Class 7: Beyond Linear Models & Midterm Review

MSA 8150: Machine Learning for Analytics

Alireza Aghasi

Institute for Insight, Georgia State University

Principal Component Regression (PCR)

- Recall the singular value decomposition (SVD)

$$\boldsymbol{X} = \boldsymbol{U}_{n\times p}\boldsymbol{\Sigma}_{p\times p}\boldsymbol{V}^{\top}_{p\times p}$$

  where

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{bmatrix}$$

- When the first r entries of $\boldsymbol{\Sigma}$ are the dominant terms, the first r columns in $\boldsymbol{U}$ form a subspace (coordinate system) that very well represent our data
- In PCR, we move from the original feature space to the new feature space represented by $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_r$
- Technically, instead of a fit like

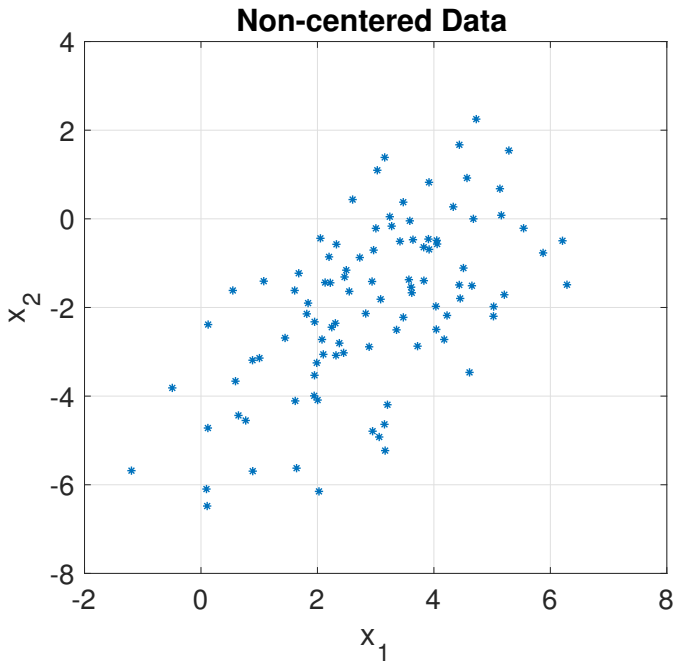
$$\boldsymbol{y} = \beta_0 + \beta_1\boldsymbol{x}_1 + \cdots \beta_p\boldsymbol{x}_p$$

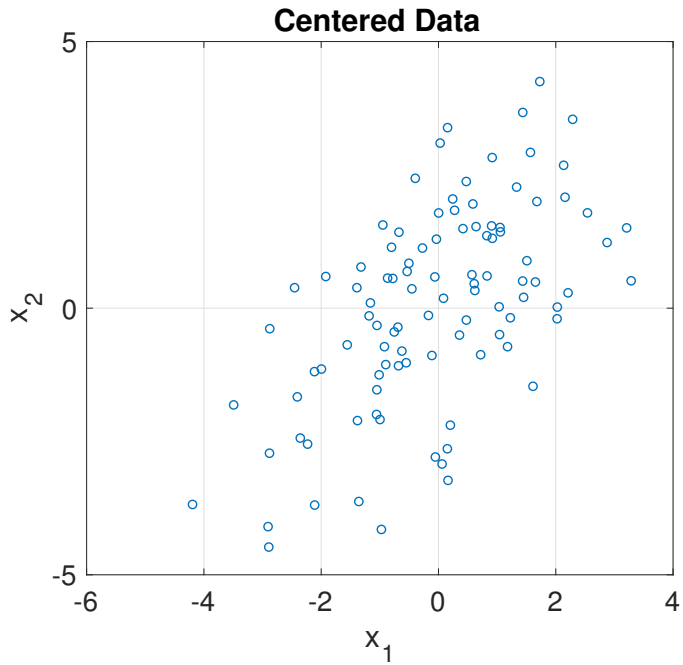
  we fit the model

$$\boldsymbol{y} = \gamma_0 + \gamma_1\boldsymbol{z}_1 + \cdots \gamma_r\boldsymbol{z}_r$$

Principal Component Regression (PCR)

- Consider $\boldsymbol{X}$ to be an $n \times p$ matrix representing our feature data (each column corresponds to a feature and each row to a sample)
- We assume that $\boldsymbol{X}$ is centered, meaning that each column is zero mean
- This technically means that our data cloud is centered around the origin and not some other point in the space





Principal Component Regression (PCR)

- Lets see what we want to do mathematically:

$$\boldsymbol{X} = \begin{pmatrix} | & | & | \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \\ | & | & | \end{pmatrix}$$

We assume that $\boldsymbol{X}$ is normalized and centered then

$$\text{corr}(\boldsymbol{X}) = \boldsymbol{X}^\top \boldsymbol{X} = \begin{pmatrix} 1 & -0.3694 & 0.9907 \\ -0.3694 & 1 & -0.4926 \\ 0.9907 & -0.4926 & 1 \end{pmatrix}$$

