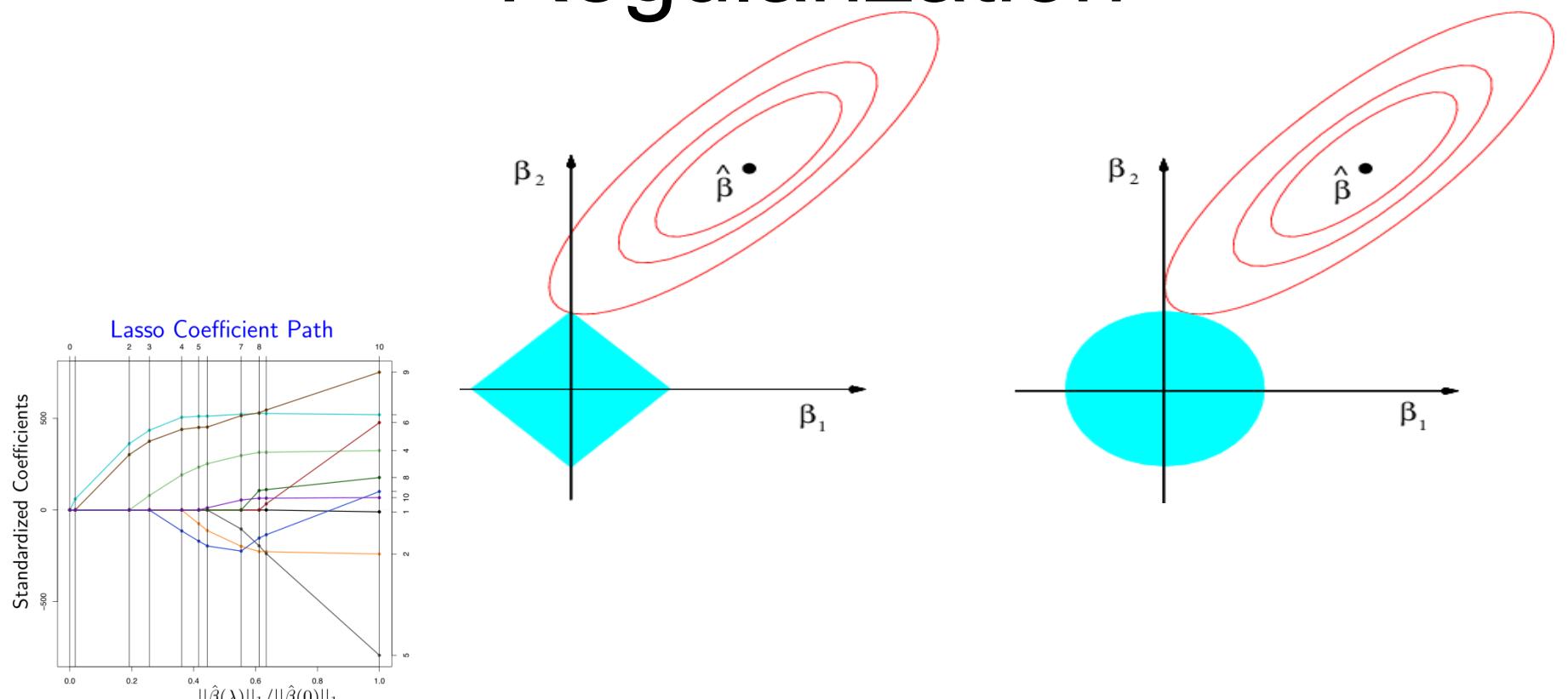
DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 5

Linear Model Selection and Regularization



$$\text{Lasso: } \hat{\beta}(\lambda) = \operatorname{argmin}_{\beta} \sum_{i=1}^N (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \|\beta\|_1$$

Linear Model Selection and Regularization

- Recall the linear model

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

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

All hail to linear models!

- Despite its simplicity, the linear model has distinct advantages in terms of its *interpretability* and often shows good (acceptable) *predictive performance*.

All hail to linear models!

- To improve the simple linear model, the ordinary least squares fitting can be replaced with some alternative fitting procedures.

Why consider alternatives to least squares?

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

Three classes of methods

- *Subset Selection*. Identifying a subset of the p predictors that are related to the response and fitting a model using least squares on the reduced set of variables.

Three classes of methods

- ***Shrinkage***. Fitting a model involving all p predictors, but the estimated coefficients are shrunken towards zero relative to the least squares estimates.
 - shrinkage (= *regularization*) reduces the variance and can perform variable selection.

Three classes of methods

- *Dimension Reduction*. The p predictors are projected into a M -dimensional subspace, where $M < p$.
 - Achieved by computing M different *linear combinations*, or *projections*, of the variables.
 - These M projections are used as predictors to fit a linear regression model by least squares.

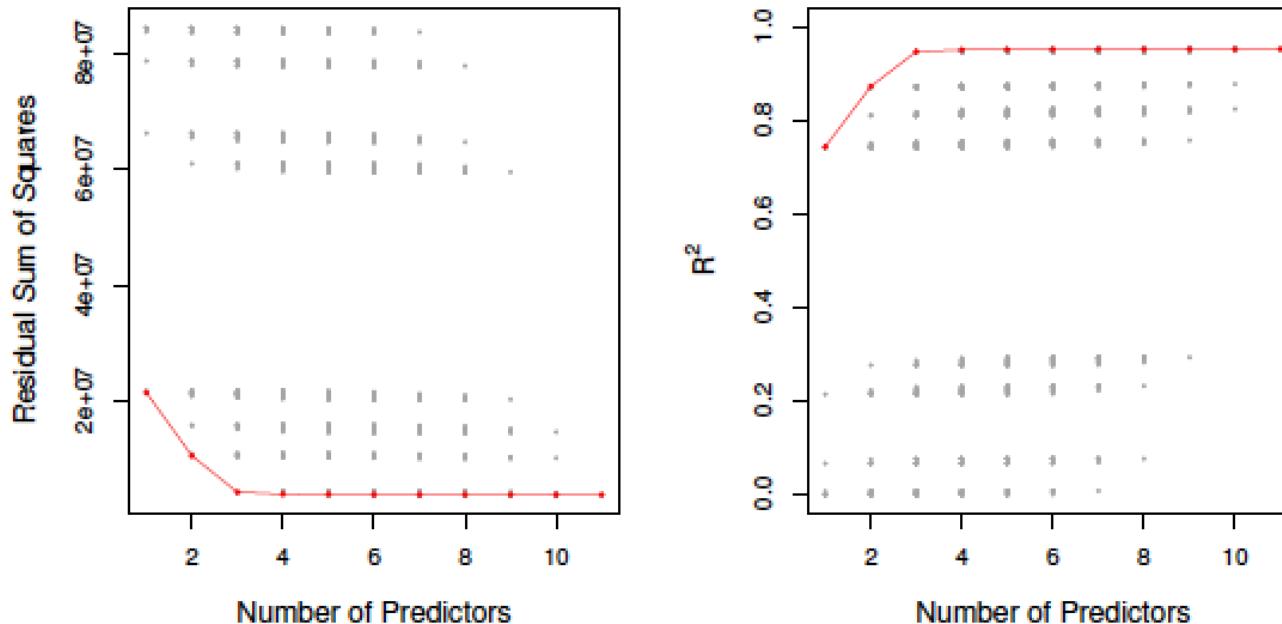
Subset Selection

Best subset and stepwise model selection procedures

Best Subset Selection

1. Let M_0 denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \dots, p$:
 - Fit all $\binom{p}{k}$ models that contain exactly k predictions
 - Pick the best among these $\binom{p}{k}$ models and call it M_k .
 - Here best is defined as having the smallest RSS, or equivalently largest R^2 .
3. Select a single best model from among M_0, \dots, M_p using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .

Example- Credit data set



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

Extensions to other models

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

Stepwise Selection

- Best subset selection cannot be applied with very large p . *Why not?*
- Best subset selection may also suffer from statistical problems when p is large: the larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data.

Stepwise Selection

- Thus an enormous search space can lead to *overfitting* and high variance of the coefficient estimates.
- For both of these reasons, *stepwise* methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

Forward Stepwise Selection

- Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model.

Forward Stepwise Selection

- In particular, at each step the variable that gives the greatest *additional* improvement to the fit is added to the model.

Forward Stepwise Selection In Detail

1. Let M_0 denote the *null* model, which contains no predictors.
2. For $k = 0, \dots, p - 1$:
 1. Consider all $p - k$ models that augment the predictors in M_k with one additional predictor.
 2. Choose the *best* among these $p - k$ models, and call it M_{k+1} . Here *best* is defined as having smallest RSS or highest R^2 .
3. Select a single best model from among M_0, \dots, M_p using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .

More on Forward Stepwise Selection

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

Why not? Give an example.

Credit data example

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

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

Backward Stepwise Selection

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

Backward Stepwise Selection: details

Backward Stepwise Selection

1. Let M_p denote the *full* model, which contains all p predictors.
2. For $k = p, p - 1, \dots, 1$:
 1. Consider all k models that contain all but one of the predictors in M_k , for a total of $k - 1$ predictors.
 2. Choose the *best* among these k models, and call it M_{k-1} . Here *best* is defined as having smallest RSS or highest R^2 .
3. Select a single best model from among M_0, \dots, M_p using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .

More on Backward Stepwise Selection

- Like forward stepwise selection, the backward selection approach searches through only $1 + p(p + 1)/2$ models, and so can be applied in settings where p is too large to apply best subset selection
- Like forward stepwise selection, backward stepwise selection is **not guaranteed** to yield the *best* model containing a subset of the p predictors.

More on Backward Stepwise Selection

- Backward selection requires that the *number of samples n is larger than the number of variables p* (so that the full model can be fit).
- In contrast, forward stepwise can be used even when $n < p$, and **so is the only viable subset method** when p is very large.

Choosing the Optimal Model

- The model containing all of the predictors will always have the smallest RSS and the largest R^2 , since these quantities are related to the training error.

Choosing the Optimal Model

- We wish to choose a model with low test error, not a model with low training error. Recall that training error is usually a poor estimate of test error.

Choosing the Optimal Model

- Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

Estimating test error: two approaches

- We can indirectly estimate test error by making an *adjustment* to the training error to account for the bias due to overfitting.

Estimating test error: two approaches

- We can *directly* estimate the test error, using either a validation set approach or a cross-validation approach, as discussed in previous lectures.
- We illustrate both approaches next.

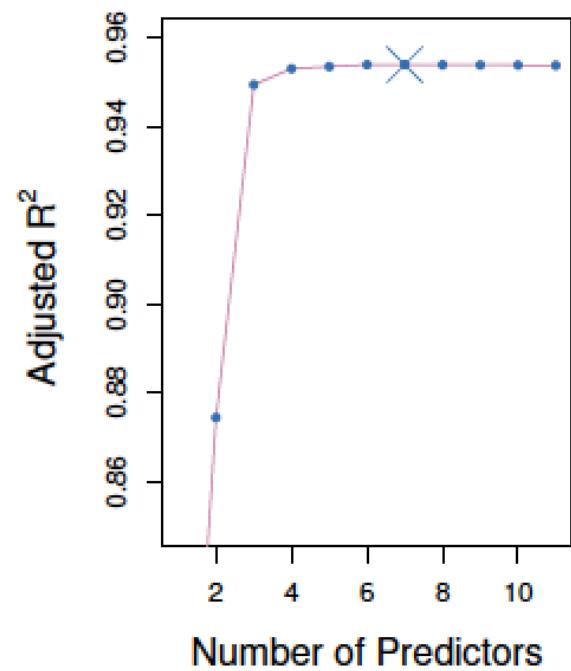
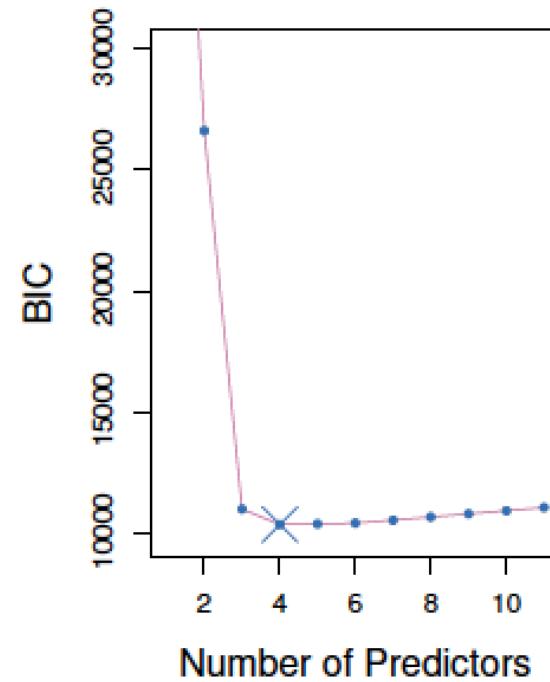
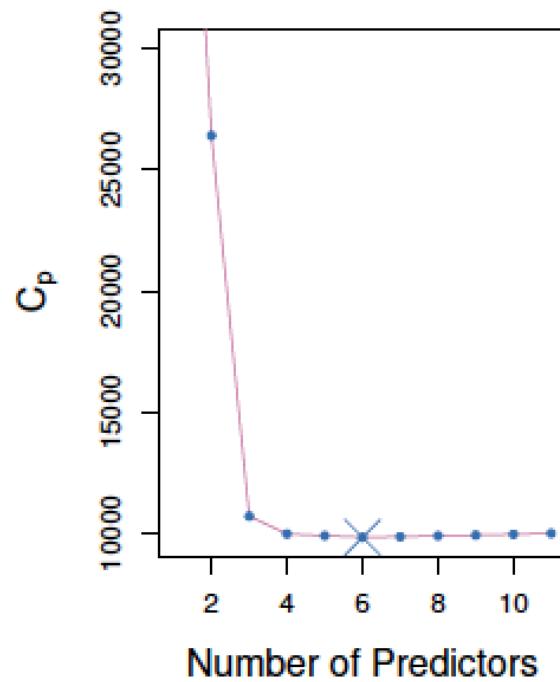
C_p , AIC, BIC, and Adjusted R^2

- These techniques adjust the training error for the model size, and can be used to select among a set of models with different numbers of variables.

C_p , AIC, BIC, and Adjusted R^2

- The next figure displays C_p , BIC, and adjusted R^2 for the best model of each size produced by best subset selection on the **Credit** data set.

Credit data example



Now for some details

Mallow's C_p :
$$C_p = \frac{1}{n} (\text{RSS} + 2d\hat{\sigma}^2)$$

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

We learned that Residual Sum of Squares, RSE, is used to estimate the variance of error:

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

Now for some details

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

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

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

Now for some details

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

Details on BIC

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

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

Adjusted R^2

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

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

where TSS is the total sum of squares.

Unlike C_p , AIC, and BIC, for which a small value indicates a model with a low test error, a large value of adjusted R^2 indicates a model with a small test error. Remember:

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

Adjusted R^2

Maximizing the adjusted R^2 is equivalent to minimizing $\text{RSS}/(n-d-1)$.

While RSS always decreases as the number of variables in the model increases, $\text{RSS}/(n-d-1)$ may increase or decrease, due to the presence of d in the denominator.

Unlike the R^2 statistic, the **adjusted R^2 statistic pays a price for the inclusion of unnecessary variables in the model.**

Validation and Cross-Validation

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

Validation and Cross-Validation

- We compute the validation set error or the cross-validation error for each model M_k under consideration, and then select the k for which the resulting estimated test error is smallest.

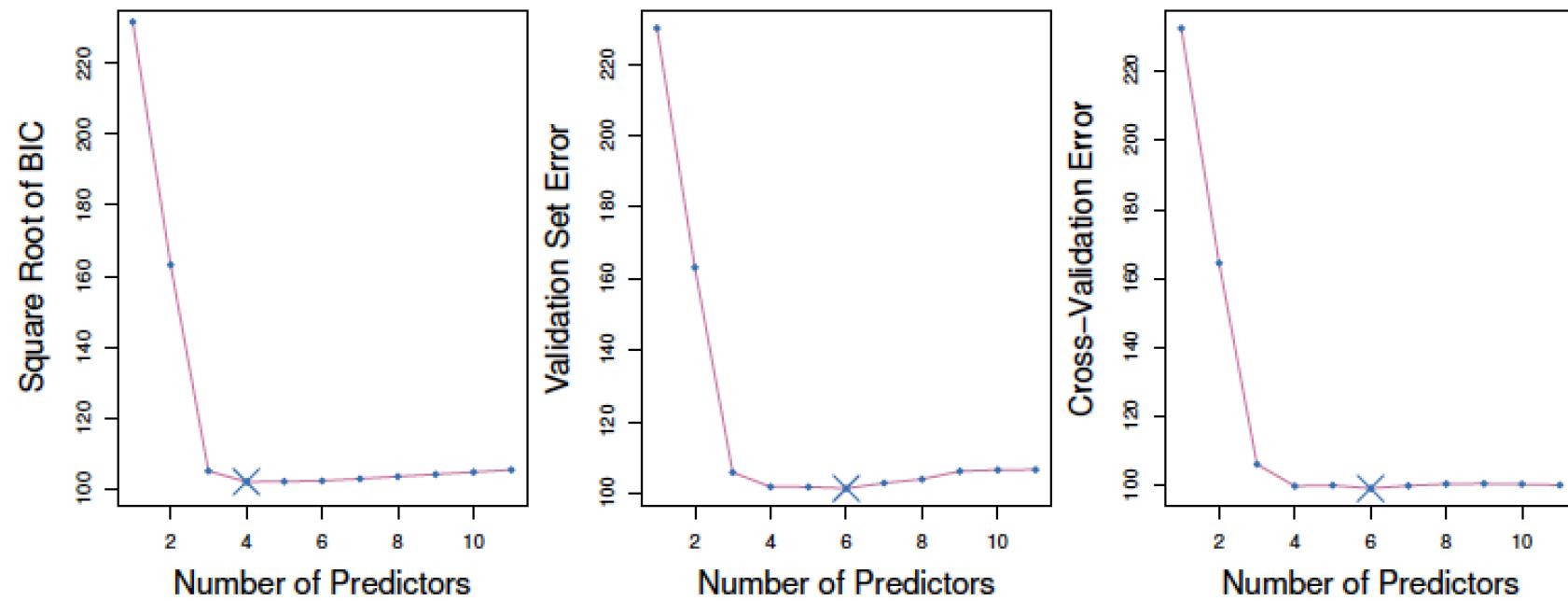
Validation and Cross-Validation

- This procedure has an advantage relative to AIC, BIC, C_p , and adjusted R^2 , in that it provides a direct estimate of the test error, and *doesn't require an estimate of the error variance σ^2 .*

Validation and Cross-Validation

- It can also be used in a wider range of model selection tasks, even in cases where **it is hard to pinpoint the model degrees of freedom** (e.g. the number of predictors in the model) or hard to estimate the error variance σ^2 .

Credit data example



Details of Previous Figure

- The validation errors were calculated by randomly selecting three-quarters of the observations as the training set, and the remainder as the validation set.

Details of Previous Figure

- The cross-validation errors were computed using $k = 10$ folds. In this case, the validation and cross-validation methods both result in a six-variable model.

Details of Previous Figure

- However, all three approaches suggest that the four-, five-, and six-variable models are roughly equivalent in terms of their test errors.

Shrinkage Methods

Ridge regression and *Lasso*

- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.

Shrinkage Methods

Ridge regression and *Lasso*

- As an alternative, we can fit a model containing all p predictors using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero.

Shrinkage Methods

Ridge regression and *Lasso*

- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can **significantly reduce their variance.**

Ridge regression

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

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

In contrast, the ridge regression coefficient estimates β^R are the values that minimize

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

where $\lambda \geq 0$ is a tuning parameter, to be determined separately. Note that β_0 is not regularized.

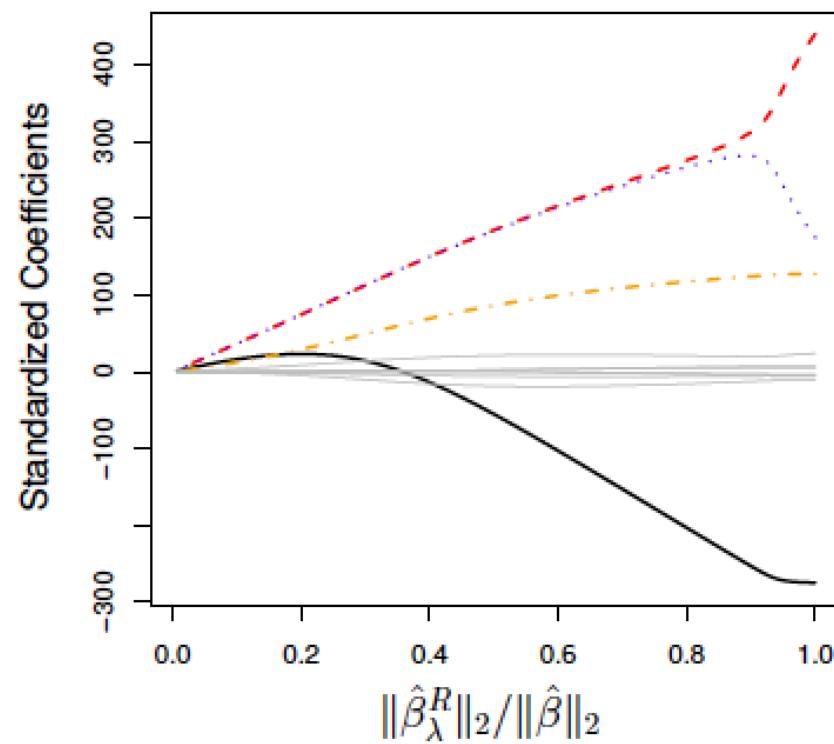
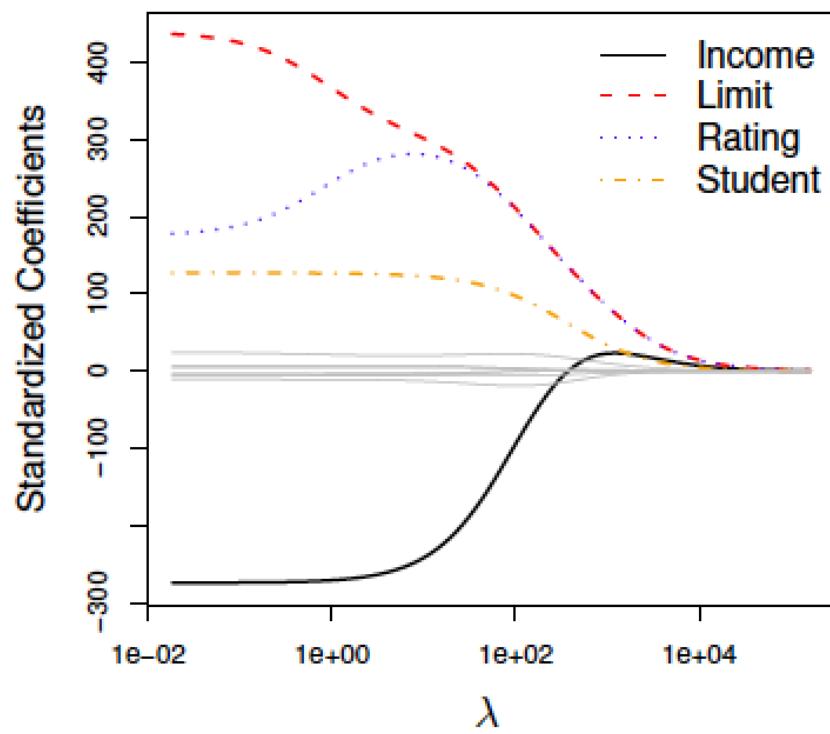
Ridge regression: continued

- As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small.
- However, the second term, $\lambda \sum_j \beta_j^2$, called a **shrinkage penalty**, is small when β_1, \dots, β_p are close to zero, and so it has the effect of **shrinking** the estimates of β_j towards zero.

Ridge regression: continued

- The tuning parameter serves to control the relative impact of these two terms on the regression coefficient estimates.
- Selecting a good value for λ is critical; cross-validation is used for this.

Credit data example



Details of Previous Figure

- In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of λ .
- The right-hand panel displays the same ridge coefficient estimates as the left-hand panel, but instead of displaying on the x-axis, we now display $\|\hat{\beta}_\lambda^R\|_2 / \|\hat{\beta}\|_2$, where β^\wedge denotes the vector of least squares coefficient estimates.
- The notation $\|\beta\|_2$ denotes the ℓ_2 norm of a vector, and is defined as

$$\|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}.$$

Ridge regression: scaling of predictors

- The standard least squares coefficient estimates are *scale equivariant*: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$.

Ridge regression: scaling of predictors

- In other words, regardless of how the j th predictor is scaled, $X_j \hat{\beta}_j$ will remain the same.

Ridge regression: scaling of predictors

- In contrast, the ridge regression coefficient estimates can change *substantially* when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.

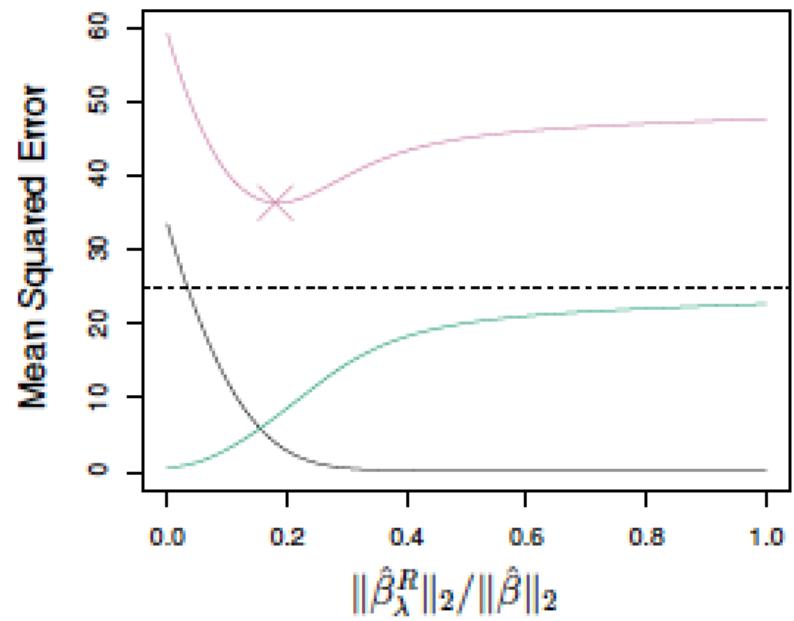
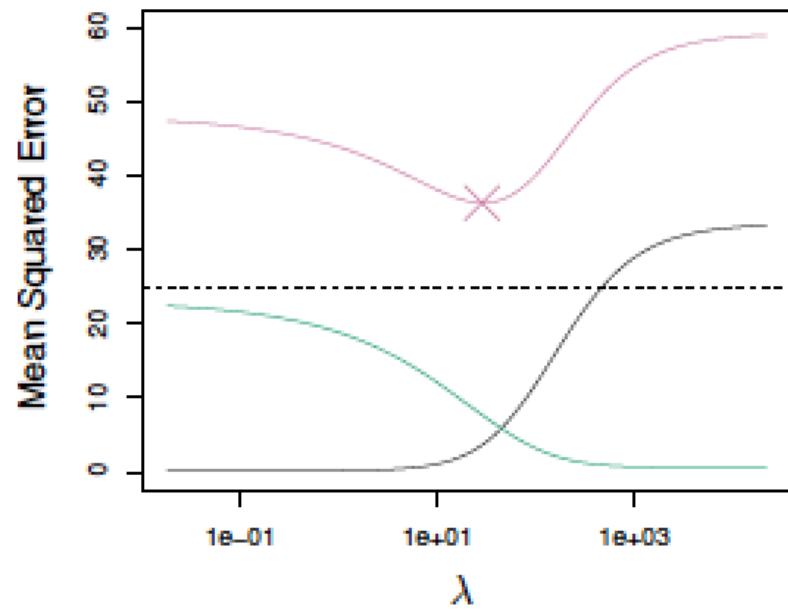
Ridge regression: scaling of predictors

- Therefore, it is best to apply ridge regression after *standardizing the predictors*, using the formula

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

- Practical note: try raw, standardized, and normalized data.

Ridge Regression Improves Over Least Squares?



The Bias-Variance tradeoff

Ridge Regression Improves Over Least Squares?

The Bias-Variance tradeoff

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

The Lasso

- Ridge regression does have one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model

The Lasso

- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso

The Lasso: continued

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.

The Lasso: continued

- However, in the case of the lasso, the L_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.

The Lasso: continued

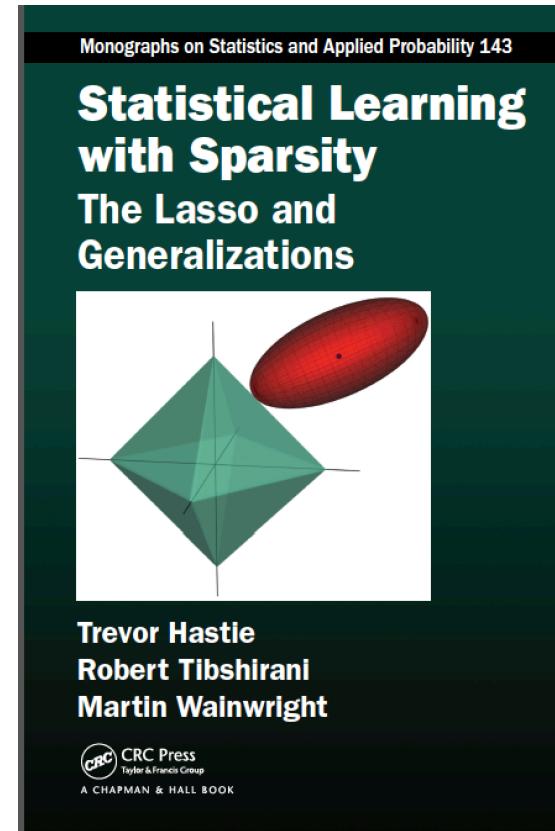
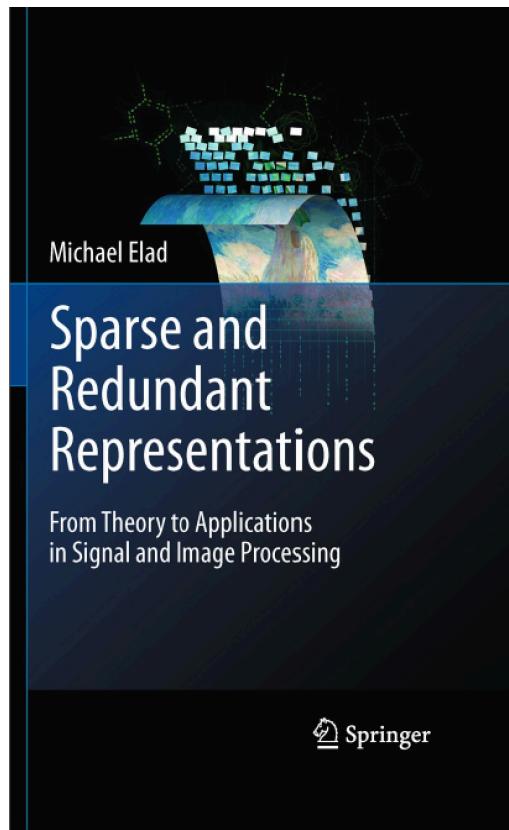
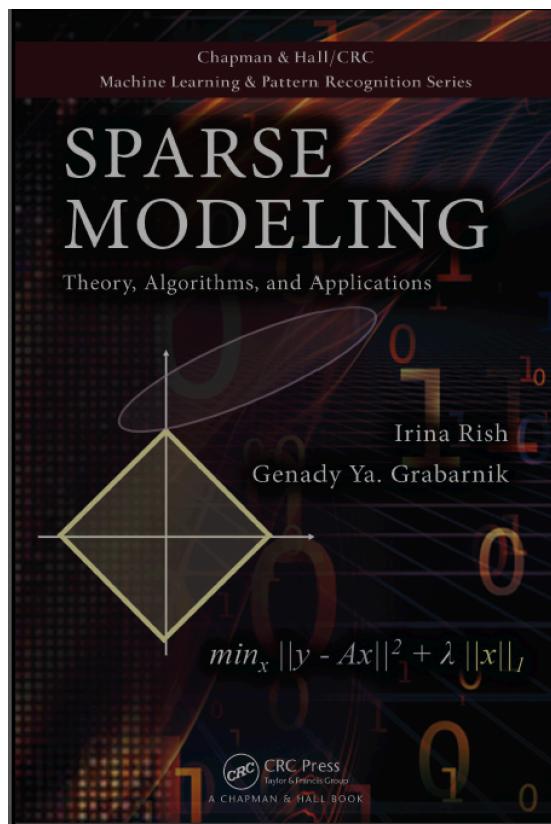
- Hence, much like best subset selection, the lasso performs *variable selection*.
- We say that the lasso yields *sparse* models — that is, models that involve only a subset of the variables.

The Lasso: continued

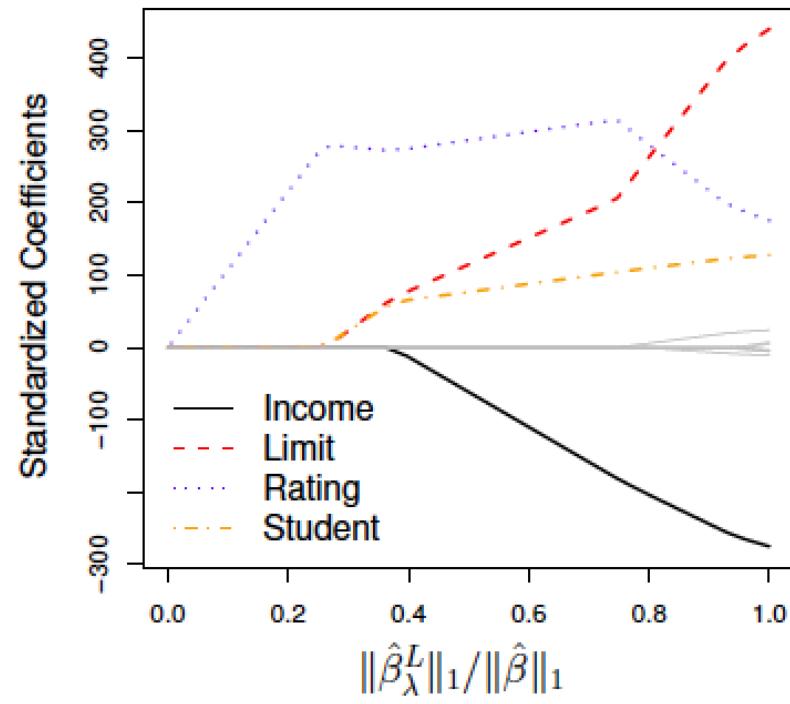
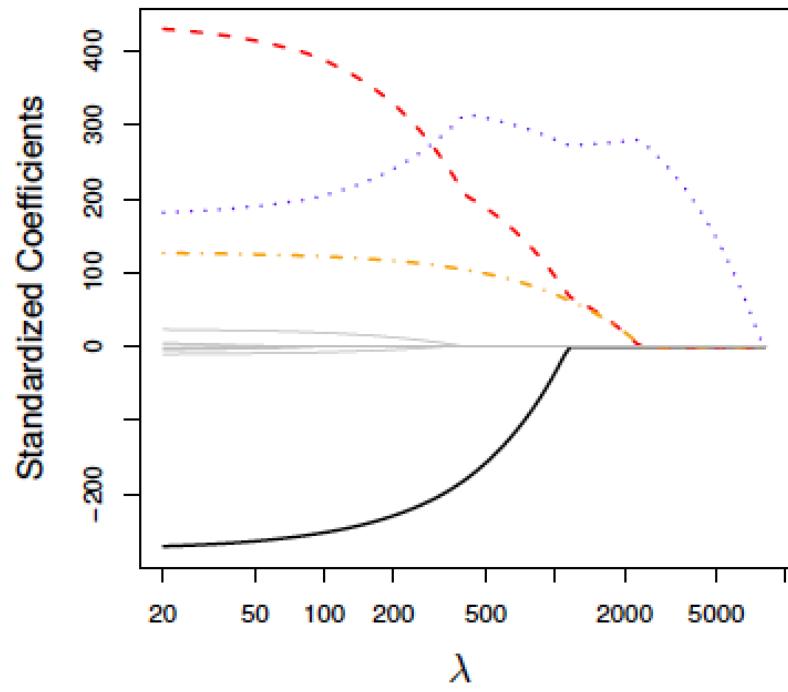
- As in ridge regression, selecting a good value of λ for the lasso is critical; cross-validation is again the method of choice.

Aside: Sparsity

Sparsity is an essential issue in Machine Learning



Example: Credit dataset



The Variable Selection Property of the Lasso

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

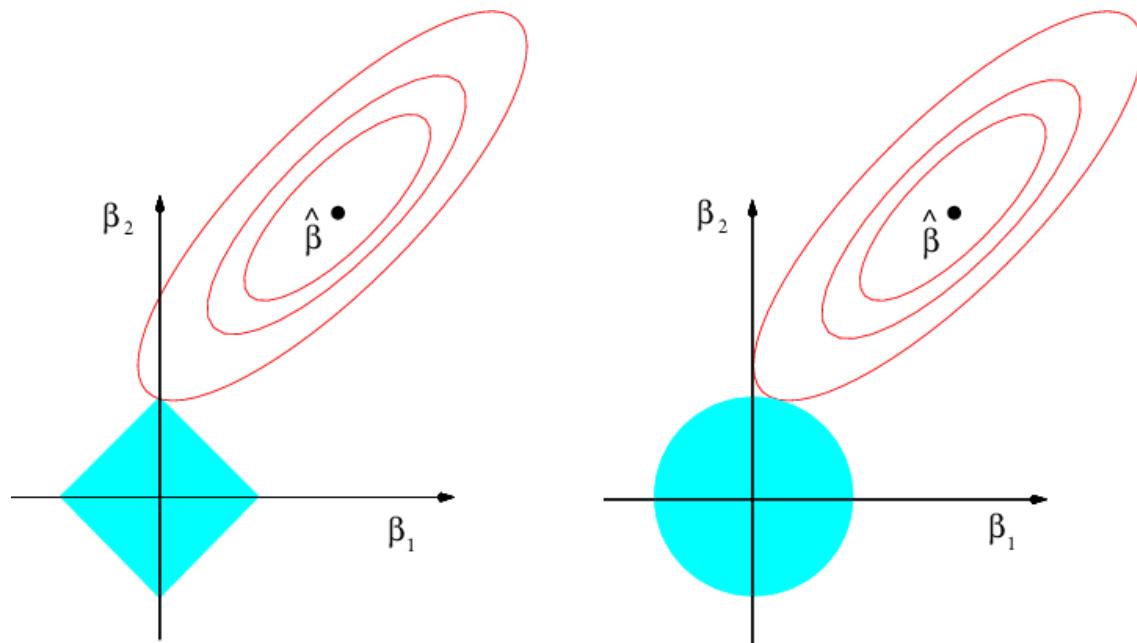
One can show that the lasso and ridge regression coefficient estimates respectively solve the problems

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

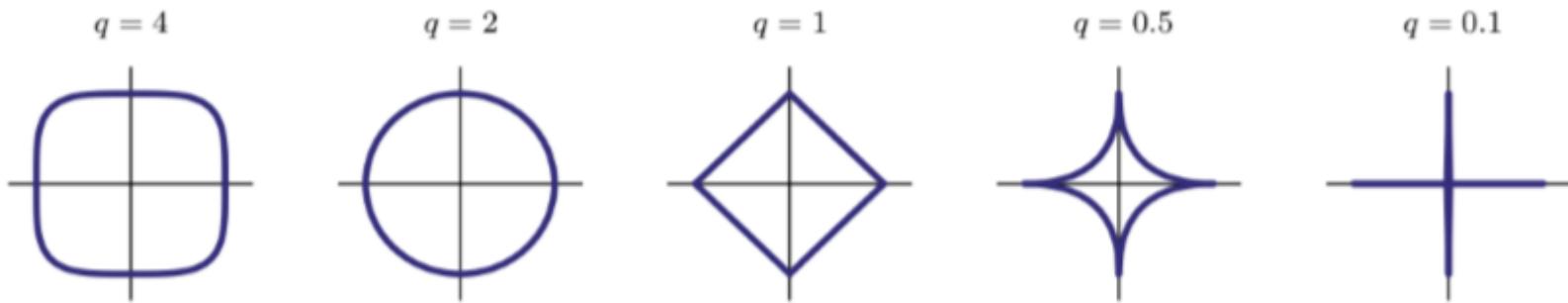
and

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

The Lasso Picture

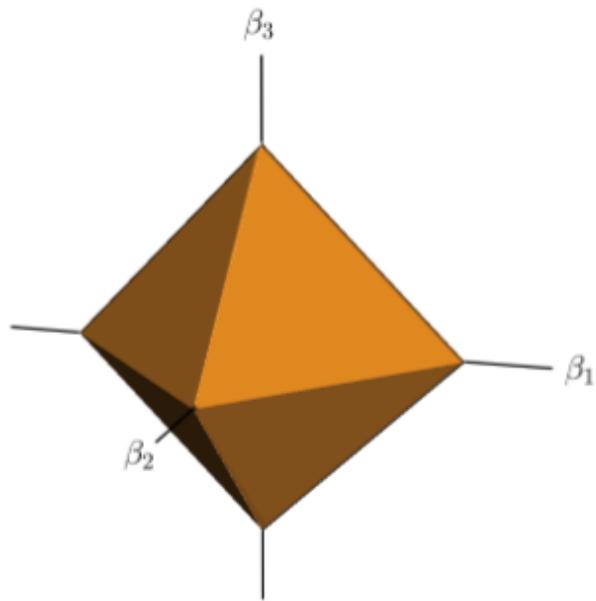


Different Constraints in 2D

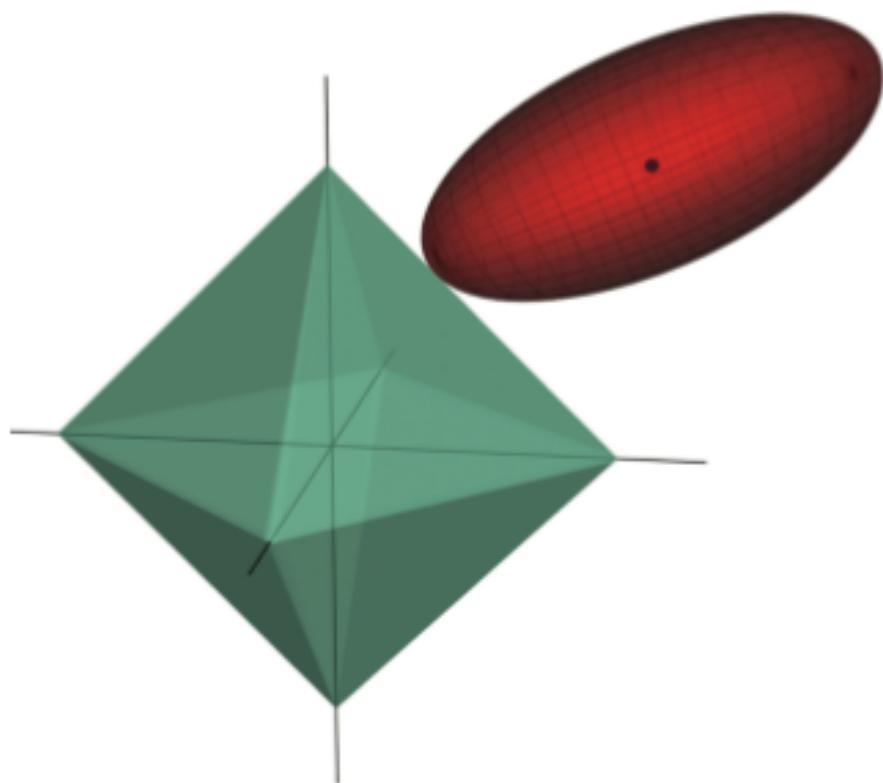


Constraint regions $\sum_{j=1}^p |\beta_j|^q \leq 1$ for different values of q . For $q < 1$, the constraint region is nonconvex.

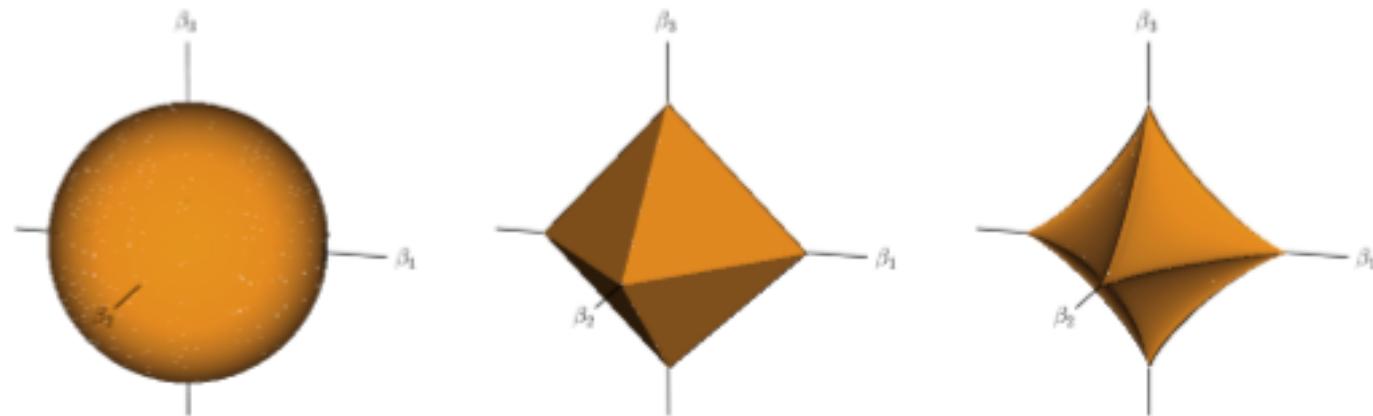
The Lasso Constraint in 3D



The Lasso in 3D

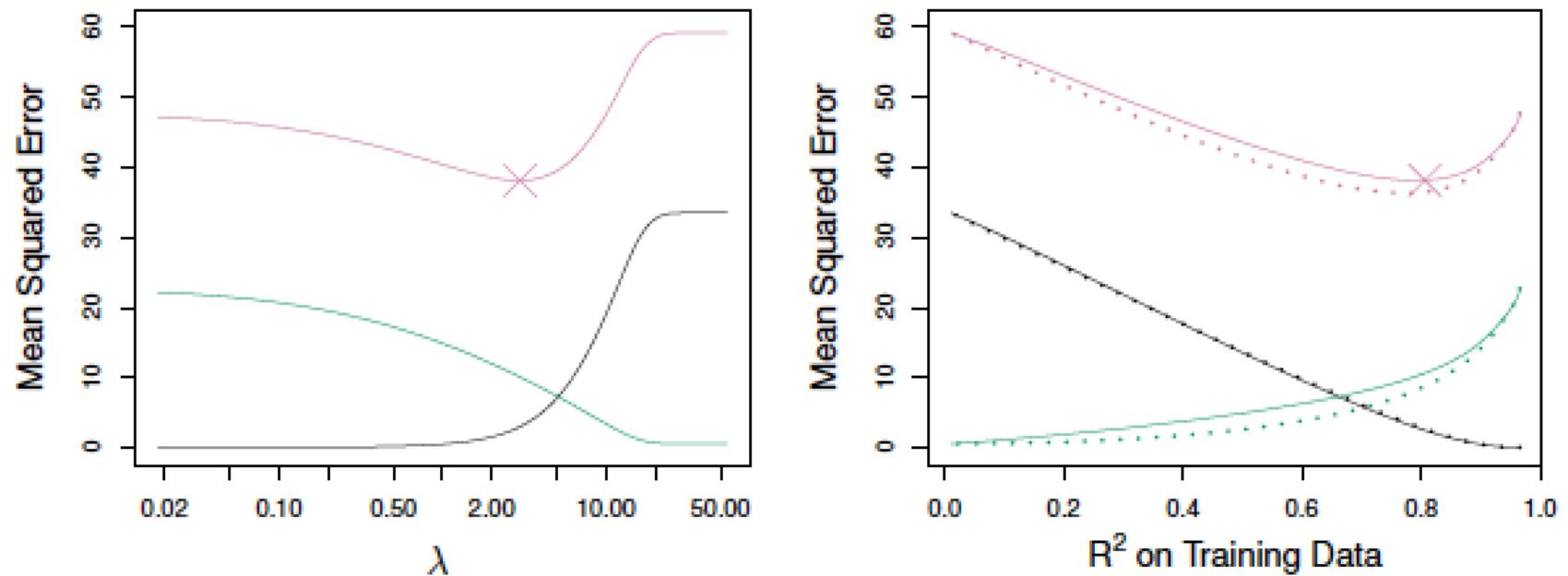


Other Constraints in 3D



The ℓ_q unit balls in \mathbb{R}^3 for $q = 2$ (left), $q = 1$ (middle), and $q = 0.8$ (right). For $q < 1$ the constraint regions are nonconvex. Smaller q will correspond to fewer nonzero coefficients, and less shrinkage. The nonconvexity leads to combinatorially hard optimization problems.

Comparing the Lasso and Ridge Regression

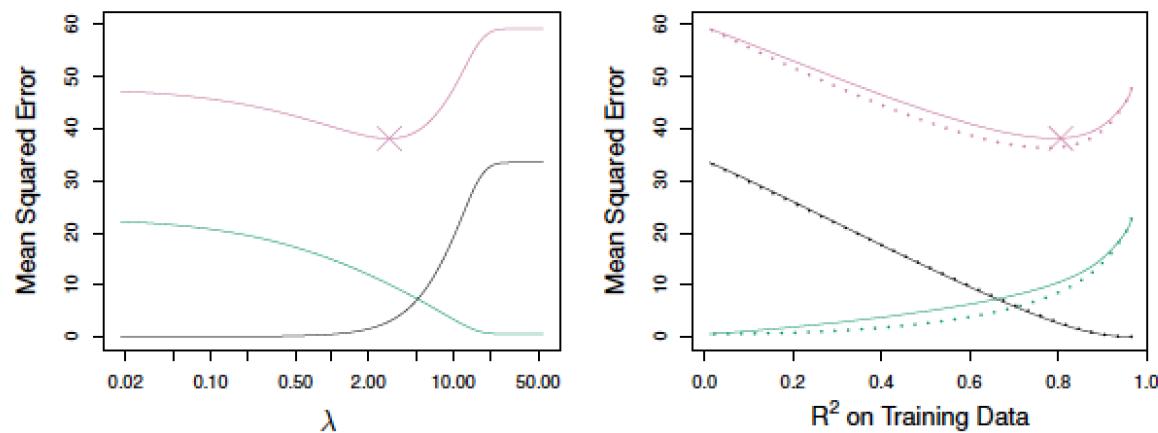


Same simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients.

Lasso: solid

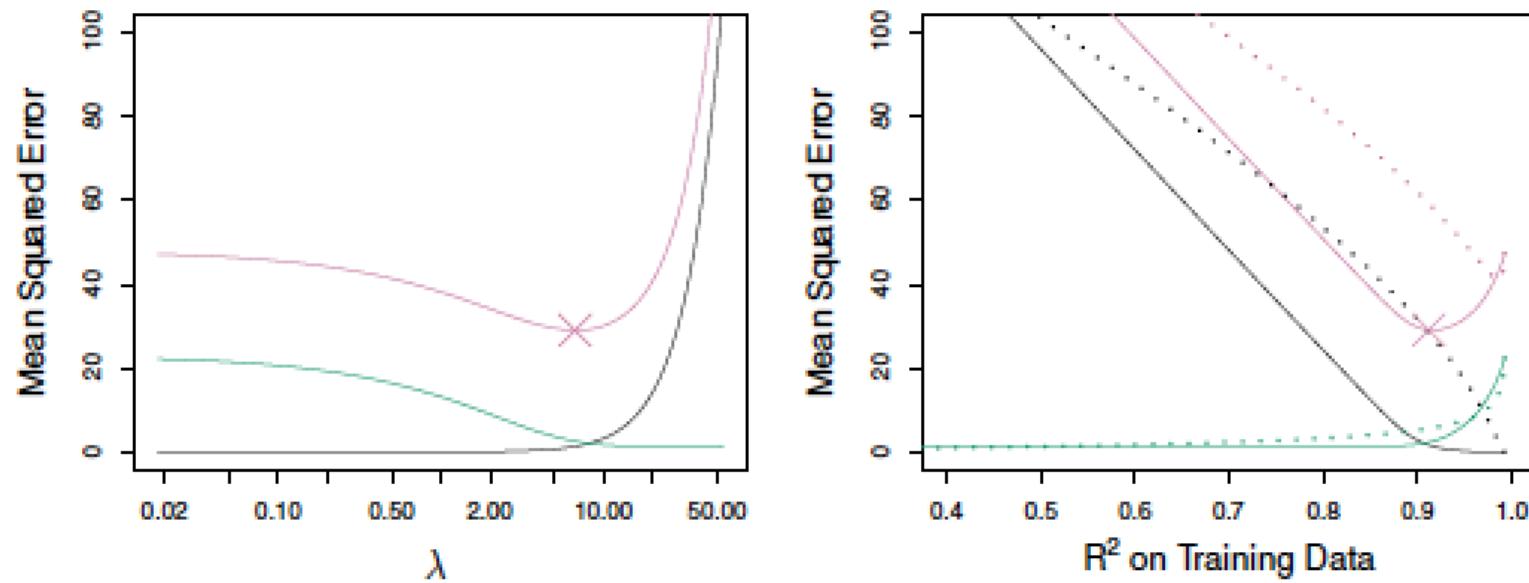
Ridge: dashed

Comparing the Lasso and Ridge Regression



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

Comparing the Lasso and Ridge Regression: continued

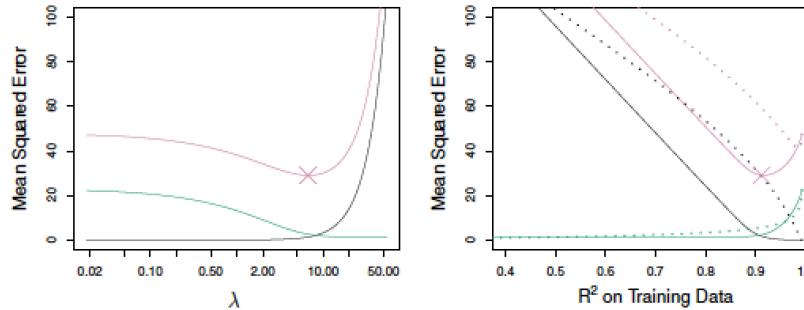


Simulated data with $n = 50$ observations, $p = 45$ predictors, **only two** having nonzero coefficients.

Lasso: solid

Ridge: Dashed

Comparing the Lasso and Ridge Regression: continued



Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to previous slides, except that now only two predictors are related to the response.

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

Conclusions

- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better when the response is a function of only a relatively small number of predictors.

Conclusions

- However, the number of predictors that is related to the response is never known *a priori* for real data sets.
- A technique such as cross-validation can be used in order to determine which approach is better on a particular data set.

Selecting the Tuning Parameter for Ridge Regression and Lasso

- As for subset selection, for ridge regression and lasso we require a method to determine which of the models under consideration is best.

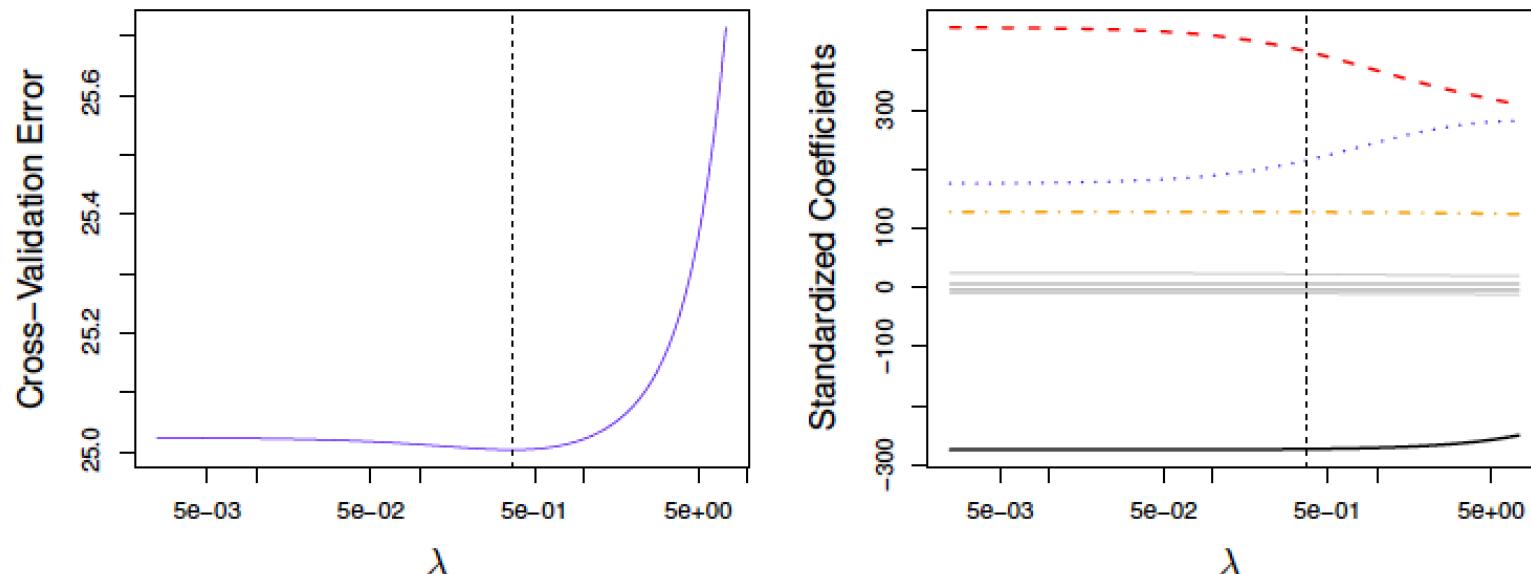
Selecting the Tuning Parameter for Ridge Regression and Lasso

- That is, we require a method selecting a value for the tuning parameter λ or equivalently, the value of the constraint s .

Selecting the Tuning Parameter for Ridge Regression and Lasso

- *Cross-validation* provides a simple way to tackle this problem. We choose a grid of λ values, and compute the cross-validation error rate for each value of λ .
- We then select the tuning parameter value for which the cross-validation error is smallest.
- Finally, the model is **re-fit using all of the available observations** and the selected value of the tuning parameter.

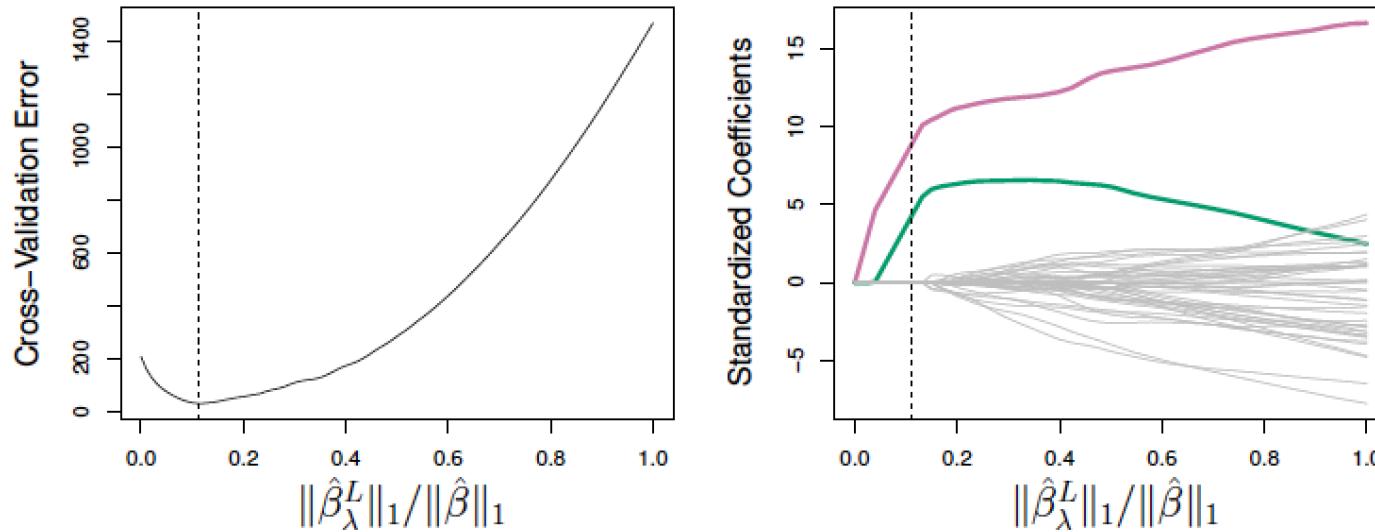
Credit data example



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

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

Simulated data example



Left: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set with two predictors.

Right: The corresponding lasso coefficient estimates are displayed. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

Elastic Net

- Although the lasso does a good job in eliminating *irrelevant* variables, it does not handle **redundant** variables (highly correlated variables) very well
 - The coefficient paths tend to be erratic and can sometimes show wild behavior (e.g. starting from zero and increasing, when λ increases)

Elastic Net

- Consider a simple but extreme example, where the coefficient for a variable X_j with a particular value for λ is $\hat{\beta}_j > 0$. If we augment our data with an identical copy $X_{j'}$ = X_j , then they can share this coefficient in infinitely many ways—any $\tilde{\beta}_j + \tilde{\beta}_{j'} = \hat{\beta}_j$ with both pieces positive—and the loss and L1 penalty are indifferent.
- So the coefficients for this pair are not defined.

Elastic Net

- A quadratic penalty, on the other hand, will divide $\hat{\beta}_j$ exactly equally between these two twins.
- In practice, we are unlikely to have an identical pair of variables, but often we do have groups of very correlated variables.

Elastic Net

- In microarray studies, groups of genes in the same biological pathway tend to be expressed (or not) together, and hence measures of their expression tend to be strongly correlated.

Elastic Net

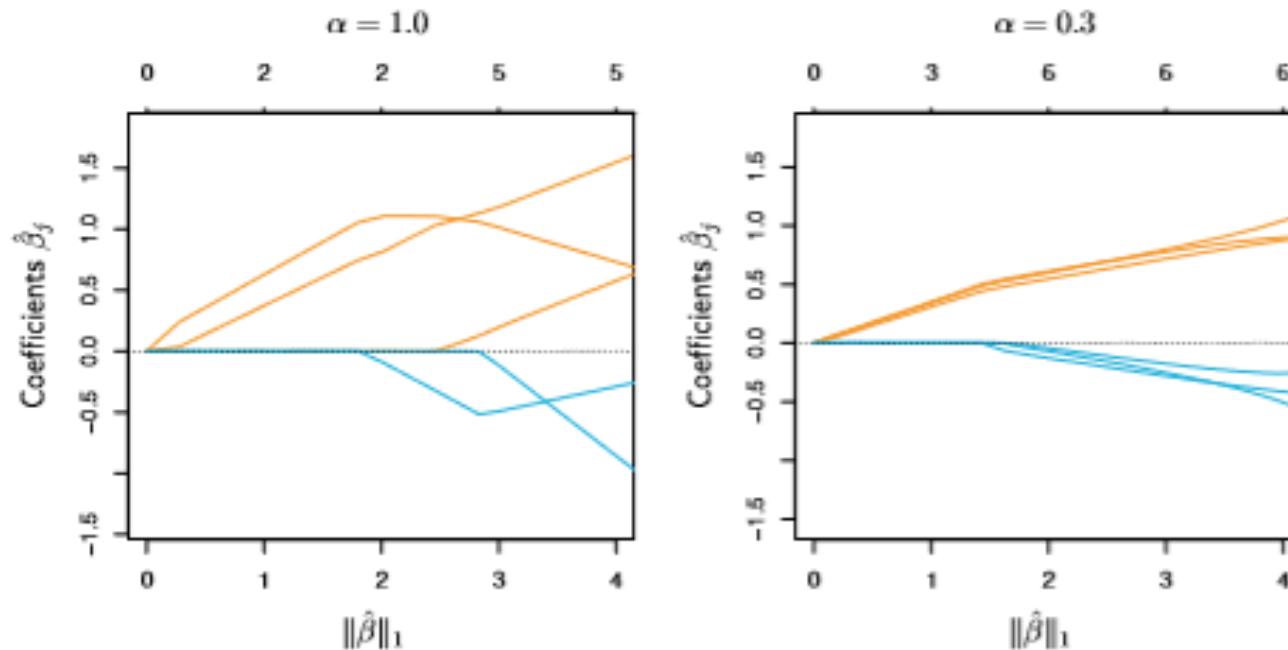
- The elastic net is a regularization method that combines L1 and L2 regularizations, and enforces the coefficients of correlated variables to vary together as λ changes. The Elastic Net penalty is:

$$\lambda \left[\frac{1}{2}(1 - \alpha) \|\beta\|_2^2 + \alpha \|\beta\|_1 \right]$$

where $\alpha \in [0, 1]$ is a parameter that can be varied. When $\alpha = 1$, it reduces to the L1-norm or lasso penalty, and with $\alpha = 0$, it reduces to the squared L2-norm, corresponding to the ridge penalty.

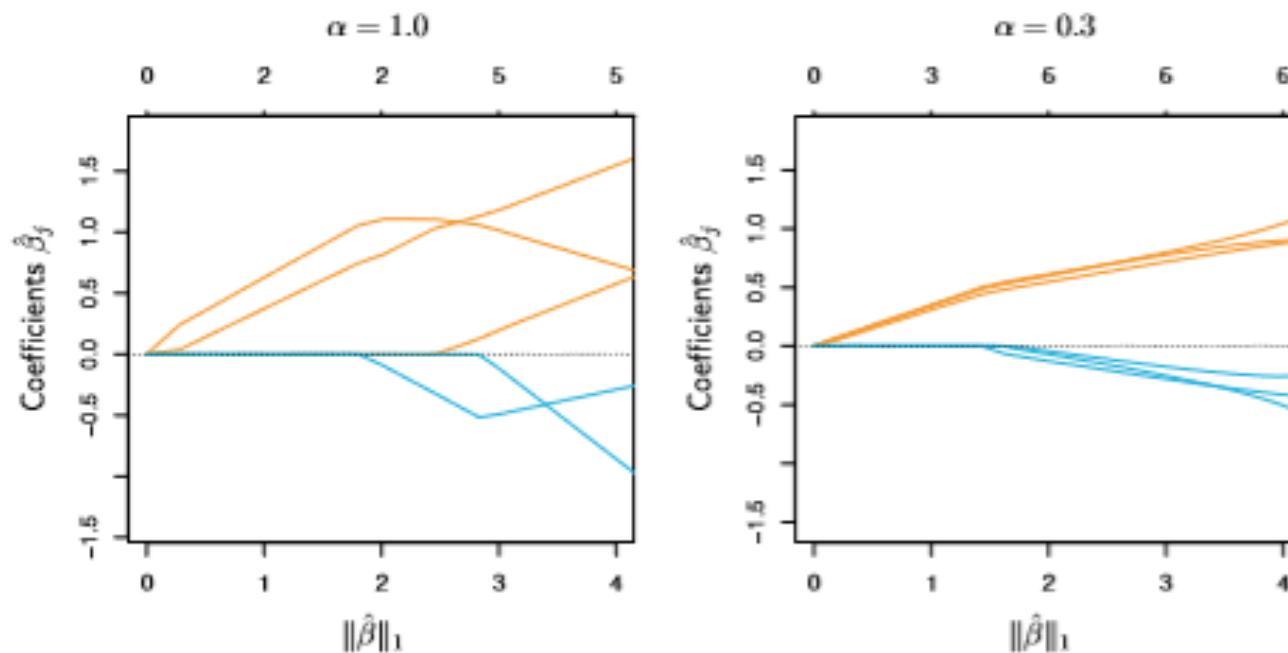
Elastic Net

- Example: Six variables, highly correlated in groups of three. The lasso estimates ($\alpha = 1$), as shown in the left panel, exhibit somewhat erratic behavior as the regularization parameter λ is varied. In the right panel, the elastic net with ($\alpha = 0.3$) includes all the variables, and the correlated groups are pulled together.



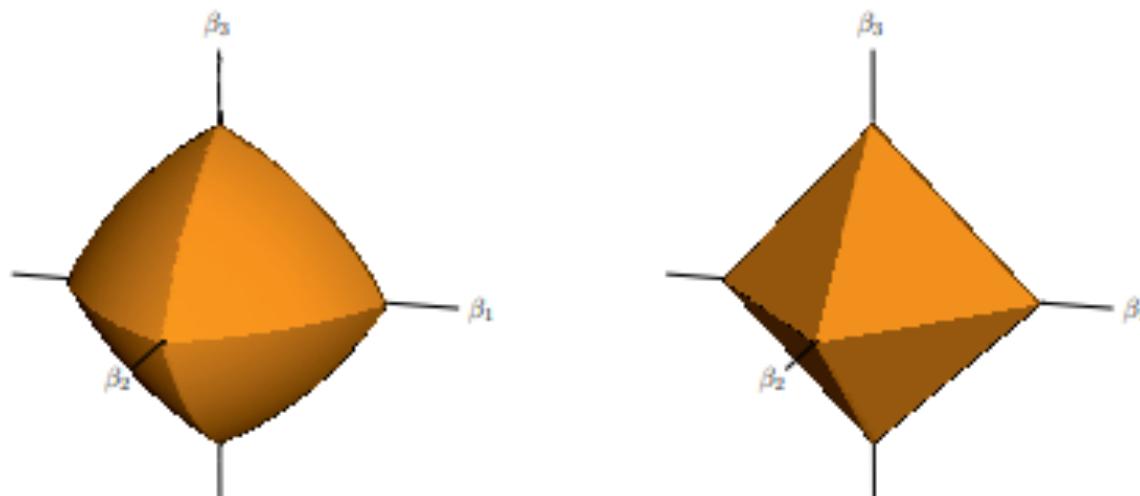
Elastic Net

- Of course, this example is idealized, and in practice the group structure will not be so cleanly evident. But by adding some component of the ridge penalty to the L1-penalty, the elastic net automatically controls for strong within-group correlations.



Elastic Net vs. Lasso Constraints

- Figure 4.2 compares the constraint region for the elastic net (left image) to that of the lasso (right image) when there are three variables. We see that the elastic-net ball shares attributes of the L2 ball and the L1 ball: the sharp corners and edges encourage selection, and the curved contours encourage sharing of coefficients.



Dimension Reduction Methods

- The methods that we have discussed so far in this chapter have involved fitting linear regression models, via least squares or a shrunken approach, using the original predictors, X_1, X_2, \dots, X_p .

Dimension Reduction Methods

- We now explore a class of approaches that *transform* the predictors and then fit a least squares model using the transformed variables.
- We will refer to these techniques as *dimension reduction* methods.

Dimension Reduction Methods

- Note that they are only feature transformation methods, **not feature selection methods.**

Dimension Reduction Methods: details

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

That is,

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

for some constants $\varphi_{m1}, \varphi_{m2}, \dots, \varphi_{mp}$.

- We can then fit the linear regression model,

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

using ordinary least squares.

Dimension Reduction Methods: details

- Note that in the model

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

the regression coefficients are given by $\theta_1, \theta_2, \dots, \theta_M$.

- If the constants $\varphi_{m1}, \varphi_{m2}, \dots, \varphi_{mp}$ are chosen wisely, then such dimension reduction approaches can often outperform OLS regression/reduce variance.

Notice that from definition (1),

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

where

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

Hence model (2) can be thought of as a special case of the original linear regression model.

- Dimension reduction serves to constrain the estimated β_j coefficients, since now they must take the form (3).
- Can win in the bias-variance tradeoff.

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

Principal Components Regression

- Here we apply principal components analysis (PCA) (discussed in Chapter 10 of the text) to define the linear combinations of the predictors, for use in our regression.

Principal Components Regression

- The first principal component is that (normalized) linear combination of the variables with the largest variance.

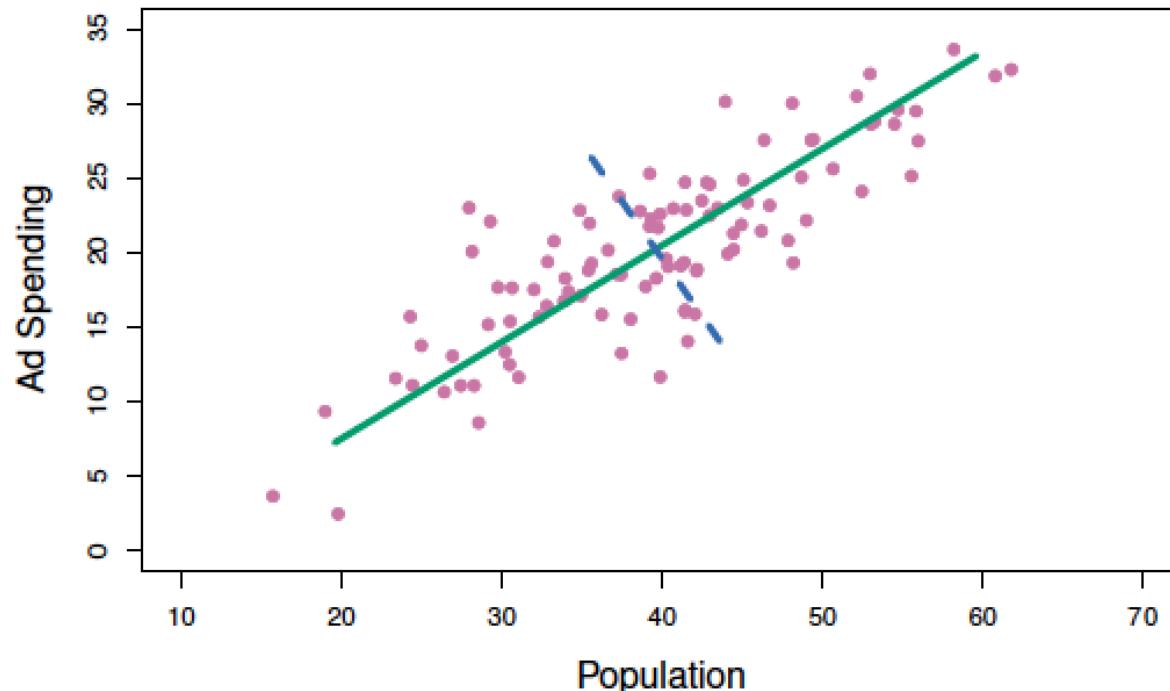
Principal Components Regression

- The second principal component has largest variance, subject to being uncorrelated with the first.
- And so on.

Principal Components Regression

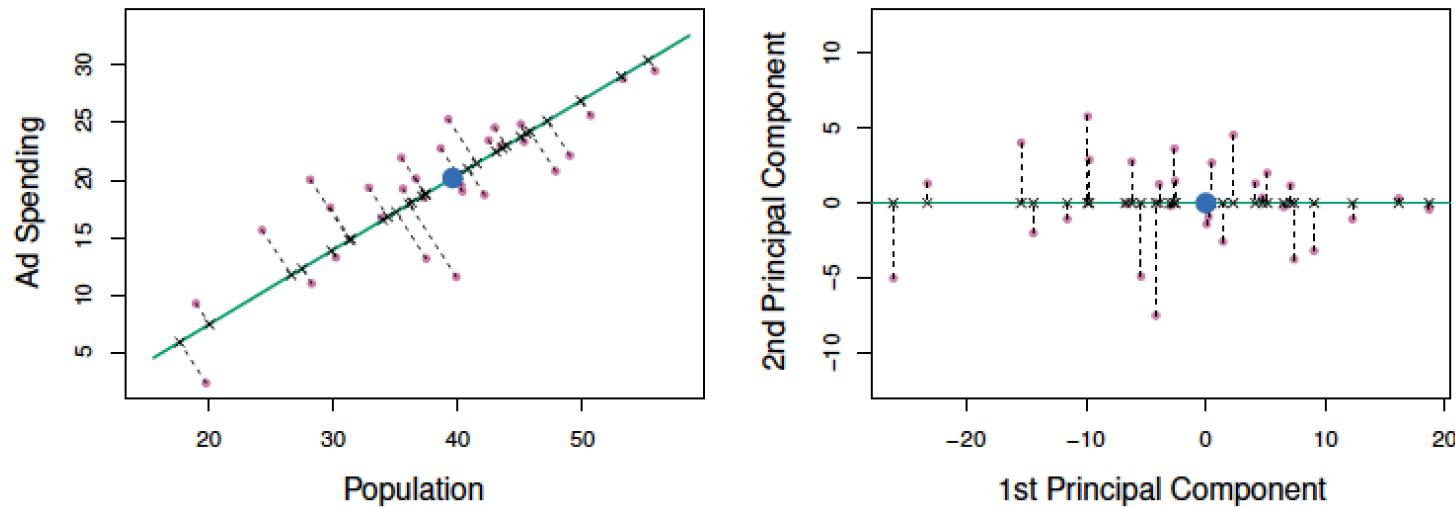
- Hence with many correlated original variables, we replace them with a small set of principal components that capture their joint variation.

Pictures of PCA



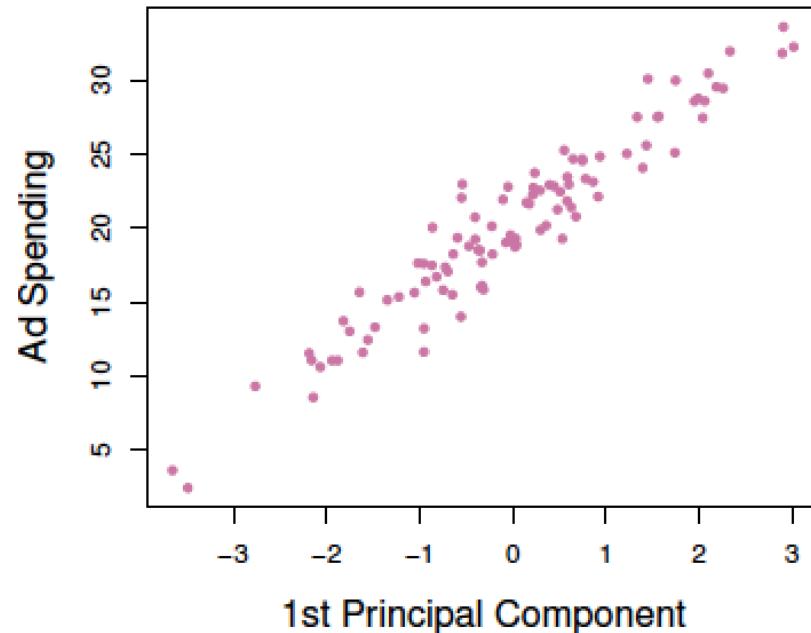
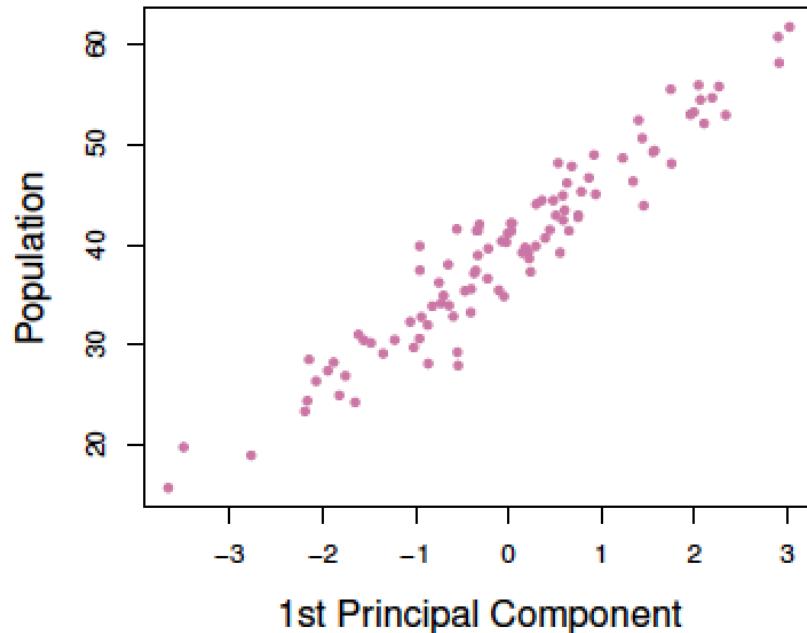
The population size (**pop**) and ad spending (**ad**) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

Pictures of PCA: continued



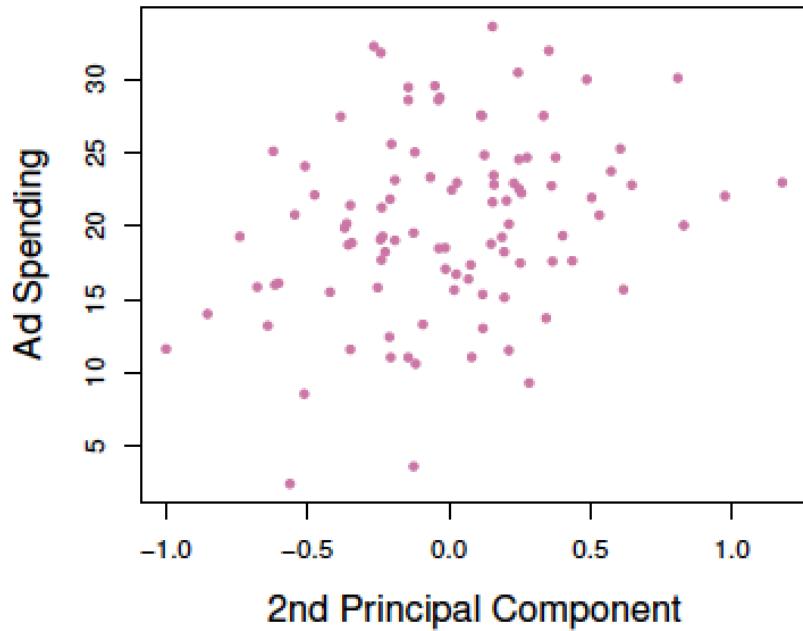
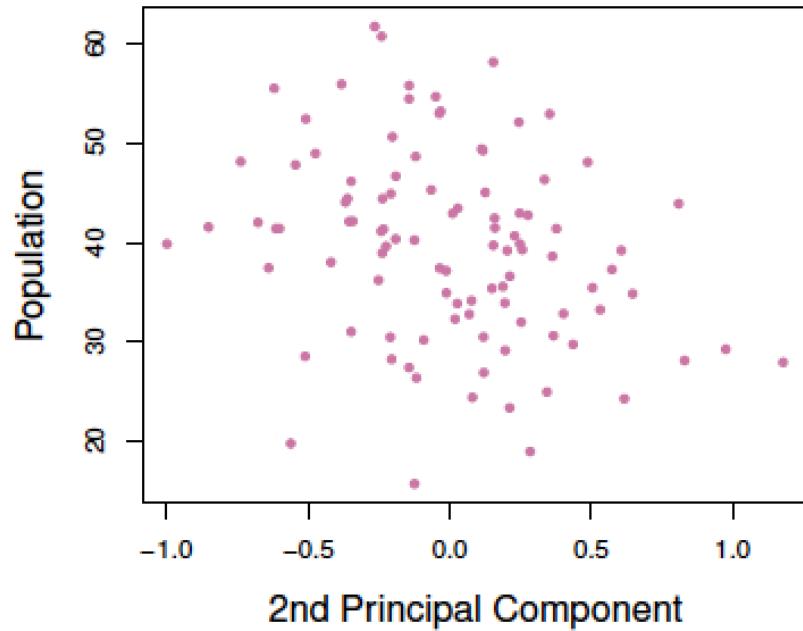
A subset of the advertising data. **Left:** The first principal component, chosen to minimize the sum of the squared perpendicular distances to each point, is shown in green. These distances are represented using the black dashed line segments. **Right:** The left-hand panel has been rotated so that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

Pictures of PCA: continued



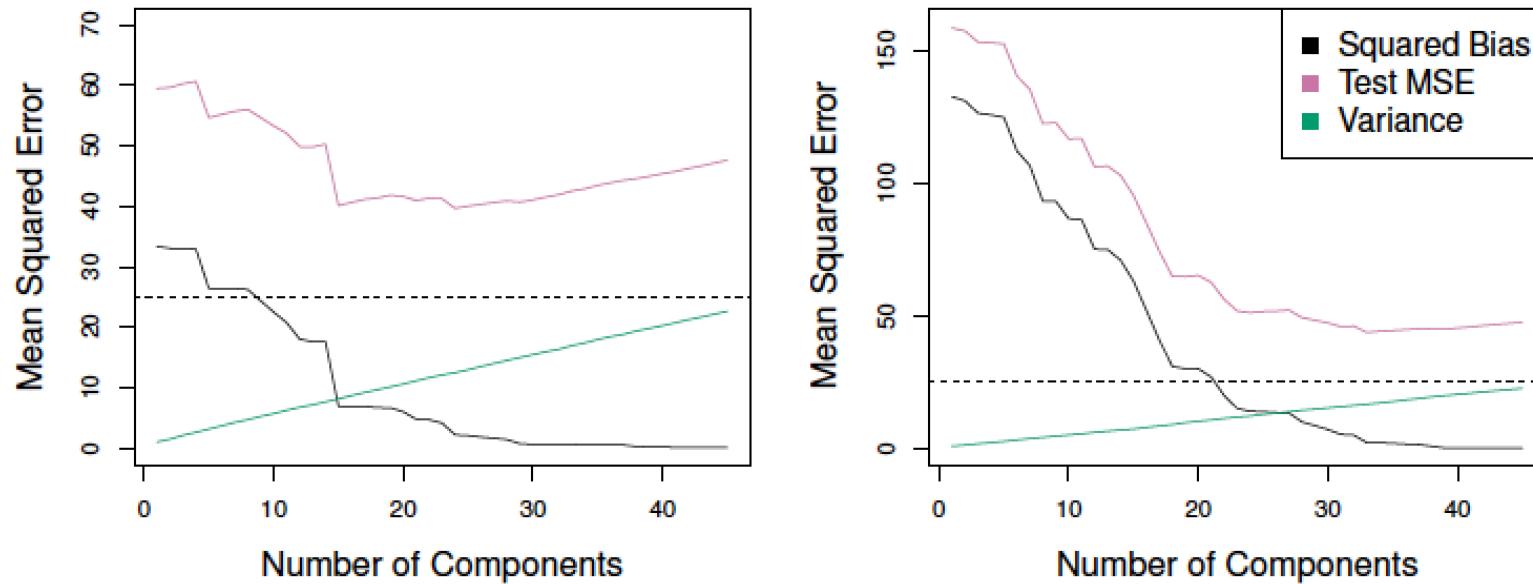
Plots of the first principal component scores z_{i1} versus pop and ad. The relationships are strong.

Pictures of PCA: continued



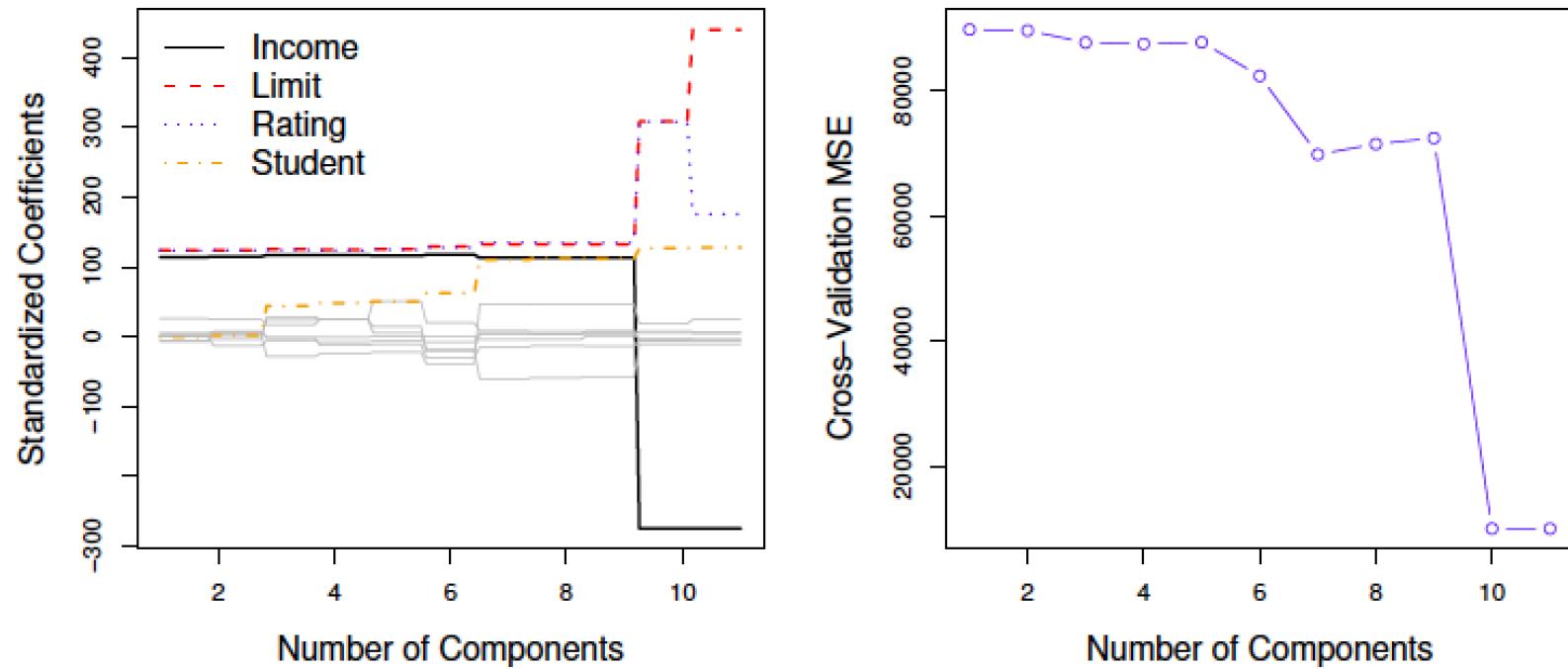
*Plots of the second principal component scores z_{i2} versus **pop** and **ad**. The relationships are weak.*

Application to Principal Components Regression



PCR was applied to two simulated data sets. The black, green, and purple lines correspond to squared bias, variance, and test mean squared error, respectively. Left: Simulated data 2 (Sparse) Right: Simulated data 1 (Non-sparse)

Choosing the number of directions M



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

Partial Least Squares

- PCR identifies linear combinations, or *directions*, that best represent the predictors X_1, \dots, X_p .
- These directions are identified in an *unsupervised* way, since the response Y is not used to help determine the principal component directions.

Partial Least Squares

- That is, the response does not *supervise* the identification of the principal components.

Partial Least Squares

- Consequently, PCR suffers from a potentially serious drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response.

Partial Least Squares: continued

- Like PCR, PLS is a dimension reduction method, which first identifies a new set of features Z_1, \dots, Z_M that are linear combinations of the original features, and then fits a linear model via OLS using these M new features.

Partial Least Squares: continued

- But unlike PCR, PLS identifies these new features in a supervised way – that is, it makes use of the response Y in order to identify new features that not only approximate the old features well, but also that *are related to the response*.

Partial Least Squares: continued

- Roughly speaking, the PLS approach attempts to find directions that help explain both the response and the predictors.

Details of Partial Least Squares

- After standardizing the p predictors, PLS computes the first direction Z_1 by setting each φ_{1j} equal to the coefficient from the simple linear regression of Y onto X_j .

Details of Partial Least Squares

- One can show that this coefficient is proportional to the correlation between Y and X_j . Hence, in computing $Z_1 = \sum_{j=1}^p \phi_{1j} X_j$, PLS places the highest weight on the variables that are most strongly related to the response.

Details of Partial Least Squares

- Subsequent directions are found by taking residuals and then repeating the above prescription.

Summary

- Model selection methods are an essential tool for data analysis, especially for big datasets involving many predictors.

Summary

- Research into methods that give *sparsity*, such as the *lasso* is an especially hot area.
- Later, we will return to sparsity in SVMs.

DSCI 552, Machine Learning for Data Science

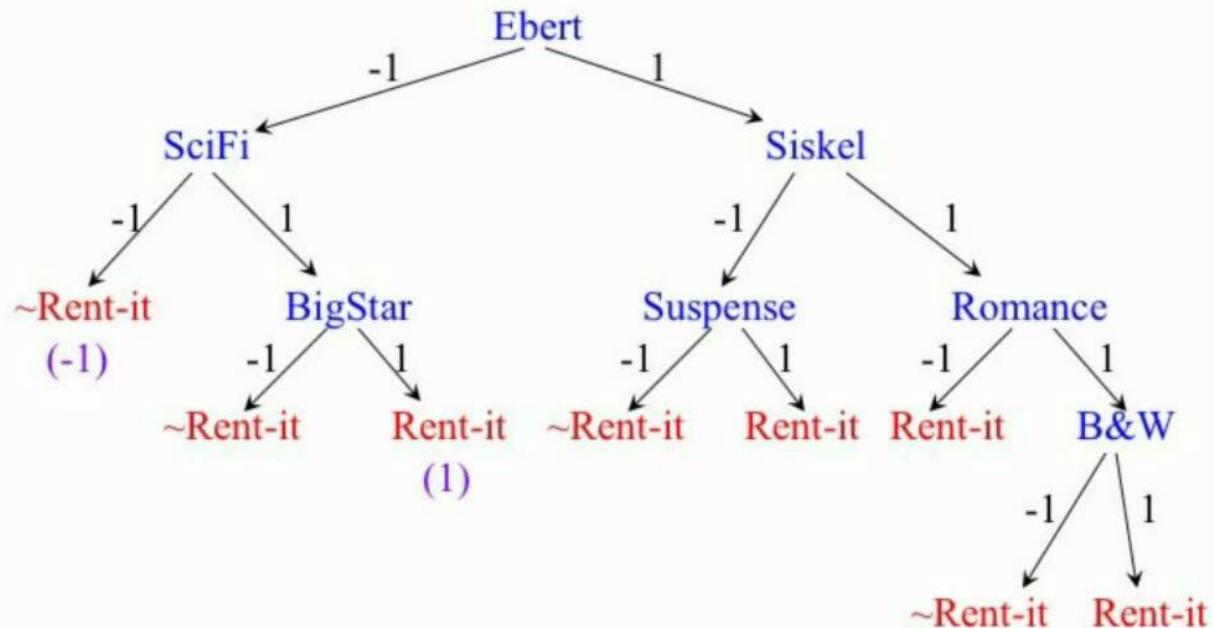
University of Southern California

M. R. Rajati, PhD

Tree-Based Methods

Decision tree classifiers

[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, ..., Rent-it??]



Tree-based Methods

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

Pros and Cons

- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.

Pros and Cons

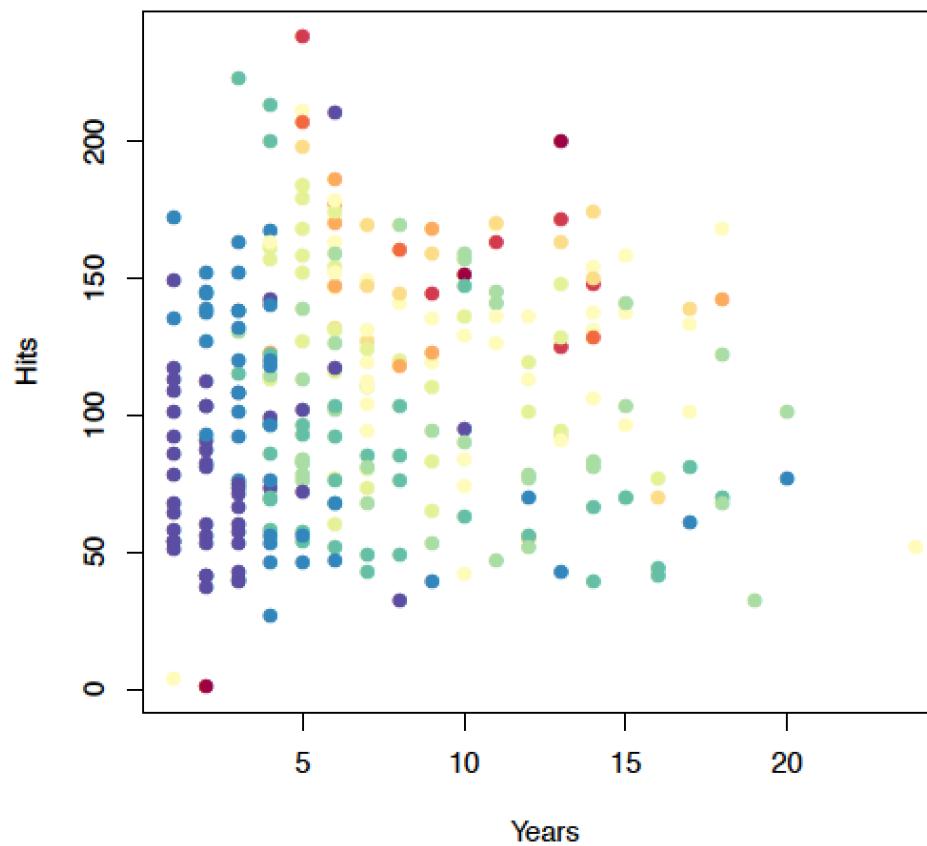
- Hence we also discuss *bagging*, *random forests*, and *boosting*. These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss of interpretation.

The Basics of Decision Trees

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

Baseball salary data: how would you stratify it?

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



Decision tree for these data



Details of previous figure

- For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.

Details of previous figure

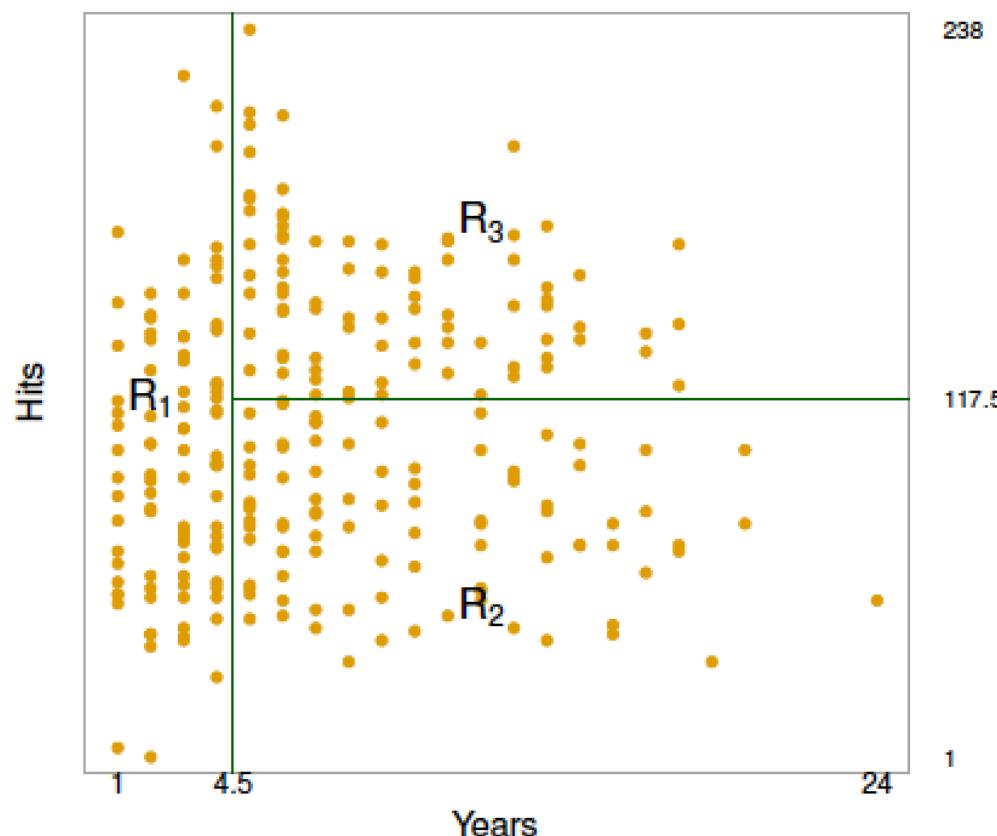
- At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to **Years** < 4.5, and the right-hand branch corresponds to **Years** ≥ 4.5.

Details of previous figure

- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

Results

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



Terminology for Trees

- In keeping with the *tree* analogy, the regions R_1 , R_2 , and R_3 are known as *terminal nodes*
- Decision trees are typically drawn *upside down*, in the sense that the leaves are at the bottom of the tree.

Terminology for Trees

- The points along the tree where the predictor space is split are referred to as *internal nodes*
- In the hitters tree, the two internal nodes are indicated by the text Years<4.5 and Hits<117.5.

Interpretation of Results

- **Years** is the most important factor in determining **Salary**, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of **Hits** that he made in the previous year seems to play little role in his **Salary**.

Interpretation of Results

- For players with an experience of five or more years, the number of **Hits** made in the previous year does affect **Salary**, and players who made more **Hits** last year tend to have higher salaries.

Interpretation of Results

- Surely an **over-simplification**,
but compared to a regression
model, it is easy to display,
interpret and explain

Details of the tree-building process

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

More details of the tree-building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.

More details of the tree-building process

- The goal is to find boxes R_1, \dots, R_J that minimize the RSS, given by

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

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

More details of the tree-building process

- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a *top-down, greedy* approach that is known as recursive binary splitting.

More details of the tree-building process

- The approach is *top-down* because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.

More details of the tree-building process

- It is *greedy* because at each step of the tree-building process, the *best* split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Details— Continued

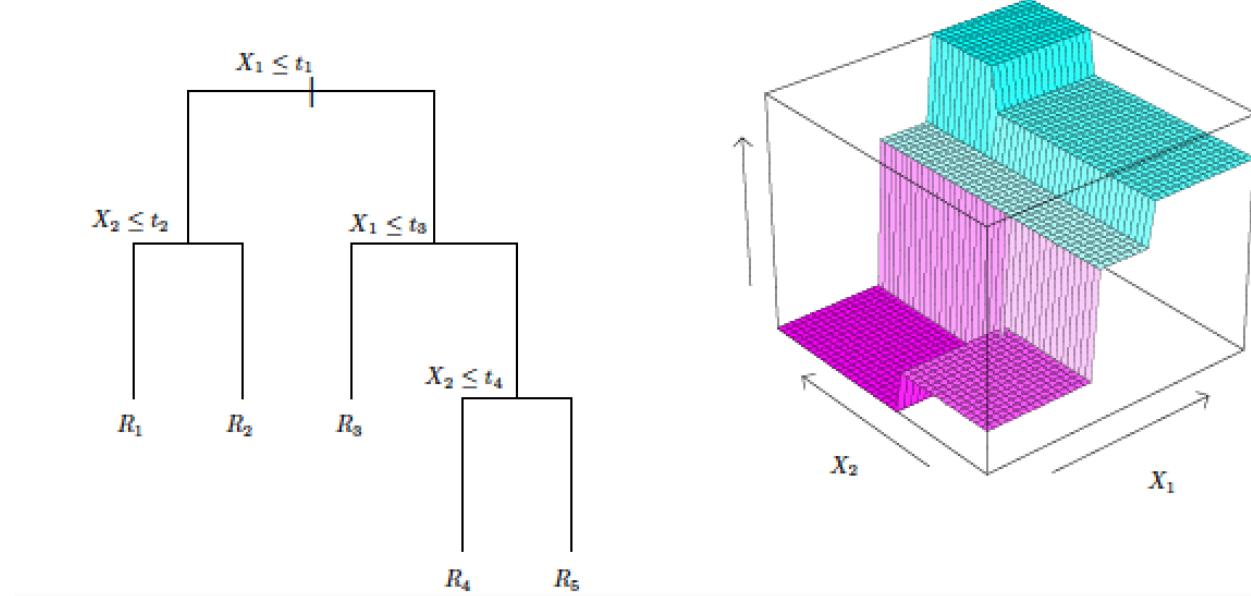
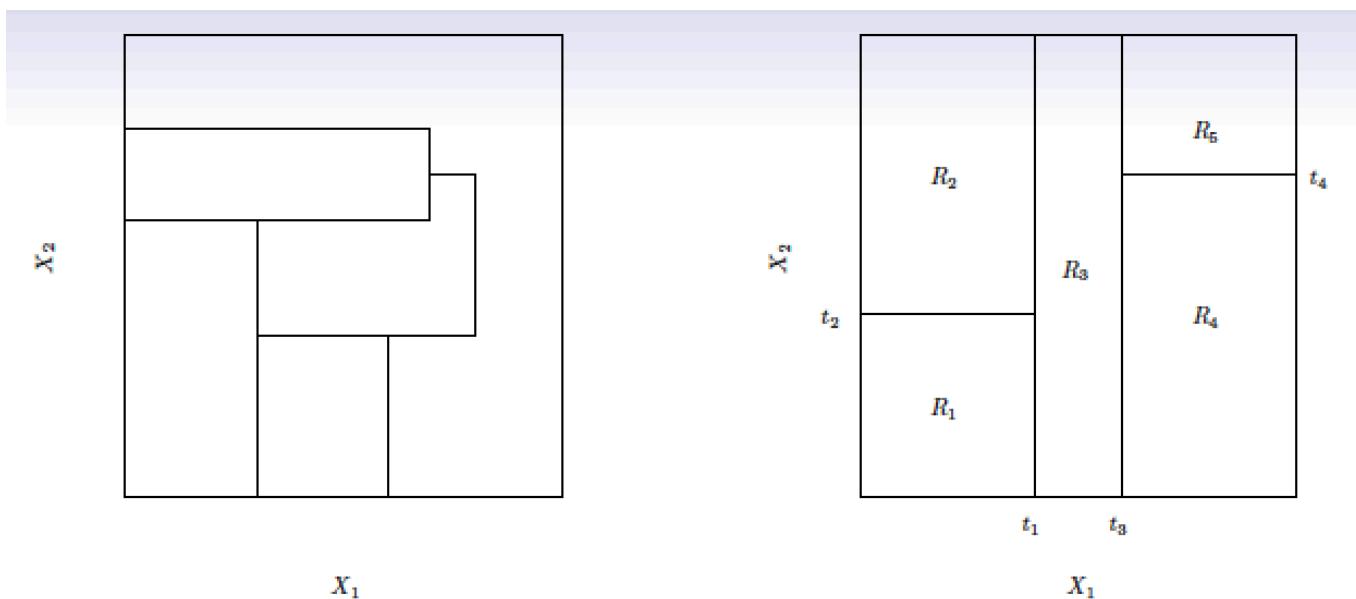
- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.