- We want to construct new features z_1, z_2 such that $z_1 = c_{11}x_1 + c_{12}x_2 + c_{13}x_3$, and $z_2 = c_{21}x_1 + c_{22}x_2 + c_{23}x_3$ and are uncorrelated, that is:

$$\boldsymbol{Z}_{n\times 2} = \boldsymbol{X}_{n\times 3} \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{pmatrix}$$

such that $\text{corr}(\boldsymbol{Z}) = \boldsymbol{Z}^\top \boldsymbol{Z} = \boldsymbol{I}$

Principal Component Regression (PCR) when $r = p$

- Based on what we learned from SVD, finding such matrix $\boldsymbol{C}$ is not hard
- In general we have a centered and normalized feature matrix $\boldsymbol{X}_{n\times p}$ where $\text{corr}(\boldsymbol{X}) \neq \boldsymbol{I}$ and we want to construct a new feature matrix $\boldsymbol{Z}_{n\times p}$ such that

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{C}, \quad \text{and} \quad \boldsymbol{Z}^\top \boldsymbol{Z} = \boldsymbol{I}$$

- Say $n > p$, and $\text{rank}(\boldsymbol{X}) = p$ then by doing SVD we have

$$\boldsymbol{X} = \boldsymbol{U}_{n\times p}\boldsymbol{\Sigma}_{p\times p}\boldsymbol{V}^\top_{p\times p}$$

then simple calculation reveals that $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ and as a result $\boldsymbol{Z} = \boldsymbol{U}$

Principal Component Regression (PCR) when $r < p$

- Based on what we learned from SVD, finding such matrix $\boldsymbol{C}$ is not hard
- In general we have a centered and normalized feature matrix $\boldsymbol{X}_{n\times p}$ where $\text{corr}(\boldsymbol{X}) \neq \boldsymbol{I}$ and we want to construct a new feature matrix $\boldsymbol{Z}_{n\times p}$ such that

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{C}, \quad \text{and} \quad \boldsymbol{Z}^\top \boldsymbol{Z} = \boldsymbol{I}$$

- Say $n > p$, and $\text{rank}(\boldsymbol{X}) > r$ then by doing SVD we have

$$\boldsymbol{X} = \boldsymbol{U}_{n\times p}\boldsymbol{\Sigma}_{p\times p}\boldsymbol{V}^\top_{p\times p}$$

and therefore

$$\boldsymbol{X} \approx \boldsymbol{U}_{n\times r}\boldsymbol{\Sigma}_{r\times r}\boldsymbol{V}^\top_{p\times r}$$

then as before we can see that setting $\boldsymbol{C} = \boldsymbol{V}_{p\times r}\boldsymbol{\Sigma}^{-1}_{r\times r}$ and $\boldsymbol{Z} = \boldsymbol{U}_{n\times r}$ would yield $\boldsymbol{Z}^\top \boldsymbol{Z} = \boldsymbol{I}$ and $\boldsymbol{Z} \approx \boldsymbol{X}\boldsymbol{C}$

Principal Component Regression (PCR) when $r < p$

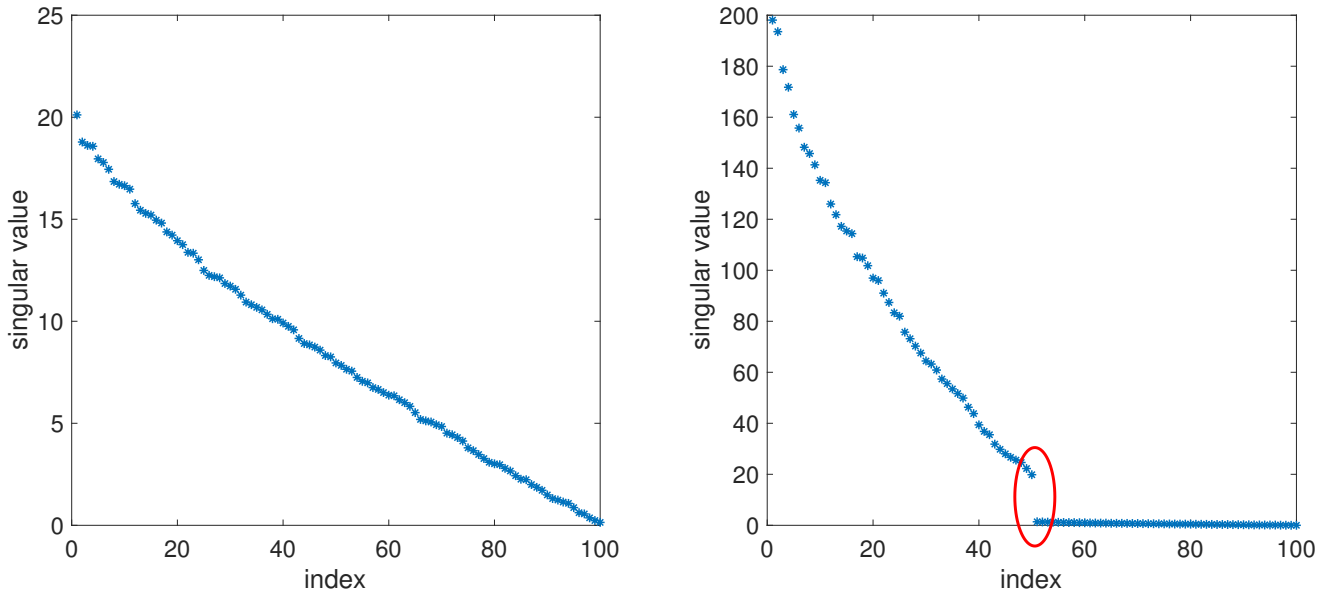
- Lets make the world simple: you are given the standardized data matrix $\boldsymbol{X}$ which is correlated, all you need to do to generate an uncorrelated data matrix is to do SVD and pick the r first columns, i.e.,

$$\boldsymbol{Z} = \boldsymbol{U}_{n \times r}$$

- Now if we have a response variable $\boldsymbol{y} \in \mathbb{R}^n$ we can use the $\boldsymbol{z}$ features to do the regression
- For the training part of the data since multiplying $\boldsymbol{X}$ by $\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ generates $\boldsymbol{U}$ we can pick $\boldsymbol{U}$ as the new features. However, for the test part of the data this does not hold and we need to multiply $\boldsymbol{X}_{test}$ by $\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ to generate the test features

How to Determine r

- Normally, the number of principal components, r, can be determined by cross validation
- If we see a good contrast between two consecutive singular values, roughly choosing r (or at least an initial value) can become easy!



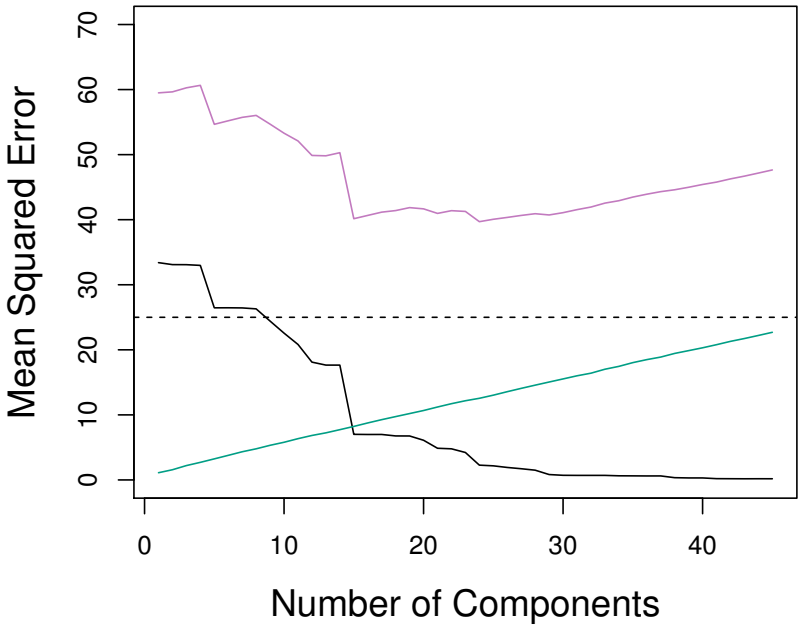
- Despite all this, a formal selection requires CV
- This is technically because such selection only relies on $\boldsymbol{X}$, and no guarantee that the selected features contribute to the response