Details— Continued

- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

Predictions

- We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- A five-region example of this approach is shown in the next slide.



Details of previous figure

Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting.

Top Right: The output of recursive binary splitting on a two-dimensional example.

Bottom Left: A tree corresponding to the partition in the top right panel.

Bottom Right: A perspective plot of the prediction surface corresponding to that tree.

Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance. *Why?*
- A smaller tree with fewer splits (that is, fewer regions R_1, \dots, R_J) might lead to lower variance and better interpretation at the cost of a little bias.

Pruning a tree

- One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.

Pruning a tree

- This strategy will result in smaller trees, but is too *short-sighted*: a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on.

Pruning a tree— continued

- A better strategy is to grow a very large tree T_0 , and then *prune* it back in order to obtain a *subtree*
- *Cost complexity pruning* — also known as *weakest link pruning* — is used to do this

Pruning a tree— continued

- We consider a sequence of trees indexed by a nonnegative tuning parameter α . For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

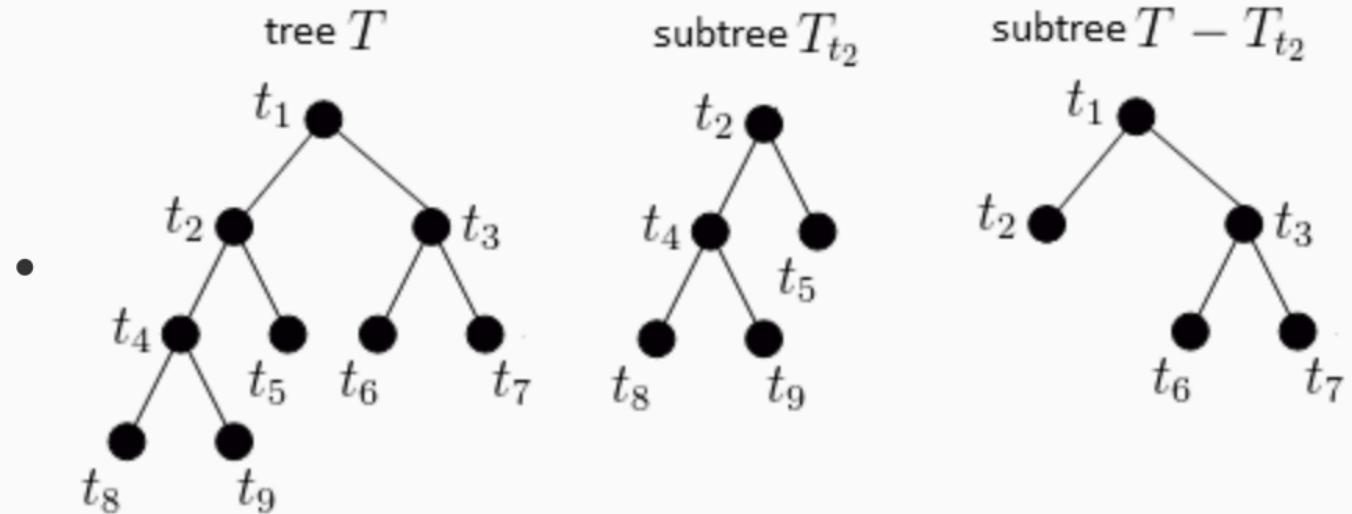
is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T , R_m is the rectangle (i.e. the subset of predictor space) corresponding to the m^{th} terminal node, and \hat{y}_{R_m} is the mean of the training observations in R_m .

Pruning a tree— continued

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Pruning Subtrees

Subtrees:



Choosing the best subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- We select an optimal value $\hat{\alpha}$ using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

Summary: tree algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α .

Summary: tree algorithm

3. Use K-fold cross-validation to choose α . For each

$k = 1, \dots, K$:

3.1 Repeat Steps 1 and 2 on the $\frac{K-1}{K}$ th fraction of the training data, excluding the k th fold.

3.2 Evaluate the mean squared prediction error on the data in the left-out k th fold, as a function of α .

Average the results, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α .

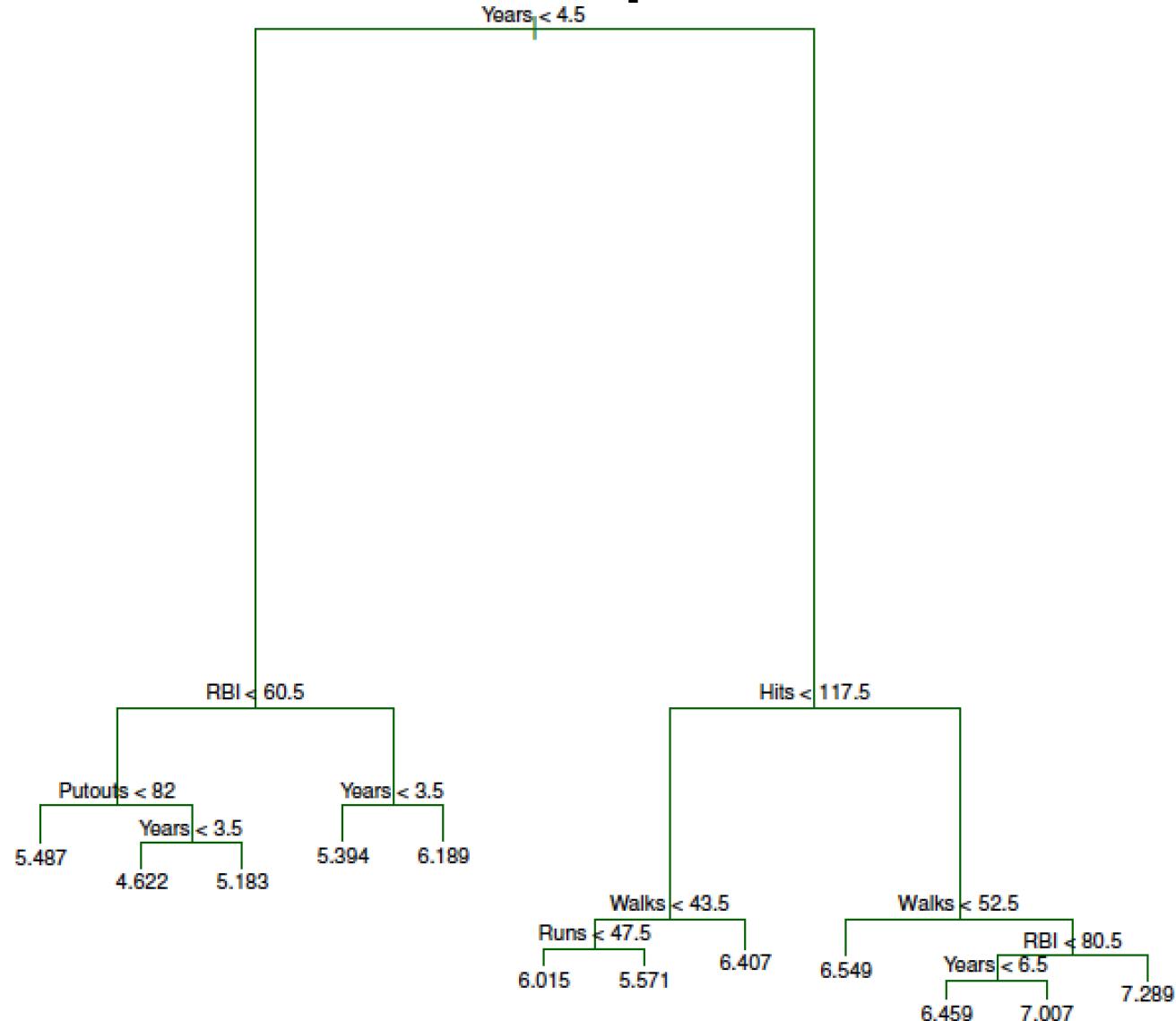
Baseball example continued

- First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set.

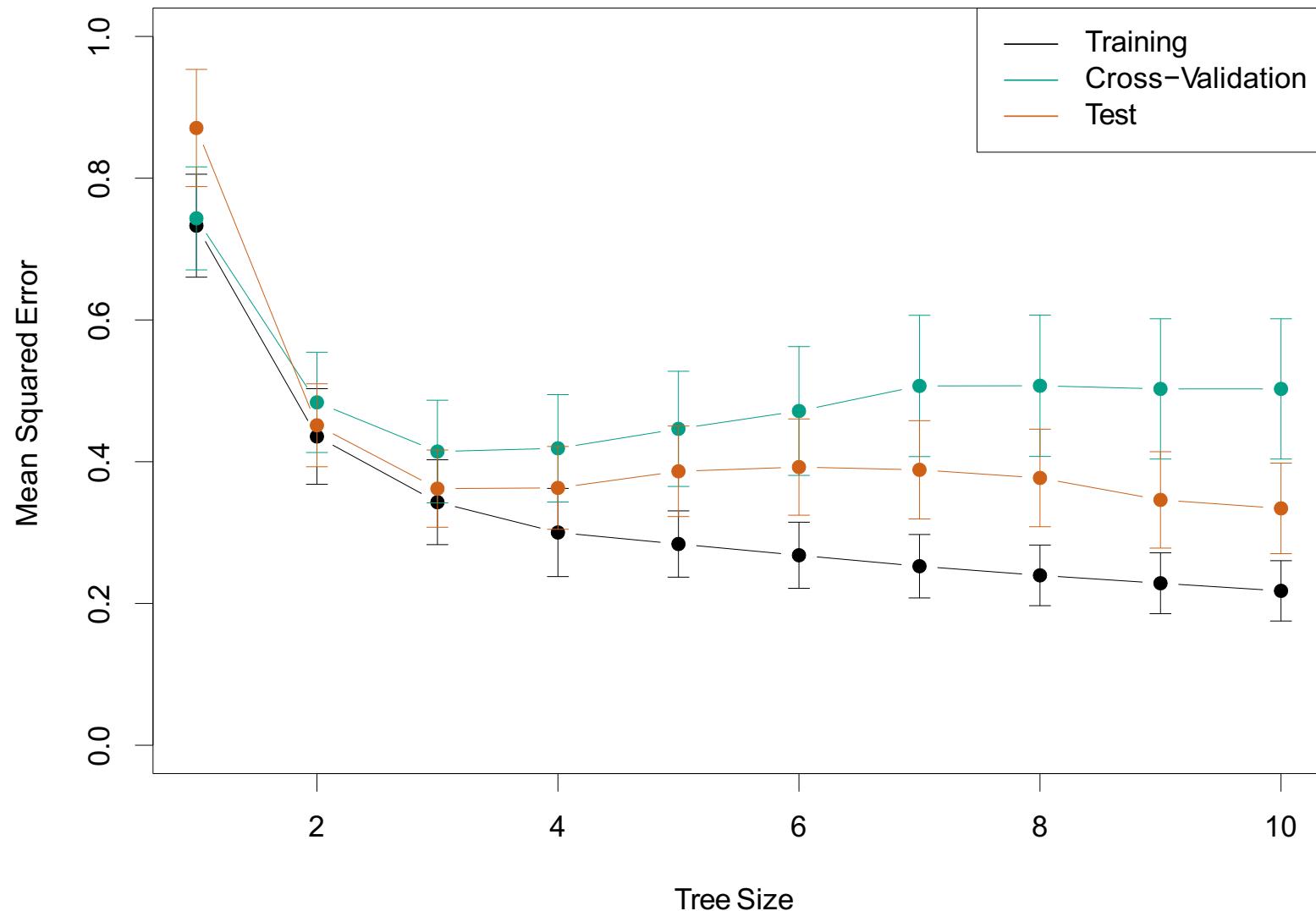
Baseball example continued

- We then built a large regression tree on the training data and varied α in order to create subtrees with different numbers of terminal nodes.
- Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of α .

Baseball example continued



Baseball example continued



Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs.

Details of classification trees

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
- In the classification setting, RSS cannot be used as a criterion for making the binary splits

Details of classification trees

- A natural alternative to RSS is the *classification error rate*, which is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max(\hat{p}_{mk}).$$

- Here \hat{p}_{mk} represents the proportion of training observations in the m^{th} region that are from the k^{th} class.

Example

A simple example:

Details of classification trees

- However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

Gini index and Deviance

- The *Gini index* is defined by

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

a measure of total variance across the K classes. The Gini index takes on a *small value* if all of the \hat{p}_{mk} 's are close to zero or one.

Gini index and Deviance

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

- For this reason the Gini index is referred to as a measure of node *purity* — a small value indicates that a node contains predominantly observations from a single class.
- The Tree Algorithm in this case is called CART (Classification and Regression Trees)

Gini index and Deviance

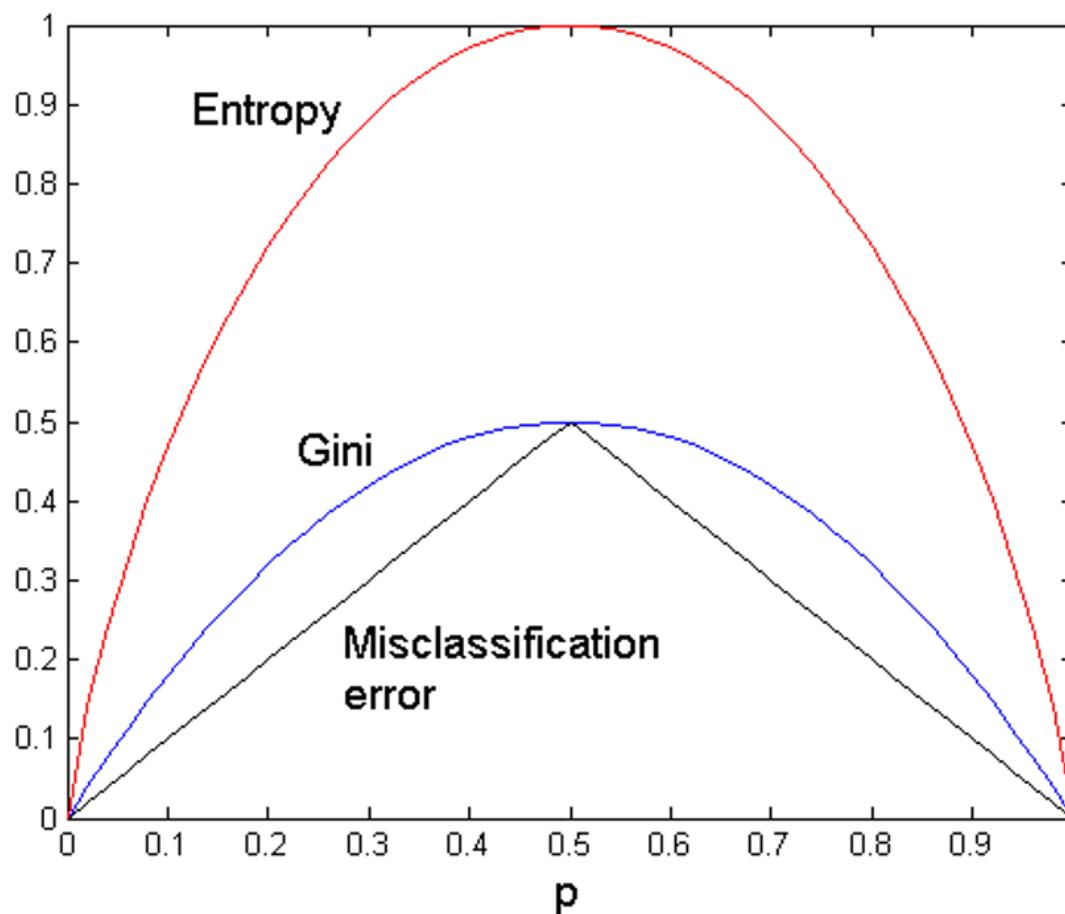
- An alternative to the Gini index is *cross-entropy*, given by

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}.$$

- The tree algorithms that use cross entropy are ID3, C4.5, and C5.0
- It turns out that the Gini index and the cross-entropy are very similar numerically.

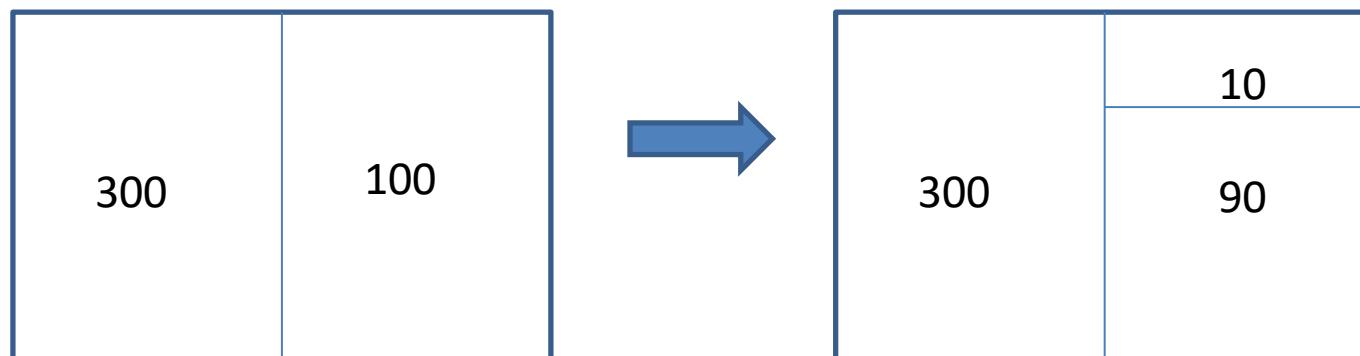
Comparison of Impurity Measures

- For a two-class problem



Weighted Impurity

- Most of the time, to be more fair, we add up the *weighted impurity* of the regions resulting from a split.
- We choose weights to be the fraction of data points in each region.

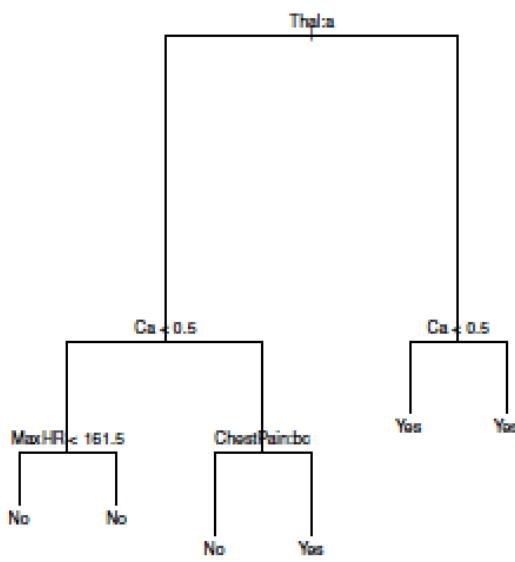
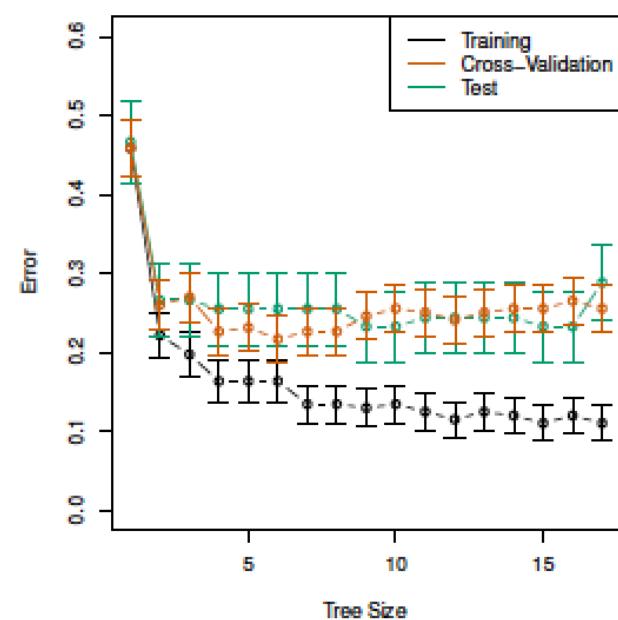
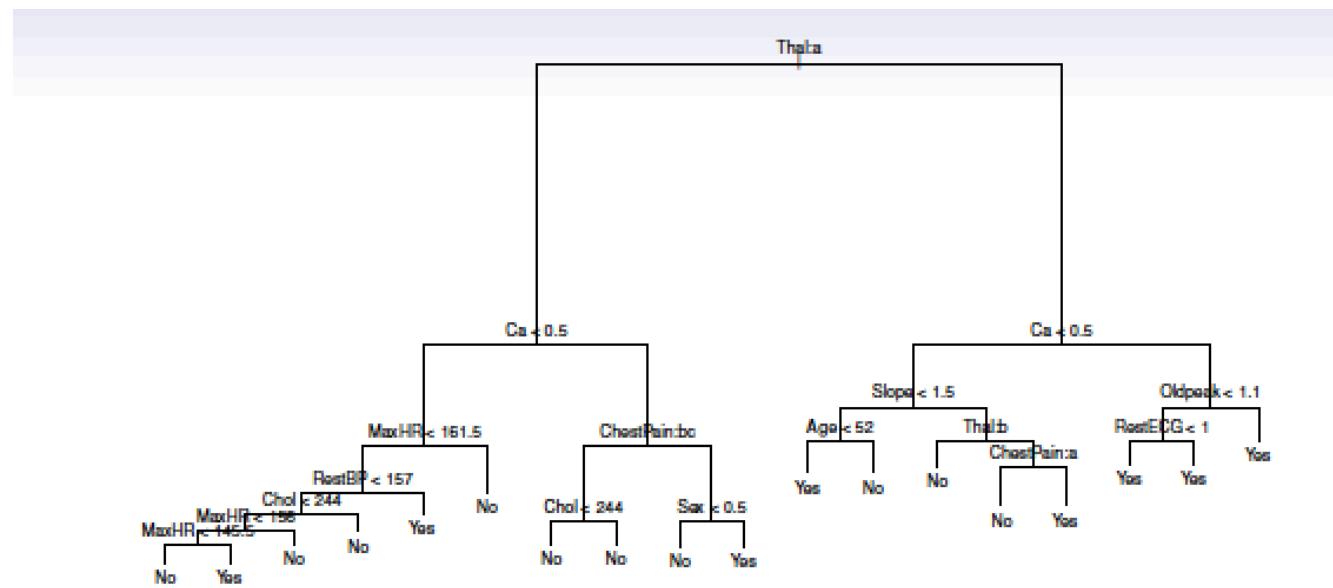


Example: heart data

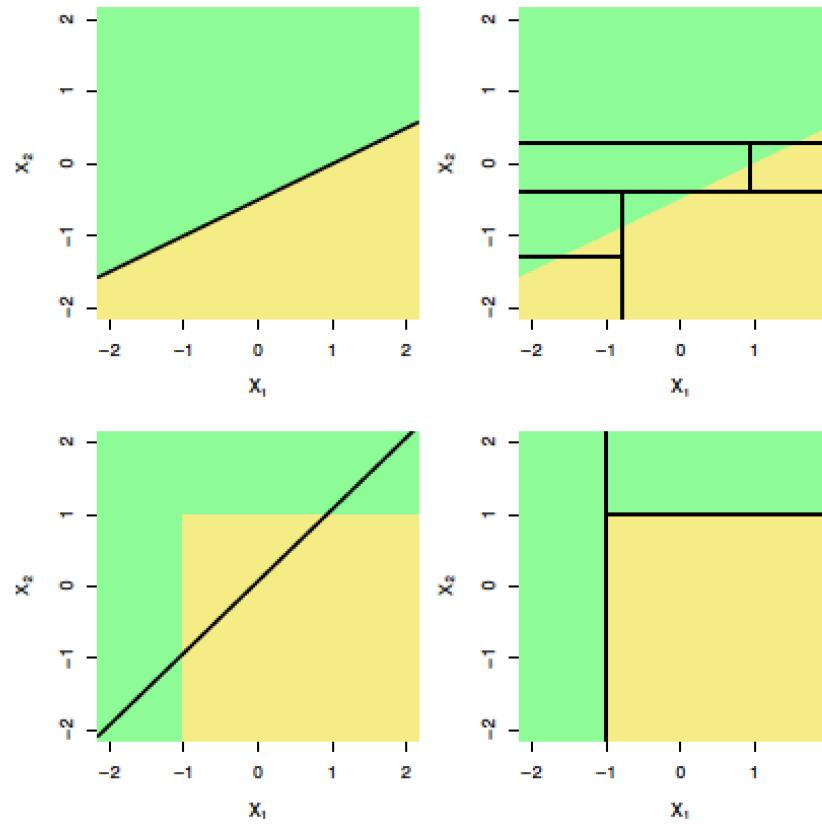
- These data contain a binary outcome **HD** for 303 patients who presented with chest pain.
- An outcome value of **Yes** indicates the presence of heart disease based on an angiographic test, while **No** means no heart disease.

Example: heart data

- There are 13 predictors including **Age**, **Sex**, **Chol** (a cholesterol measurement), and other heart and lung function measurements.
- Cross-validation yields a tree with six terminal nodes. See next figure.



Trees Versus Linear Models



Top Row: True linear boundary; Bottom row: true non-linear boundary.

Left column: linear model; Right column: tree-based model

Advantages and Disadvantages of Trees

- Trees are very easy to explain to people. In fact, they are sometimes even easier to explain than linear regression!

Advantages and Disadvantages of Trees

- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.

Advantages and Disadvantages of Trees

- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Advantages and Disadvantages of Trees

- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

Bagging

- *Bootstrap aggregation*, or *bagging*, is a general-purpose (wrapper) procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- It is an *ensemble method*.

Bagging

- Recall that given a set of n independent observations Z_1, \dots, Z_n , each with variance σ^2 , the variance of the mean \bar{Z} of the observations is given by σ^2/n .
- In other words, *averaging a set of observations reduces variance*. Of course, this is not practical because we generally do not have access to multiple training sets.

Bagging— continued

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate B different bootstrapped training data sets. We then train our method on the b^{th} bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x . We then average all the predictions to obtain

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x).$$

This is called *bagging*.

Bagging classification trees

- The above prescription can be applied to regression trees
- For classification trees: for each test observation, we record the class predicted by each of the B trees, and take a *majority vote*: the overall prediction is the most commonly occurring class among the B predictions.

Out-of-Bag Error Estimation

- It turns out that there is a very straightforward way to estimate the test error of a bagged model.

Out-of-Bag Error Estimation

- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations.
- One can show that on average, each bagged tree makes use of around two-thirds of the observations.

Out-of-Bag Error Estimation

- The remaining one-third of the observations not used to fit a given bagged tree are called the *out-of-bag* (OOB) observations.

Out-of-Bag Error Estimation

- We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the i^{th} observation, which we average.
- This estimate is essentially the LOO cross-validation error for bagging, if B is large.

Random Forests

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.

Random Forests

- But when building these decision trees, each time a split in a tree is considered, *a random selection of m predictors* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.

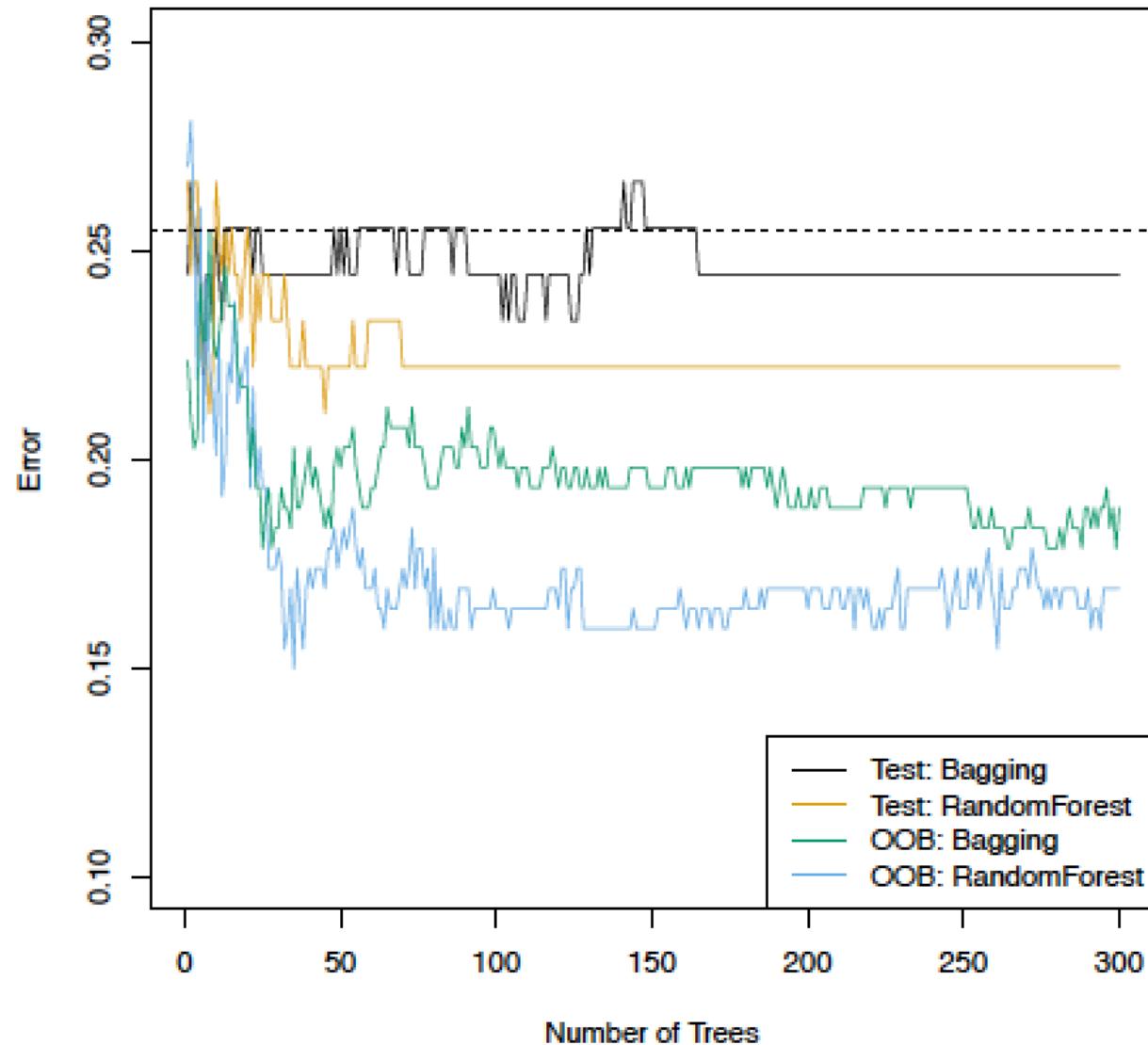
Random Forests

- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx p^{1/2}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Random Forests: An Alternative Version

- Alternatively, one can build B decision trees, each of which on a random subset of m features.
- Instead of a bootstrap sample from the training set, one can even use a subsample with replacement from the training set, which is equivalent to a bootstrap sample with a size different from the training set.

Bagging the heart data



Details of previous figure

Bagging and random forest results for the Heart data.

- The test error (black and orange) is shown as a function of B , the number of bootstrapped training sets used.
- Random forests were applied with $m = p^{1/2}$.
- The dashed line indicates the test error resulting from a single classification tree.
- The green and blue traces show the OOB error, which in this case is considerably lower

Example: gene expression data

- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.

Example: gene expression data

- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.

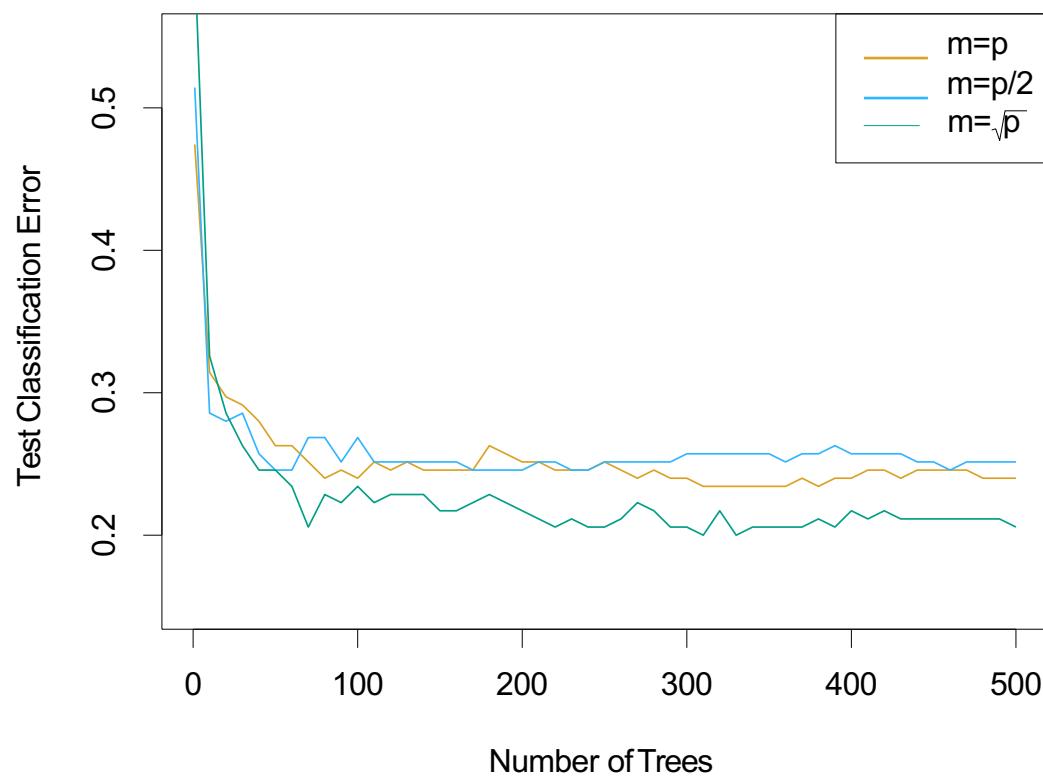
Example: gene expression data

- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.

Example: gene expression data

- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m .

Results: gene expression data



Details of previous figure

- Results from random forests for the fifteen-class gene expression data set with $p = 500$ predictors.
- The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m , the number of predictors available for splitting at each interior tree node.

Details of previous figure

- Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We only discuss boosting for decision trees.

Boosting

- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.

Boosting

- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown *sequentially*: each tree is grown using information from previously grown trees.

Boosting algorithm for regression trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \dots, B$, repeat:
 1. Fit a tree \hat{f}^b with d splits ($d + 1$ terminal nodes) to the training data (X, r) .
 2. Update f by adding in a shrunken version of the new tree:

$$f(x) \leftarrow f(x) + \lambda \hat{f}^b(x).$$

3. Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

Boosting algorithm for regression trees

Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x).$$

What is the idea behind this procedure?

- Unlike fitting a single large decision tree to the data, which amounts to *fitting the data hard* and potentially overfitting, the boosting approach instead *learns slowly*.

What is the idea behind this procedure?

- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.

What is the idea behind this procedure?

- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.

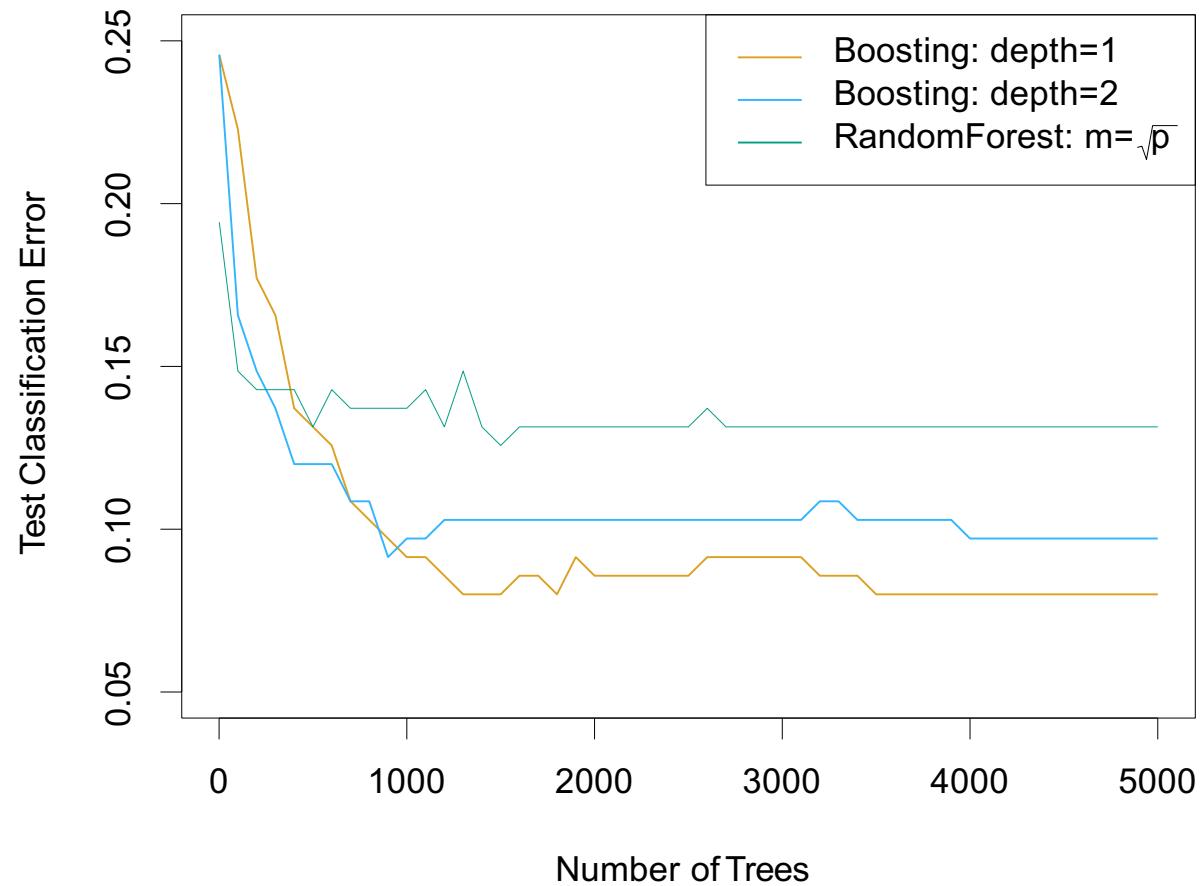
What is the idea behind this procedure?

- By fitting small trees to the residuals, we slowly improve f in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Boosting for classification

- Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex. We will not go into detail here (see appendix), nor does the text book.
- More details in *Elements of Statistical Learning, chapter 10 or Appendix.*
- The R package `gbm` (gradient boosted models) handles a variety of regression and classification problems.

Gene expression data continued



Details of previous figure

- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict *cancer* versus *normal*.

Details of previous figure

- The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.
- The test error rate for a single tree is 24%.

Tuning parameters for boosting

1. The *number of trees* B . Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B .

Tuning parameters for boosting

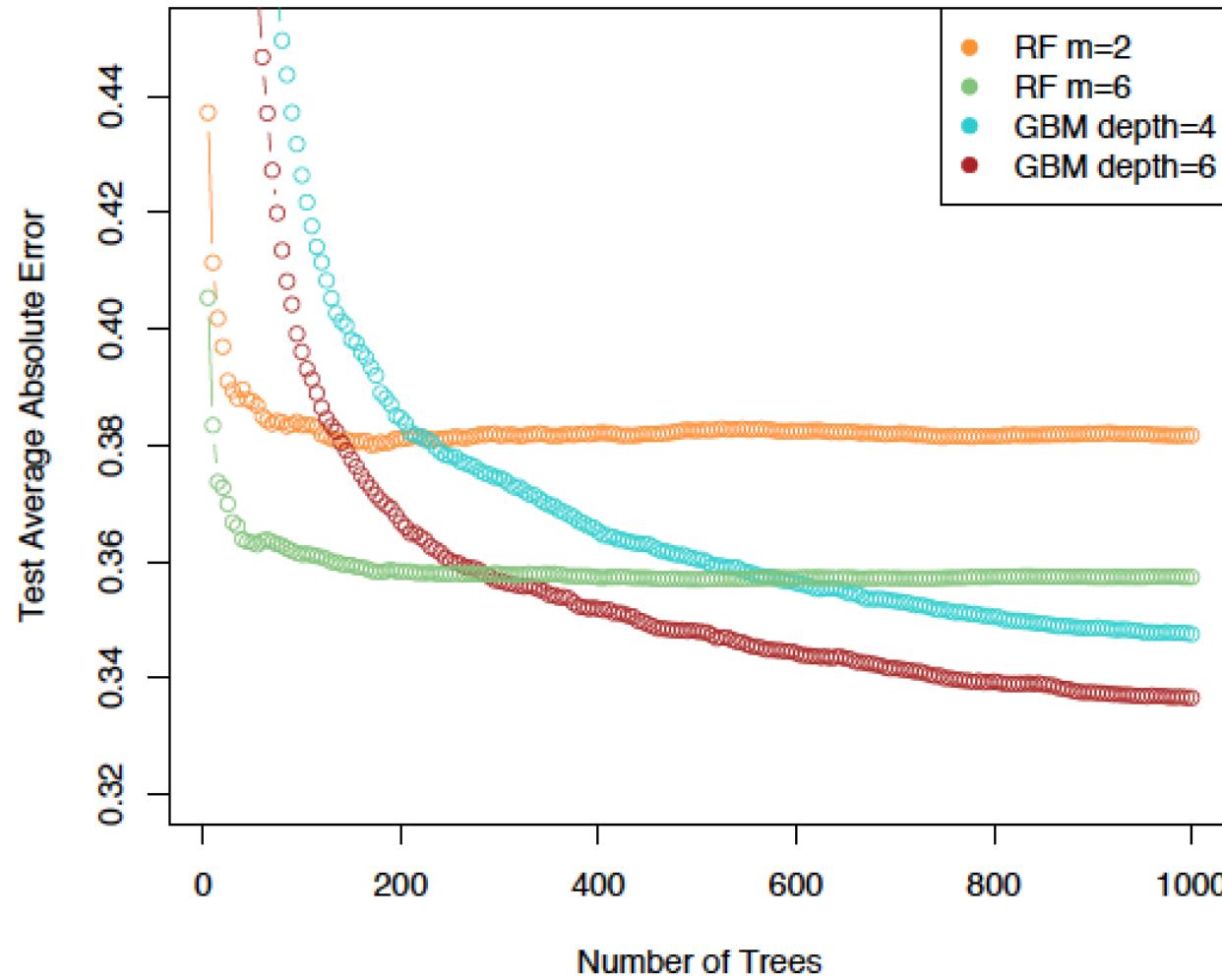
2. The *shrinkage parameter* λ , a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.

Tuning parameters for boosting

3. The *number of splits* d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a *stump*, consisting of a single split and resulting in an additive model. More generally d is the *interaction depth*, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

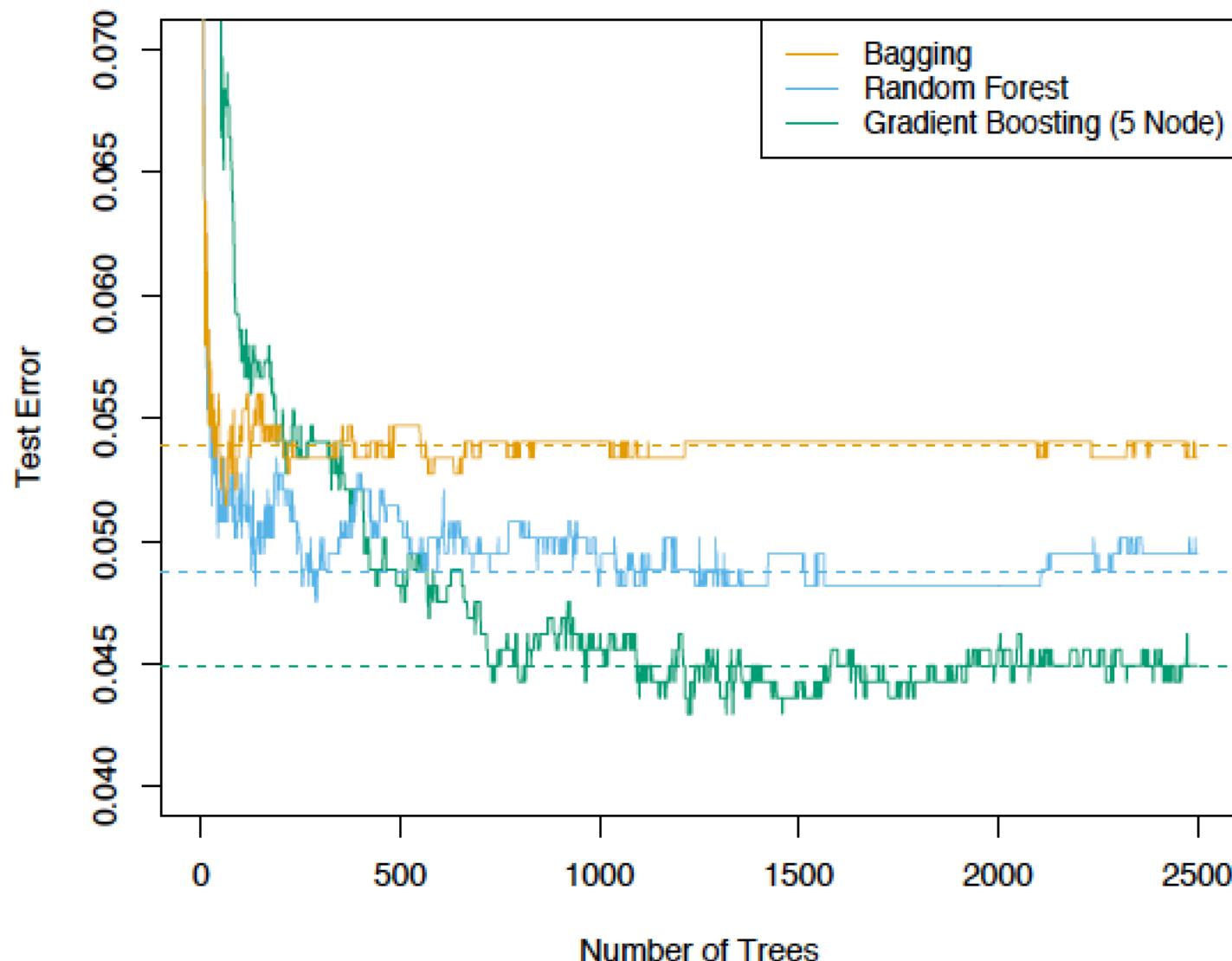
Another regression example

California Housing Data



Another classification example

Spam Data



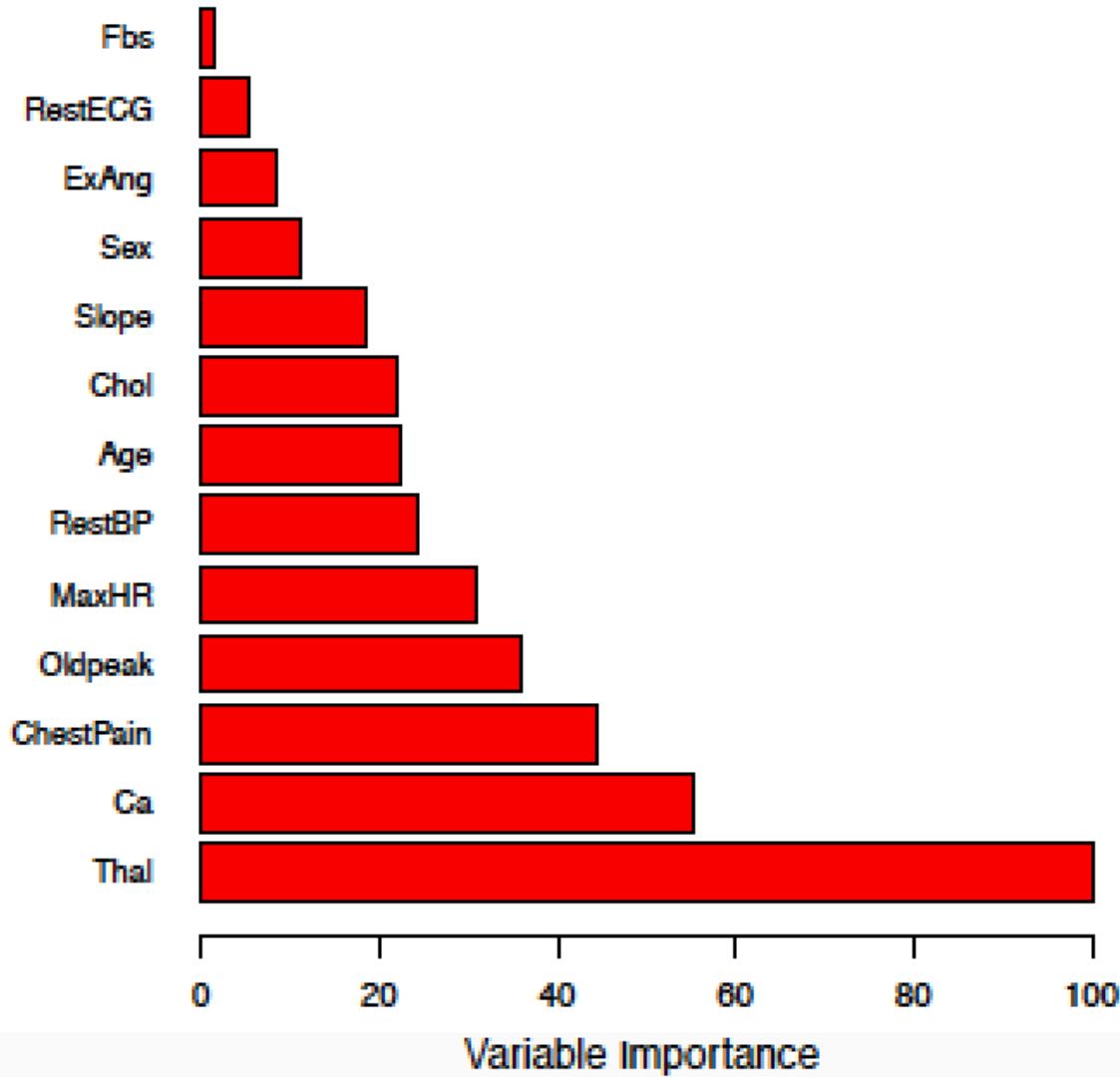
Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.

Variable importance measure

- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

Variable importance measure



Variable
importance
plot for the
Heart data

Summary

- Decision trees are simple and interpretable models for regression and classification
- However they are often not competitive with other methods in terms of prediction accuracy

Summary

- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods— random forests and boosting— are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

Appendix

Extra Trees

Boosting (Appendix)

- Extremely Randomized Trees were proposed by Geurts et al. in 2006.
- The idea is to make the trees even weaker, but to compensate that with the large number of estimators in the ensemble.

Boosting (Appendix)

More specifically, not only the features and samples shown to the tree are randomized, but also the growing of individual trees is random.

Boosting (Appendix)

- In particular the split point i.e., the thresholds of comparisons is randomized. This makes training each tree faster.
- Implemented as `sklearn.ensemble.ExtraTreesClassifier`.

Boosting (Appendix)

Boosting for Classification

Boosting iteratively learns weak classifiers; a weak classifier is one whose error rate is only slightly better than random guessing.

Boosting (Appendix)

Boosting for Classification

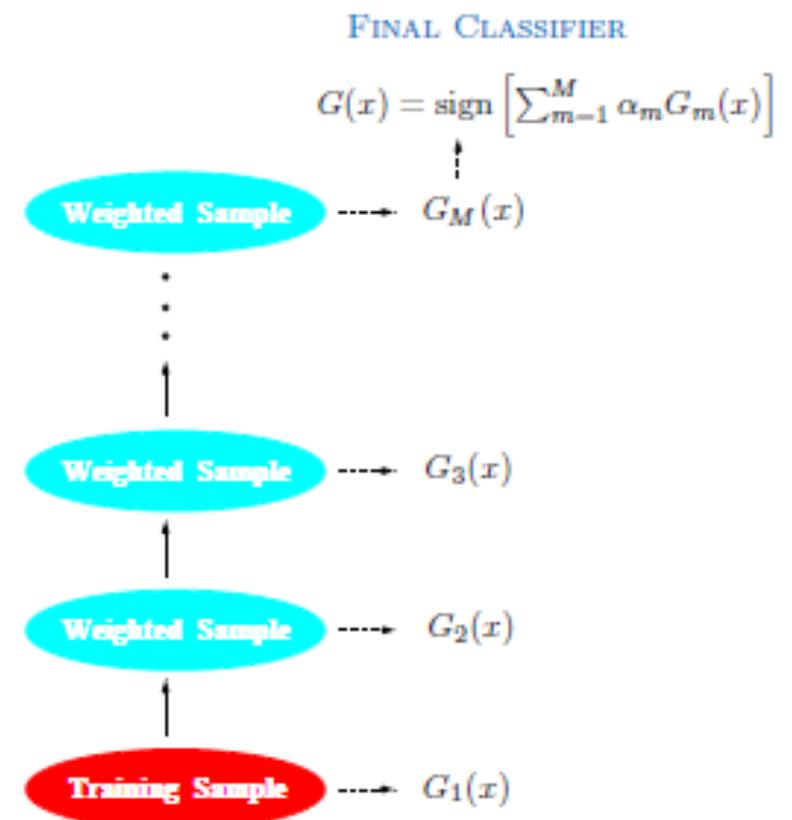
The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers.