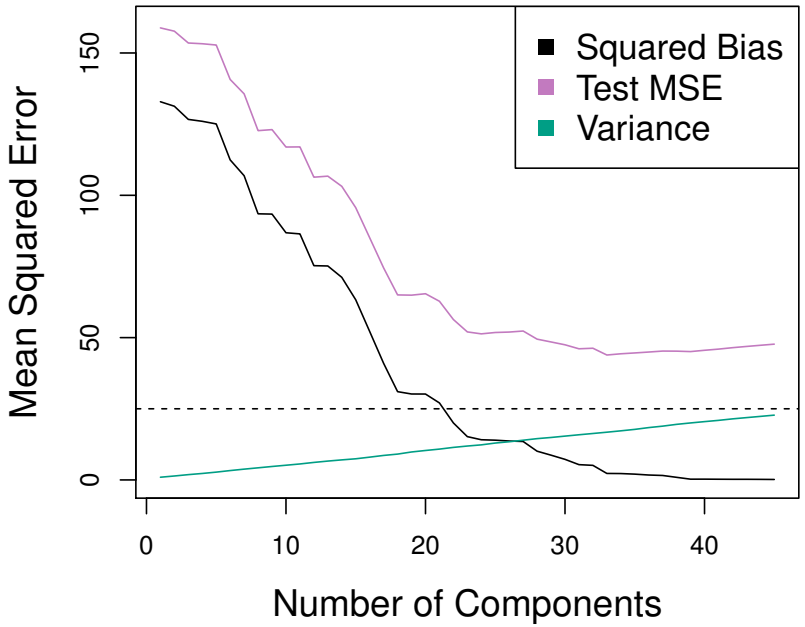
More on PCR

- PCR identifies linear combinations, or directions, that best represent the predictors $x_1, \cdots, x_p$
- These directions are identified in an unsupervised way, since the response Y is not used to help determine the principal component directions
- That is, **the response does not supervise the identification of the principal components**
- Consequently, PCR suffers from a potentially serious drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response
- **Partial Least Squares** (PLS, details of which are skipped in this course) can be considered as a remedy to this

Example

Example Running CV and PCR on two different data sets (left: $r \approx 14$ could be a good selection; right: CV suggests using almost all features)





Because PCR does not take into account the relation to the response, despite the possibility of expressing the response with only two features on the right panel, PCR suggests almost using all features

Let's try some coding examples!

Beyond Linear Models

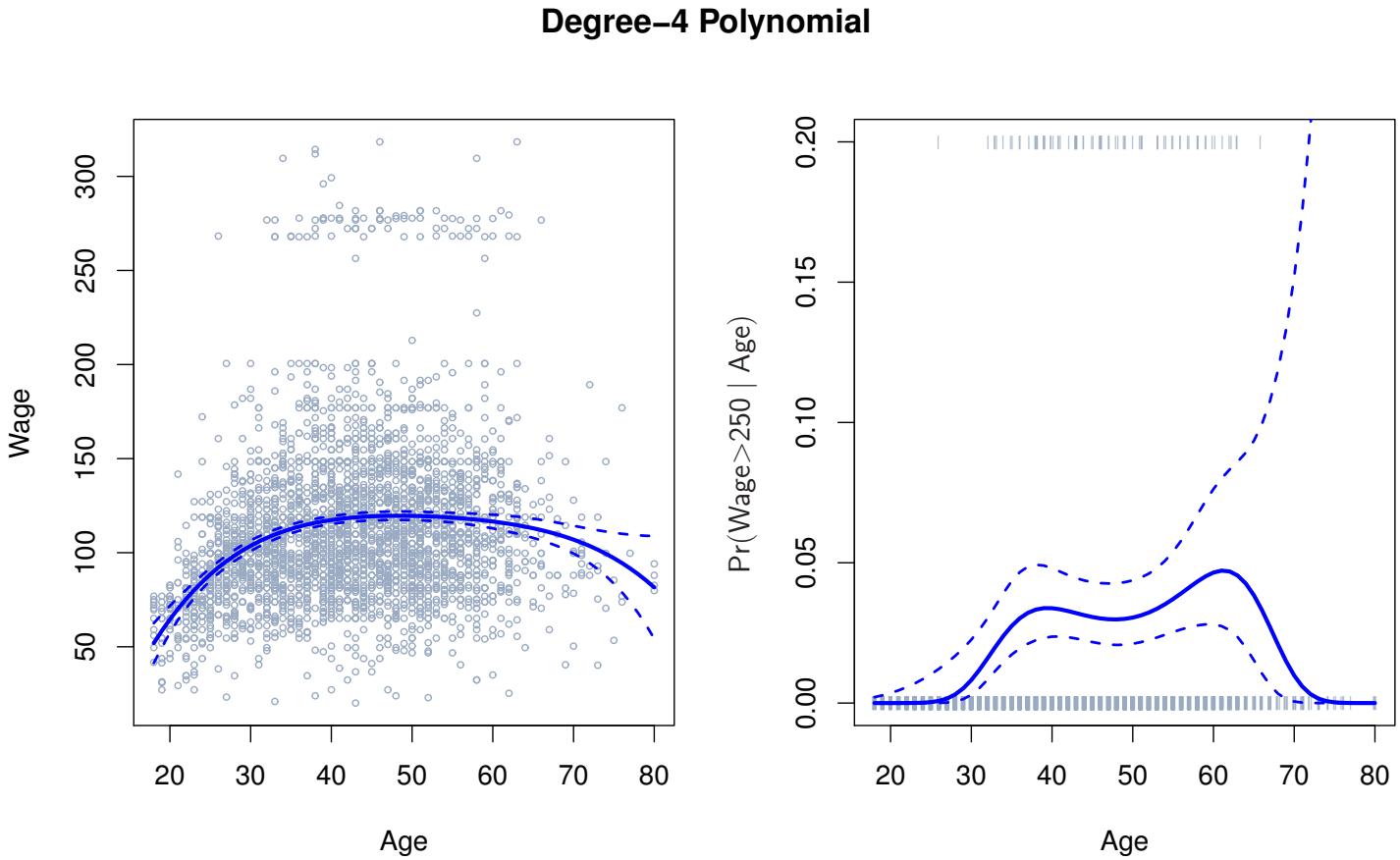
Polynomial Regression

- We learned that linear models can also produce nonlinear relationships between the features and the response
- All we need to do is including nonlinear functions of the features among the covariates
- For example, in the case of a single feature we can consider the model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \epsilon$$