The predictions from all of them are then combined through a weighted majority vote to produce the final prediction.

Boosting (cont.)

Boosting for Classification (cont'd)

Thus, the final result is the weighted sum of the results of weak classifiers.

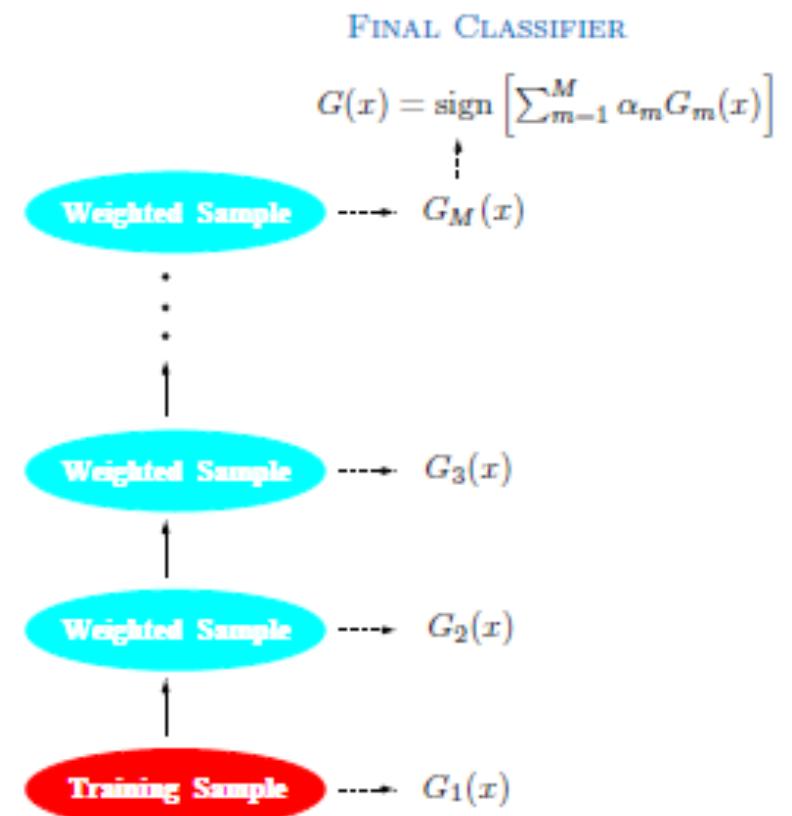


Boosting (cont.)

Boosting for Classification (cont'd)

The alphas in the final classifier are computed by the boosting algorithm, which weight the contribution to give higher influence to more accurate classifiers in the sequence.

Weights are modified at each boosting step!



Boosting (cont.)

Up-weight data that are difficult; incorrectly classified in the previous round. **Down-weight** data that are easy; correctly classified in the previous round.

Boosting (cont.)

There are many different boosting algorithms for classification:

AdaBoost, LPBoost, BrownBoost,
LogitBoost, Gradient Boosting, etc.

Boosting (cont.)

Example of Boosting Algorithm (AdaBoost):

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
 2. For $m = 1$ to M :
 - (a) Fit a classifier $G_m(x)$ to the training data using weights w_i .
 - (b) Compute
$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
 - (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
 - (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \dots, N$.
 3. Output $G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$.
-

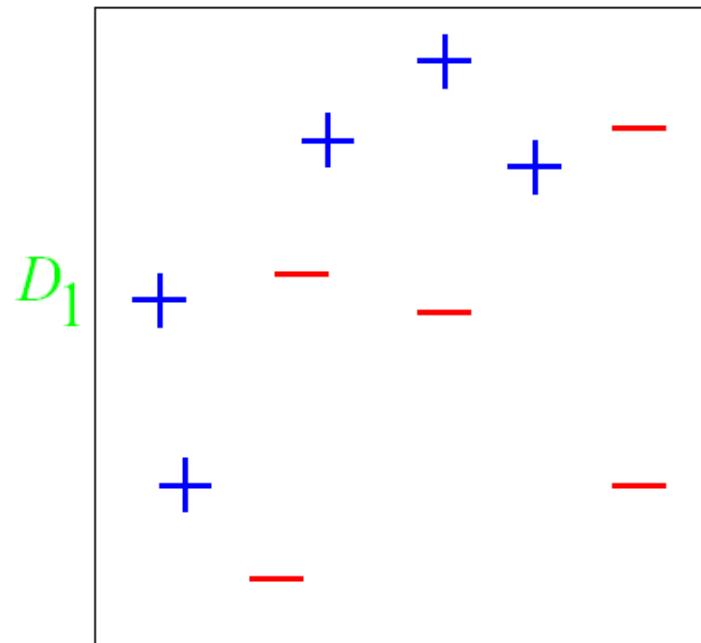
Boosting Intuition

- We adaptively weigh each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them)
- Each boosting round learns a new (simple) classifier on the weighed dataset.

Boosting Intuition

- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that obtain low training error rate have high weight.
- We stop by using monitoring a hold out set (cross-validation).

An Illustrative Example



Original training set: equal weights to all training samples

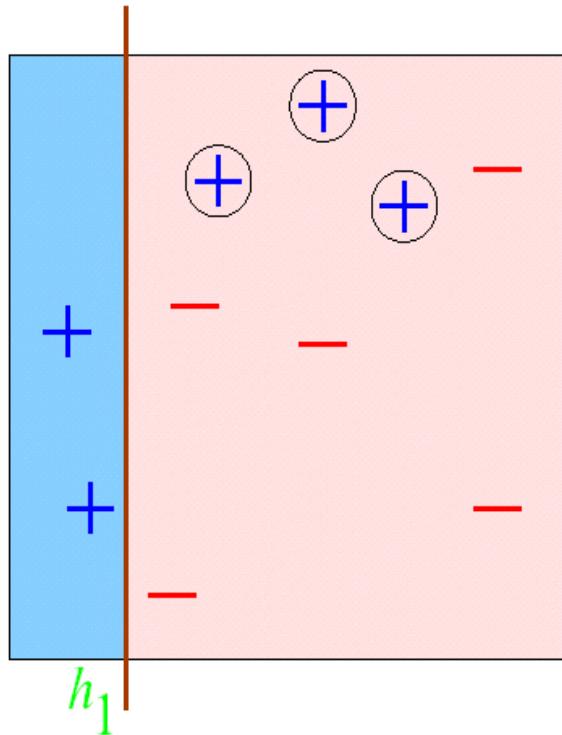
Taken from “A Tutorial on Boosting” by Yoav Freund and Rob Schapire

AdaBoost example

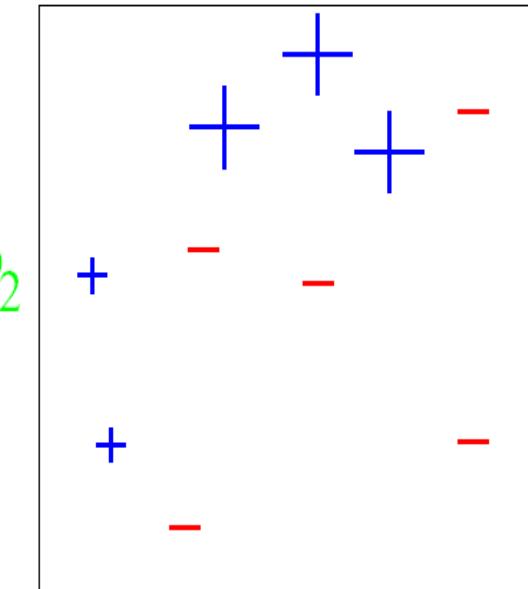
ε = error rate of
classifier

α = weight of classifier

ROUND 1

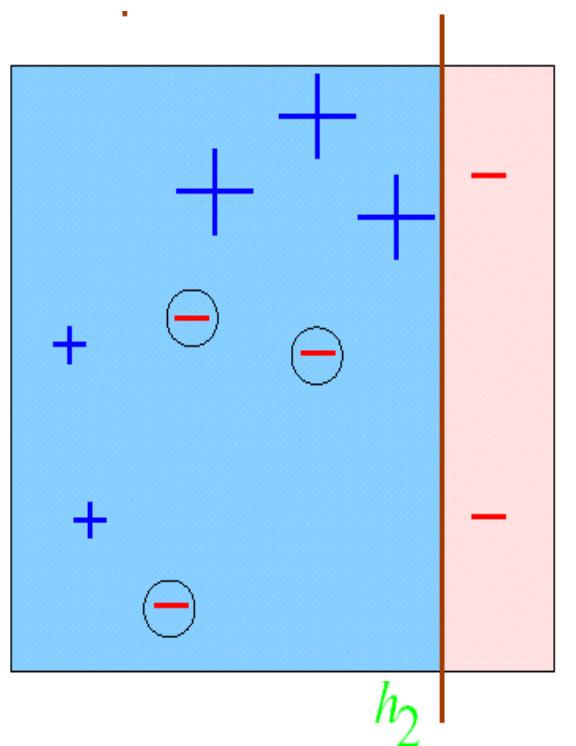


$$\begin{aligned}\varepsilon_1 &= 0.30 \\ \alpha_1 &= 0.42\end{aligned}$$

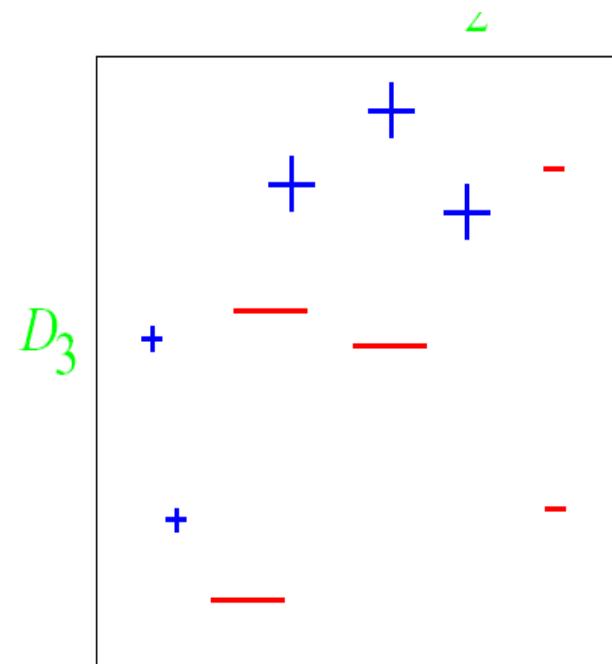


AdaBoost example

ROUND 2

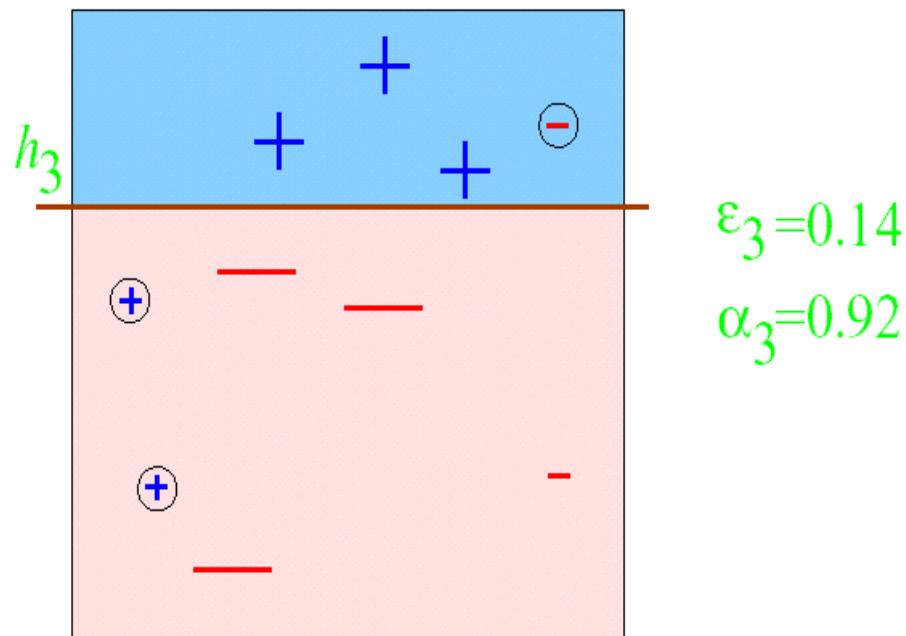


$$\begin{aligned}\varepsilon_2 &= 0.21 \\ \alpha_2 &= 0.65\end{aligned}$$

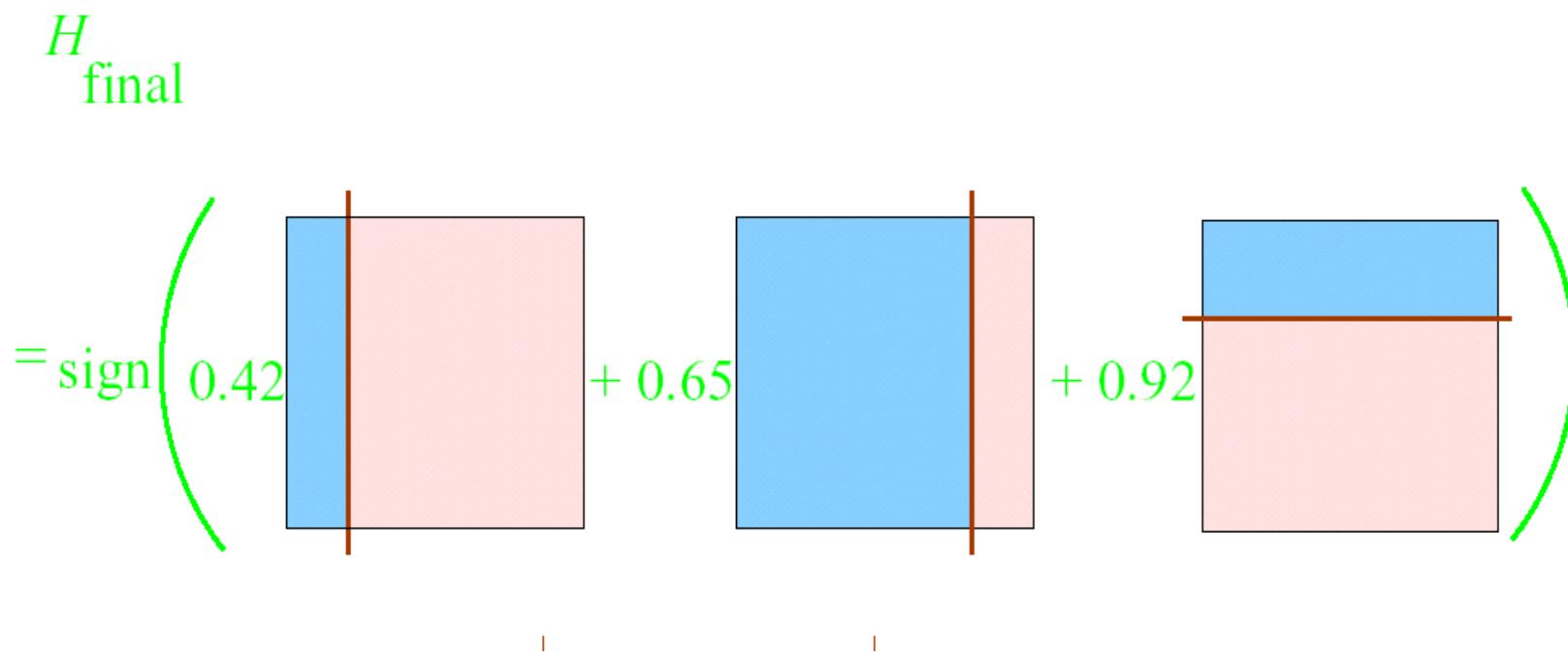


AdaBoost example

ROUND 3



AdaBoost example



Boosting (cont.)

Boosting is remarkably resistant to overfitting, and it is fast and simple.

In fact, it can often continue to improve even when the training error has gone to zero.

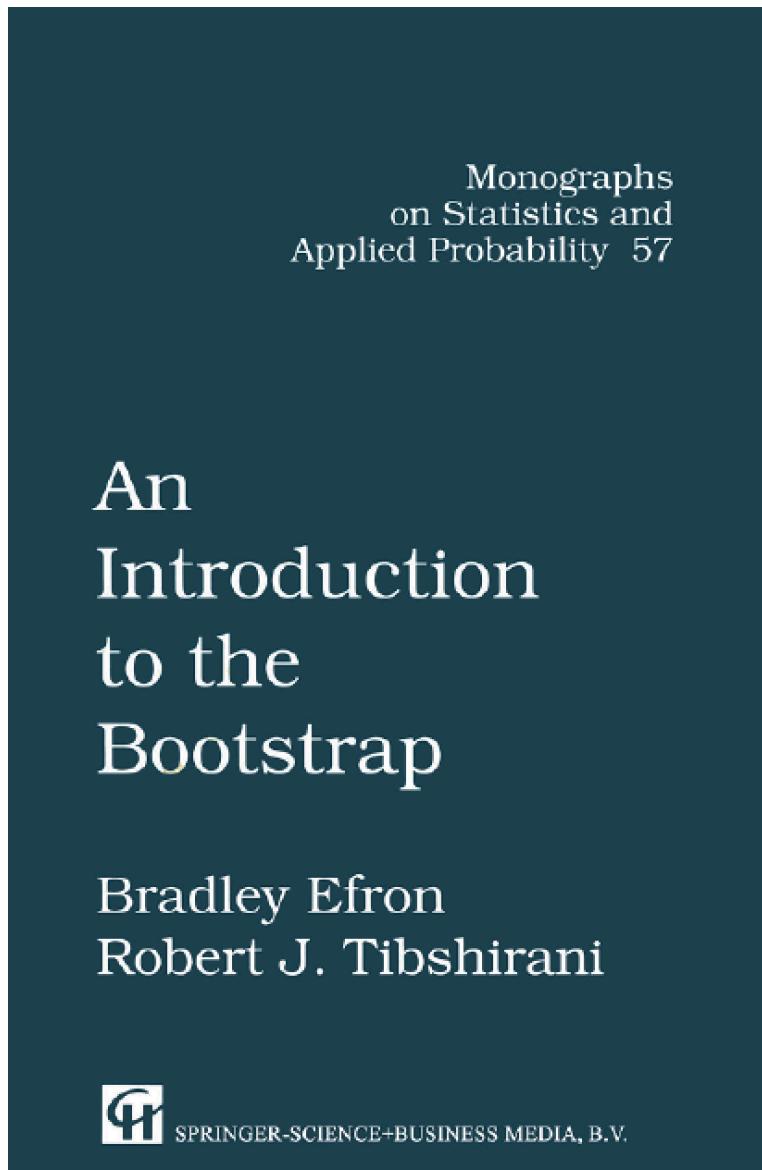
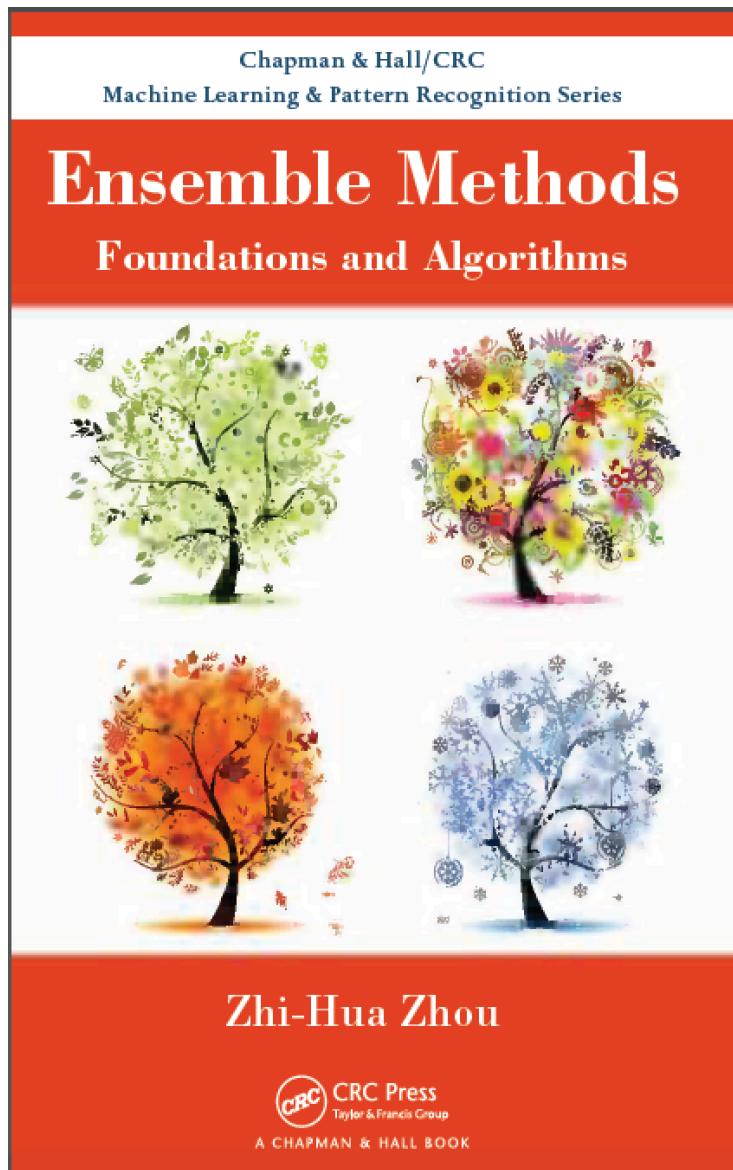
Boosting (cont.)

It improves the performance of many kinds of machine learning algorithms.

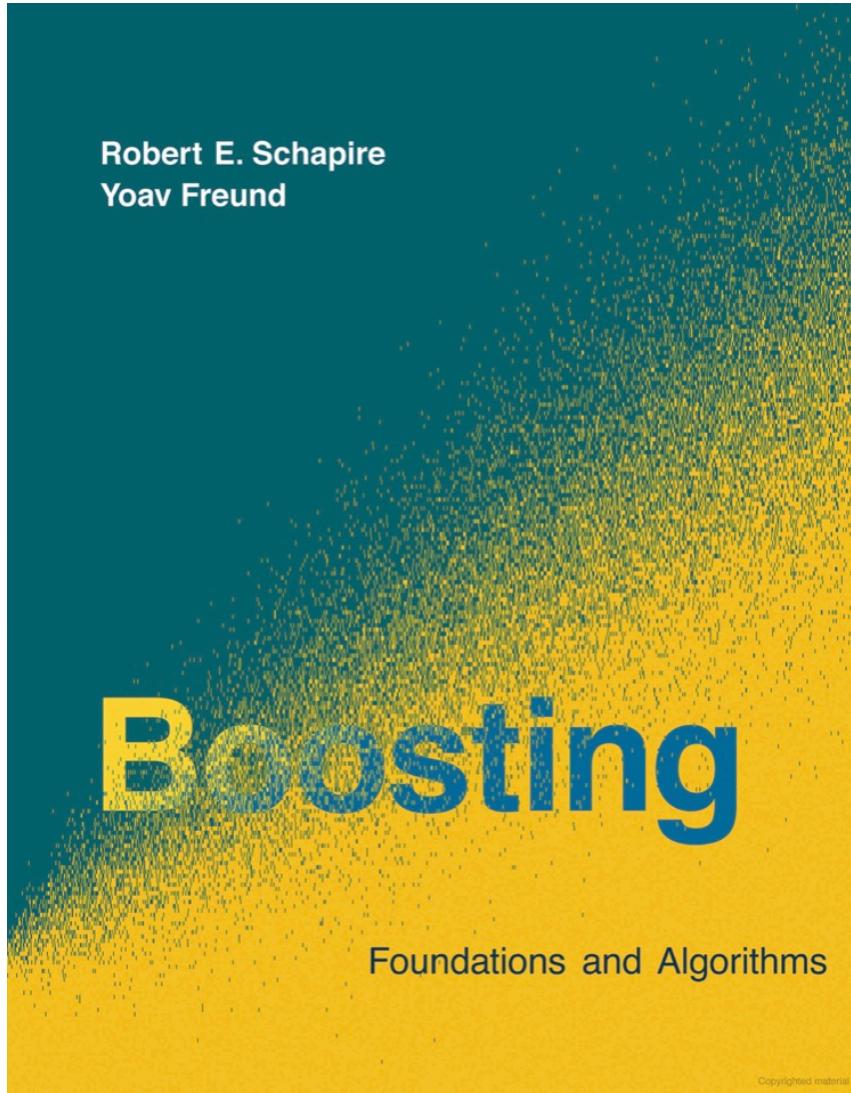
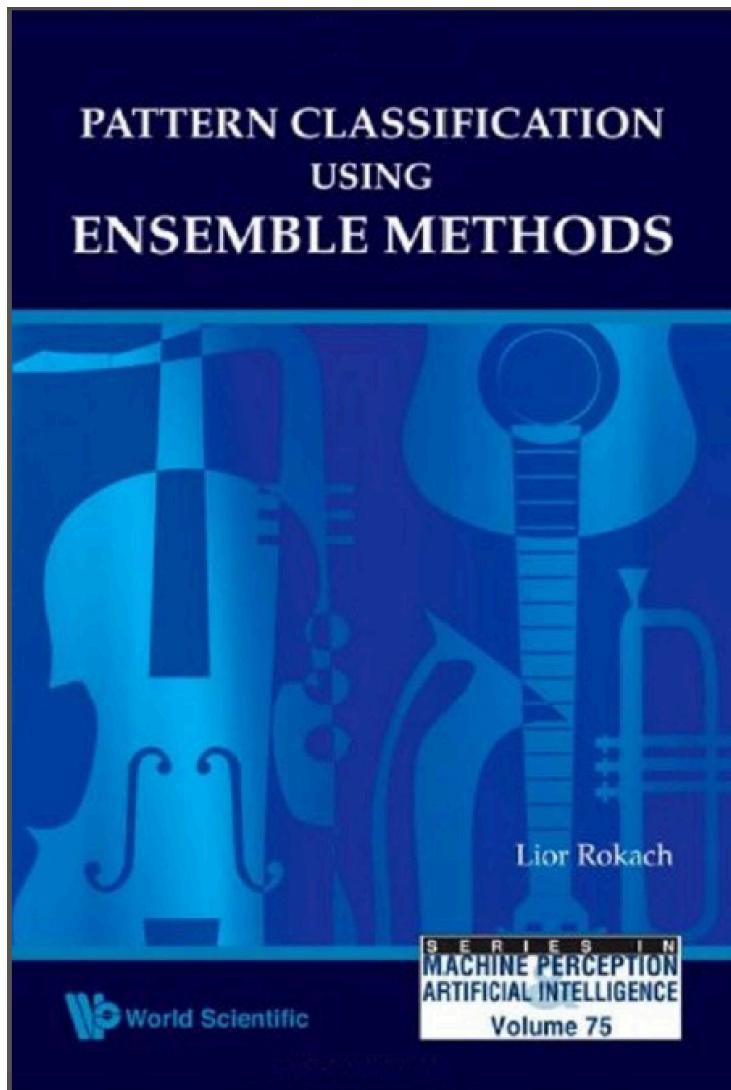
Boosting does not work when:

Not enough data, base learner is too weak or too strong, and/or susceptible to noisy data.

More



More



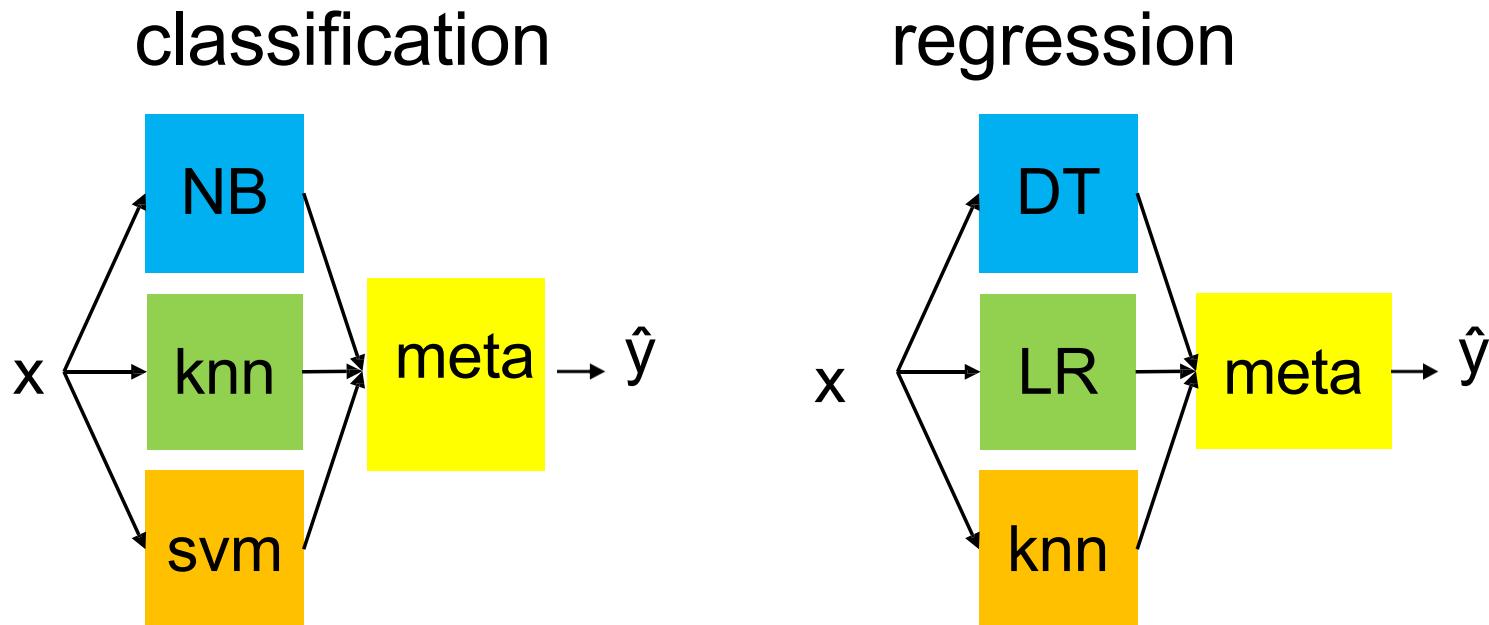
Appendix

More on Ensemble Methods

Stacking

Stacking

- Basic idea: use the output of multiple classifiers as input to a **meta-model (meta-learner)**.
- We ‘stack’ the meta-model on top of the base models

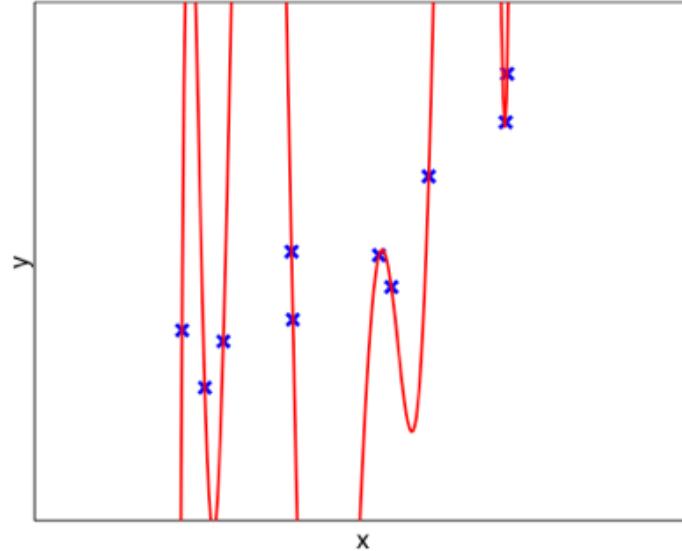
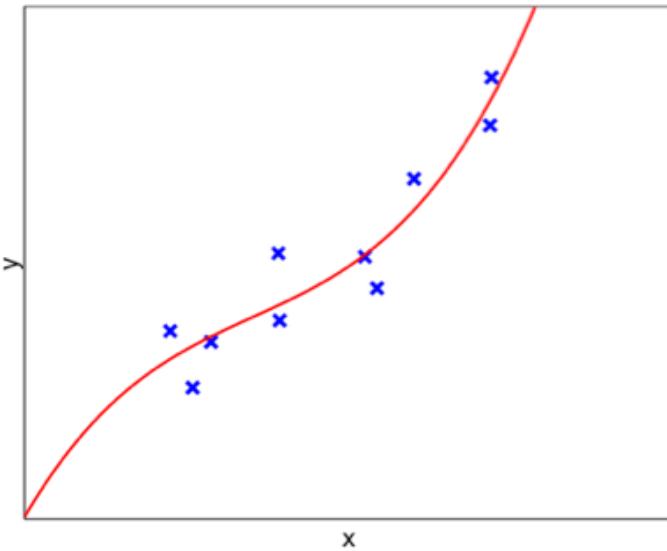


Stacking – a naïve approach

- Let's consider the regression case
- Base model predictions are $f_1(\mathbf{x}), \dots, f_L(\mathbf{x})$
- Meta learner could be a linear regression model
$$f_{\text{meta}}(\mathbf{x}) = \sum_{i=1}^L w_i f_i(\mathbf{x})$$
- If we could choose w_i to minimize true error, the stacked model would always be at least as good as any base model!
 - Worst case we set all w_i to 0 except that of the best base model
- But what if we minimize the train error instead?

Stacking – a naïve approach

- Consider the following base models



- What weights would minimize train set error?
- Does that yield good generalization error for the meta model?

Stacking – a naïve approach

- Naïve implementation of stacking prefers over-fitted models
- Underlying problem: the outputs of the base models have been adapted to the labels.

Stacking – a naïve approach

- Thus, inputs of the meta model are **not representative** of the inputs it will get at test-time.
- To avoid preference for overfitted models, inputs to the meta- model should not have seen the labels for the data points themselves

Stacking – second attempt

- Now, we can train the meta-model on the data in the base model outputs paired with the target label
- Any base-model output is now a good indication of test-time behavior
- If the meta-model has free parameters itself, we can cross-validate using the same folds

Stacking – second attempt

- Usually, the meta-model is relatively simple (e.g. linear regression or logistic regression)
- Empirical Recommendation: Do not have your base-learners as the meta-learner.
For example, if you use logistic regression as the base-learner, use some other classifier (e.g. SVM) as the meta-learner.

Testing the stacked model

- To test the stacked model, again we set aside a test set from the very beginning
- Have several versions of the base models from cross-validation!

Testing the stacked model

- Two approaches:
 - Retrain the base models on the whole dataset
 - Possible disadvantage: slightly different input to meta-model
 - Use an average of the trained base models
 - Possible disadvantage: time cost
- Then feed the base model predictions into the trained meta- model

Comparison to model selection

- If we force meta-learner to use just one base model (with weight 1) and set all other weights to 0, this is equivalent to selecting the best model with cross-validation
- More expressive meta-models (e.g. linear / logistic regression) can leverage the relative strength of multiple models

Comparison to model selection

- A very complex meta-model (e.g. decision tree) could again easily overfit
- Could use cross-validation on the meta-level to ensure good generalization properties

Effectiveness of stacking

- Stacking generally improves performance, but not by much
- Additional cost of training and evaluating multiple models

Effectiveness of stacking

- If **interpretability or speed are important** consideration, stacking might not help you much.
- In competitions where **a small gain is important** and time cost is not so much of an issue, it is usually effective!
- Quite **useful in collaborative approaches** where everyone can integrate their own model in overall system

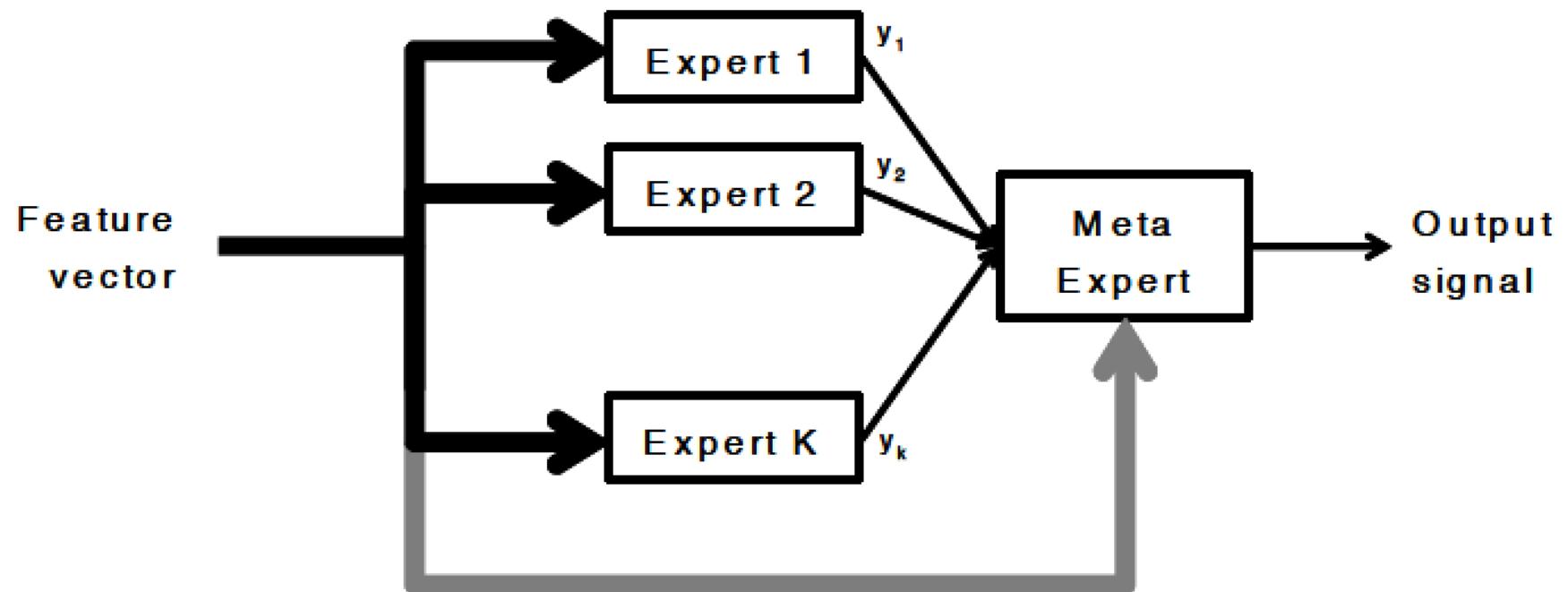
Adaptive Meta-Learners

- The meta-learner is a function that depends on the input feature vector
- Thus, the ensemble implements a function that is local to each region in feature space
- This **divide-and-conquer approach** leads to **modular ensembles** where relatively simple classifiers **specialize** in different parts of I/O space

Adaptive Meta-Learners

- In contrast with static-combiner ensembles, the individual experts here do not need to perform well for all inputs, only in their region of expertise
- Representative examples of this approach are **Mixture of Experts (ME)** and **Hierarchical ME** [Jacobs et al., 1991; Jordan and Jacobs, 1994]

Adaptive Meta-Learners

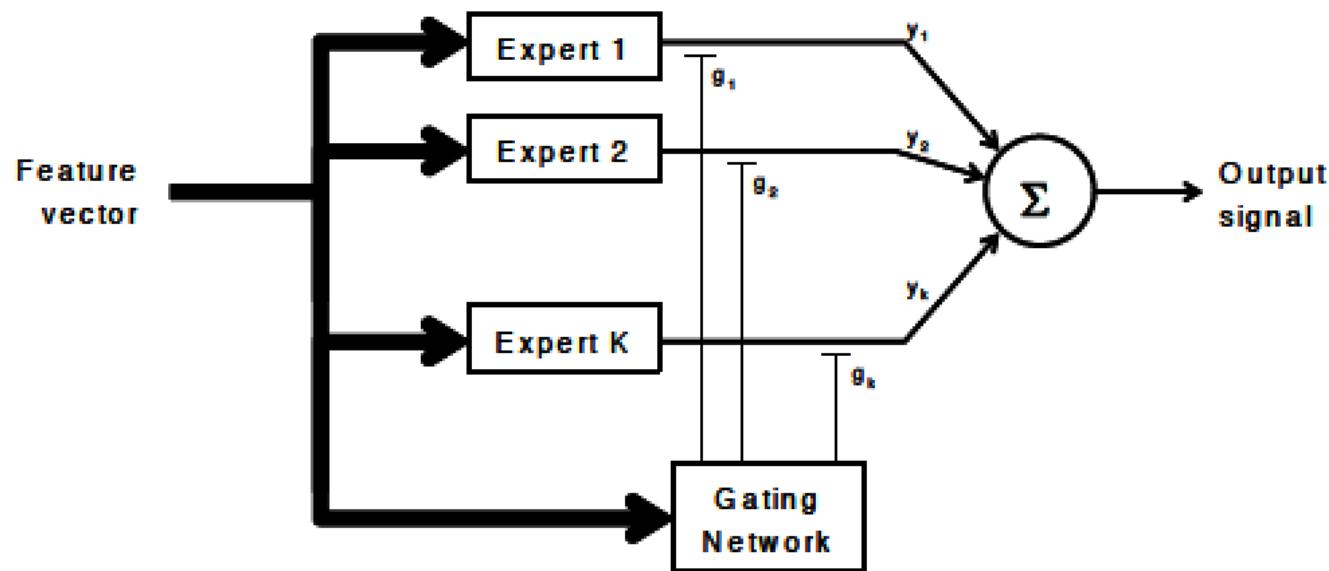


Mixture of Experts

- ME is the classical adaptive ensemble method
- A **gating network** is used to partition feature space into different regions, with one expert in the ensemble being responsible for generating the correct output within that region [Jacobs et al., 1991]

Mixture of Experts

- The experts in the ensemble and the gating network **are trained simultaneously**.
- ME can be extended to a multi-level hierarchical structure, where each component is itself a ME.



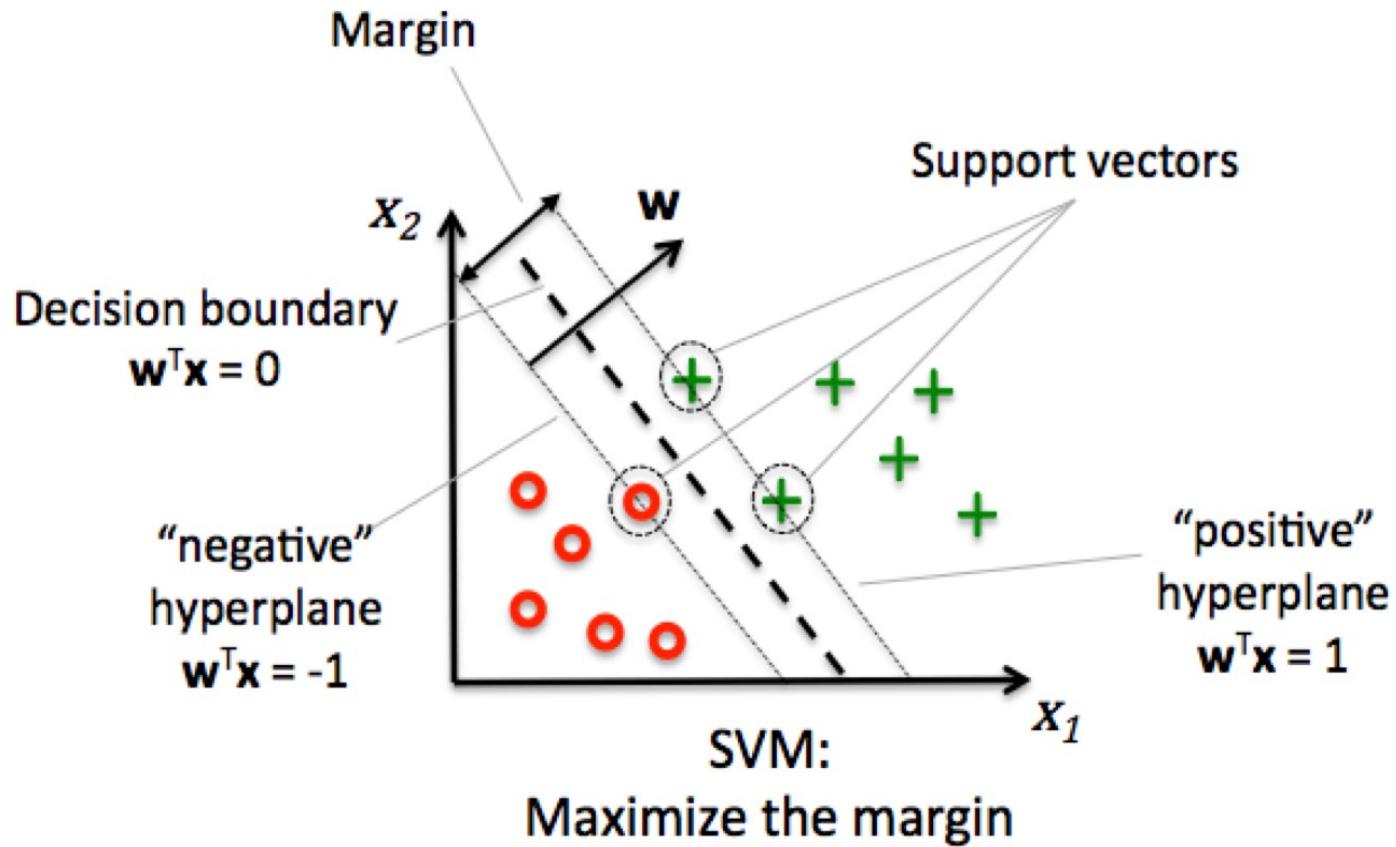
DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 7

Support Vector Machines



Support Vector Machines

Here we approach the two-class classification problem in a direct way:

We try and find a plane that separates the classes in feature space.

If we cannot, we get creative in two ways:

- We soften what we mean by “separates”, and
- We enrich and enlarge the feature space so that separation is possible.

What is a Hyperplane?

- A hyperplane in p dimensions is a flat affine subspace of dimension $p - 1$.
- In general the equation for a hyperplane has the form

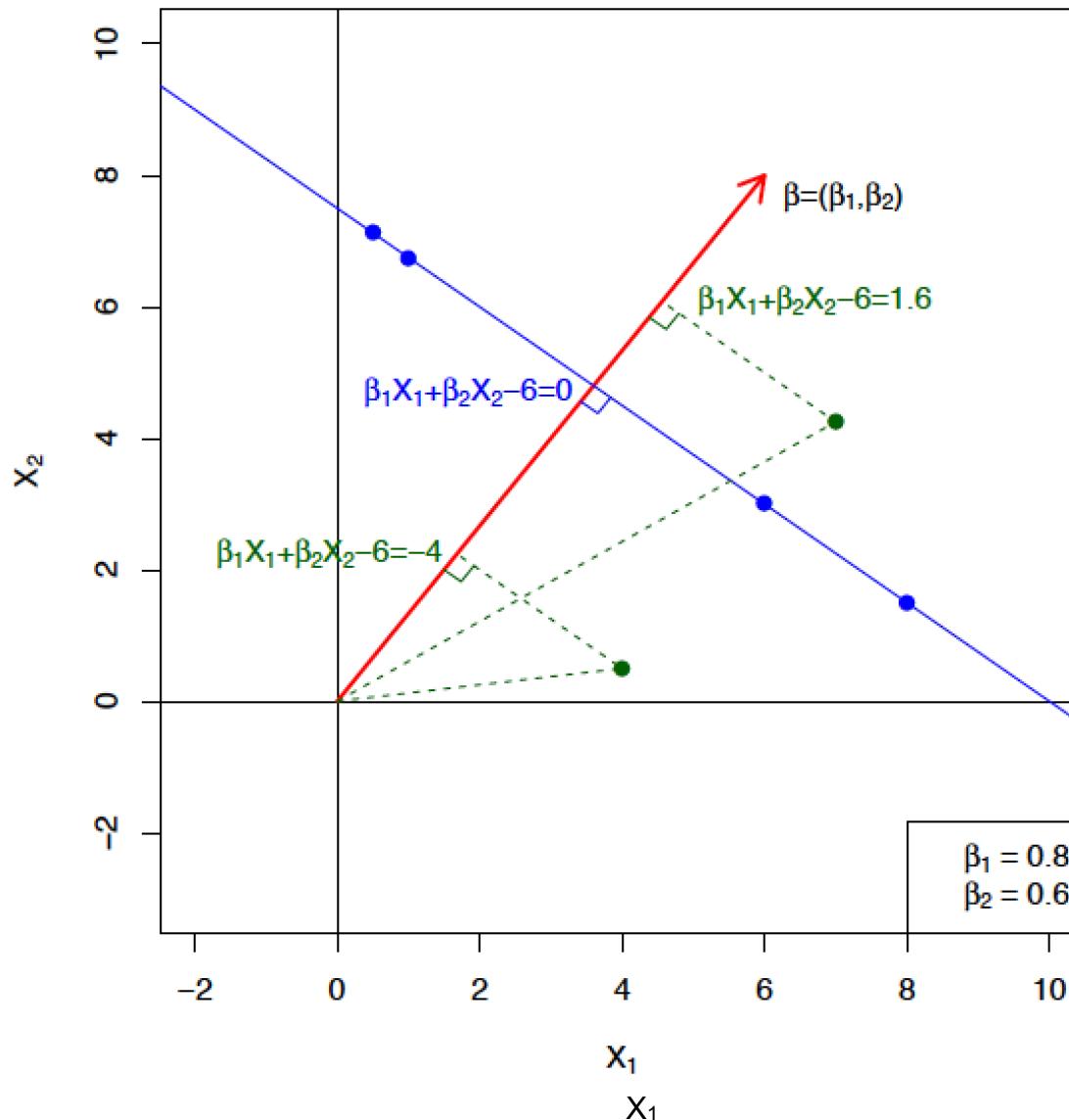
$$\begin{aligned}\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \\ = 0\end{aligned}$$

- In $p = 2$ dimensions a hyperplane is a line.

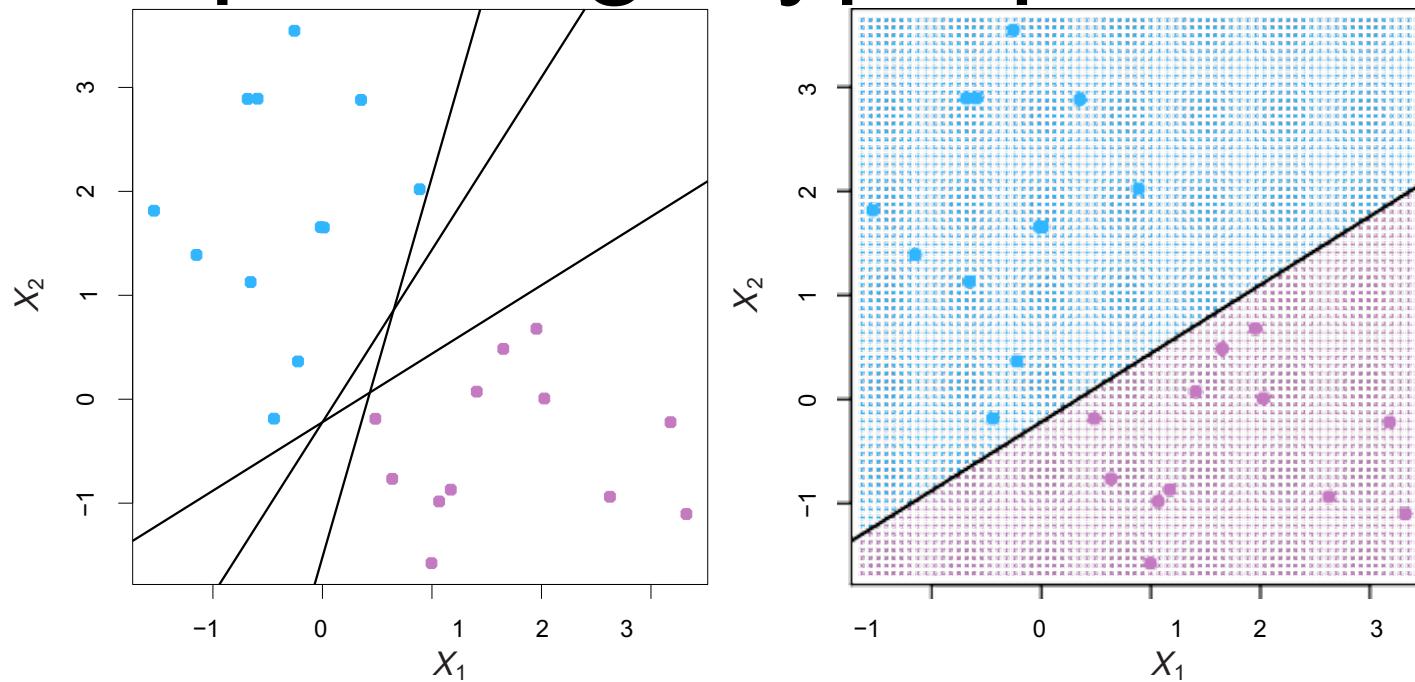
What is a Hyperplane?

- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \dots, \beta_p)$ is called the normal vector
 - it points in a direction orthogonal to the surface of a hyperplane.

Hyperplane in 2 Dimensions



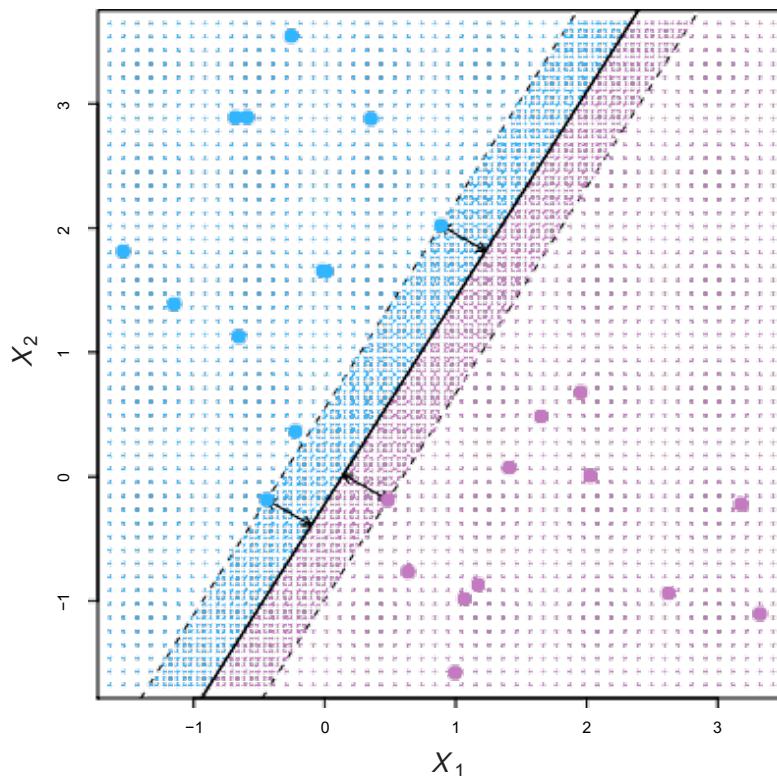
Separating Hyperplanes



- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all i , $f(X) = 0$ defines a *separating hyperplane*.

Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$\begin{aligned} & \text{maximize } M \\ & \beta_0, \beta_1, \dots, \beta_p \end{aligned}$$

$$\text{subject to } \sum_{j=1}^p \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$$

for all $i = 1, \dots, N$.

This can be rephrased as a convex quadratic program, and solved efficiently.

Maximal Margin Classifier

maximize M
 $\beta_0, \beta_1, \dots, \beta_p$

subject to $\sum_{j=1}^p \beta_j^2 = 1,$

$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$
for all $i = 1, \dots, N.$

The constraint $\sum_{j=1}^p \beta_j^2 = 1$ makes
sure that a unique solution for
 β_i 's exists.

Combined with

$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$
for all $i = 1, \dots, N.$

it guarantees that the minimum
margin is M if M is positive.

Maximal Margin Classifier

For each observation to be on the correct side of the hyperplane we would simply need

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0$$

so the constraint

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$$

for all $i = 1, \dots, N$.

in fact requires that each observation be on the correct side of the hyperplane, *with some cushion*, provided that M is positive.

Maximal Margin Classifier

Since if

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} = 0$$

Defines a hyperplane, then

$$k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) = 0$$

also defines a hyperplane for any nonzero k

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

guarantees a unique set of β 's exists,

Maximal Margin Classifier

Moreover, the constraint

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

adds meaning to

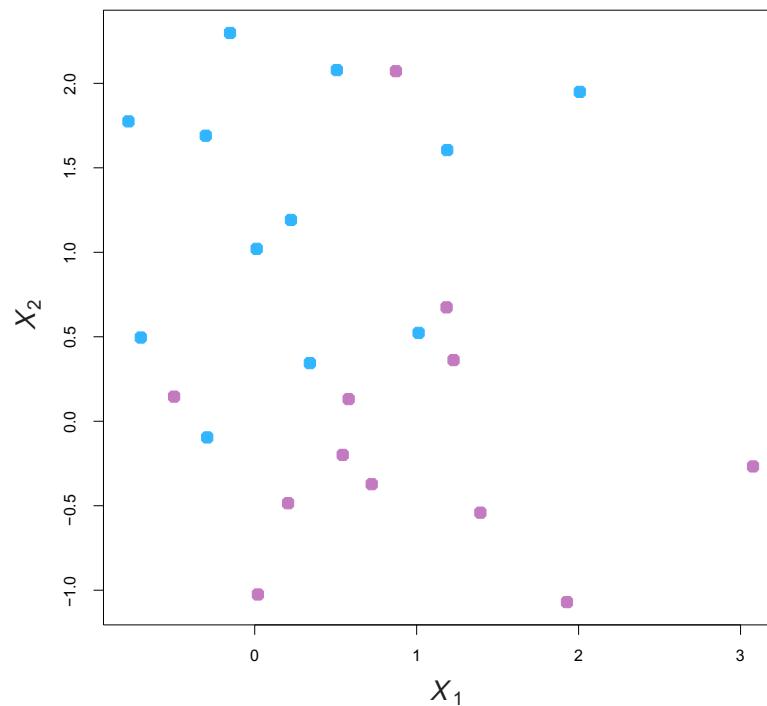
$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$$

for all $i = 1, \dots, N.$

one can show that with this constraint the perpendicular distance from the i^{th} observation to the hyperplane is given by

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip})$$

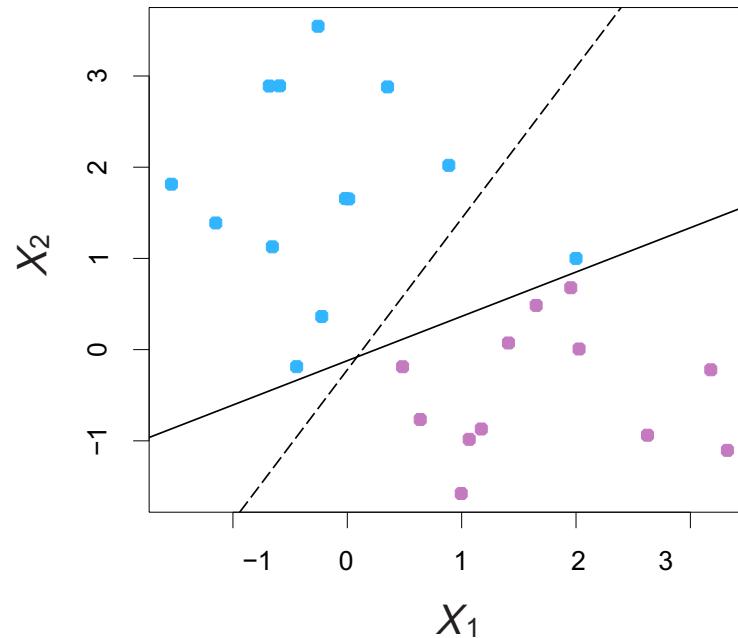
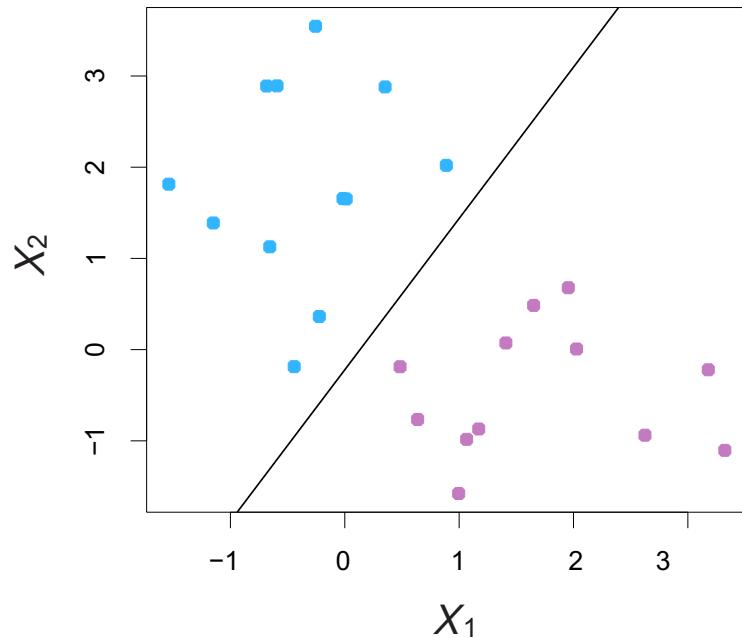
Non-separable Data



The data on the left are not separable by a linear boundary.

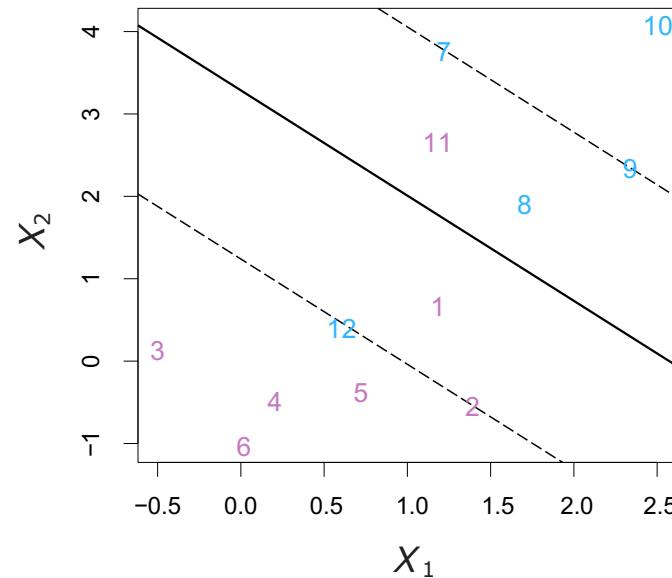
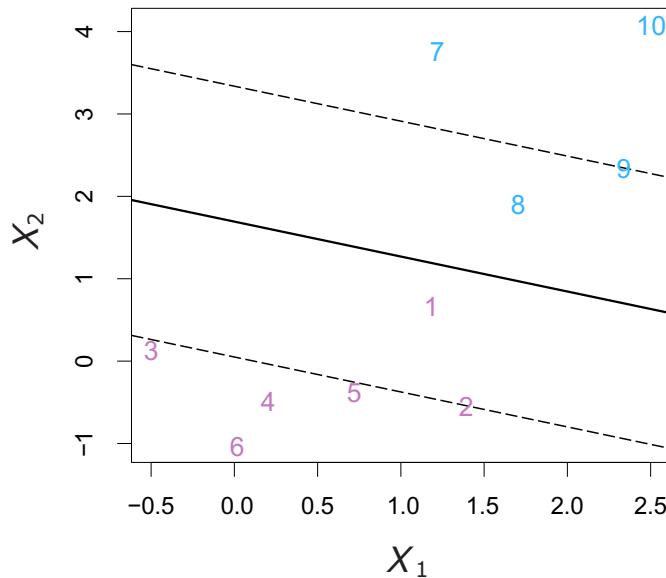
This is often the case, unless $N < p$.

Noisy Data



Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier. The *support vector classifier* maximizes a *soft* margin.

Support Vector Classifier



$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C,$$

Support Vector Classifier

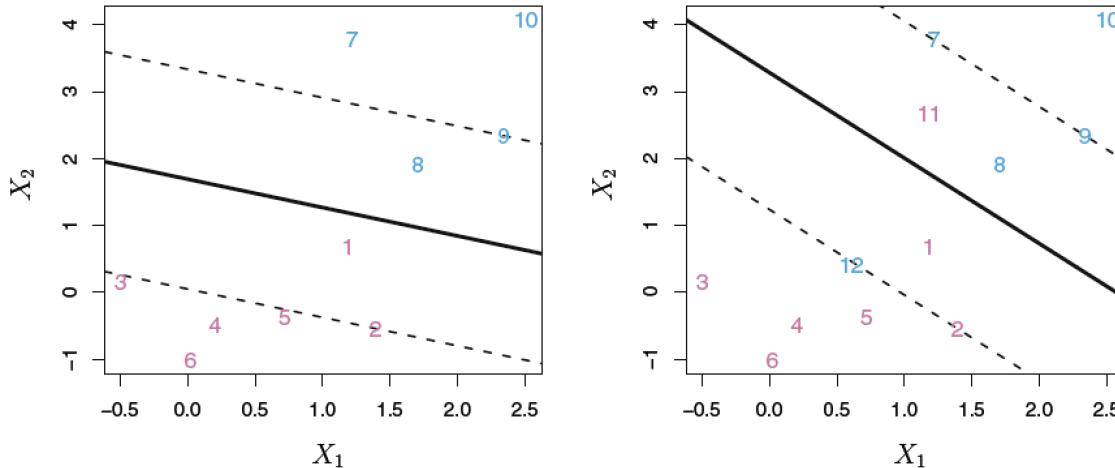


FIGURE 9.6. Left: A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines. Purple observations: Observations 3, 4, 5, and 6 are on the correct side of the margin, observation 2 is on the margin, and observation 1 is on the wrong side of the margin. Blue observations: Observations 7 and 10 are on the correct side of the margin, observation 9 is on the margin, and observation 8 is on the wrong side of the margin. No observations are on the wrong side of the hyperplane. Right: Same as left panel with two additional points, 11 and 12. These two observations are on the wrong side of the hyperplane and the wrong side of the margin.

Details of Optimization Problem

The slack variable ε_i tells us where the i^{th} observation is located, relative to the hyperplane and relative to the margin.

- If $\varepsilon_i = 0$ then the i^{th} observation is on the **correct side** of the margin

Details of Optimization Problem

The slack variable ε_i tells us where the i^{th} observation is located, relative to the hyperplane and relative to the margin.

- If $\varepsilon_i > 0$ then the i^{th} observation is on the **wrong side** of the margin, and we say that the i^{th} observation has *violated* the margin.

Details of Optimization Problem

The slack variable ε_i tells us where the i^{th} observation is located, relative to the hyperplane and relative to the margin.

- If $\varepsilon_i > 1$ then it is on the **wrong side of the hyperplane**.

Details of Optimization Problem

- C bounds the sum of the ε_i 's, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.

Details of Optimization Problem

- We can think of C as a budget for the amount that the margin can be violated by the N observations.
- If $C = 0$ then there is no budget for violations to the margin, so all ε_i 's = 0, which leads to the maximal margin hyperplane optimization problem

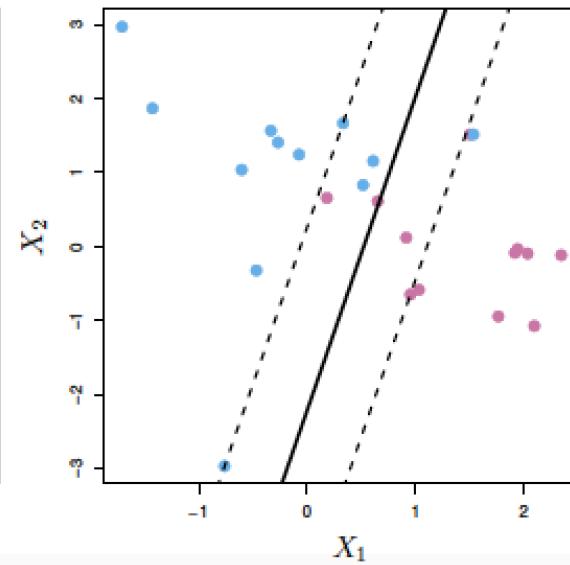
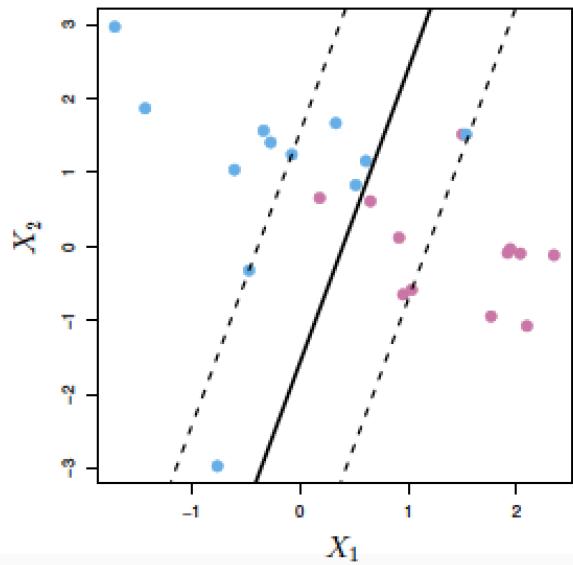
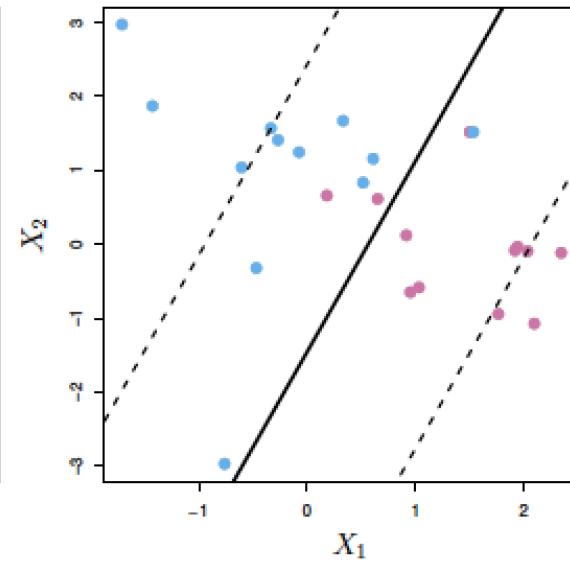
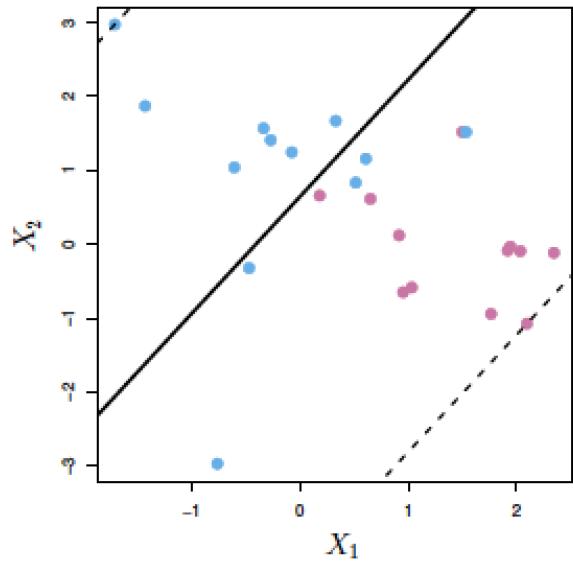
Details of Optimization Problem

For $C > 0$ no more than C observations can be on the wrong side of the hyperplane, because if an observation is on the wrong side of the hyperplane then $\varepsilon_i > 1$, and the optimization problem requires that the sum of ε_i 's be less than C .

Details of Optimization Problem

- As the budget C increases, we become more tolerant of violations to the margin, and so the margin will widen.
- Conversely, as C decreases, we become less tolerant of violations to the margin and so the margin narrows.
 - An example is shown in the next slide.

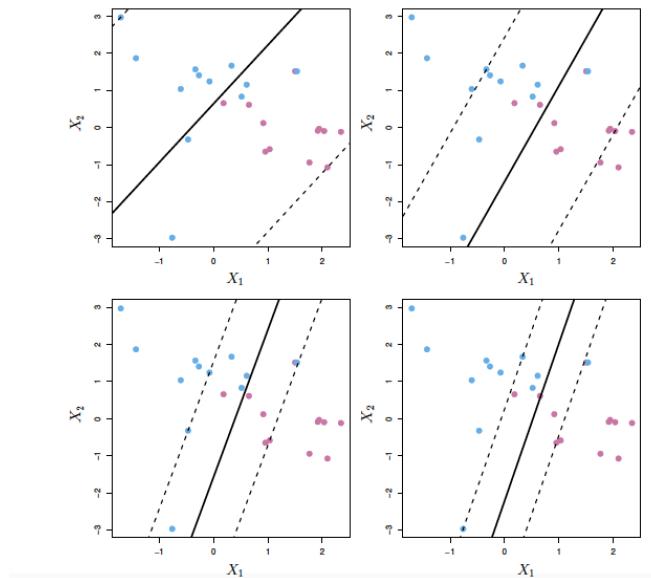
C is a regularization parameter



C is a regularization parameter

A support vector classifier was fit using four different values of the tuning parameter C.

- The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels.



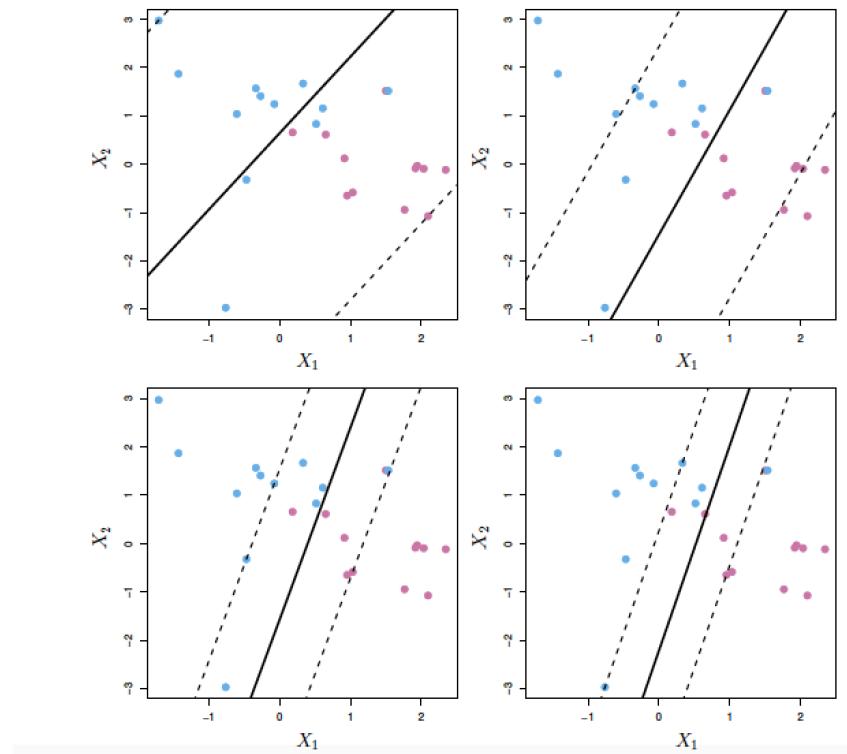
C is a regularization parameter

A support vector classifier was fit using four different values of the tuning parameter C.

- When C is large, there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large.
- As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

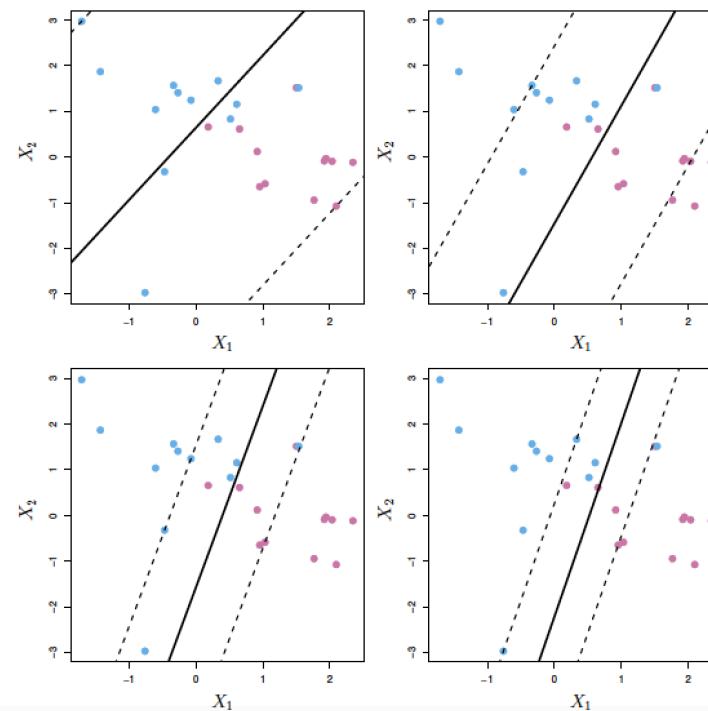
C is a regularization parameter

- C is treated as a tuning parameter generally chosen via cross-validation.
- C controls the bias-variance trade-off of the statistical learning technique.



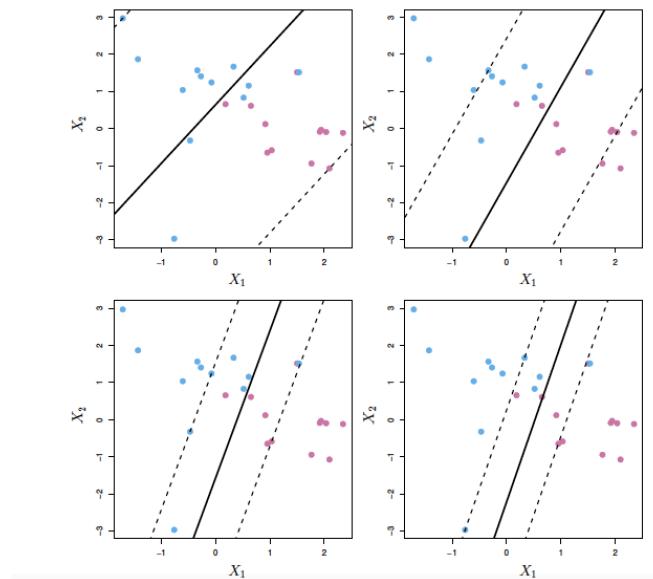
C is a regularization parameter

- When C is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.



C is a regularization parameter

- On the other hand, when C is larger, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.



Support Vectors

- Only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained.
- In other words, an observation that lies strictly on the correct side of the margin does not affect the support vector classifier!

Support Vectors

- Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin.

Support Vectors

- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors . These observations do affect the support vector classifier.

Support Vectors

The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane.

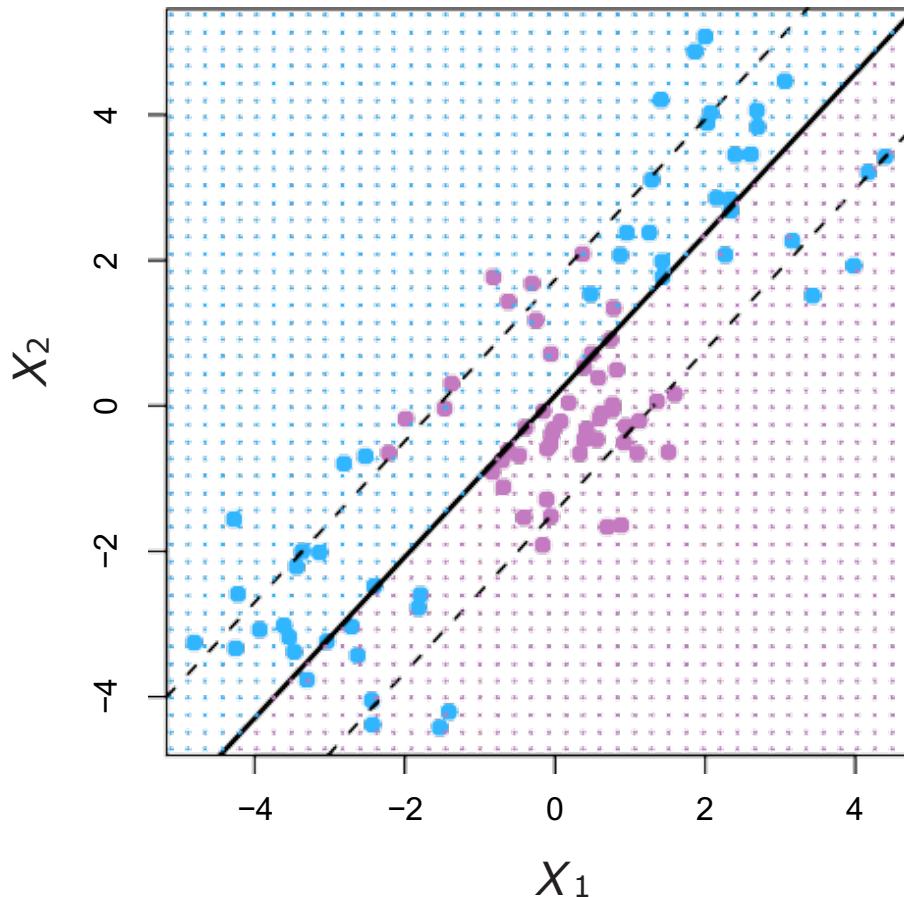
Comparison with LDA

- This property is distinct from some of the other classification methods that we have seen, such as linear discriminant analysis.
- The LDA classification rule depends on the mean of all of the observations within each class, as well as the within-class covariance matrix computed using all of the observations.

Comparison with LR

In contrast, logistic regression, unlike LDA, has very low sensitivity to observations far from the decision boundary. In fact we will see that the support vector classifier and logistic regression are closely related.

Linear boundary can fail



Sometimes a linear boundary simply won't work, no matter what value of C .

The example on the left is such a case.

What to do?

Feature Expansion

Enlarge the space of features by including transformations; e.g. $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \dots$. Hence go from a p -dimensional space to a $M > p$ dimensional space.

- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

Feature Expansion

Example: Suppose we use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1, X_2) . Then the decision boundary would be of the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

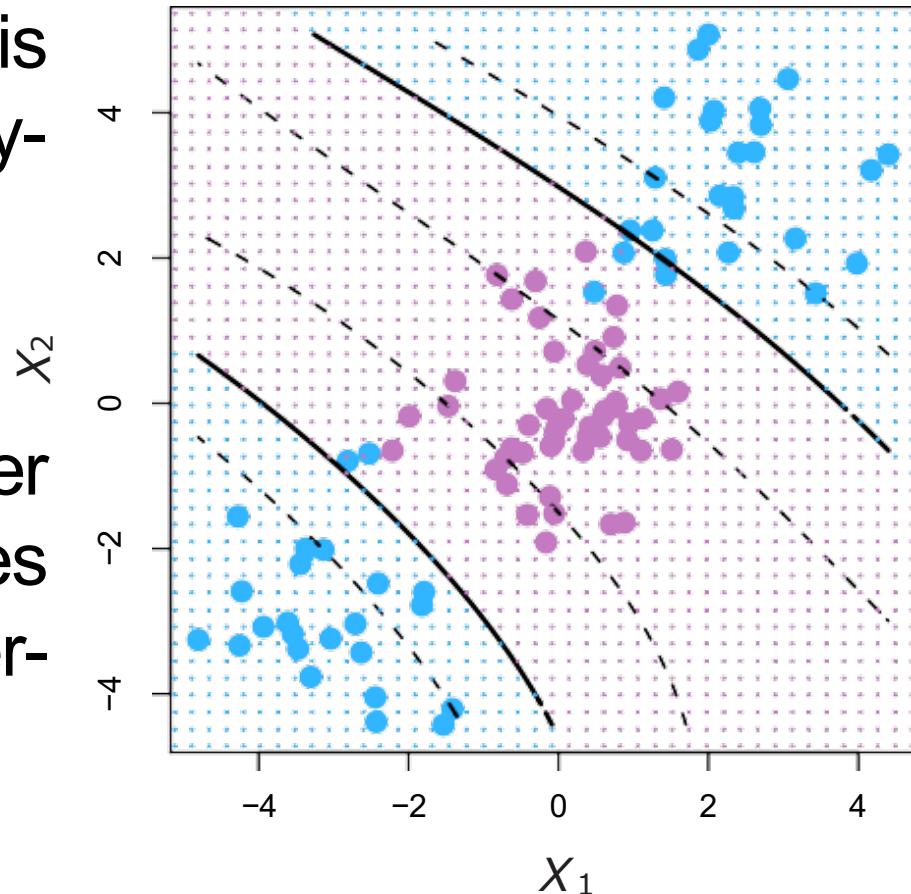
This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space



$$\begin{aligned} & \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 \\ & + \beta_7 X_2^3 + \beta_8 X_1 X_2^2 + \beta_9 X_1^2 X_2 = 0 \end{aligned}$$

Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of *kernels*.
- Before we discuss these, we must understand the role of *inner products* in support-vector classifiers.

Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j}$$

inner product between vectors

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

with n parameters

Subject to

$$\sum_{i=1}^n \alpha_i = 0$$

Inner products and support vectors

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

The intercept can be calculated as:

$$\beta_0 = \frac{1}{y_j} - \sum_{i=1}^N \alpha_i \langle x_i, x_j \rangle$$

where x_j is one of the support vectors and y_j is its label.

Inner products and support vectors

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle \quad \sum_{i=1}^n \alpha_i = 0$$

with n parameters

in order to evaluate the function $f(x)$, we need to compute the inner product between the new point x and each of the training points x_i .

Inner products and support vectors

To estimate the parameters $\alpha_1, \dots, \alpha_n$ and β_0 , all we need are the $\binom{n}{2}$ inner products $\langle x_i, x_j \rangle$ between all pairs of training observations.

However, it turns out that α_i is nonzero only for the support vectors in the solution—that is, if a training observation is not a support vector, then its α_i equals zero.

Inner products and support vectors

In other words

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle x, x_i \rangle \quad \sum_{i \in S} \hat{\alpha}_i = 0$$

$$\beta_0 = \frac{1}{y_j} - \sum_{i \in S} \hat{\alpha}_i \langle x_i, x_j \rangle$$

where S is the support set of indices i for support vectors.

Inner products and support vectors

To expand the feature space, one can use a transformed set of features $u = \varphi(x)$. Note that u does not need to be of the same dimension as x .

The classifier in the new feature space can be represented as

$$f_1(x) = f(\varphi(x)) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle \varphi(x), \varphi(x_i) \rangle$$

Inner products and support vectors

It can be shown that a class of functions called *kernels* can be represented as inner products $\langle \varphi(x), \varphi(x_i) \rangle$

The interesting point is that we do not need to explicitly know the function $\varphi(x)$ to use the kernel that correspond to it!

They are generalizations of the inner product.

Kernels and Support Vector Machines

Now suppose that every time the inner product appears in our equations, we replace it with

$$K(x_i, x_{i'})$$

where K is a *kernel*.

A kernel is a function that quantifies the similarity of two observations.

For SVC, the Kernel is the usual inner product, which is called a *linear kernel*, because $\varphi(x) = x$.

Aside: Mercer's Condition

- A function $K(x_i, x_j)$ is a Kernel function, i.e. it can be written in the following form

$$K(x_i, x_j) = [\varphi(x_i)]^T [\varphi(x_j)]$$

if and only if it satisfies Mercer's condition.

Aside: Mercer's Condition

- Mercer's condition: for all finite sequences (x_1, x_2, \dots, x_n) and all choices from the sequence of real numbers (c_1, c_2, \dots, c_n) :

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j K(x_i, x_j) \geq 0$$

Kernels and Support Vector Machines

For example

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij} x_{i'j} \right)^d$$

computes the inner-products needed for d dimensional polynomials-basis functions!

Kernels and Support Vector Machines

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij} x_{i'j} \right)^d$$

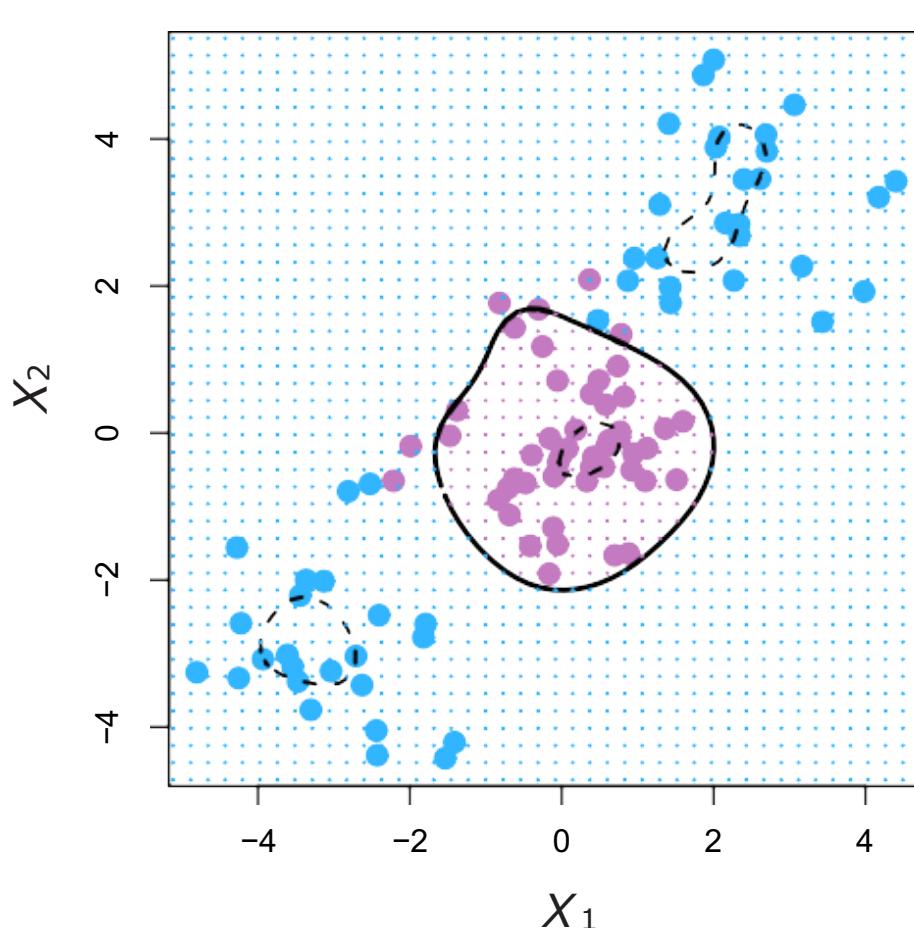
Try it for $p = 2$ and $d = 2$.

- The solution has the form

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i).$$

Radial Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2).$$



$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space;
very high dimensional.

Controls variance by
squashing down most
dimensions severely.

RBF Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2).$$

- If a given test observation $x^* = (x_1^* \dots x_p^*)^T$ is far from a training observation x_i in terms of Euclidean distance, then the exponent will be very negative, and so $K(x^*, x_i)$ will be very tiny.

RBF Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2).$$

- Therefore, x_i will play virtually no role in $f(x^*)$.
- Recall that the predicted class label for the test observation x^* is based on the sign of $f(x^*)$.
- The radial kernel has very local behavior: only nearby training observations have an effect on the class label of a test observation. (Similar to?)

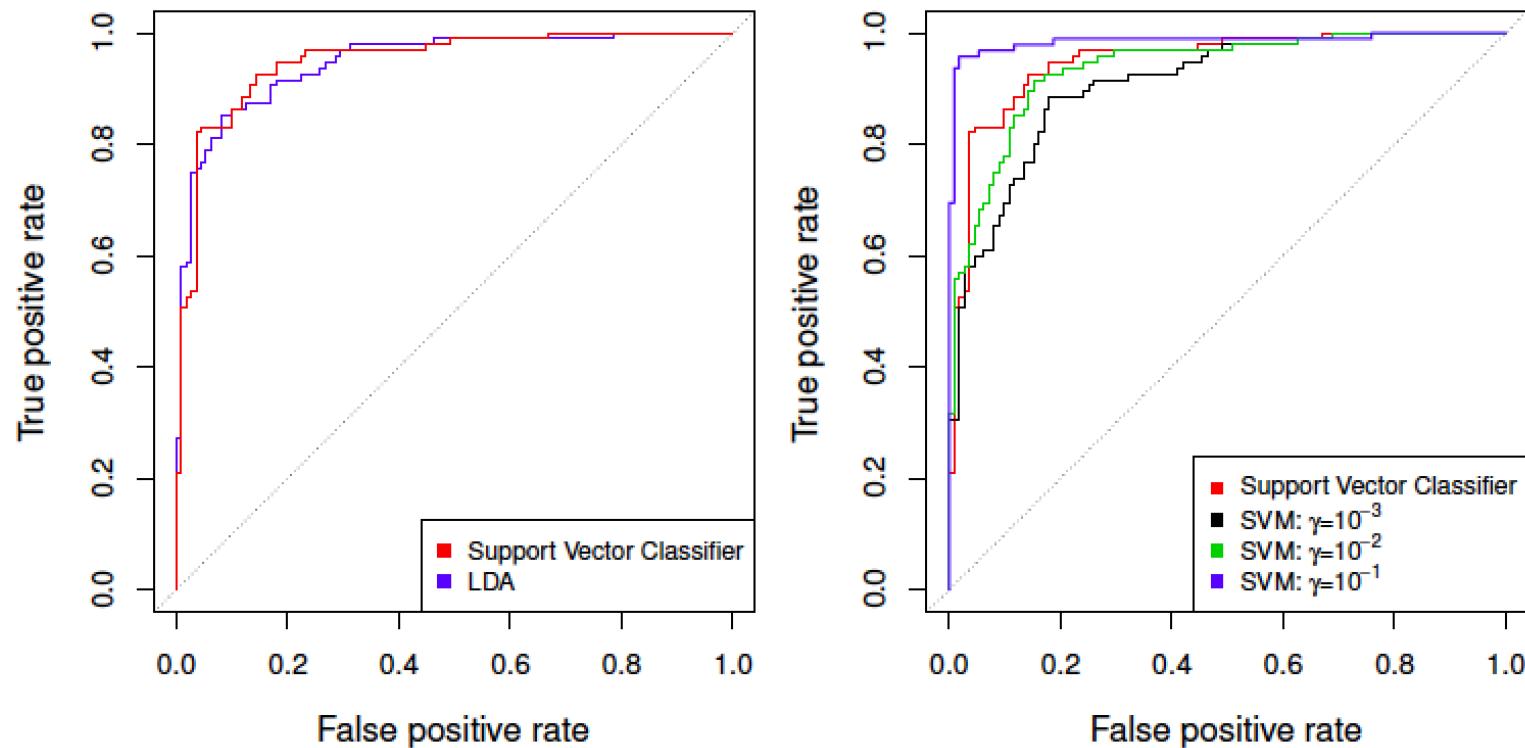
Computational Advantage

- What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features?
- One advantage is computing without explicitly working in the enlarged feature space.

Computational Advantage

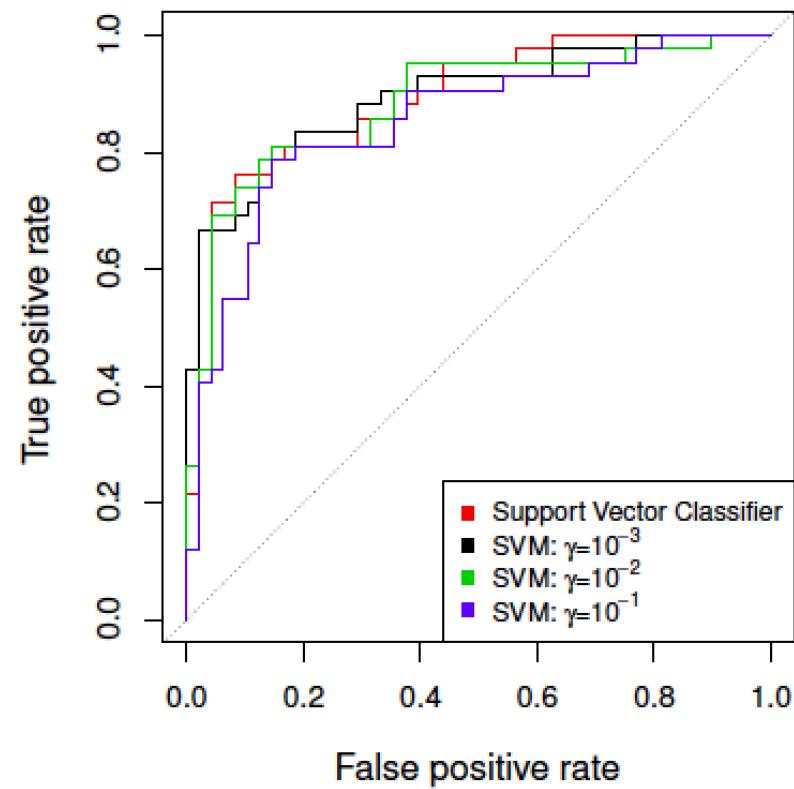
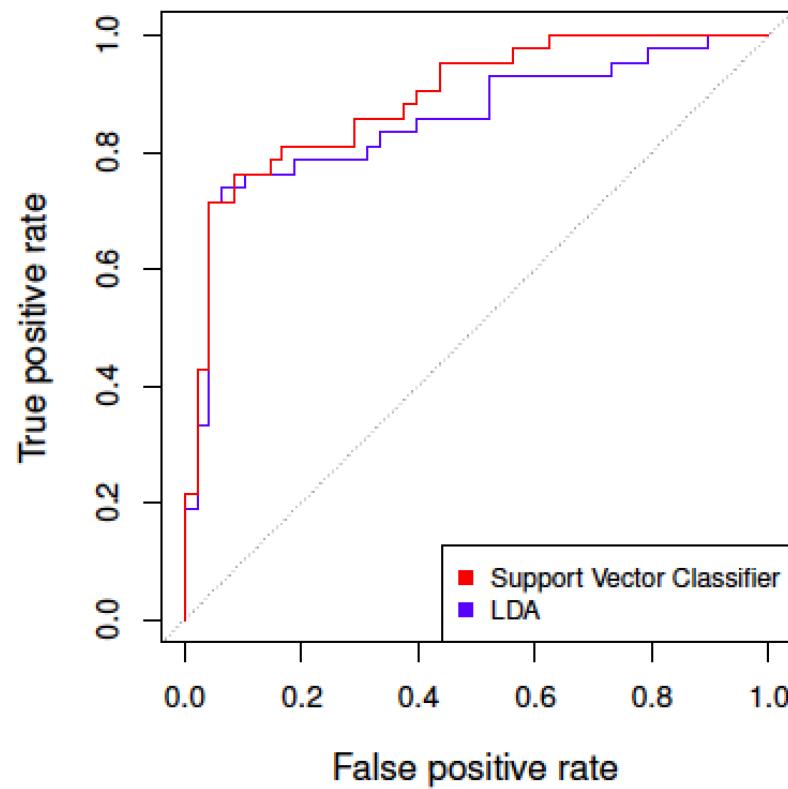
- This is important because in many applications of SVMs, the enlarged feature space is so large that computations are intractable.
- For some kernels, such as the radial, the feature space is implicit and infinite-dimensional, so we could never do the computations there anyway!

Example: Heart Data



ROC curve is obtained by changing the threshold 0 to threshold t in $f(X) > t$, and recording *false positive* and *true positive* rates as t varies. Here we see ROC curves on training data.

Example continued: Heart Test Data



Multi-Class and Multi-Label Classification Revisited

Multiclass classification means a classification task with more than two classes; e.g., classify a set of images of animals which may be horses, birds, or fish.

Multiclass classification makes the assumption that each sample is **assigned to one and only one label**: an animal can be either a horse or a bird but not both at the same time.

SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

- **OVA** One versus All (The rest). Fit K different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \dots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.

SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

- **OVO** One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{kl}(x)$. Classify x^* to the class that wins the most pairwise competitions.

Which to choose? If K is not too large, use OVO.

Multi-Class and Multi-Label Problems

Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document.

A text might be about any of religion, politics, finance or education at the same time or none of these.

Multi-Class and Multi-Label Classification Revisited

- Three methods to solve a multi-label classification problem:
 - Problem Transformation
 - Transform multi-label problem into single-label problem(s)
 - Adapted Algorithms
 - adapting conventional algorithms to directly perform multi-label classification
 - Ensemble approaches
 - Combining multiple classifiers

Multi-Class and Multi-Label Classification Revisited

- Problem Transformation
 - Binary Relevance
 - Classifier Chains
 - Label Powerset

Multi-Class and Multi-Label Classification Revisited

- **Binary Relevance**
- The simplest technique, which treats each label as a separate binary or multi-class classification problem.
- **Example:** consider a case as shown below. X is the independent feature and Y's are the target variable.

X	Y ₁	Y ₂	Y ₃	Y ₄
x ⁽¹⁾	0	1	1	0
x ⁽²⁾	1	0	0	0
x ⁽³⁾	0	1	0	0
x ⁽⁴⁾	1	0	0	1
x ⁽⁵⁾	0	0	0	1

Multi-Class and Multi-Label Classification Revisited

- **Binary Relevance**
- **Solution:** The problem is broken into 4 different single class classification problems.

X	Y ₁	Y ₂	Y ₃	Y ₄
x ⁽¹⁾	0	1	1	0
x ⁽²⁾	1	0	0	0
x ⁽³⁾	0	1	0	0
x ⁽⁴⁾	1	0	0	1
x ⁽⁵⁾	0	0	0	1



X	Y ₁
x ⁽¹⁾	0
x ⁽²⁾	1
x ⁽³⁾	0
x ⁽⁴⁾	1
x ⁽⁵⁾	0

X	Y ₂
x ⁽¹⁾	1
x ⁽²⁾	0
x ⁽³⁾	1
x ⁽⁴⁾	0
x ⁽⁵⁾	0

X	Y ₃
x ⁽¹⁾	1
x ⁽²⁾	0
x ⁽³⁾	0
x ⁽⁴⁾	0
x ⁽⁵⁾	0

X	Y ₄
x ⁽¹⁾	0
x ⁽²⁾	0
x ⁽³⁾	0
x ⁽⁴⁾	1
x ⁽⁵⁾	1

Multi-Class and Multi-Label Classification Revisited

- **Binary Relevance**
- **Solution:** For a new data point \mathbf{x}^* , each of the labels Y_1, \dots, Y_4 is predicted separately.
- **Note1:** Any classifier (e.g. Naïve Baye's, Logistic Regression, and SVM) can be used for predicting each label
- Note 2: Each label may give rise to a binary or multi-class classification problem.



X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

X	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

X	Y_2
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	1
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

X	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

X	Y_4
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	1

Multi-Class and Multi-Label Classification Revisited

- **Classifier Chains**
- In this approach, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain.

Multi-Class and Multi-Label Classification Revisited

- **Example:** X as the input space and Y 's as the labels.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Multi-Class and Multi-Label Classification Revisited

- **Solution:** In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below. Here yellow colored is the input space and the white part represent the target variable.

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

Multi-Class and Multi-Label Classification Revisited

- **Note:** This is quite similar to binary relevance, the only difference being it forms chains in order to preserve label correlation.

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

Multi-Class and Multi-Label Classification Revisited

- [Label Powerset](#)
- This method transforms the problem into a multi-class problem with one multi-class classifier trained on all unique label combinations found in the training data.

Multi-Class and Multi-Label Classification Revisited

- **Example:** X are features, Y_1, \dots, Y_4 are labels

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

Multi-Class and Multi-Label Classification Revisited

- **Solution:** x_1 and x_4 have the same labels, similarly, x_3 and x_6 have the same set of labels. So, label powerset transforms this problem into a single multi-class problem as shown below.

The diagram illustrates the transformation of a multi-label classification matrix into a multi-class classification matrix. On the left, a 6x5 matrix represents multi-label data. The columns are labeled X, y1, y2, y3, and y4. The rows are labeled x1 through x6. The values in the matrix indicate which labels are present for each sample. An orange arrow points from this matrix to a smaller 6x2 matrix on the right, representing the transformed multi-class data. The columns are labeled X and y1. The rows are labeled x1 through x6. The values in the second column (y1) represent the class index for each sample based on its original labels.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

Multi-Class and Multi-Label Classification Revisited

- **Solution:** The label powerset method has given a unique class to every possible label combination that is present in the training set.

The diagram illustrates a transformation from a multi-label classification dataset to a single-class dataset. On the left, a table shows six data points (x1 to x6) with binary labels for four categories (y1 to y4). An orange arrow points to the right, where another table shows the same six data points, but each is now assigned a unique integer value (1 through 6) based on its label combination. This mapping represents the label powerset method, where each unique combination of labels is assigned a distinct class index.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

Multi-Class and Multi-Label Classification Revisited

- **Adapted Algorithms**
 - The most famous adapted algorithm is Multilabel kNN (MLkNN)

Multi-Class and Multi-Label Classification Revisited

- ML-kNN uses the kNN algorithm independently for each label ℓ .
- It finds the k nearest examples to the test instance and considers those that are labeled at least with ℓ as positive and the rest as negative.

Multi-Class and Multi-Label Classification Revisited

- What mainly differentiates this method from other binary relevance (BR) methods is the use of prior probabilities. ML-kNN can also rank labels.

Multi-Class and Multi-Label Classification Revisited

- **Adapted Algorithms**
 - Decision Trees can be modified to perform multi label classification.
 - The main modification is in calculating purities for each region.