- This could be easily done for both regression and classification

Polynomial Regression



- Regressing the wage in terms of the age (regression)
- Logistic regression:

$$P(Wage > 250|Age = x) = \frac{\exp(\beta_0 + \beta_1 x + \cdots + \beta_d x^d)}{1 + \exp(\beta_0 + \beta_1 x + \cdots + \beta_d x^d)}$$

Use of Step Functions

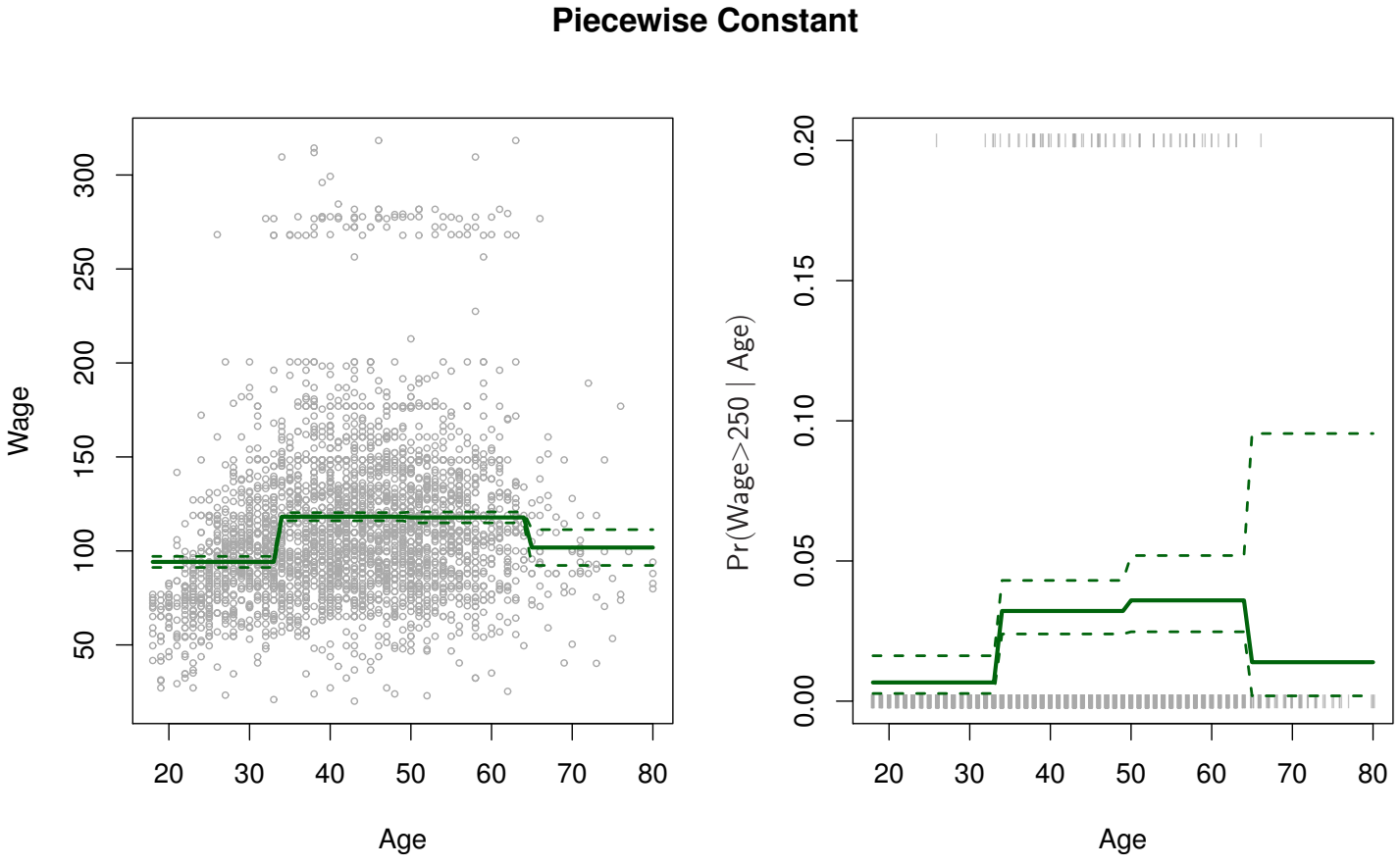
- There is no reason for not exploring other forms of nonlinearities
- One other popular function to use is a step function:

$$C_{[a,b]}(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & else \end{cases},$$

- Now if we split the domain of x into d intervals $[a_0, a_1], [a_1, a_2], \cdots, [a_{d-1}, a_d]$ we may consider the fit

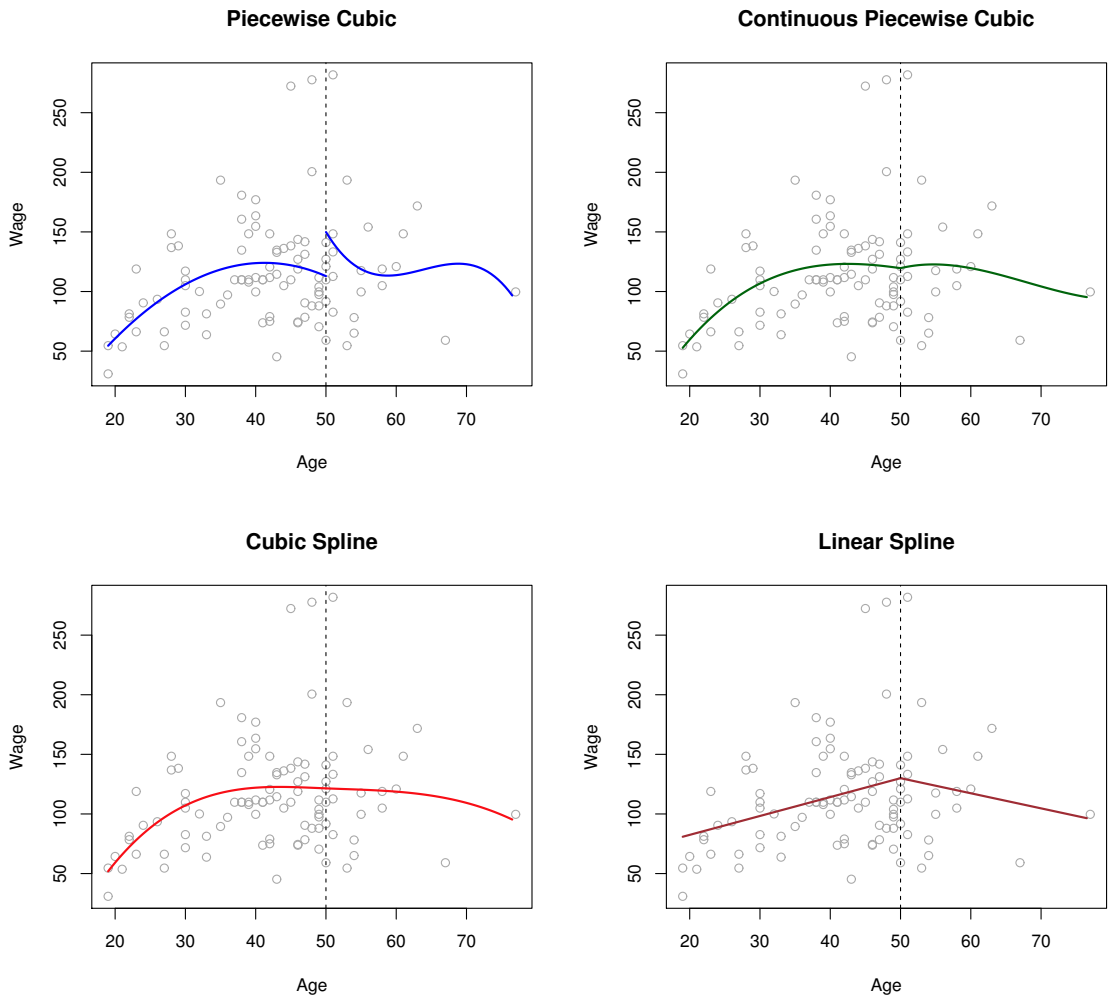
$$y = \beta_0 + \beta_1 C_{[a_0,a_1]}(x) + \beta_2 C_{[a_1,a_2]}(x) + \cdots \beta_d C_{[a_{d-1},a_d]}(x)$$

Example: Use of Step Functions



- The choice of cut-points $a_0, a_1, \cdots a_d$ can be uniform or adaptive

Piecewise Polynomials



- Similar to the idea of step functions, we can split the domain of x into multiple intervals and fit a polynomial in each

Spline Representations

- One immediate idea would be to fit a cubic polynomial at each interval and impose continuity conditions at the cutting points
- Equivalently, we can fit models of the form $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$ and at the same time impose q orders of continuity at K points, $\xi_1, \cdots, \xi_K$, by introducing the additional basis functions

$$b_{3+k} = (x - \xi_k)_+^q = \begin{cases} (x - \xi_k)^q & x \geq \xi_k \\ 0 & else \end{cases}, \qquad k = 1, \cdots, K$$