Multi-Class and Multi-Label Classification Revisited

- Ensemble Algorithms
 - AdaBoost.MH and AdaBoost.MR

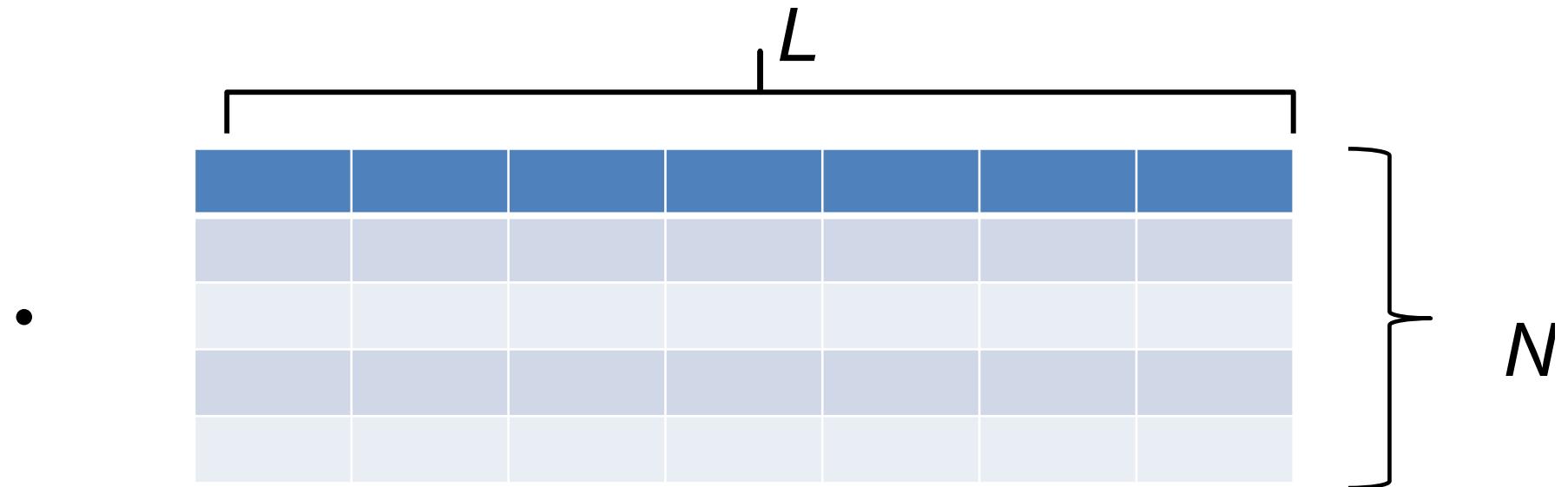
Evaluation Metrics

- One cannot simply use our normal metrics to calculate the accuracy of the predictions.
- **Accuracy score/ Exact match metric:** This function calculates subset accuracy meaning the predicted set of labels should **exactly match** with the true set of labels.

Evaluation Metrics

- **Hamming Loss:** The fraction of the wrong labels to the total number of labels, which is:

$$\frac{1}{NL} \sum_{i=1}^N \sum_{j=1}^L I(\hat{y}_{ij} \neq y_{ij})$$



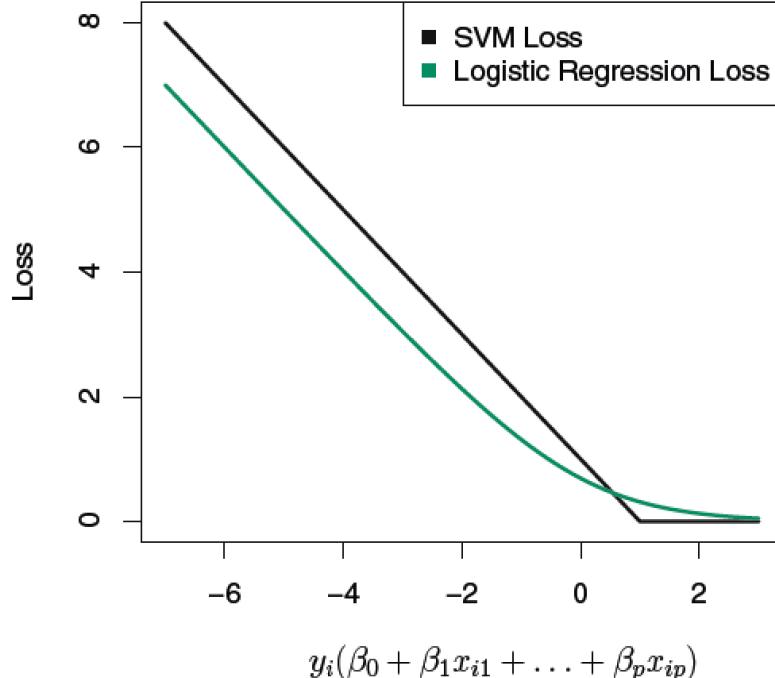
Multi-Class and Multi-Label Classification Revisited

- For an interesting overview, see:
- <https://arxiv.org/pdf/1703.08991.pdf>

SVC versus Logistic Regression?

With $f(X) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$ one can rephrase support-vector classifier optimization as

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^n \max [0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

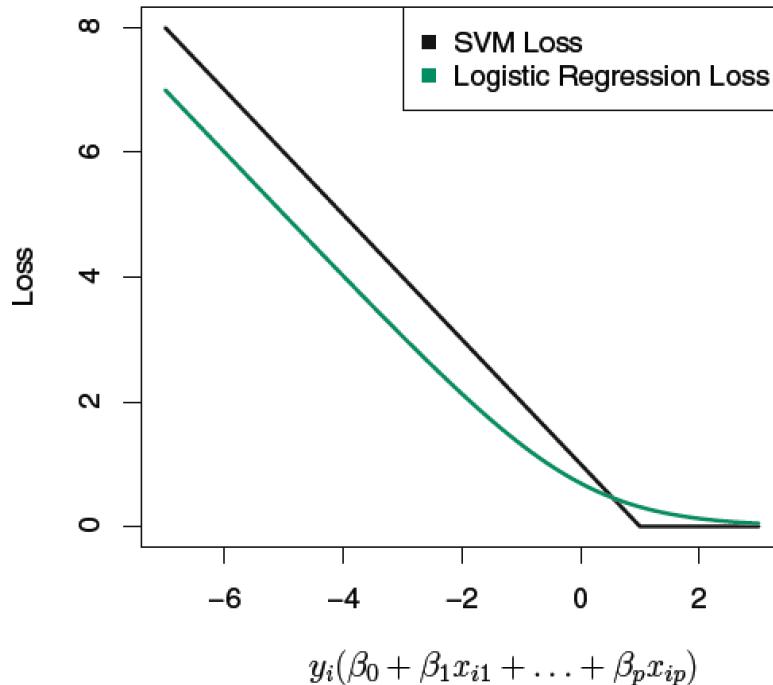


This has the form *loss plus penalty*.
The loss is known as the *hinge loss*.
Very similar to “loss” in logistic regression
(negative log-likelihood).

SVC versus Logistic Regression?

Similarly, one can rewrite logistic regression when Y is -1 or 1 (instead of 0,1) as

$$P(Y = y_i|X = x_i) = \frac{1}{1 + \exp(-y_i f(x_i))}$$



The negative log likelihood loss function is:

$$\begin{aligned} & -\log[P(Y = y_i|X = x_i)] \\ &= \log[(1 + \exp(-y_i f(x_i)))] \end{aligned}$$

Loss Plus Penalty: L1 Regularization

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^n \max [0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

The above optimization problem has the form of *loss plus penalty*.

Note that the original penalty in SVC is a ridge penalty.

One can change the penalty with L1 penalty and perform variable selection along with Support Vector Classification.

Loss Plus Penalty: L1 Regularization

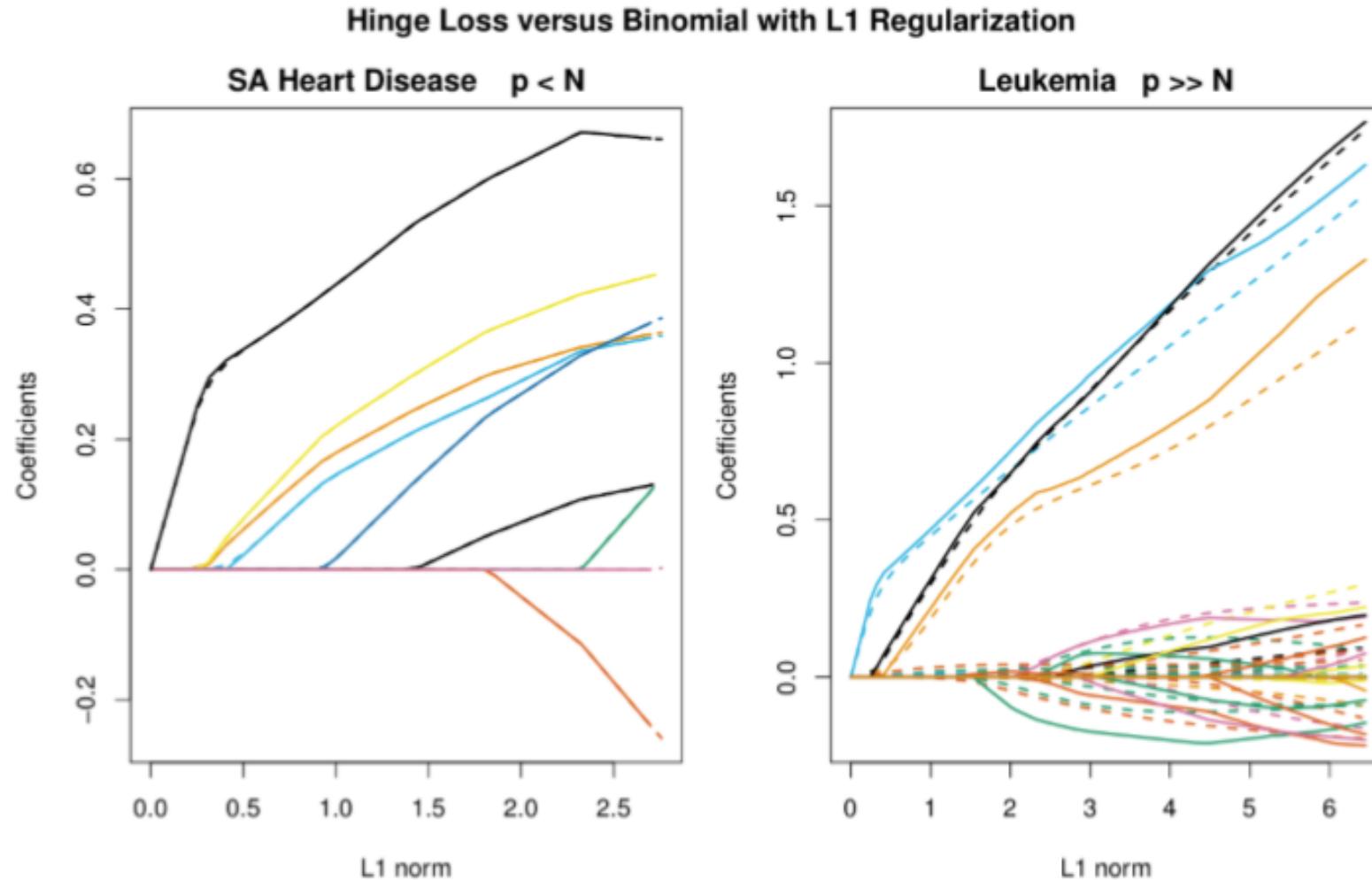
$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^n \max [0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

Because the hinge loss function is not differentiable, the solution paths for different values of λ may have jumps.

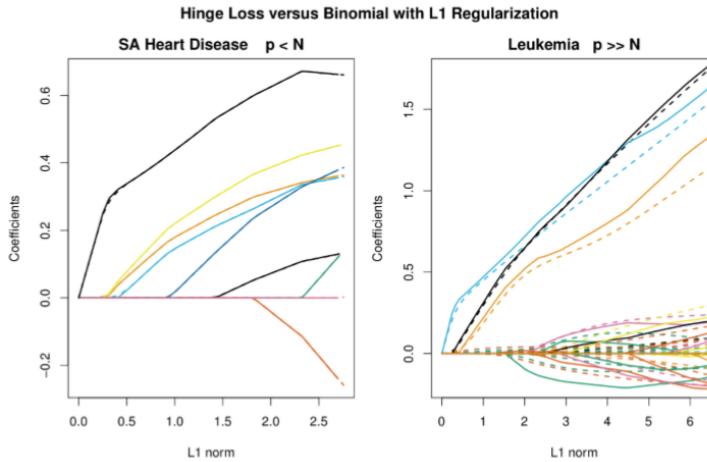
Therefore, some authors replace the hinge loss with a differentiable **squared hinge loss**.

Different combinations of hinge loss and squared hinge loss with L1 and L2 penalty result in various versions of SVM.

Comparison of L1 Regularized SVM and Logistic Regression



Details of Previous Figure



A comparison of the coefficient paths for the L1- SVM versus L1 logistic regression on two examples.

In the left we have the South African heart disease data ($N = 462$ and $p = 9$), and on the right the Leukemia data ($N = 38$ and $p = 6087$). The dashed lines are the SVM coefficients, the solid lines logistic regression. The similarity is striking in the left example, and strong in the right.

Which to use: SVM or Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.

Which to use: SVM or Logistic Regression

- SVMs are popular in high-dimensional classification problems with $p \gg n$, since the computations are $O(pn^2)$ for both linear and nonlinear kernels. (Some references say between $O(n^2)$ and $O(n^3)$).
- Additional efficiencies can be realized for linear SVMs (Shalev-Shwartz, Singer and Srebro 2007).

Which to use: SVM or Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not, LR (**with ridge penalty**) and SVM very similar.

Which one to use: SVM or Logistic Regression

- If you wish to estimate probabilities, LR is the choice.
- However, one can estimate probabilities from distances of data points from the SVC hyperplane:

Which one to use: SVM or Logistic Regression

- For nonlinear boundaries, kernel SVMs are popular, as we will see. Can use kernels with LR and Bayesian LDA as well, but computations are more expensive.
- Also, Kernels try to find a feature space in which decision boundaries are linear. That's not commensurate with characteristics of LR.

The Vapnik-Chervonenkis Dimension

- The **VC dimension** is a measure of the **capacity** (complexity, expressive power, richness, or flexibility) of a space of functions that **can be learned** by a classification algorithm.

The Vapnik-Chervonenkis Dimension

- A classification model f with some parameter vector β is said to *shatter* a set of data points $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ if for all assignments of labels to those points, there exists a β such that the model f makes no errors when evaluating that set of data points.

The Vapnik-Chervonenkis Dimension

- The **VC dimension** of a model f is the maximum number of points that can be arranged so that f shatters them.

The Vapnik-Chervonenkis Dimension

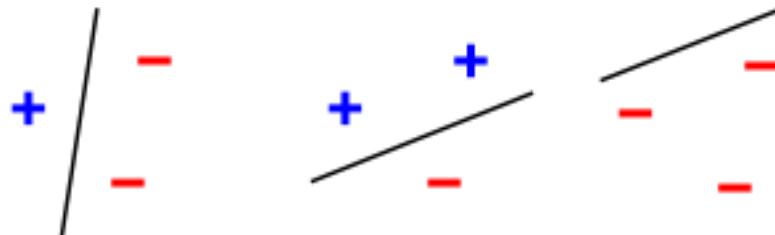
- Example: f is a straight line as a classification model on points in a two-dimensional plane (this is the model used by a perceptron).
- The line should separate positive data points from negative data points.

The Vapnik-Chervonenkis Dimension

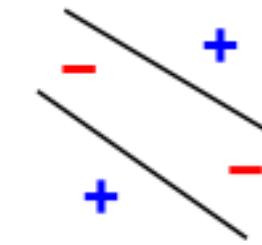
- There exist sets of 3 points that can indeed be shattered using this model (any 3 points that are not collinear can be shattered). However, no set of 4 points can be shattered. Thus, the VC dimension of this particular classifier is 3.

The Vapnik-Chervonenkis Dimension

- Note, only 3 of the $2^3 = 8$ possible label assignments are shown for the three points.



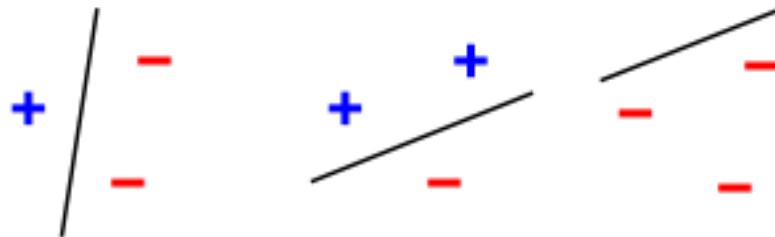
3 points shattered



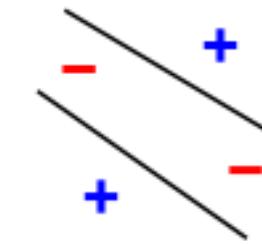
4 points impossible

The Vapnik-Chervonenkis Dimension

- Note, only 3 of the $2^3 = 8$ possible label assignments are shown for the three points.



3 points shattered



4 points impossible

The Vapnik-Chervonenkis Dimension

- The VC Dimension of affine classifiers of the form $f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x} + \beta_0$, $\boldsymbol{\beta} \in \mathbb{R}^p$ is $p+1$.
- This includes SVC, a linear SVM.
- The VC Dimension of an SVM equipped with an RBF kernel is infinite.

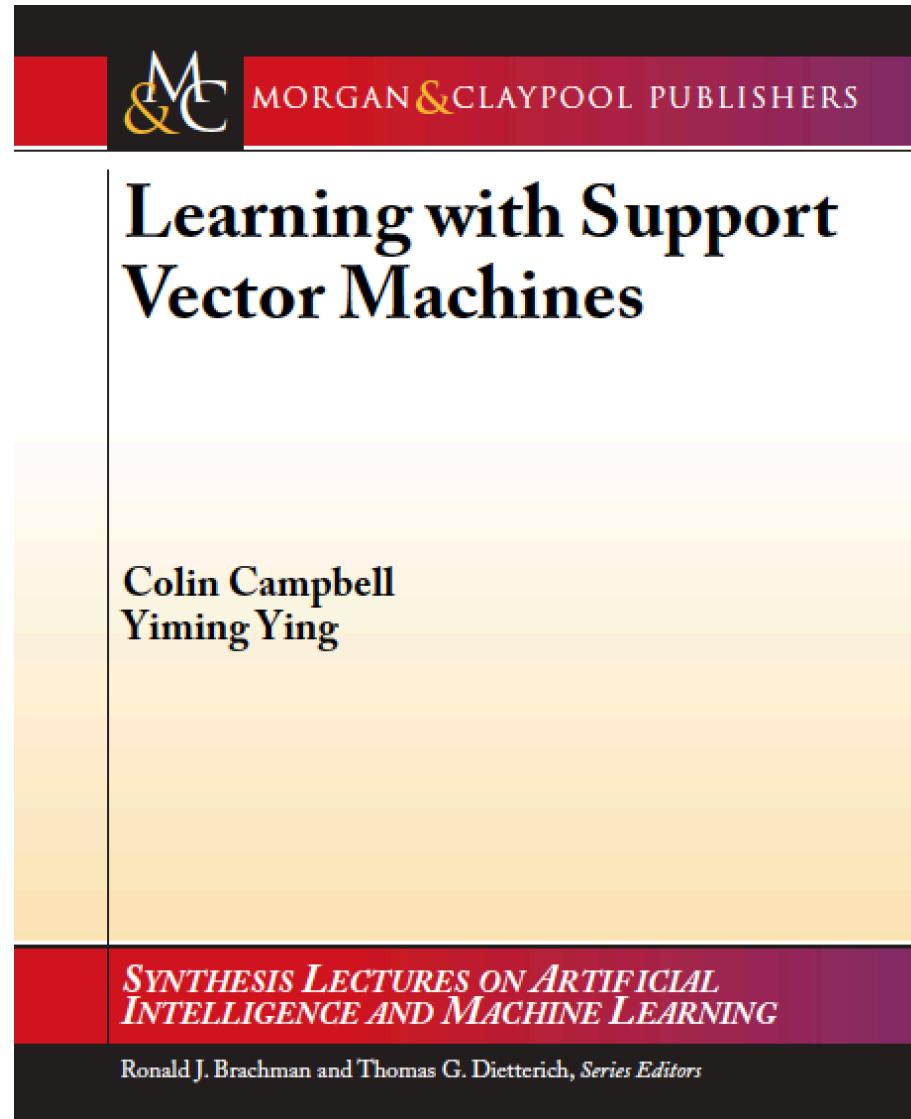
Support Vector Regression

- There is an extension of the SVM for regression, called support vector regression (SVR) .
- We saw that least squares regression seeks coefficients $\beta_0, \beta_1, \dots, \beta_p$ such that the sum of squared residuals is as small as possible.

Support Vector Regression

- Support vector regression instead seeks coefficients that minimize a different type of loss, where only residuals larger in absolute value than some positive constant contribute to the loss function.
- This is an extension of the margin used in support vector classifiers to the regression setting.

SVM Learning



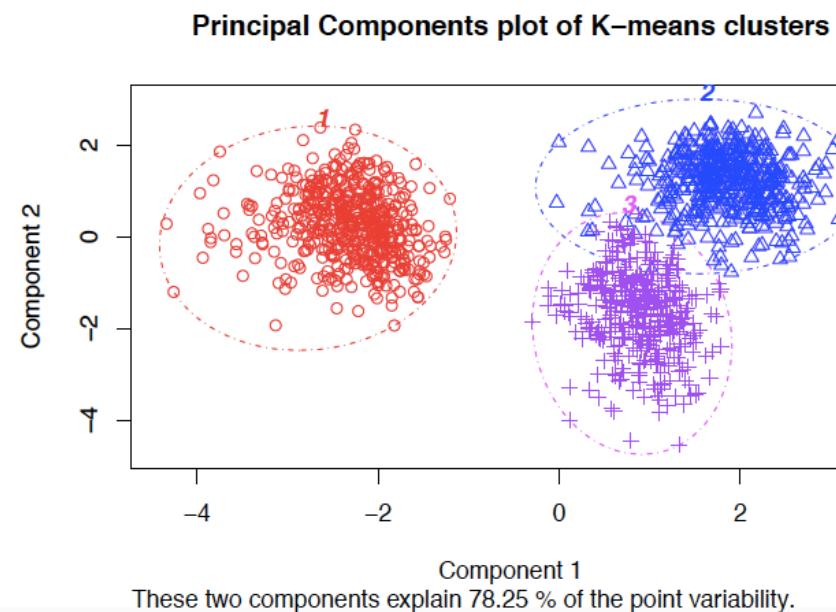
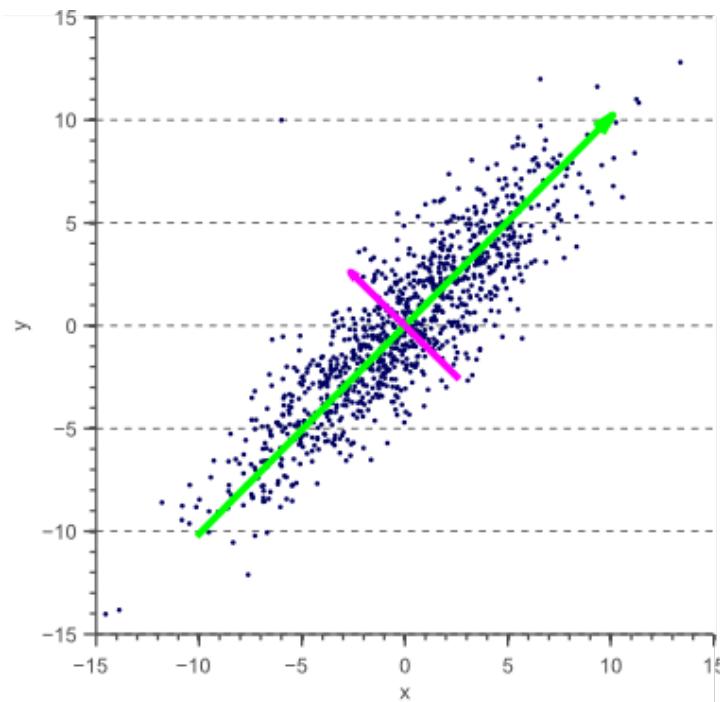
DSCI 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 8

Unsupervised Learning



Unsupervised Learning

Unsupervised vs Supervised Learning:

- Most of this course focuses on *supervised learning* methods such as regression and classification.
- In that setting we observe both a set of features X_1, X_2, \dots, X_p for each object, as well as a response or outcome variable Y . The goal is then to predict Y using X_1, X_2, \dots, X_p .

Unsupervised Learning

Unsupervised vs Supervised Learning:

- Here we instead focus on *unsupervised learning*, where we observe only the features X_1, X_2, \dots, X_p .
- We are not interested in prediction, because we do not have an associated response variable Y .

The Goals of Unsupervised Learning

- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We discuss two methods:
 - *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied, and
 - *clustering*, a broad class of methods for discovering unknown **subgroups** in data.

The Challenge of Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of growing importance in a number of fields:
 - subgroups of breast cancer patients grouped by their gene expression measurements,
 - groups of shoppers characterized by their browsing and purchase histories,
 - movies grouped by the ratings assigned by movie viewers.

Another advantage

- It is often easier to obtain *unlabeled data* — from a lab instrument or a computer — than *labeled data*, which can require human intervention.
- For example it is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?

Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other.

Clustering

- We must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

Clustering for Market Segmentation

- Suppose we have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people.

Clustering for Market Segmentation

- Our goal is to perform *market segmentation* by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.

Clustering for Market Segmentation

- The task of performing market segmentation amounts to clustering the people in the data set.

Two clustering methods

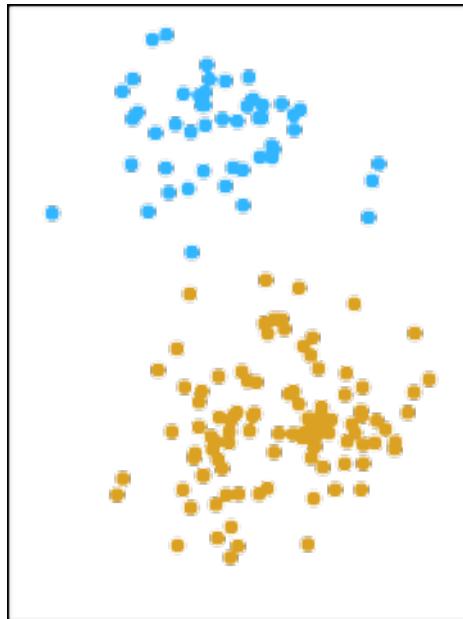
- In *K-means clustering*, we seek to partition the observations into a pre-specified number of clusters.

Two clustering methods

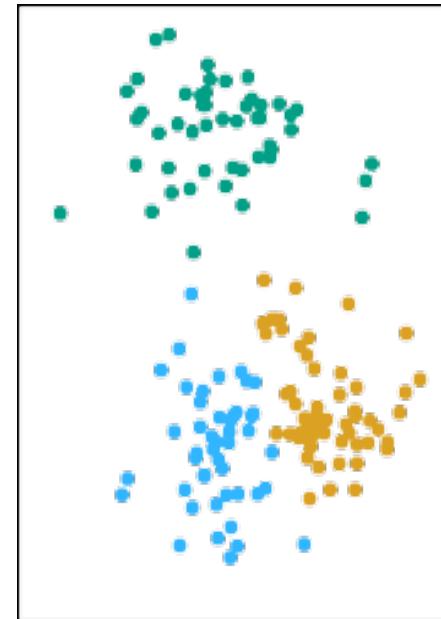
- In *hierarchical clustering*, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n .

K -means clustering

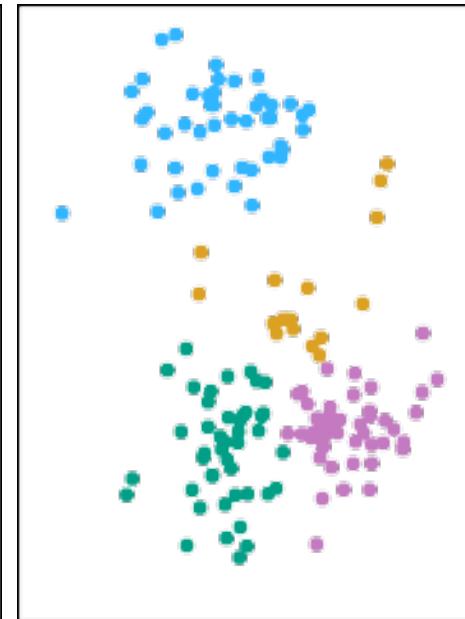
$K=2$



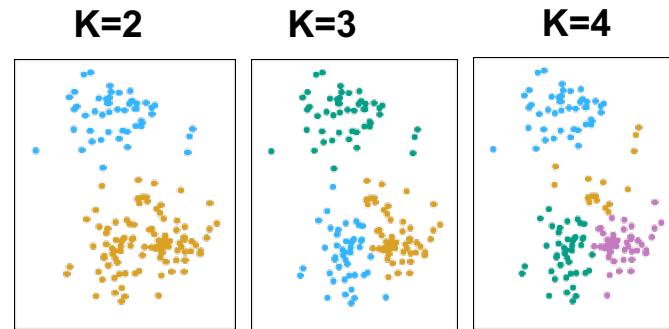
$K=3$



$K=4$



K -means clustering



A simulated data set with 150 observations in 2-dimensional space. Panels show the results of applying K-means clustering with different values of K , the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.

Details of K -means clustering

Let C_1, \dots, C_K denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_{k'} = \emptyset$ for all distinct k and k' . In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the i^{th} observation is in the k^{th} cluster, then $i \in C_k$.

Details of K -means clustering: continued

- The idea behind K -means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.

Details of K -means clustering: continued

- The within-cluster variation for cluster C_k is a measure $\text{WCV}(C_k)$ of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \text{WCV}(C_k) \right\}$$

Details of K -means clustering: continued

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

How to define within-cluster variation?

- Typically we use Euclidean distance

$$\text{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$$

where $|C_k|$ denotes the number of observations in the k^{th} cluster.

How to define within-cluster variation?

- Combining the above equations gives the optimization problem that defines K-means clustering,

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\}$$

K-Means Clustering Algorithm

1. Randomly assign a number, from 1 to K , to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
 1. For each of the K clusters, compute the cluster *centroid*.
The k^{th} cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster.
 2. Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

Properties of the Algorithm

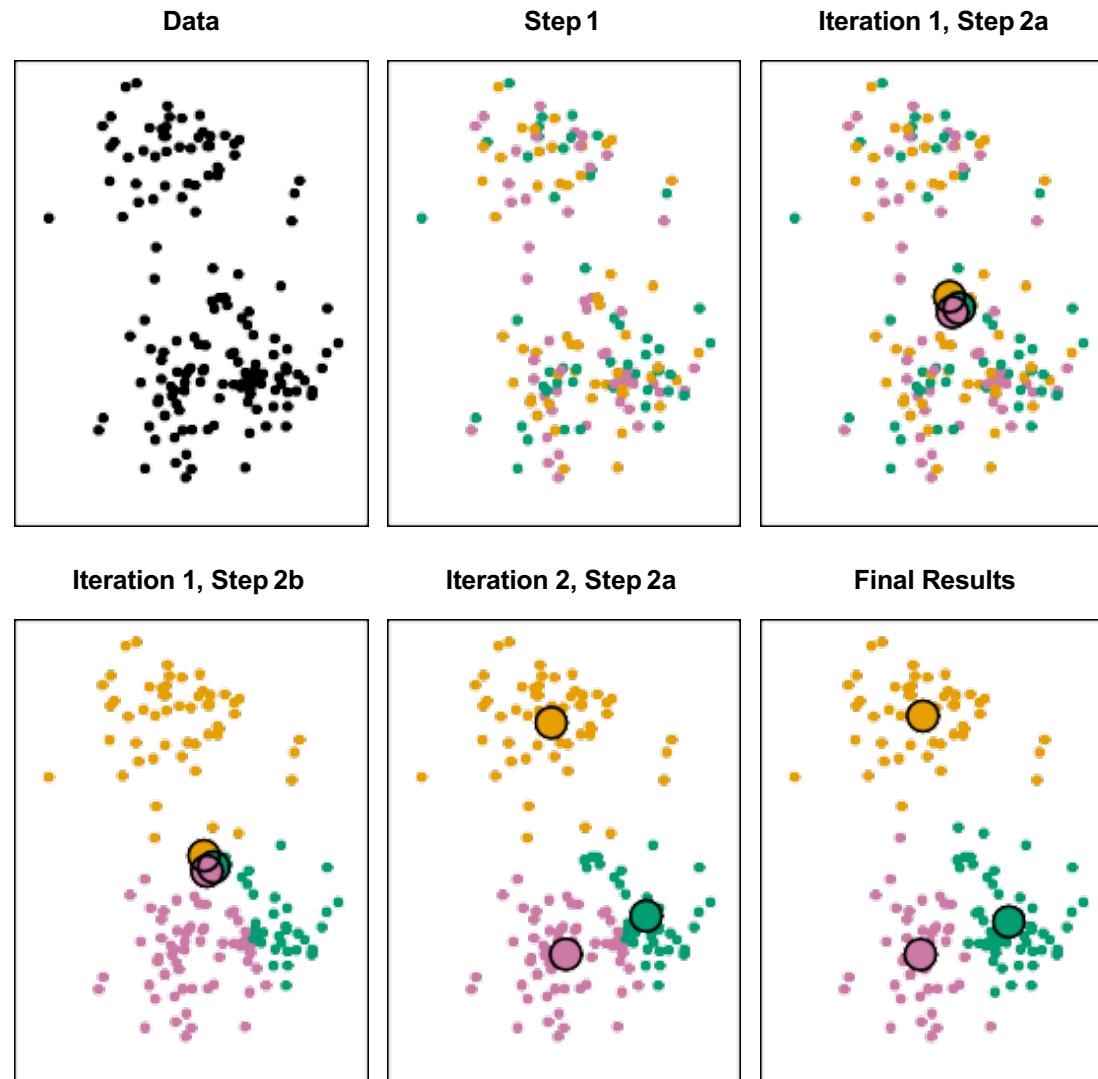
- This algorithm is guaranteed to decrease the value of the objective function at each step.
Why? Note that

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2$$

where $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$ is the mean for feature j in cluster C_k .

- however it is not guaranteed to give the global minimum. *Why not?*

Example



Details of Previous Figure

The progress of the K-means algorithm with $K=3$.

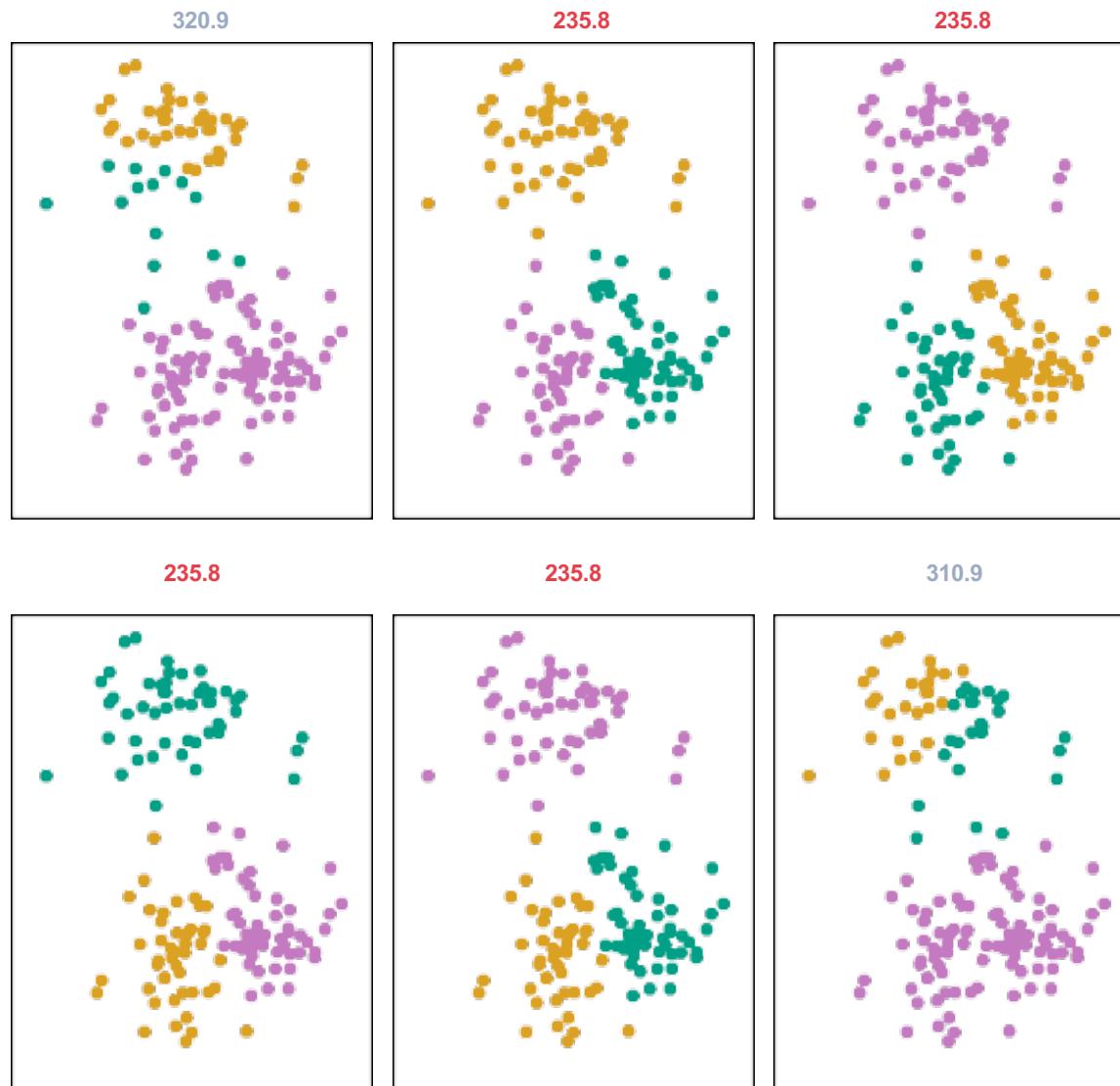
- *Top left:* The observations are shown.
- *Top center:* In Step 1 of the algorithm, each observation is randomly assigned to a cluster.
- *Top right:* In Step 2(a), the cluster centroids are computed. These are shown as large colored disks. Initially the centroids are almost completely overlapping because the initial cluster assignments were chosen at random.

Details of Previous Figure

The progress of the K-means algorithm with $K=3$.

- *Bottom left:* In Step 2(b), each observation is assigned to the nearest centroid.
- *Bottom center:* Step 2(a) is once again performed, leading to new cluster centroids.
- *Bottom right:* The results obtained after 10 iterations.

Example: different starting values



Details of Previous Figure

K -means clustering performed six times on the data from previous figure with $K = 3$, each time with a different random assignment of the observations in Step 1 of the K -means algorithm.

Details of Previous Figure

Above each plot is the value of the objective

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\}$$

Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters.

Those labeled in red all achieved the same best solution, with an objective value of 235.8

K-Medoids Clustering

- Similar to K-means, but in each step we do not compute the centroid of each cluster.
- We compute the **medoid**, which is a data point that has the smallest average dissimilarity with other data points in the cluster.
- This way you limit the center of your cluster to be one of your data points.

Hierarchical Clustering

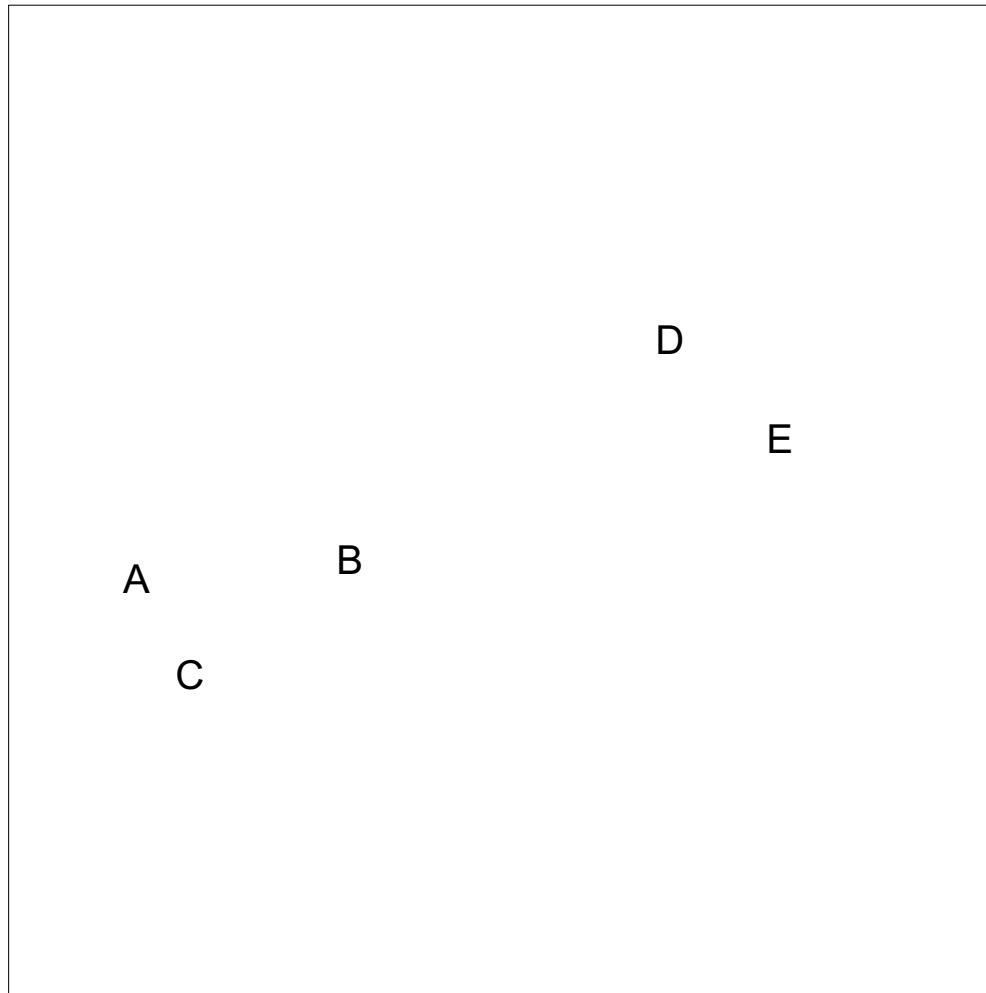
- K -means clustering requires us to pre-specify the number of clusters K . This can be a disadvantage (later we discuss strategies for choosing K)
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of K .

Hierarchical Clustering

- In this section, we describe *bottom-up* or *agglomerative* clustering. This is the most common type of hierarchical clustering, and refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

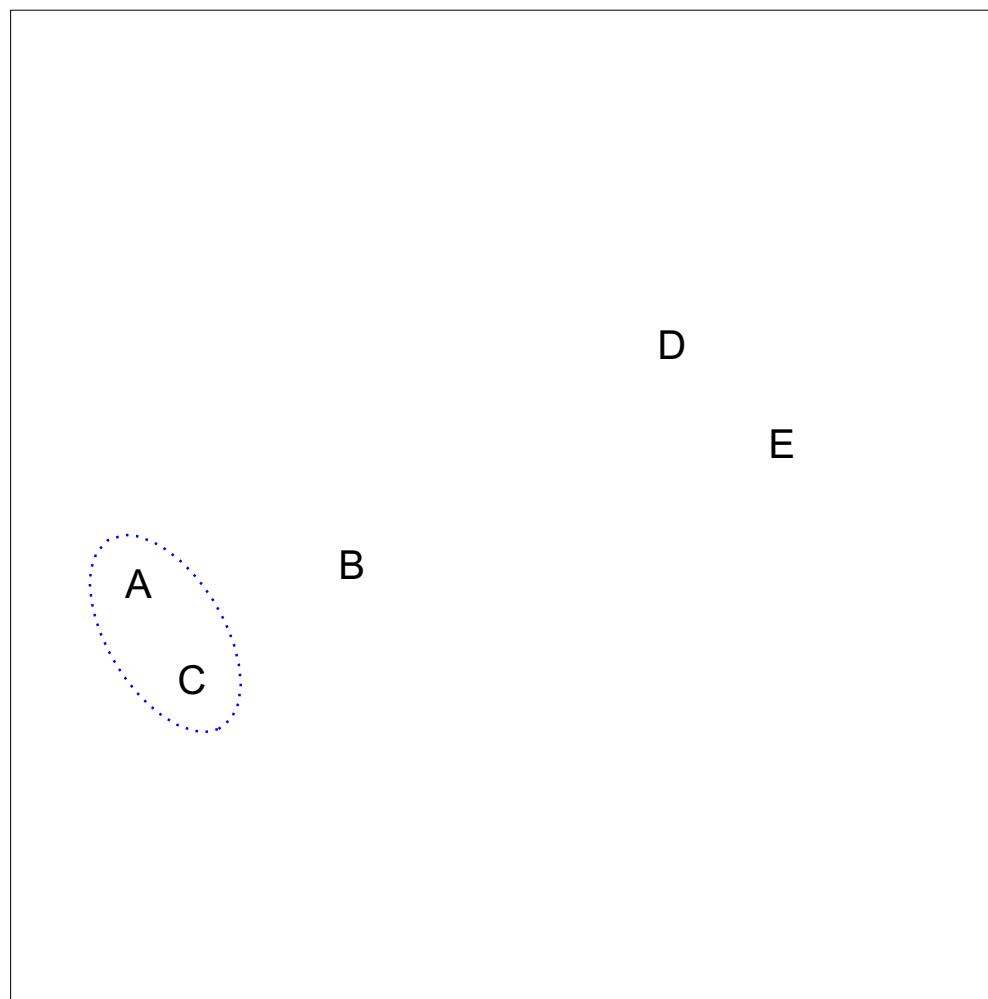
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



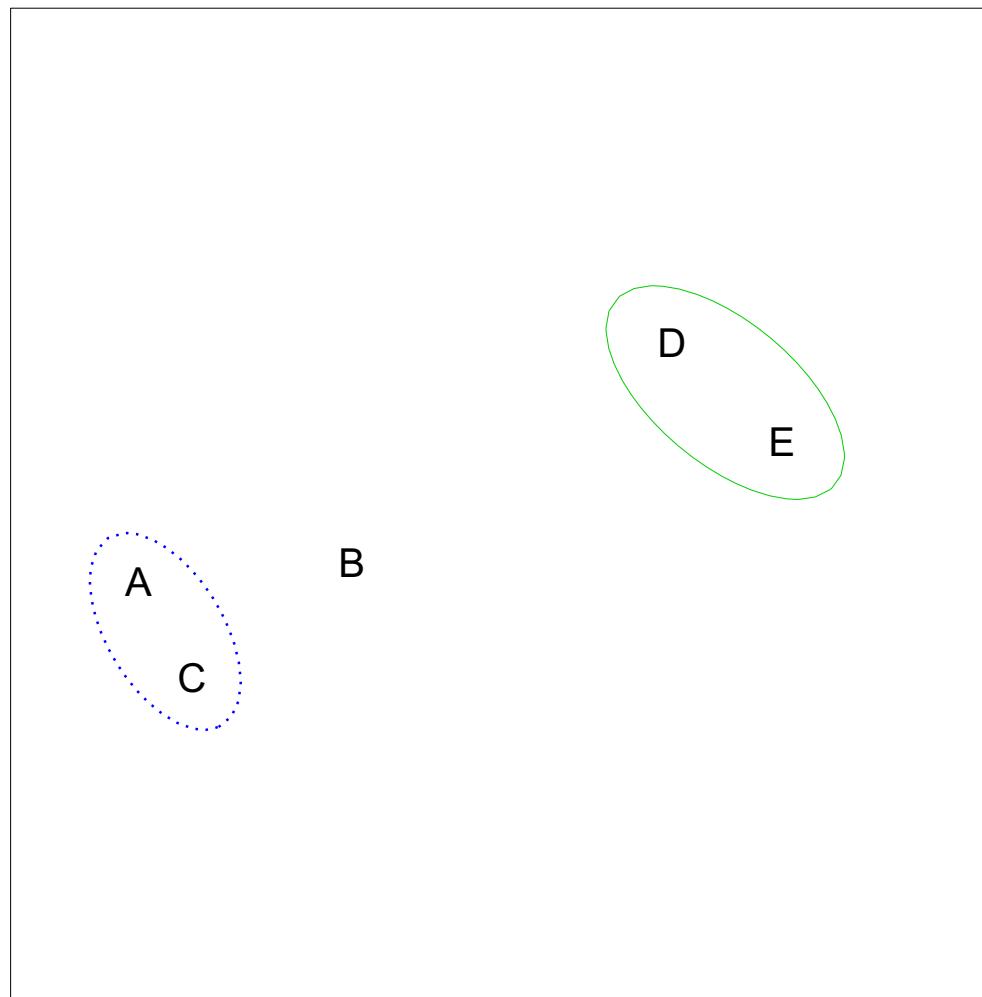
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



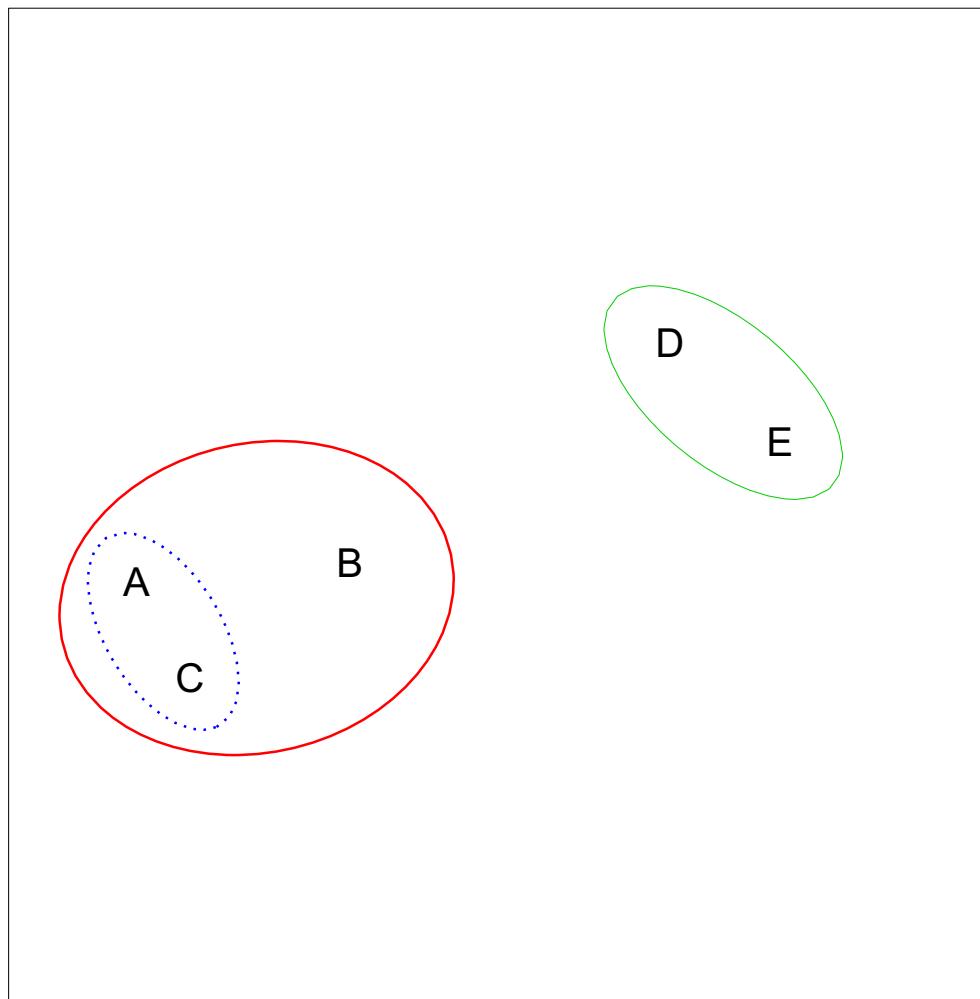
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



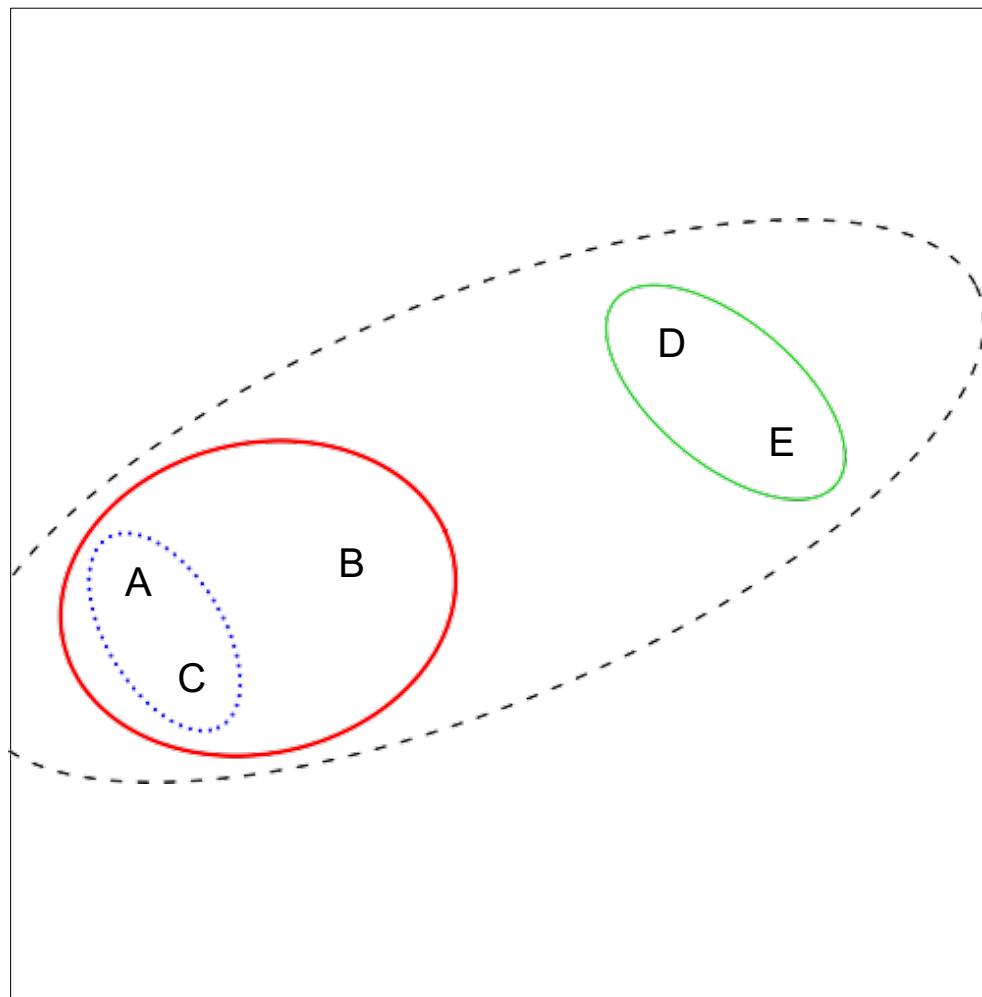
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



Hierarchical Clustering: the idea

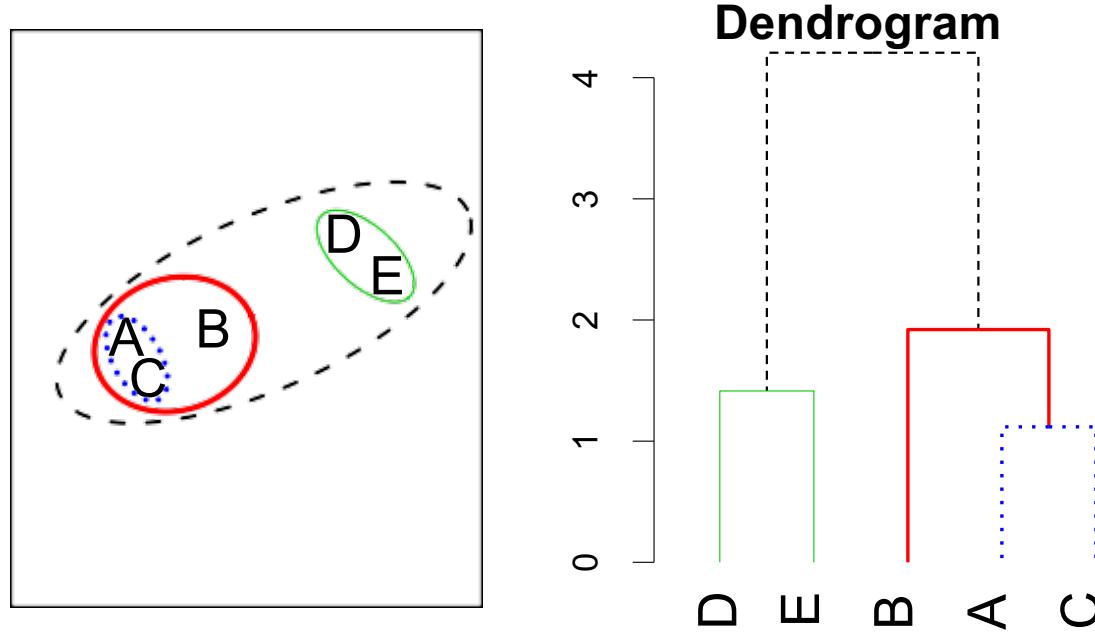
Builds a hierarchy in a “bottom-up” fashion...



Hierarchical Clustering Algorithm

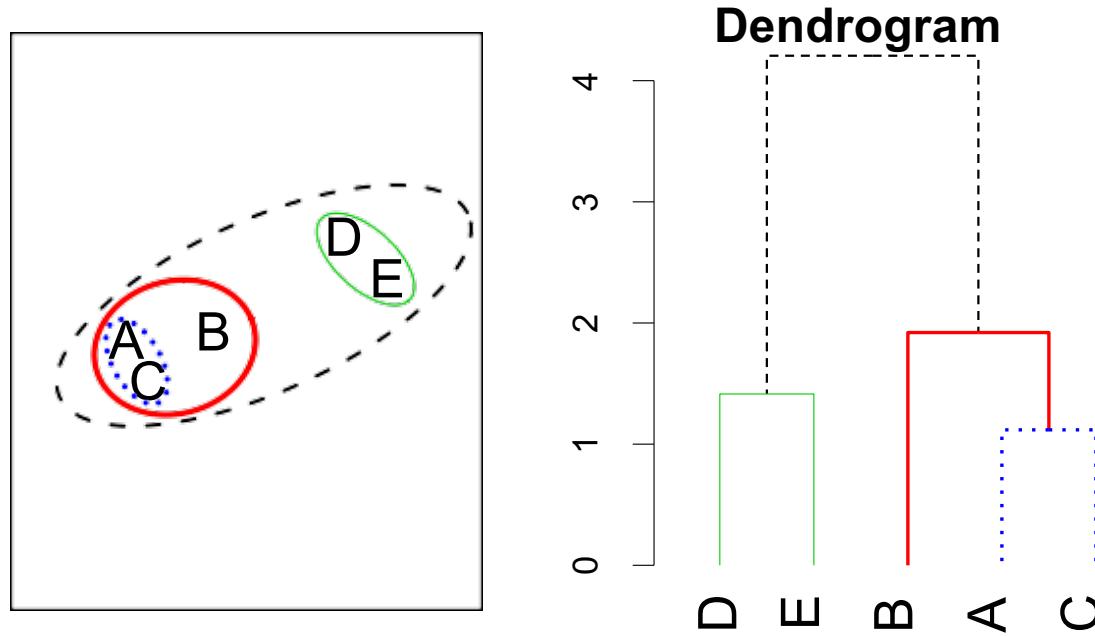
The approach in words:

- Start with each point in its own cluster.
- Identify the **closest** two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

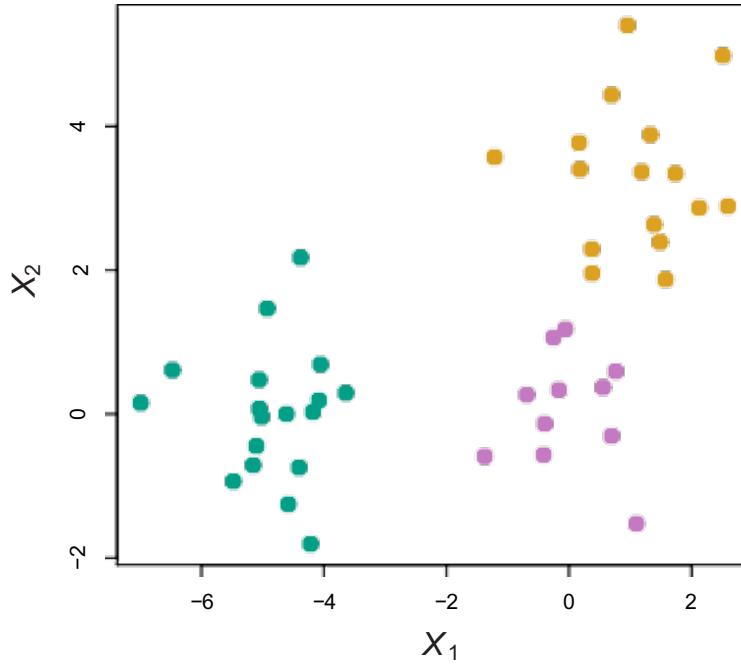


Hierarchical Clustering Algorithm

The height of each node in the plot is proportional to the value of the intergroup dissimilarity between its two daughters



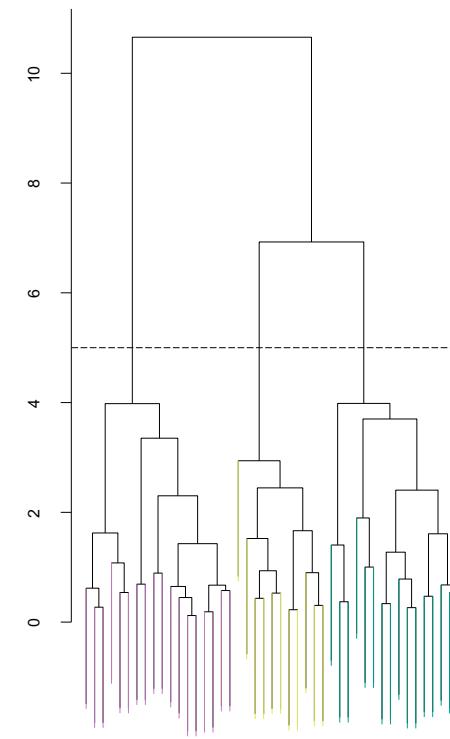
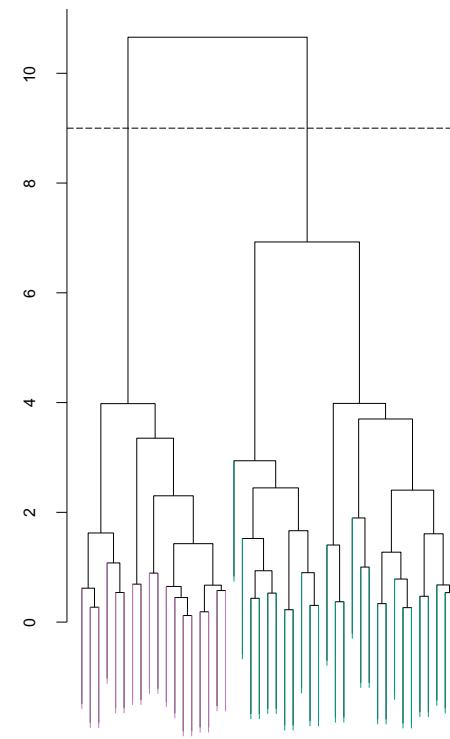
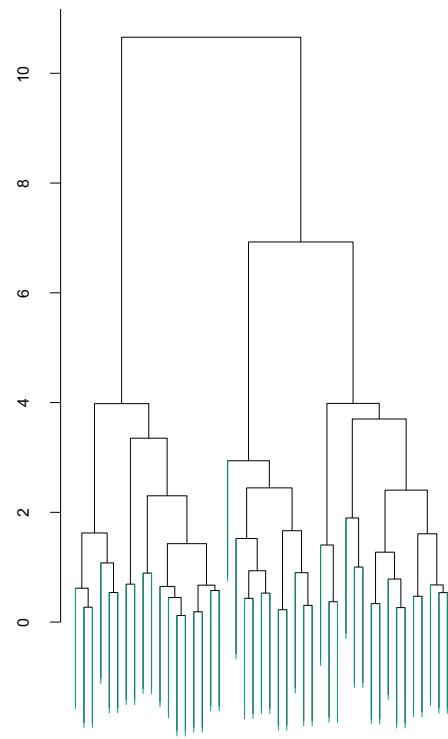
An Example



45 observations generated in 2-dimensional space. In reality there are three distinct classes, shown in separate colors.

However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

Application of hierarchical clustering



Details of previous figure

- *Left*: Dendrogram obtained from hierarchically clustering the data from previous slide, with complete linkage and Euclidean distance.
- *Center*: The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.

Details of previous figure

- *Right:* The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in different colors.
- Note that the colors were not used in clustering, but are simply used for display purposes in this figure

Algorithm

Algorithm 10.2 *Hierarchical Clustering*

1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n - 1)/2$ 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. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
 - (b) Compute the new pairwise inter-cluster dissimilarities among the $i - 1$ remaining clusters.
-

Nested Clusters

- The term hierarchical refers to the fact that clusters obtained by cutting the dendrogram at a given height are necessarily **nested** within the clusters obtained by cutting the dendrogram at any greater height.
- However, on an arbitrary data set, this assumption of hierarchical structure might be unrealistic
- Hierarchical clustering can sometimes yield worse (i.e. less accurate) results than K - means clustering for a given number of clusters

Nested Clusters: Example

- Observations: a 50–50 split of males and females, evenly split among Americans, Japanese, and French.
- The best division into two groups might split these people by gender.
- The best division into three groups might split them by nationality.

Nested Clusters: Example

- The true clusters are not nested
- The best division into three groups does not result from taking the best division into two groups and splitting up one of those groups.
- This situation could not be well-represented by hierarchical clustering.

Dissimilarity between Groups

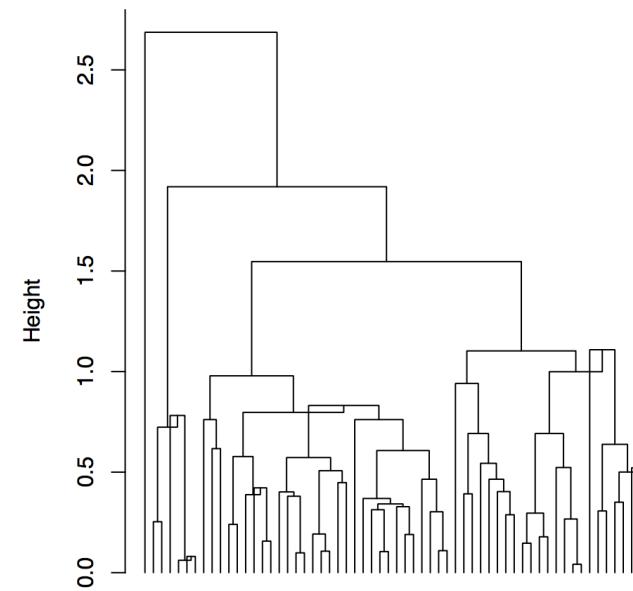
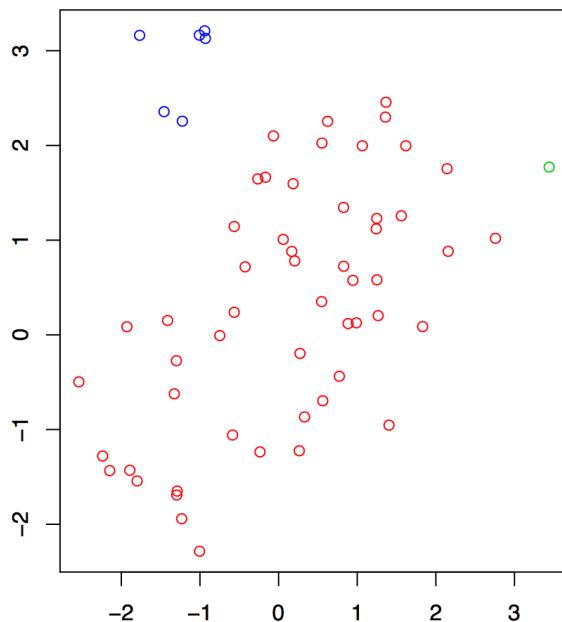
- The concept of **dissimilarity** between a pair of observations needs to be extended to a pair of groups of observations .
- This extension is achieved by developing the notion of **linkage**, which defines the dissimilarity between two groups of observations. The four most common types of linkage—**complete**, **average**, **single**, and **centroid**

Types of Linkage

<i>Linkage</i>	<i>Description</i>
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>largest</i> of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>smallest</i> of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>average</i> of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable <i>inversions</i> .

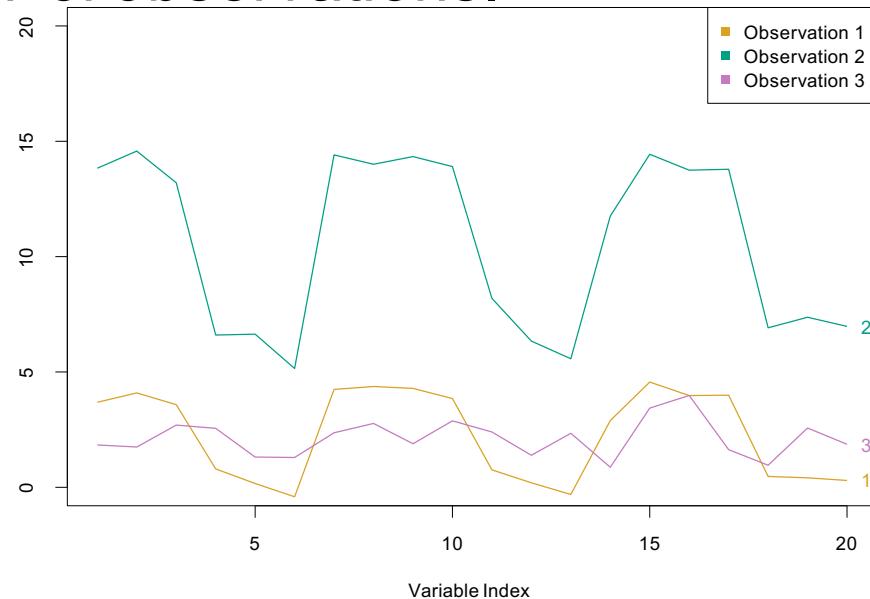
Types of Linkage

- For centroid dissimilarity, **inversion** can occur, whereby two clusters are fused at a height below either of the individual clusters in the dendrogram. This can lead to difficulties in visualization as well as in interpretation of the dendrogram.



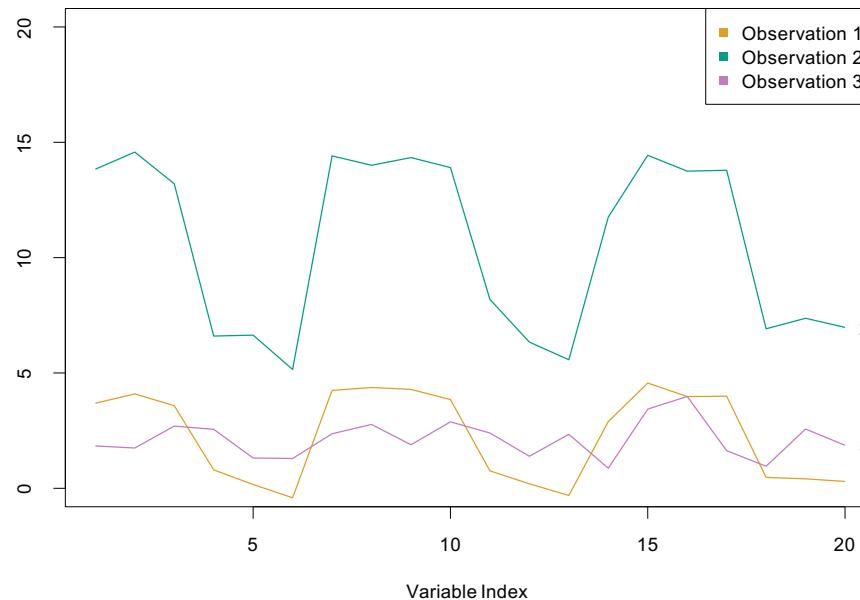
Choice of Dissimilarity Measure

- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.



Choice of Dissimilarity Measure

- Correlation-based distance focuses on the shapes of observation profiles rather than their magnitudes.
- Observations 1, 3 have small Euclidean distance but weak correlation (low correlation similarity).
- Observations 1, 2 have large Euclidean distance but strong correlation (high correlation similarity).



Practical issues

- Should the observations or features first be standardized in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering,
 - What dissimilarity measure should be used?
 - What type of linkage should be used?
- How many clusters to choose? (in both K -means or hierarchical clustering). Difficult problem. No agreed-upon method. See Elements of Statistical Learning, chapter 13 for more details.

How many clusters?

- Sometimes, we might have no problem specifying the number of clusters K ahead of time, e.g.,
 - Segmenting a client database into K clusters for K salesmen
- Other times, K is implicitly defined by cutting a hierarchical clustering tree at a given height
- In most exploratory applications, K is unknown.
- What is the “right” value of K ?

This is a hard problem

Determining the number of clusters is a
hard problem!

Why is it hard?

- Determining the number of clusters is a hard task for humans to **perform** (unless the data are low-dimensional). Not only that, it's just as hard to **explain** what it is we're looking for. Usually, statistical learning is successful when at least one of these is possible

This is a hard problem

- Why is it important?
 - E.g., it might mean a big difference scientifically if we were convinced that there were $K = 2$ subtypes of breast cancer vs. $K = 3$ subtypes
 - One of the (larger) goals of data mining/statistical learning is automatic inference; choosing K is certainly part of this.

Reminder: within-cluster variation

We focus on K-means, but most ideas apply to other settings

Recall: given the number of clusters K , the K -means algorithm approximately minimizes the within-cluster variation:

$$W = \sum_{k=1}^K \sum_{C(i)=k} \|X_i - \bar{X}_k\|_2^2$$

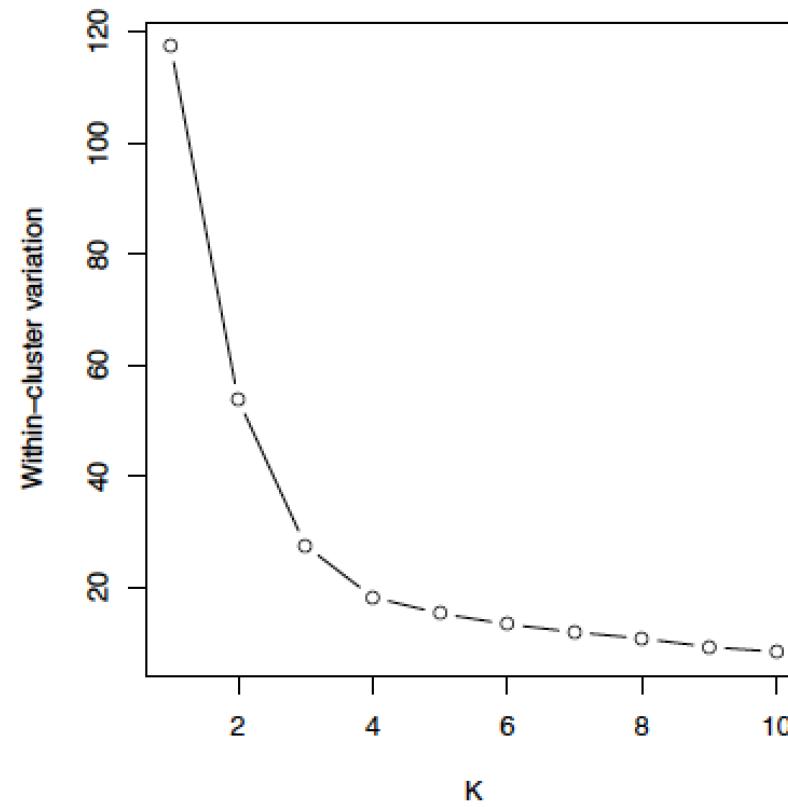
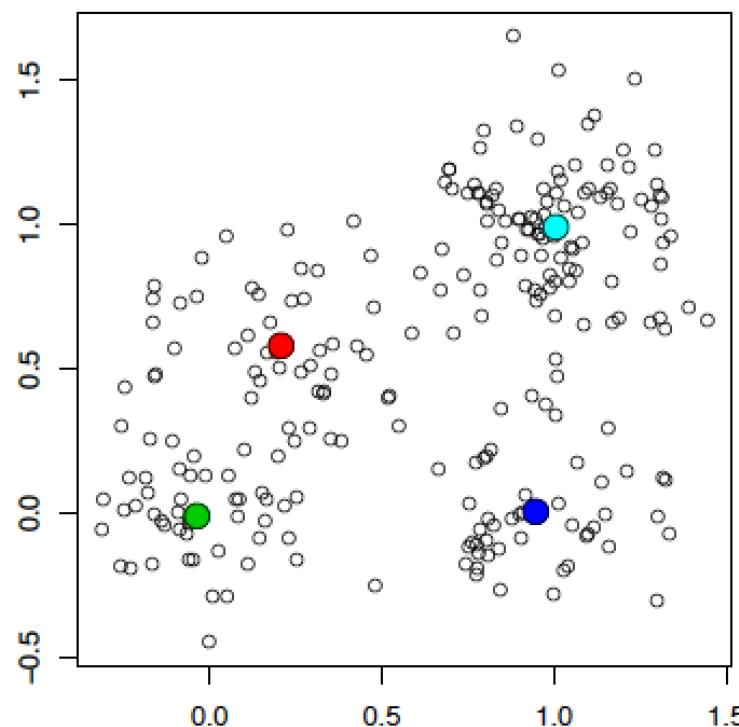
over clustering assignments C , where \bar{X}_k is the average of points in group k

Clearly a lower value of W is better. So why not just run K -means for a bunch of different values of K , and choose the value of K that gives the smallest $W(K)$?

That's not going to work

Problem: within-cluster variation just keeps decreasing: **scree plot**

Example: $n = 250$, $p = 2$, $K = 1, \dots, 10$



Between-cluster variation

Within-cluster variation measures how **tightly grouped** the clusters are. As we increase the number of clusters K , this just keeps going down. What are we missing?

Between-cluster variation measures how spread apart the groups are from each other:

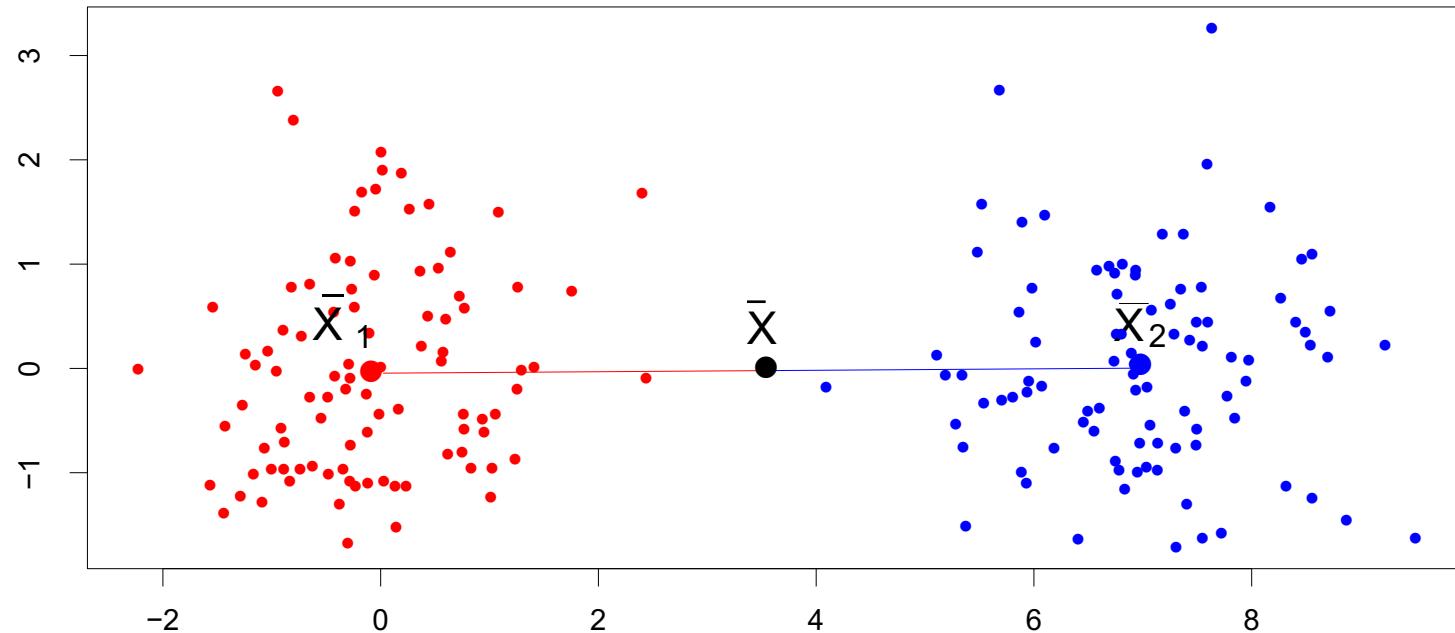
$$B = \sum_{k=1}^K n_k \|\bar{X}_k - \bar{X}\|_2^2$$

where as before \bar{X}_k is the average of points in group k , and \bar{X} is the overall average, i.e.

$$\bar{X}_k = \frac{1}{n_k} \sum_{C(i)=k} X_i \quad \text{and} \quad \bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

Example: between-cluster variation

Example: $n = 100$, $p = 2$, $K = 2$



$$B = n_1 \|\bar{X}_1 - \bar{X}\|_2^2 + n_2 \|\bar{X}_2 - \bar{X}\|_2^2$$

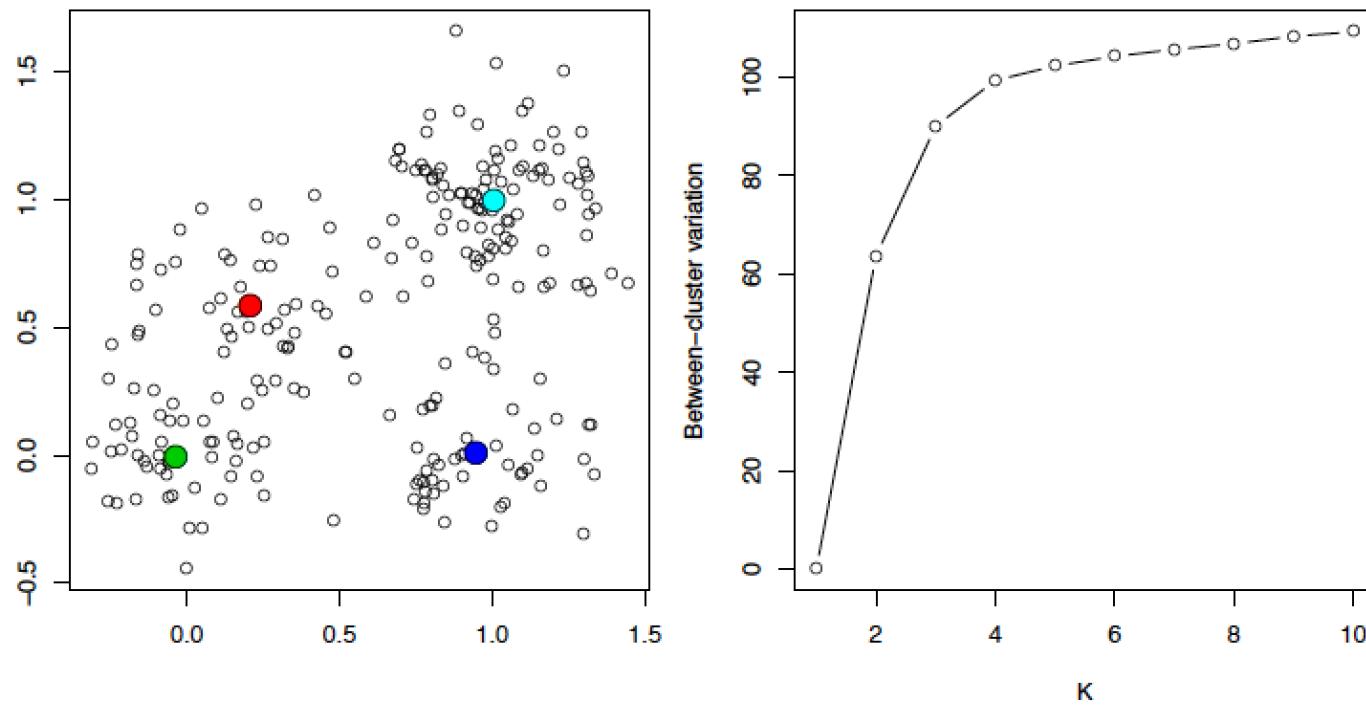
$$W = \sum_{C(i)=1} \|X_i - \bar{X}_1\|_2^2 + \sum_{C(i)=2} \|X_i - \bar{X}_2\|_2^2$$

Still not going to work

Bigger B is better, can we use it to choose K ?

Problem: between- cluster variation just keeps increasing

Running example: $n = 250$, $p = 2$, $K = 1, \dots, 10$



CH index

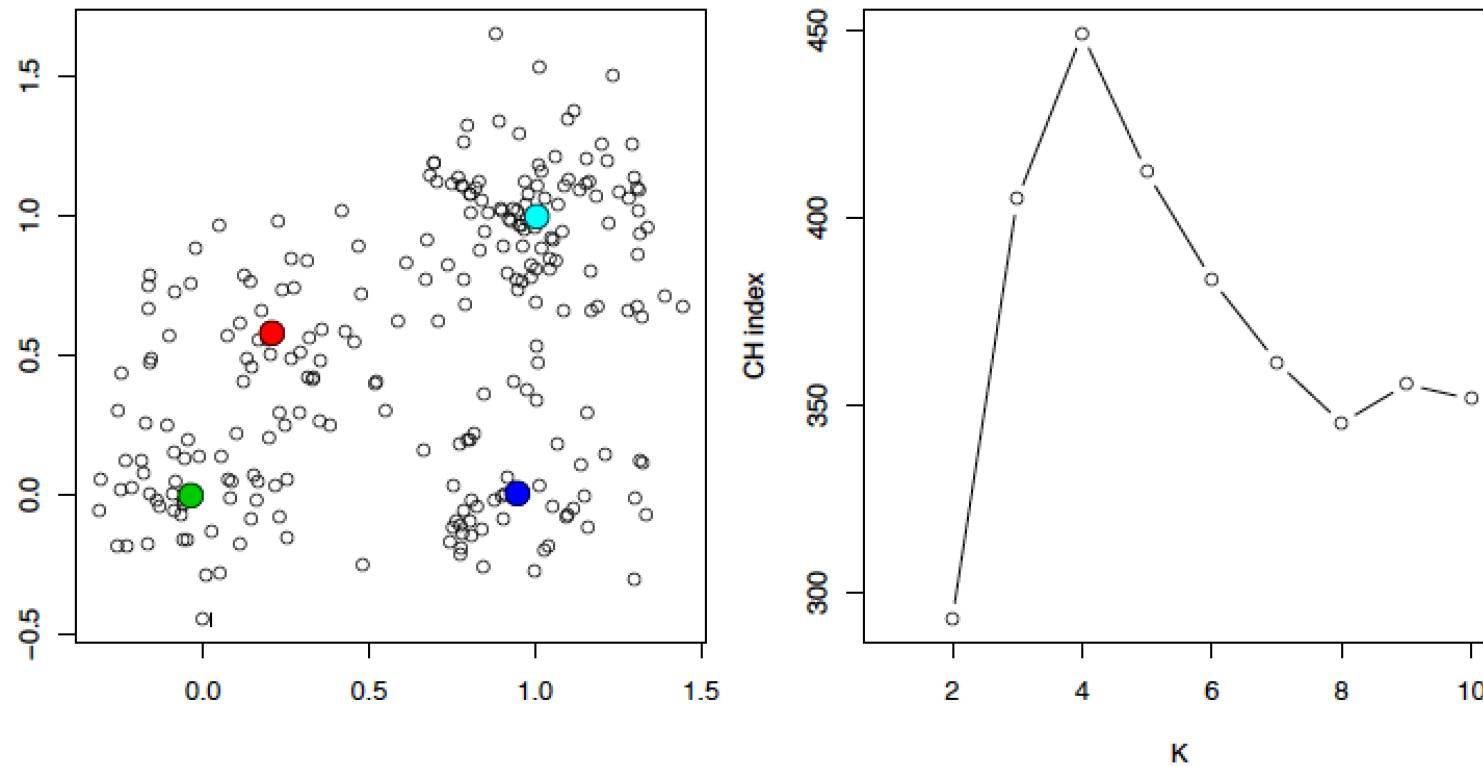
- Ideally we'd like our clustering assignments C to **simultaneously** have a small W and a large B
- This is the idea behind the **CH index** proposed by Calinski and Harabasz (1974), in “A dendrite method for cluster analysis”
- For clustering assignments coming from K clusters, we record CH score:

$$CH(K) = \frac{B(K)/(K - 1)}{W(K)/(n - K)}$$

- To choose K , just pick some maximum number of clusters to be considered K_{\max} (e.g., $K = 20$), and choose the value of K with the largest score $CH(K)$.

Example: CH index

Running example: $n = 250$, $p = 2$, $K = 2, \dots, 10$.



We would choose $K = 4$ clusters, which seems reasonable

General problem: the CH index is not defined for $K = 1$. We could never choose just one cluster (the null model)!

Gap statistic

It's true that $W(K)$ keeps dropping, but
how much it drops at any one K should
be informative

The gap statistic was introduced by
Tibshirani et al. (2001), “Estimating the
number of clusters in a data set via the
gap statistic”

Gap statistic

It is based on this idea. We compare the observed within-cluster variation $W(K)$ to $W_{\text{unif}}(K)$, the within-cluster variation we'd see if we instead had points distributed uniformly (over an *encapsulating box*). The gap for K clusters is defined as

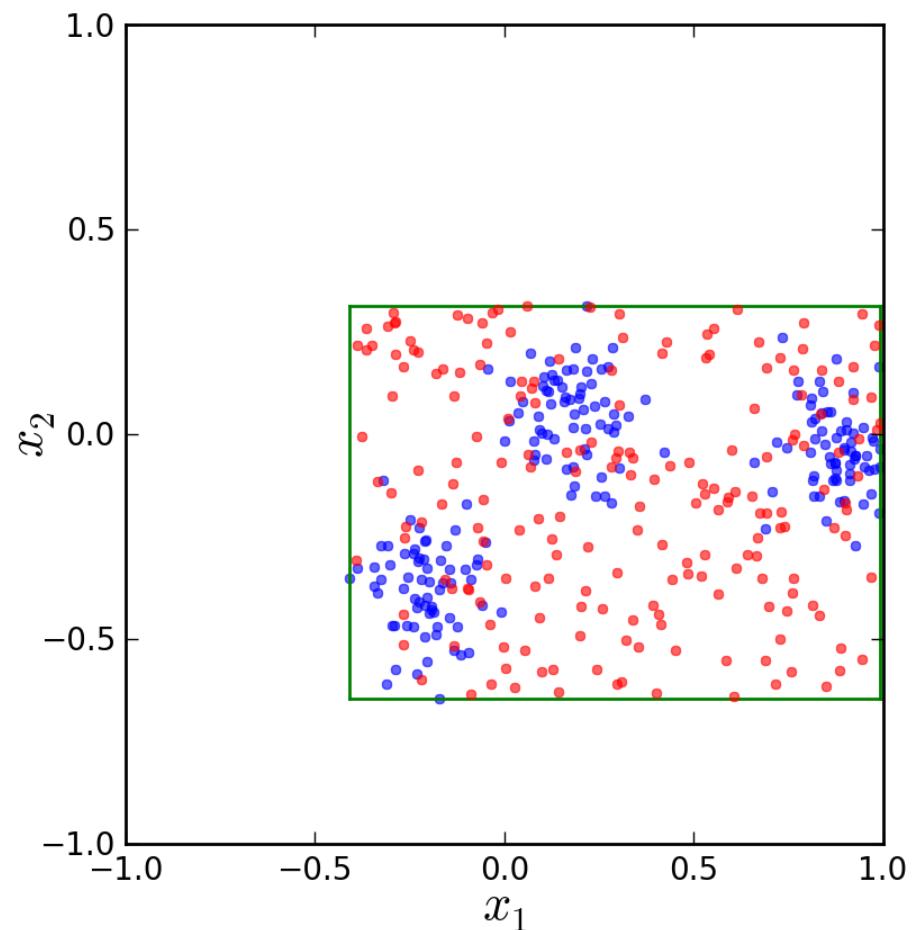
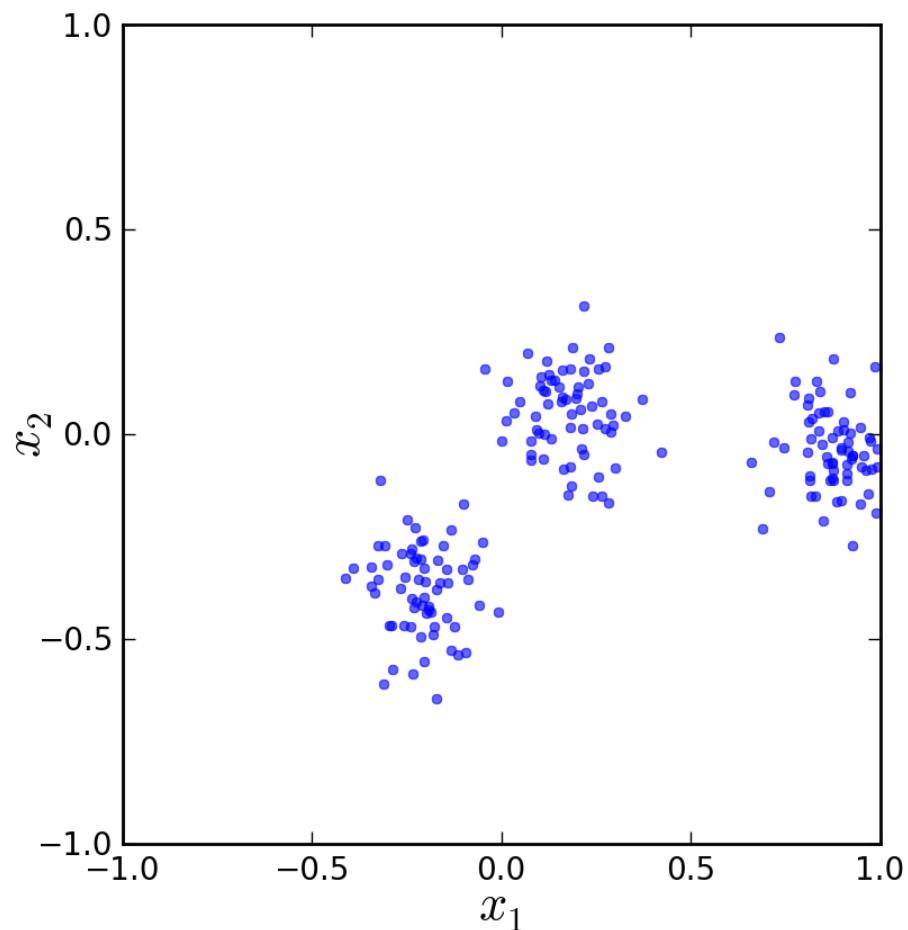
$$\text{Gap}(K) = \log W_{\text{unif}}(K) - \log W(K)$$

Gap statistic

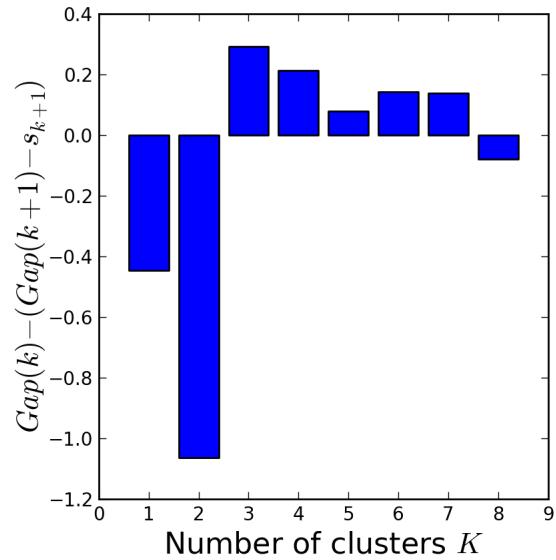
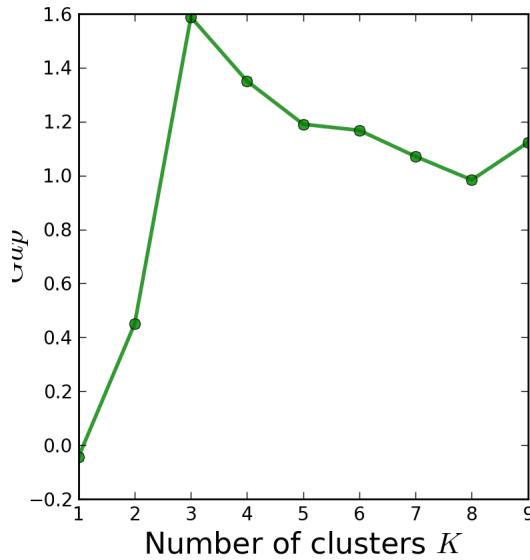
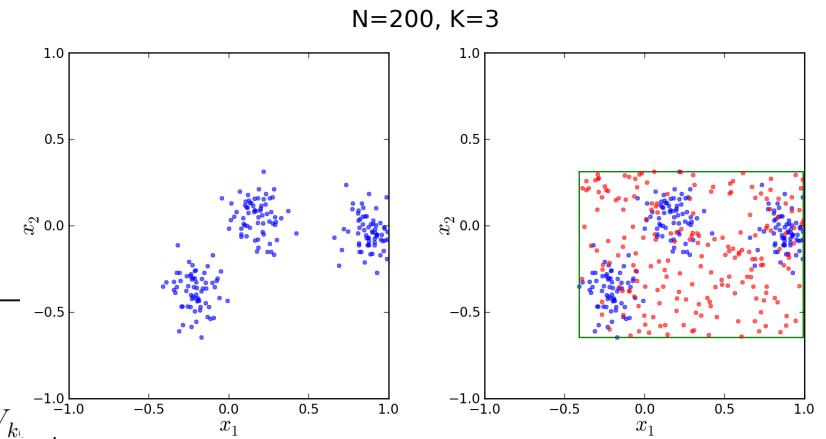
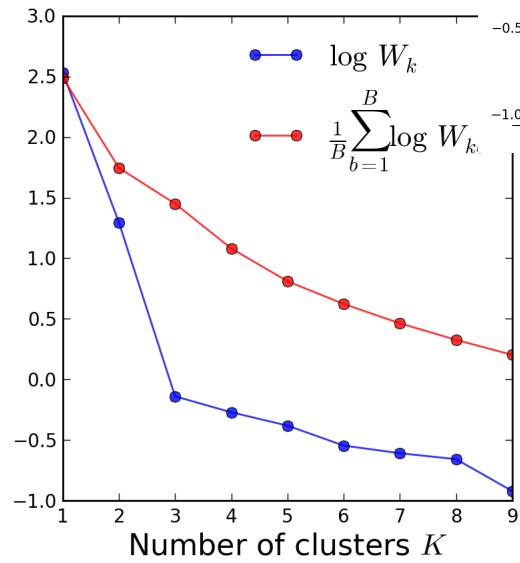
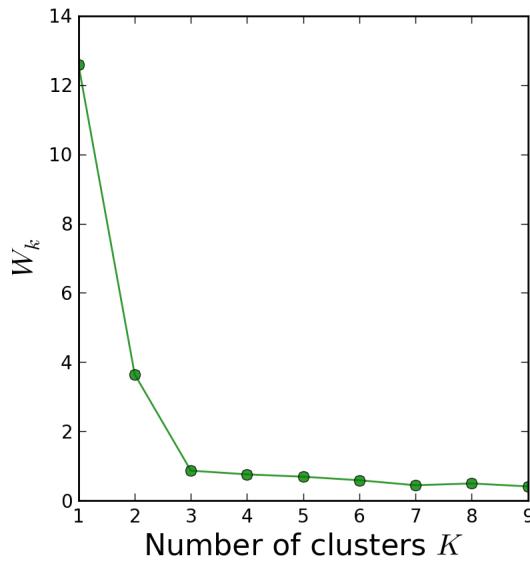
The reference datasets are in our case generated by sampling uniformly from the original dataset's bounding box (see green box in the upper right plot of the figures below). To obtain the estimate $\log W_{\text{unif}}(K)$, we compute the average of B copies of $\log W(K)$, each of which is generated from the uniform sample, and their standard deviation $s(K)$.

Gap statistic

N=200, K=3



Gap statistic



Gap statistic

Then we choose K by:

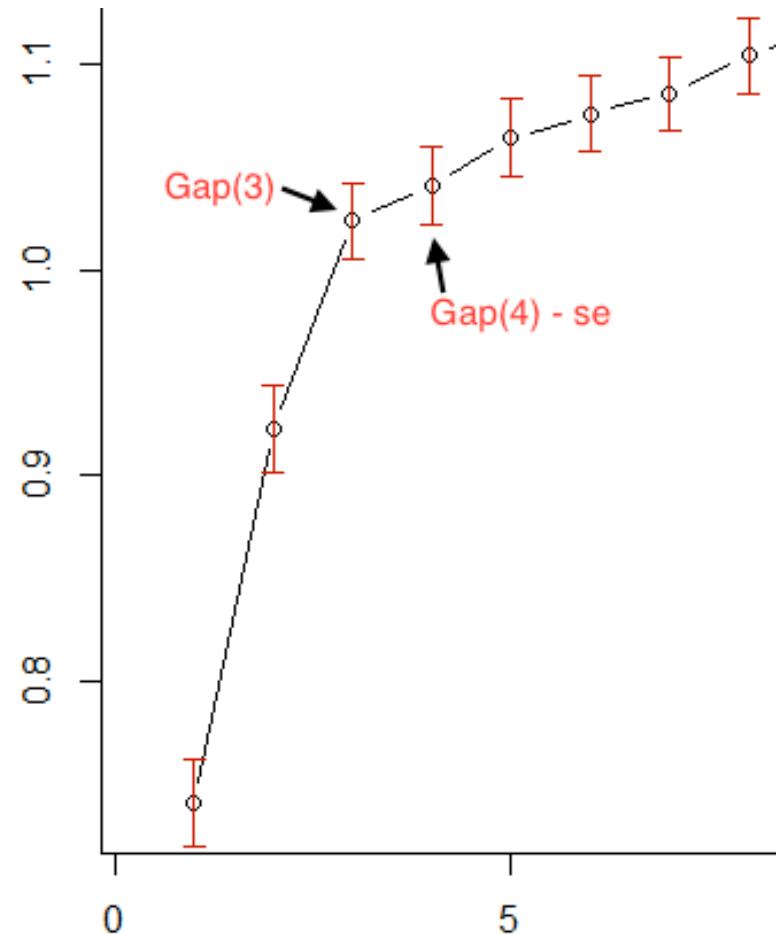
$$\hat{K} = \min_{K \in \{1, \dots, K_{\max}\}} : \text{Gap}(K) \geq \text{Gap}(K+1) - s(K+1),$$

Gap statistic

This means:

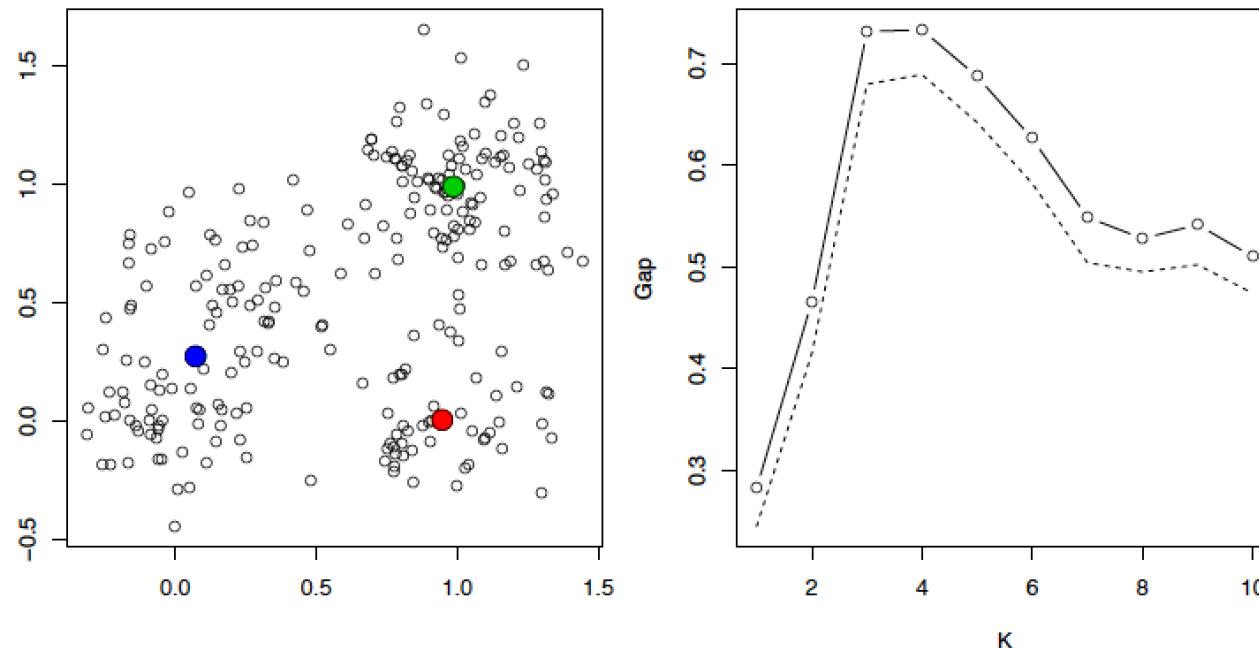
Choose the cluster size \hat{k} to be the smallest k such that

$$\text{Gap}(k) \geq \text{Gap}(k+1) - s(k+1)$$



Example: gap statistic

Running example: $n = 250$, $p = 2$, $K = 1, \dots, 10$



We would choose $K = 3$ clusters, which is also reasonable

The gap statistic does especially well when the data **fall into one cluster**. (Why? Hint: think about the null distribution that it uses)

CH index and gap statistic in R

The CH index can be computed using the kmeans function in the base distribution, which returns both the within-cluster variation and the between-cluster variation

E.g.,

$k = 5$

```
km = kmeans(x, k, alg="Lloyd")
names(km)
# Now use some of these return items to compute ch
```

The gap statistic is implemented by the function gap in the package lga, and by the function gap in the package SAGx. (Beware: these functions are poorly documented ... it's unclear what clustering method they're using)

Silhouette Analysis

- A method of interpretation and validation of consistency within clusters of data
- Provides a succinct graphical representation of how well each object lies within its cluster

Silhouette Analysis

- Assume the data have been clustered via any technique, such as k-means, into k clusters.
- For each sample \mathbf{x}_i , let a_i be the average distance between \mathbf{x}_i and all other data within the same cluster.
- Can interpret a_i as a measure of **how well \mathbf{x}_i is assigned to its cluster** (the smaller the value, the better the assignment).