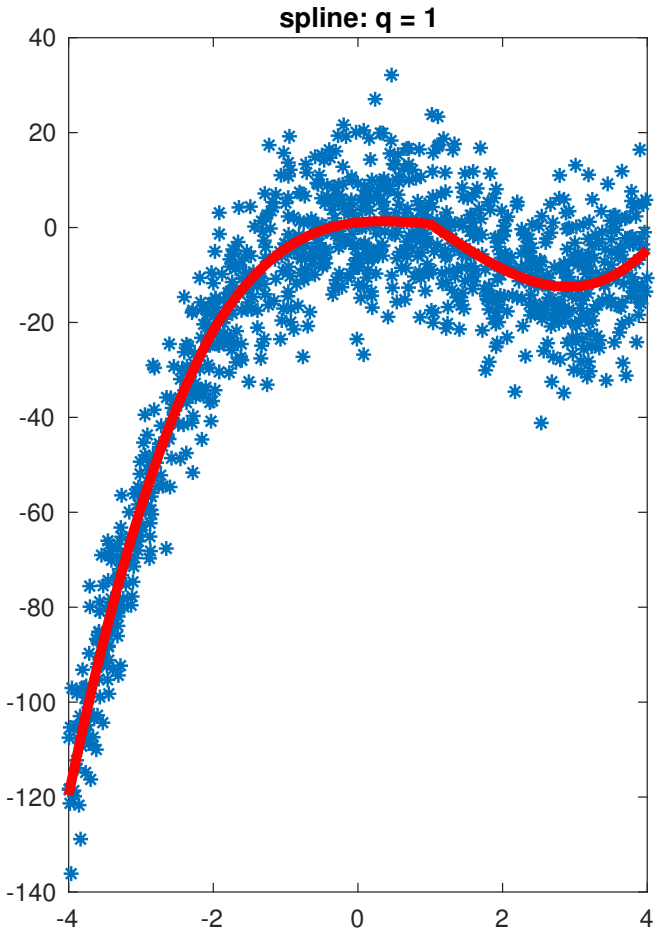
- So we ultimately end up with a regression with $K + 4$ coefficients as follows:

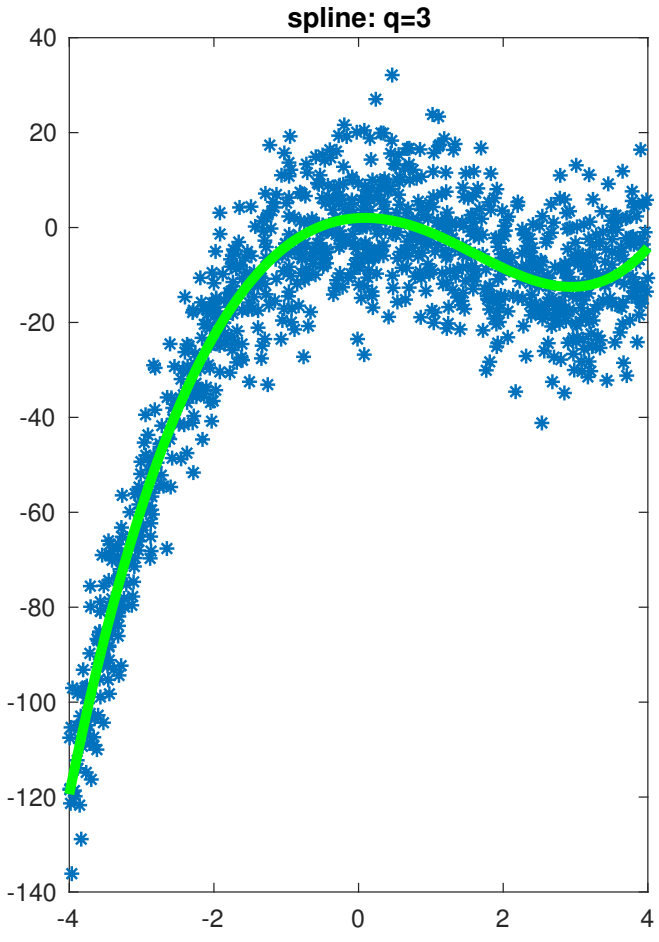
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \xi_1)_+^q + \cdots + \beta_{K+3} (x - \xi_K)_+^q$$