Silhouette Analysis

- We then define the average dissimilarity of point \mathbf{x}_i to a cluster c as the average of the distance from \mathbf{x}_i to all points in c .
- Let b_i be the lowest average distance of \mathbf{x}_i to all points in any other cluster, of which \mathbf{x}_i is not a member.

Silhouette Analysis

- The cluster with this lowest average dissimilarity is said to be the "**neighboring cluster**" of \mathbf{x}_i , because it is the next best fit cluster for point \mathbf{x}_i . We now define a silhouette:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- or

$$s_i = \begin{cases} 1 - a_i / b_i & a_i < b_i \\ 0 & a_i = b_i \\ b_i / a_i - 1 & a_i > b_i \end{cases}$$

Silhouette Analysis

- From

$$s_i = \begin{cases} 1 - a_i / b_i & a_i < b_i \\ 0 & a_i = b_i \\ b_i / a_i - 1 & a_i > b_i \end{cases}$$

it is obvious that $-1 \leq s_i \leq 1$.

Silhouette Analysis

- s_i close to 1 requires $a_i \ll b_i$.
- a_i is a measure of dissimilarity of \mathbf{x}_i to its own cluster
- Thus a small a_i means \mathbf{x}_i is well matched.
- Large b_i implies that \mathbf{x}_i is badly matched to its neighboring cluster.
- Thus an s_i close to one means that \mathbf{x}_i is appropriately clustered.

Silhouette Analysis

- s_i close to negative one, by the same logic means that \mathbf{x}_i would be more appropriate if it was clustered in its neighboring cluster.
- An s_i near zero means that \mathbf{x}_i is on the border of two natural clusters.

Silhouette Analysis

- The average s_i over a cluster measures how tightly grouped all the data in the cluster are.
- Thus the average s_i over the entire dataset is a measure of how appropriately the data have been clustered.

Silhouette Analysis

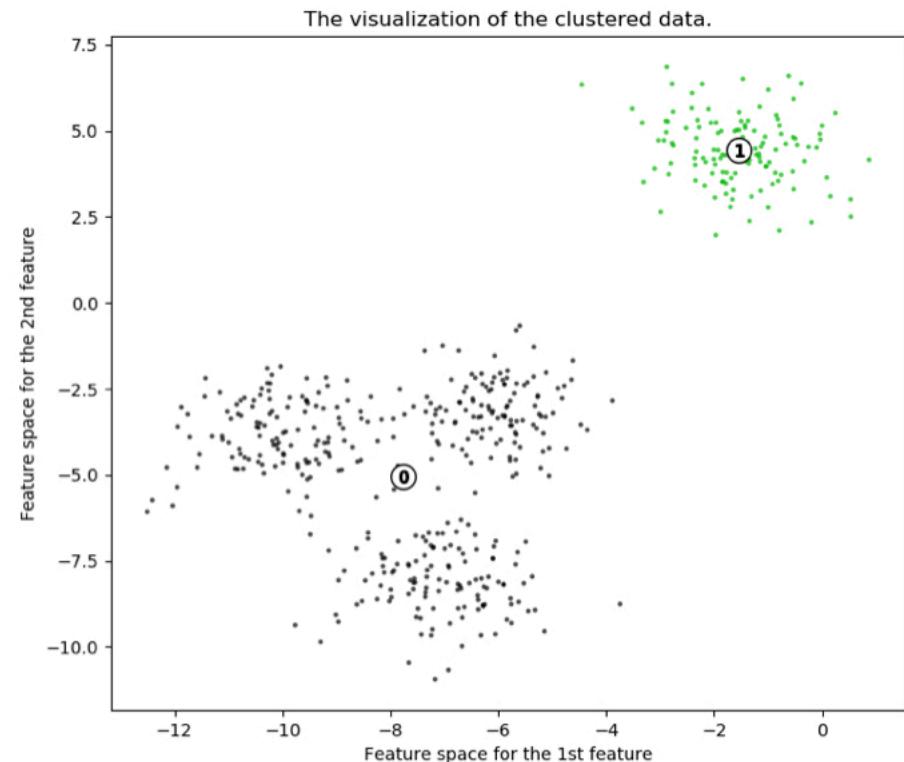
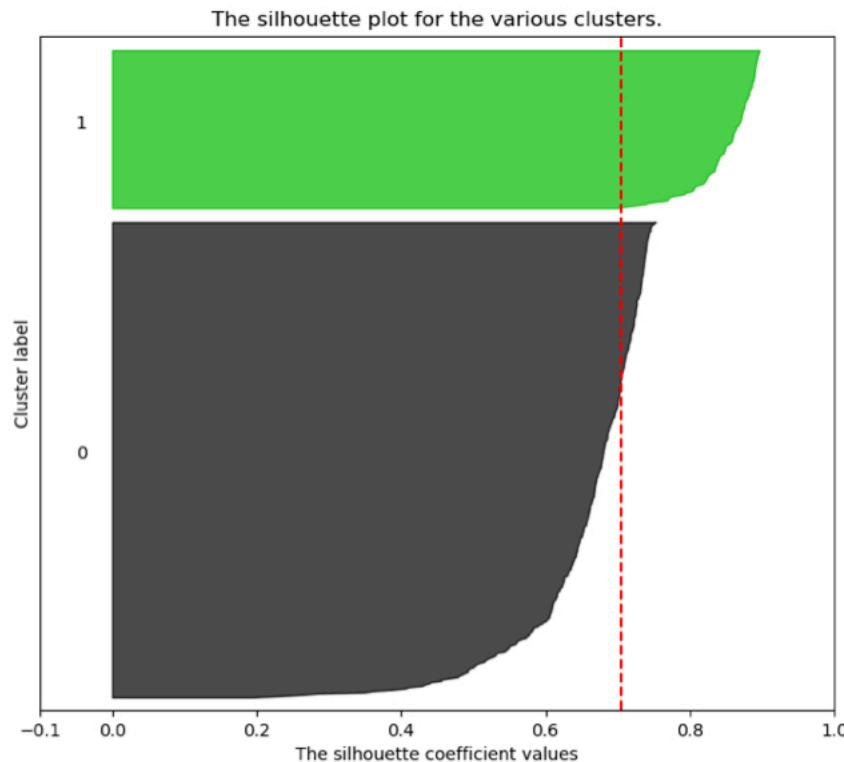
- If there are too many or too few clusters, some of the clusters will display narrower silhouettes than the rest.
- So silhouette plots and averages may be used to determine the natural number of clusters within a dataset.

Silhouette Analysis

- One can also increase the likelihood of the silhouette being maximized at the correct number of clusters by re-scaling the data using feature weights that are cluster specific.

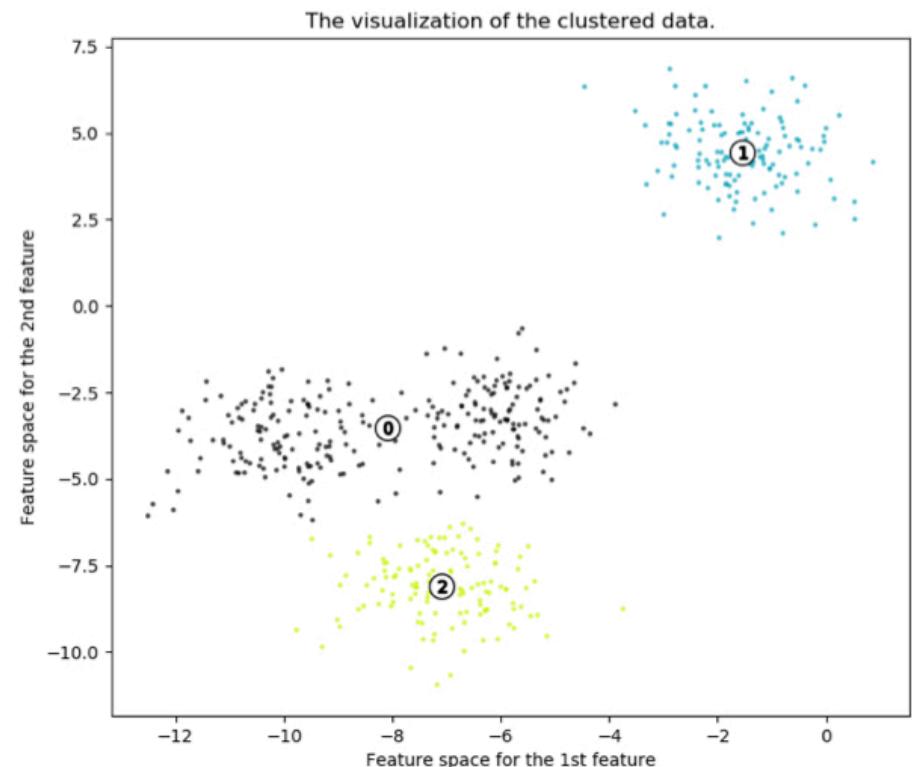
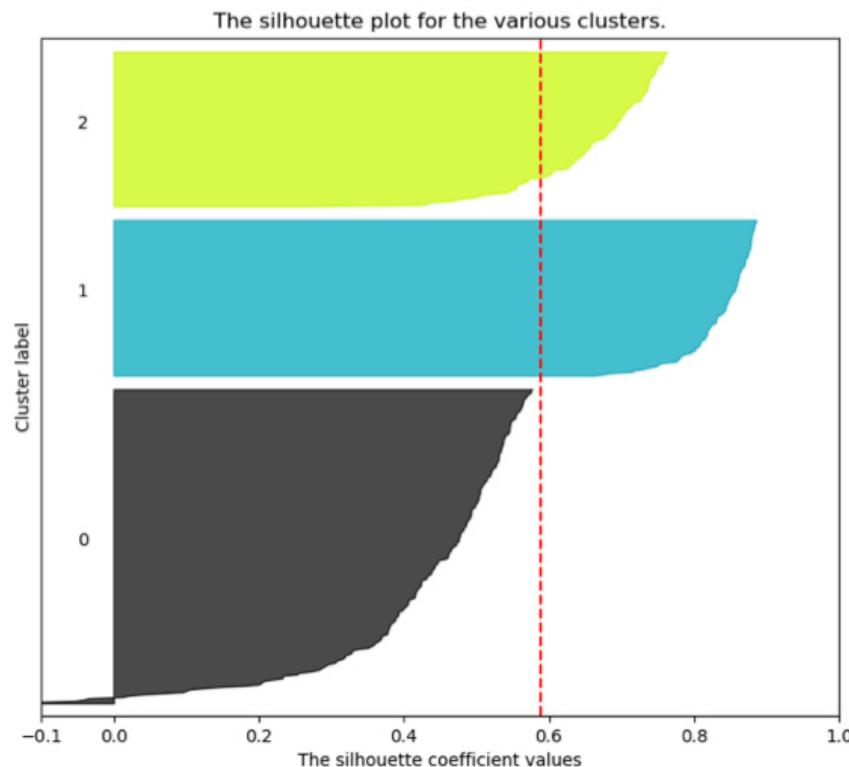
Silhouette Plot

Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



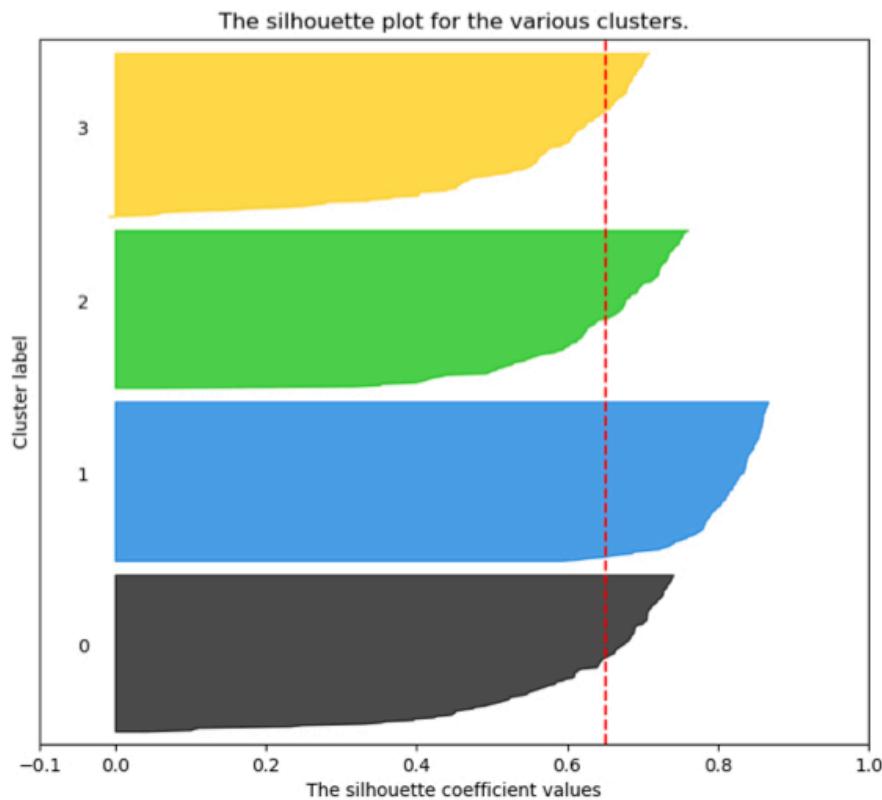
Silhouette Plot

Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



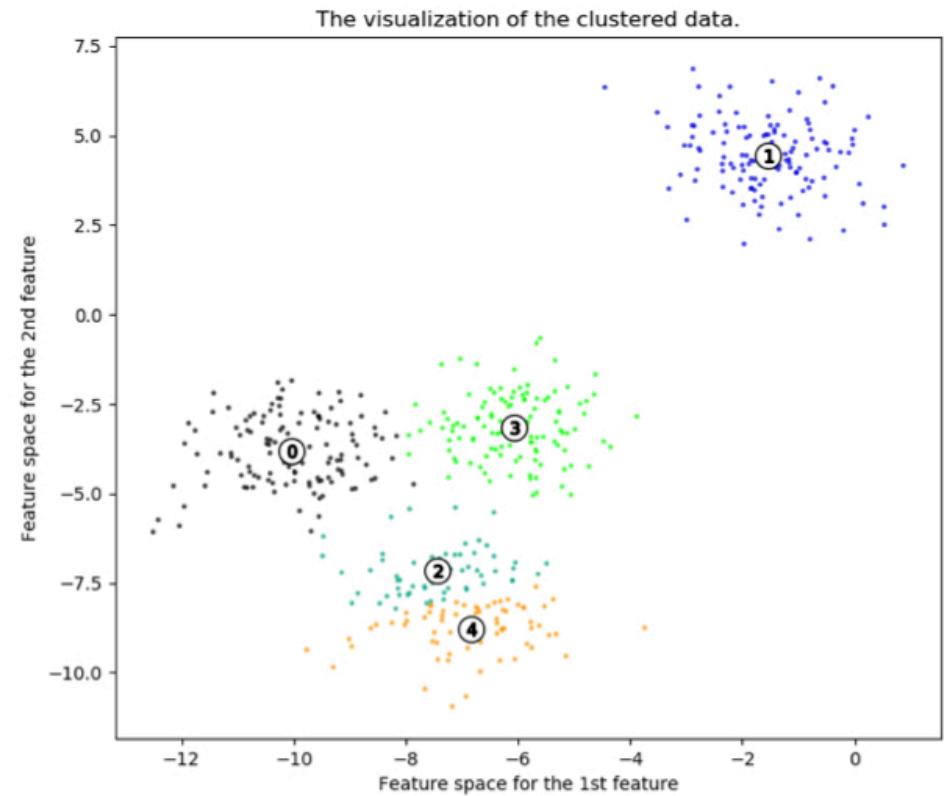
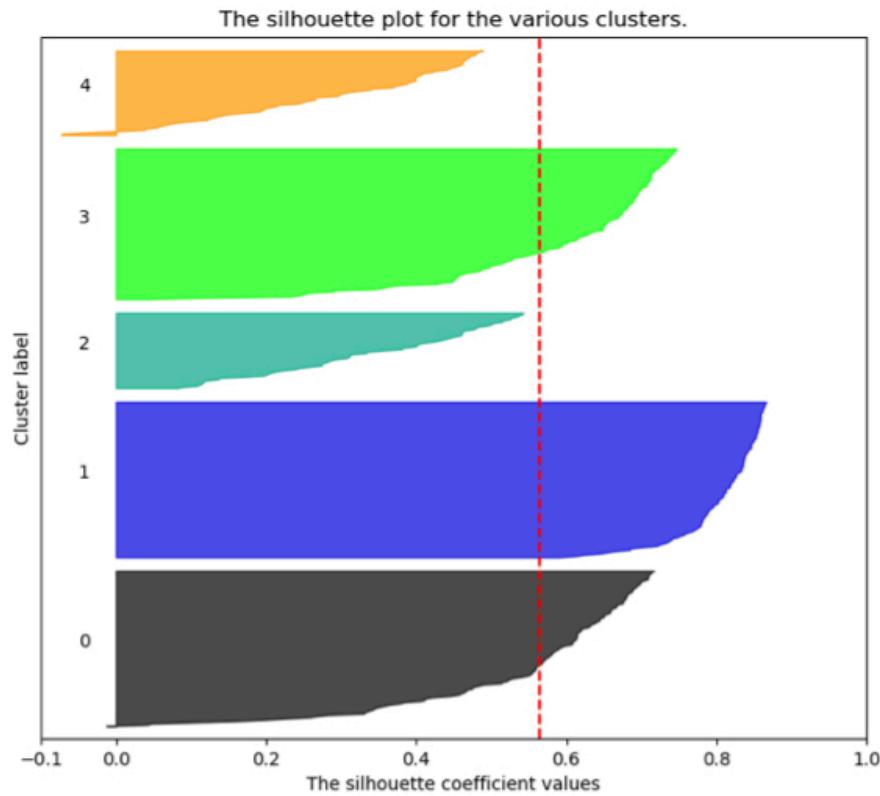
Silhouette Plot

Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



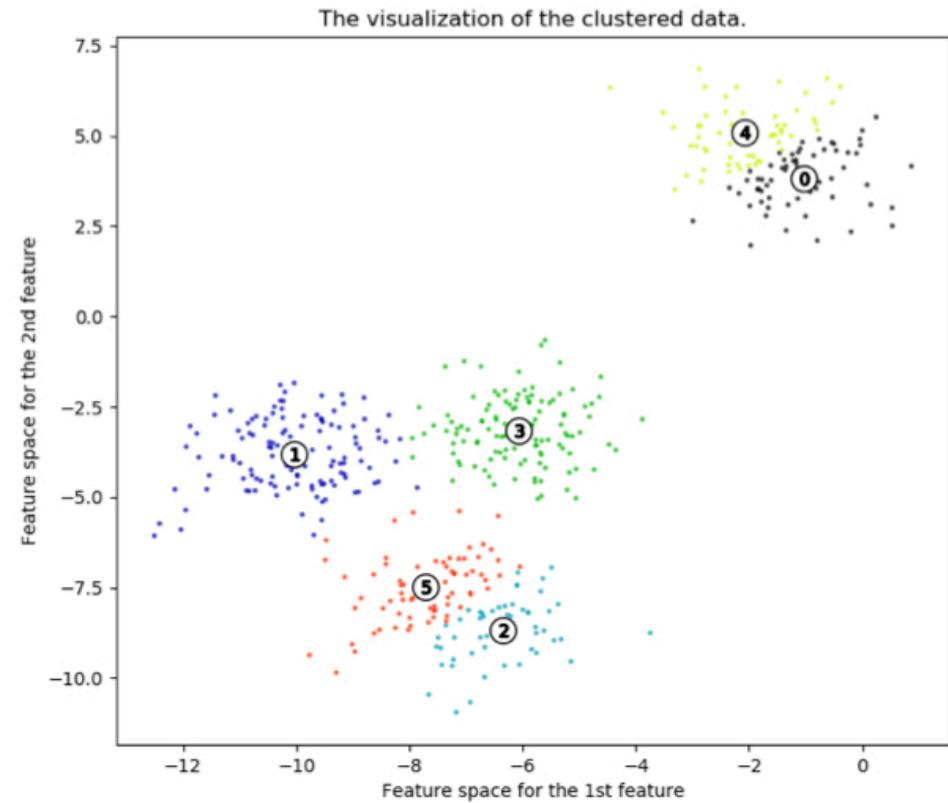
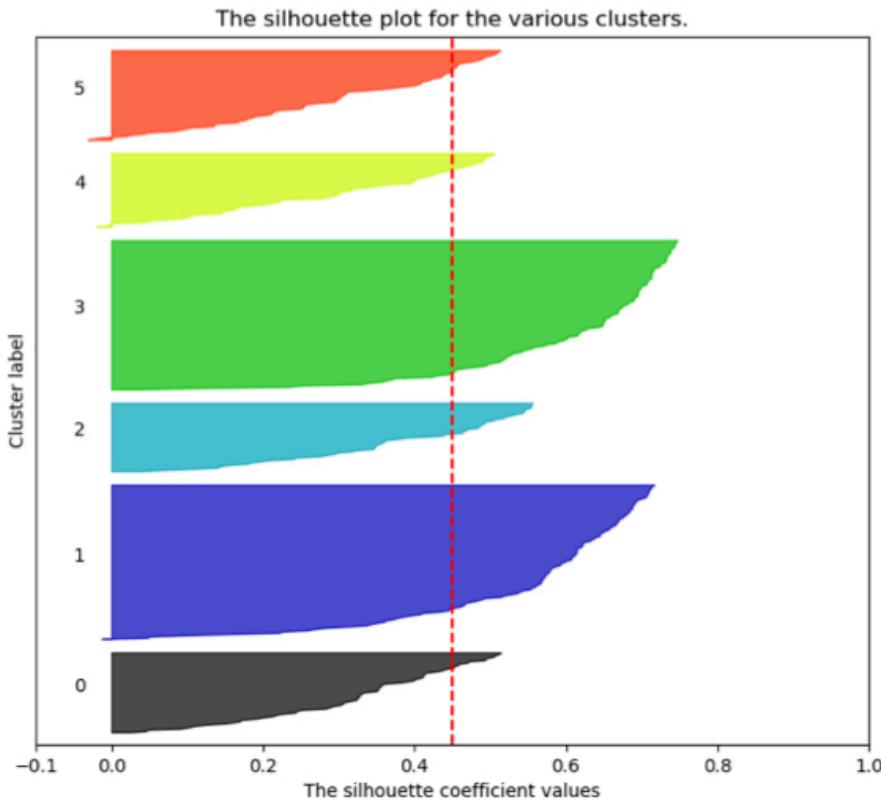
Silhouette Plot

Silhouette analysis for KMeans clustering on sample data with n_clusters = 5



Silhouette Plot

Silhouette analysis for KMeans clustering on sample data with n_clusters = 6



Silhouette Plot

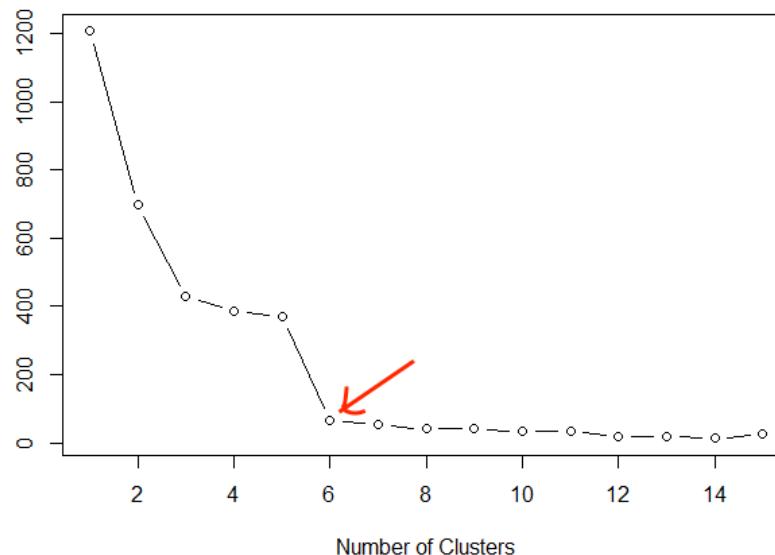
- The silhouette plot shows that $k= 3, 5$ and 6 are a bad pick for the given data due to the presence of clusters with below average silhouette scores and also due to wide fluctuations in the size of the silhouette plots.
- Silhouette analysis is more ambivalent in deciding between 2 and 4 .

Cross Validation

- The data is partitioned into v folds.
- Each of the folds is then held out at turn as a test set, a clustering model computed on the other $v - 1$ training sets
- The value of an objective function is calculated for the test set.
- Example of objective function: the sum of the squared distances to the centroids for k -means

Cross Validation

- These v values are calculated and averaged for each alternative number of clusters c , and the cluster number selected such that further increase in number of clusters leads to only a small reduction in the objective function (scree plots)



More

- <https://stackoverflow.com/questions/15376075/cluster-analysis-in-r-determine-the-optimal-number-of-clusters/15376462#15376462>

More

- <https://stats.stackexchange.com/questions/3685/where-to-cut-a-dendrogram>

Once again, it really is a hard problem

Background

Just How Many Clusters are there in the Galaxy Data?

- ▶ Galaxy Data from Postman *et al.* (1986): measurements of velocities in 10^3 km/sec of 82 galaxies from a survey of the Corona Borealis region.
- ▶ Roeder (1990): at least 3, no more than 7 modes (Confidence set)
- ▶ Others are in consensus

9172	9350	9483	9558	9775	10227
10406	16084	16170	18419	18552	18600
18927	19052	19070	19330	19343	19349
19440	19473	19529	19541	19547	19663
19846	19856	19863	19914	19918	19973
19989	20166	20175	20179	20196	20215
20221	20415	20629	20795	20821	20846
20875	20986	21137	21492	21701	21814
21921	21960	22185	22209	22242	22249
22314	22374	22495	22746	22747	22888
22914	23206	23241	23263	23484	23538
23542	23666	23706	23711	24129	24285
24289	24366	24717	24990	25633	26960
26995	32065	32789	34279		

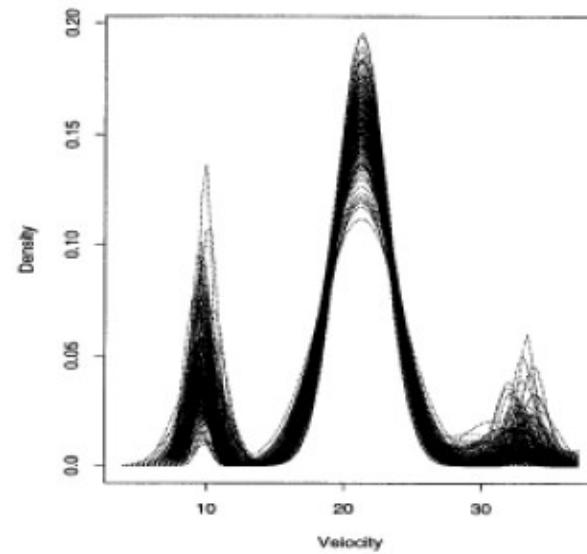
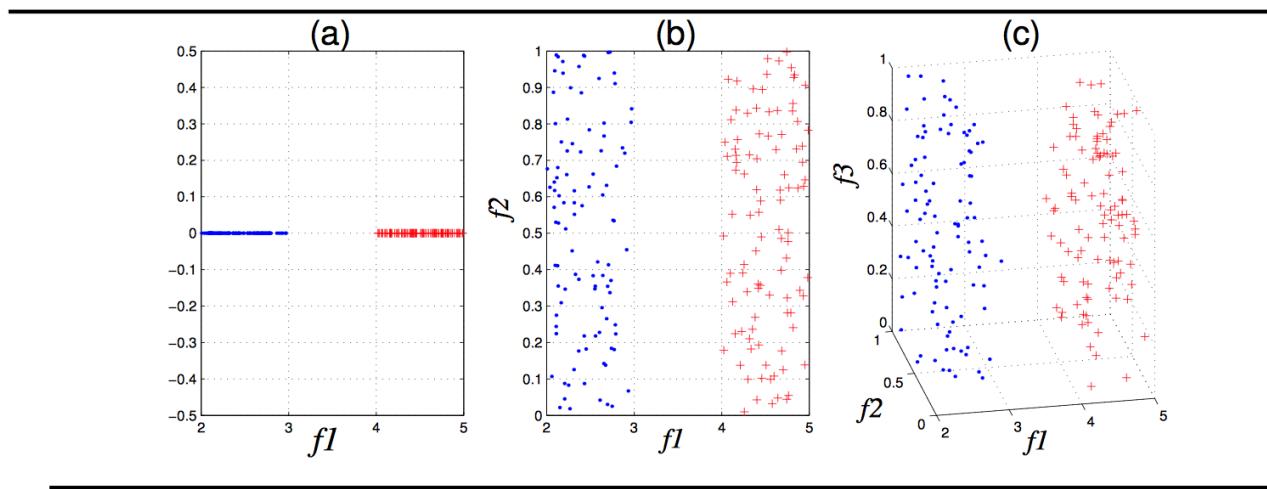


Figure 1. Densities Obtained From the Markov Chain Monte Carlo Sampler Using the Astronomy Data From Roeder (1992).

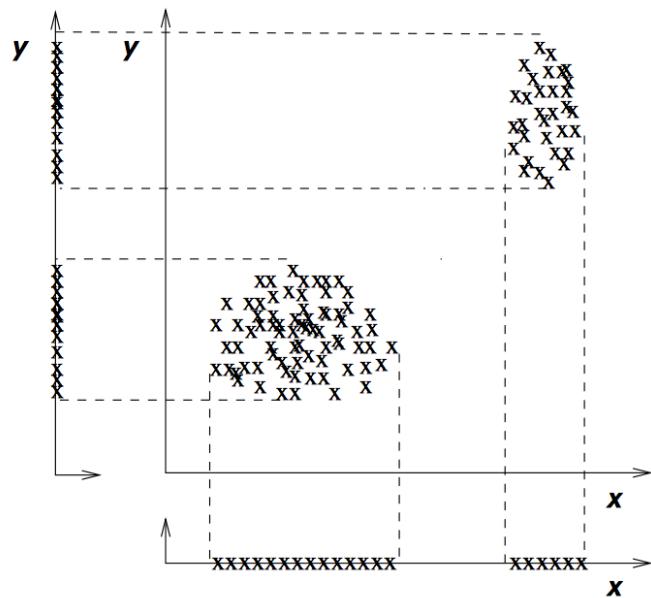
- ▶ Histogram from Roeder and Wasserman (1997)

Variable Selection for Clustering



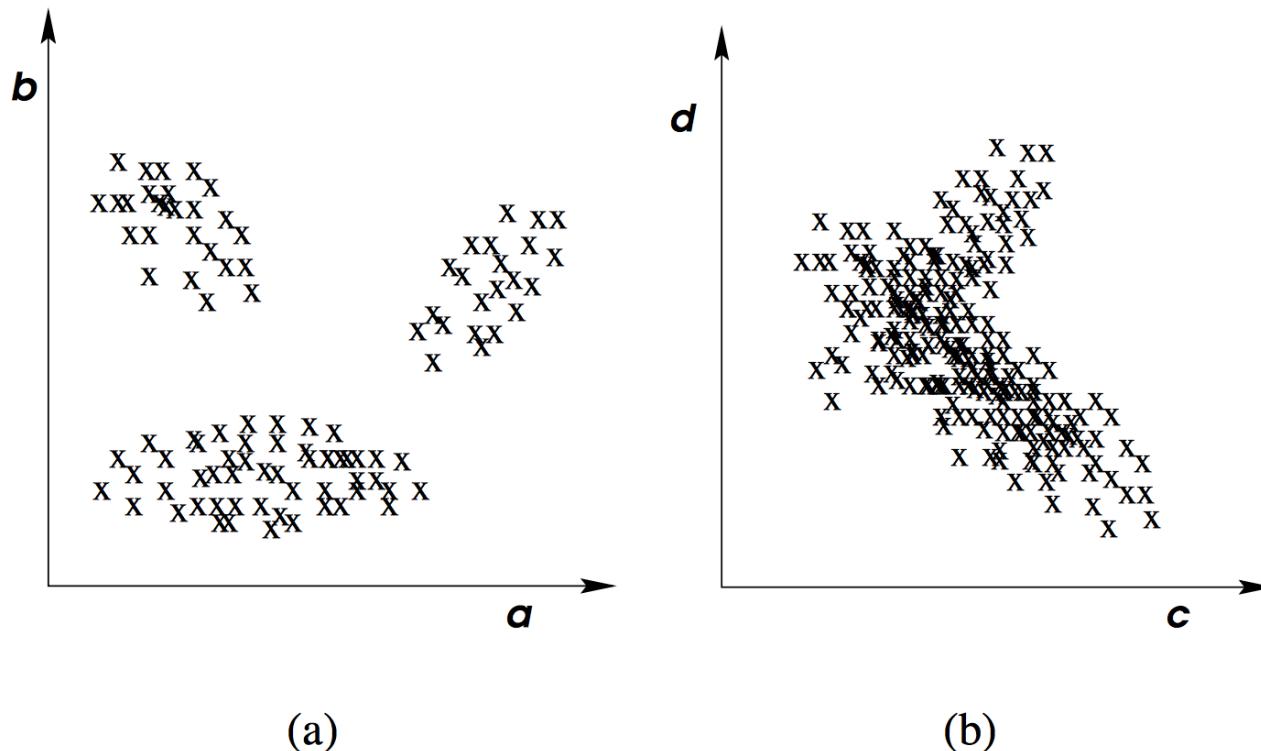
Feature f_1 is relevant while f_2 and f_3 are irrelevant. We are able to distinguish the two clusters from f_1 only. Thus, removing f_2 and f_3 will not effect the accuracy of clustering.

Variable Selection for Clustering



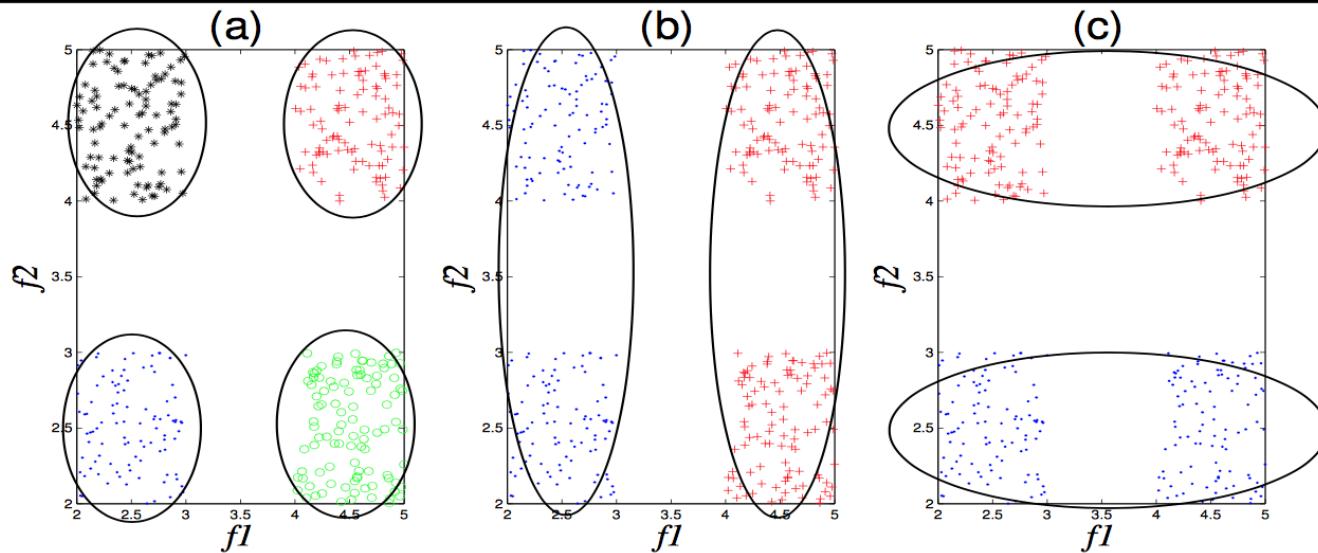
In this example, features x and y are redundant, because feature x provides the same information as feature y with regard to discriminating the two clusters.

Variable Selection for Clustering



A more complex example. Figure a is the scatterplot of the data on features a and b . Figure b is the scatterplot of the data on features c and d .

Variable Selection for Clustering



Different sets of features may produce different clustering.

Variable Selection for Clustering

- The goal of feature selection for unsupervised learning is to find the smallest feature subset that best uncovers “interesting natural” groupings (clusters) from data according to the chosen criterion.

Subset Selection

- Two paradigms:
 - feature subset selection criteria should be the criteria used for clustering
 - The two criteria need not be the same

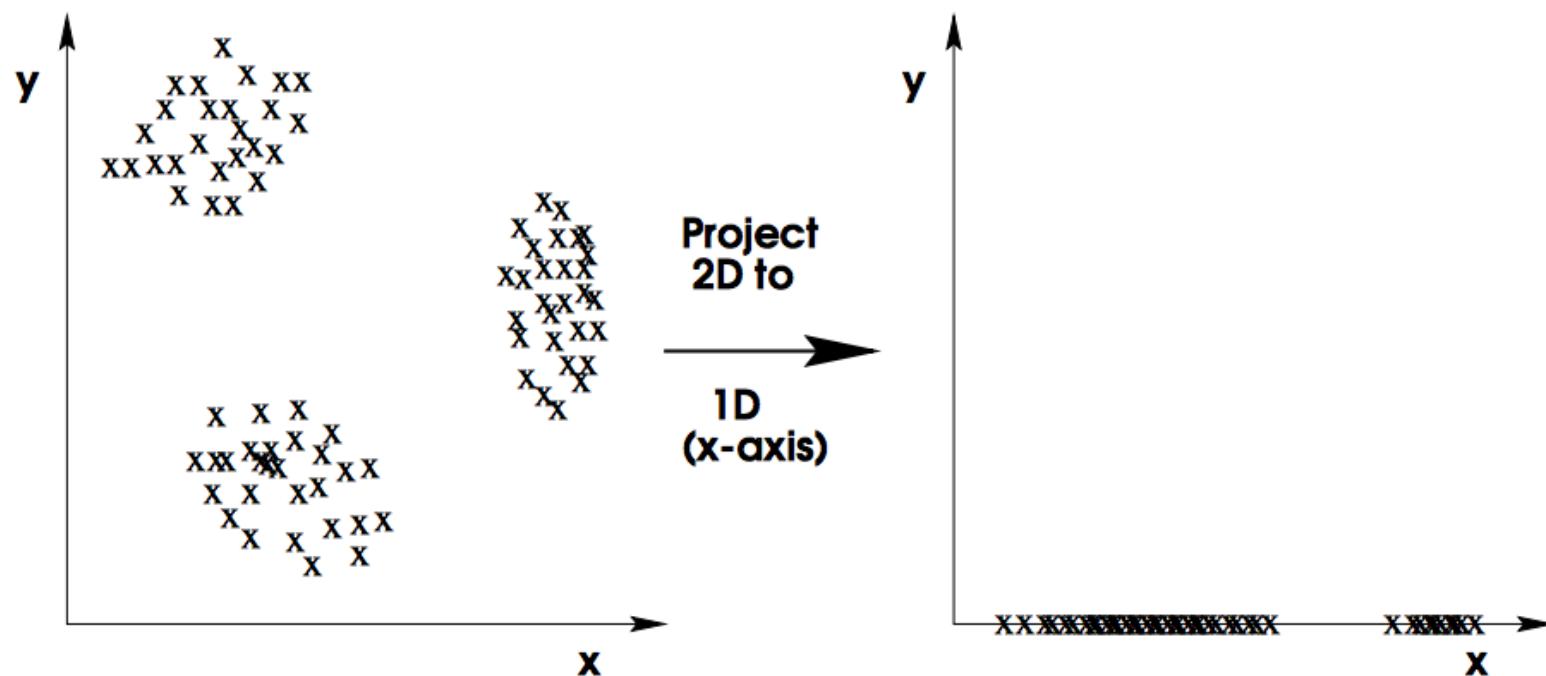
Subset Selection

Searching for the best subset of features,
we run into a new problem:
k depends on the feature subset.

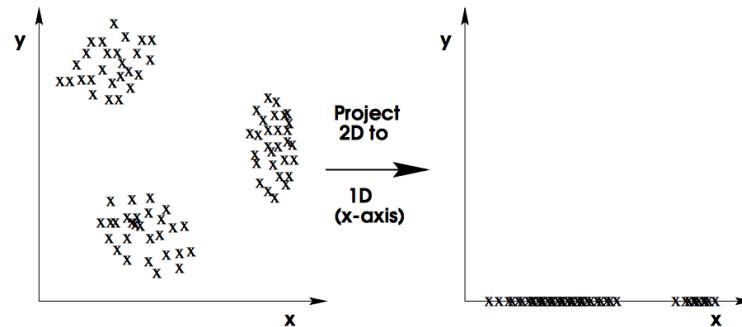
Therefore, in each step of subset
selection, **the best k has to be found.**

Many ways are conceivable for best
subset selection for clustering. The
details are left to the students.

Subset Selection



Subset Selection



Two dimensions: three clusters

One-dimension: only two clusters.

A fixed number of clusters for all feature sets does not model the data in the respective subspace correctly.

K-Means

Large number of k-means variations were proposed to handle feature selection

Most start cluster the data into k clusters. Then, assign weight to each feature.

The feature that minimizes the within-cluster distance / maximizes between-cluster distance is preferred, hence, gets higher weight.

Sparse Methods Based on L1 Norm

Witten and Tibshirani formulate clustering using a parameter vector w , and use L1 penalty and optimization to obtain a sparse set of features.

See

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2930825/pdf/nihms201124.pdf>

Principal Components Analysis

- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

Principal Components Analysis: details

- The *first principal component* of a set of features X_1, X_2, \dots, X_p is the normalized linear combination of the features

$$Z_1 = \varphi_{11}X_1 + \varphi_{21}X_2 + \dots + \varphi_{p1}X_p$$

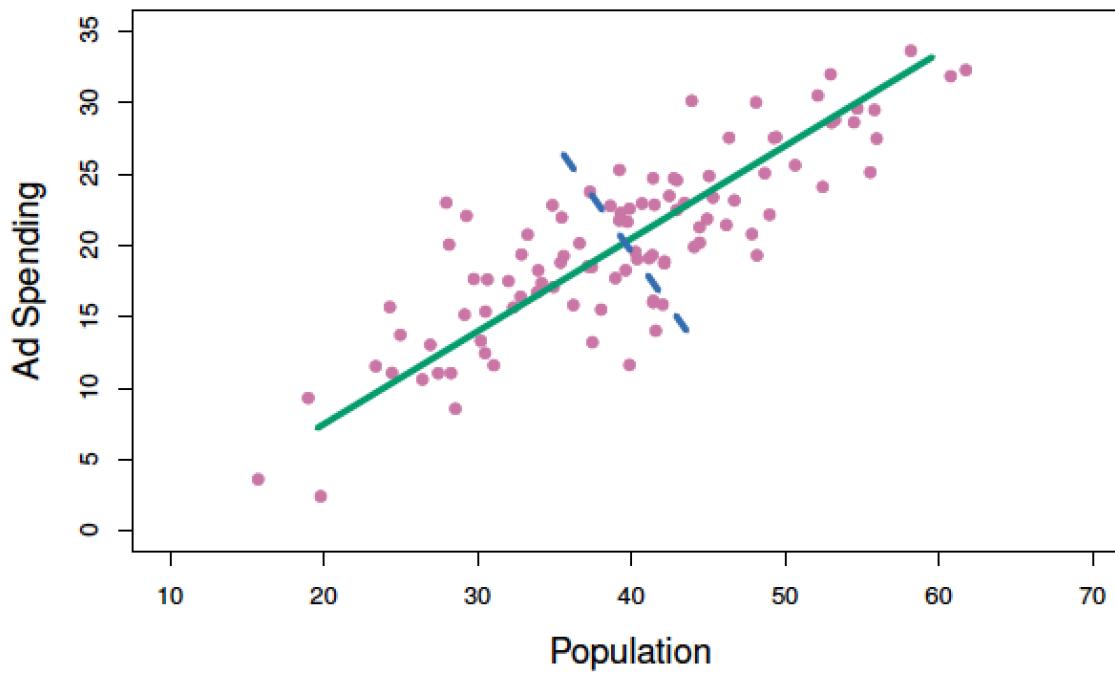
that has the largest variance. By normalized, we mean that

$$\sum_{j=1}^p \phi_{j1}^2 = 1$$

Principal Components Analysis: details

- We refer to the elements $\varphi_{11}, \dots, \varphi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector,
 $\varphi_1 = (\varphi_{11} \varphi_{21} \dots \varphi_{p1})^T$.
- We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

PCA: example



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

Computation of Principal Components

- Suppose we have a $n \times p$ data set. Since we are only interested in variance, we assume \mathbf{X} that each of the variables in \mathbf{X} has been centered to have mean zero (that is, the column means of \mathbf{X} are zero).
- We then look for the linear combination of the sample feature values of the form $z_{i1} = \varphi_{11}x_{i1} + \varphi_{21}x_{i2} + \dots + \varphi_{p1}x_{ip}$ for $i = 1, \dots, n$ that has largest sample variance, subject to the constraint that

$$\sum_{j=1}^p \phi_{j1}^2 = 1$$

Computation of Principal Components

- Since each of the x_{ij} has mean zero, then so does z_{i1} (for any values of φ_{j1}). Hence the sample variance of the z_{i1} can be written as

$$\frac{1}{n} \sum_{i=1}^n z_{i1}^2.$$

Computation: continued

- Therefore, the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11}, \dots, \phi_{p1}}{\text{maximize}} \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j1} x_{ij} \right)^2 \text{ subject to } \sum_{j=1}^p \phi_{j1}^2 = 1$$

- This problem can be solved via a singular-value decomposition of the matrix \mathbf{X} , a standard technique in linear algebra.
- We refer to Z_1 as the first principal component, with realized values z_{11}, \dots, z_{n1}

Geometry of PCA

- The loading vector φ_1 with elements $\varphi_{11}, \varphi_{21}, \dots, \varphi_{p1}$ defines a direction in feature space along which the data vary the most.
- If we project the n data points x_1, \dots, x_n onto this direction, the projected values are the principal component scores z_{11}, \dots, z_{n1} themselves.

Further principal components

- The second principal component is the linear combination of X_1, \dots, X_p that has maximal variance among all linear combinations that are *uncorrelated* with Z_1 .
- The second principal component scores $Z_{12}, Z_{22}, \dots, Z_{n2}$ take the form

$$Z_{i2} = \varphi_{12}X_{i1} + \varphi_{22}X_{i2} + \dots + \varphi_{p2}X_{ip},$$

where φ_2 is the second principal component loading vector, with elements $\varphi_{12}, \varphi_{22}, \dots, \varphi_{p2}$.

Further principal components: continued

- It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction φ_2 to be orthogonal (perpendicular) to the direction φ_1 . And so on.

Further principal components: continued

- The principal component directions $\varphi_1, \varphi_2, \varphi_3, \dots$ are the ordered sequence of right singular vectors of the matrix \mathbf{X} , and the variances of the components are $1/n$ times the squares of the singular values. There are at most $\min(n - 1, p)$ principal components.

Illustration

- **USAarrests** data: For each of the fifty states in the United States, the data set contains the number of arrests per 100,000 residents for each of three crimes: **Assault**, **Murder**, and **Rape**. We also record **UrbanPop** (the percent of the population in each state living in urban areas).

Illustration

- The principal component score vectors have length $n = 50$, and the principal component loading vectors have length $p = 4$.
- PCA was performed after standardizing each variable to have mean zero and standard deviation one.

USAarrests data: PCA plot

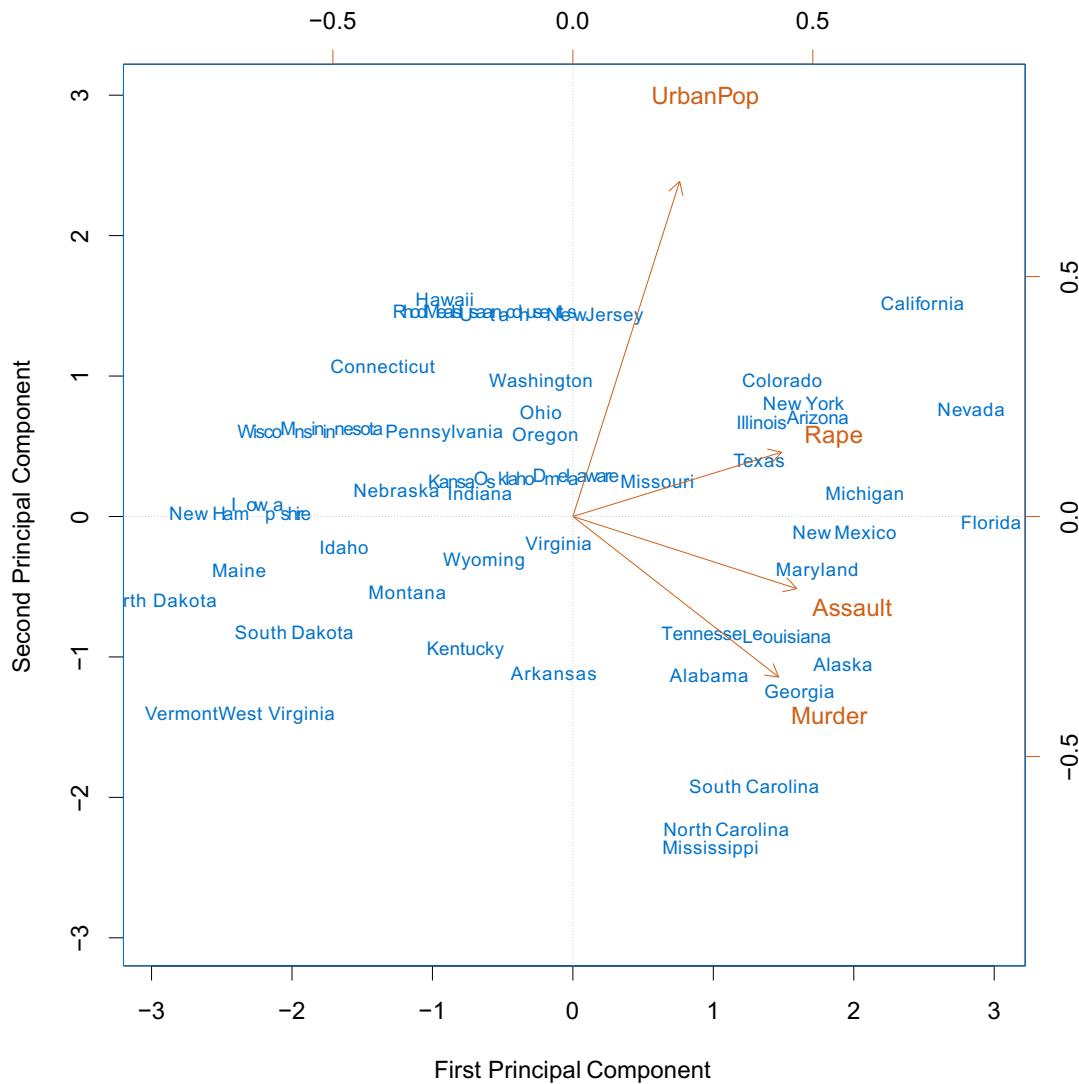


Figure details

The first two principal components for the USArrests data.

- The blue state names represent the scores for the first two principal components.

PCA loadings

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

The contribution of each variable in each of the principal components PC1 and PC2 is shown as a vector. For example, Assault is shown as $[0.58, -0.19]^T$

Figure details

- The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for **Rape** on the first component is 0.54, and its loading on the second principal component 0.17 [the word **Rape** is centered at the point (0.54, 0.17)].

Figure details

- This figure is known as a *biplot*, because it displays both the principal component scores and the principal component loadings.

Figure details

- The first loading vector places approximately equal weight on **Assault**, **Murder**, and **Rape**, with much less weight on **UrbanPop**.
- Hence this component roughly corresponds to a measure of overall rates of serious crimes.

Figure details

- The second loading vector places most of its weight on **UrbanPop** and much less weight on the other three features.
- Hence, this component roughly corresponds to the level of urbanization of the state.

Figure details

- The crime-related variables (**Murder**, **Assault**, and **Rape**) are located close to each other
- The **UrbanPop** variable is far from the other three.