- Linear splines: $q = 1$
- Cubic splines: $q = 3$

Example: Spline Fitting Linear vs Cubic

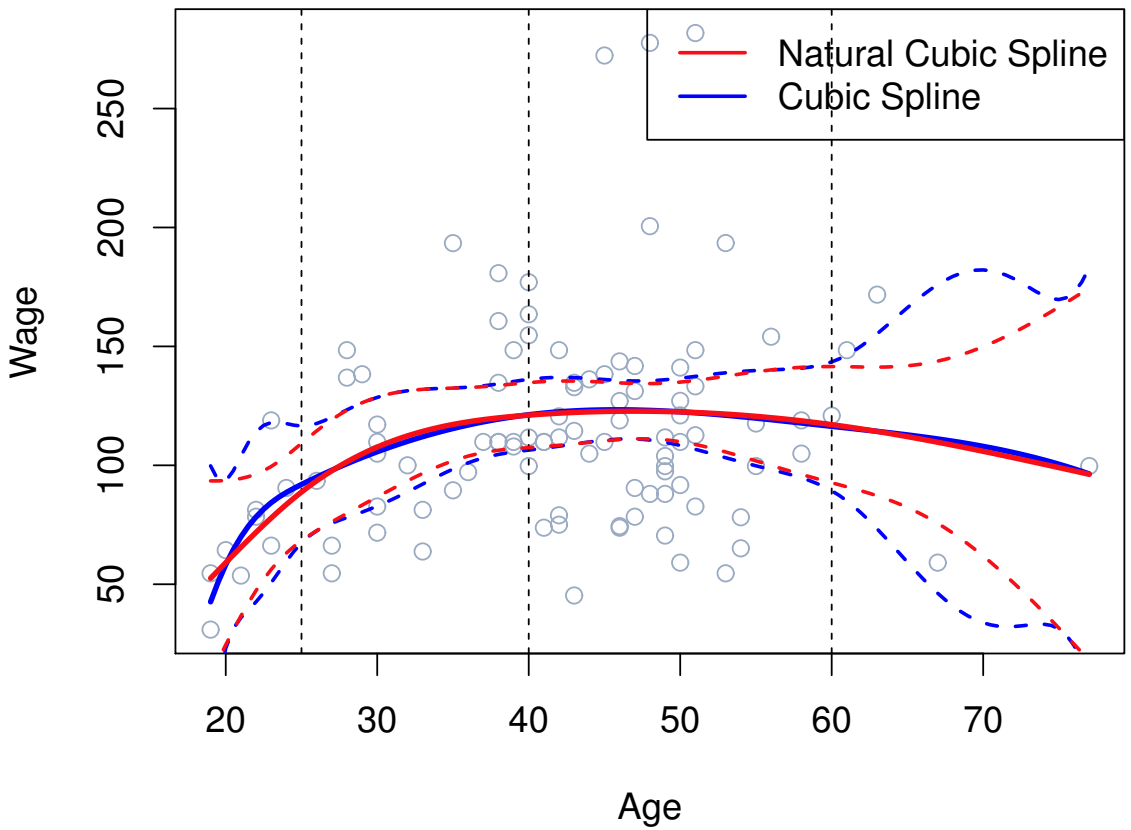
- Please follow the programming example given in the class





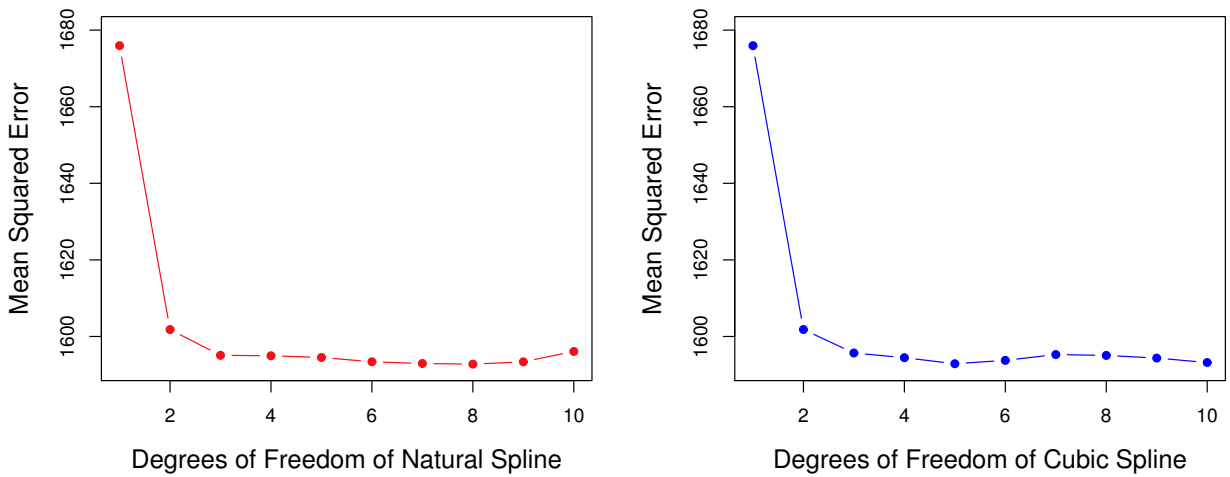
Natural Cubic Splines

- Similar to the cubic splines, only difference is beyond the boundary knots ξ_1 and ξ_K the fit is linear (requires adding 4 more constraints)
- This can avoid large deviations around the extreme boundaries



How to Decide the Number and Location of the Knots?

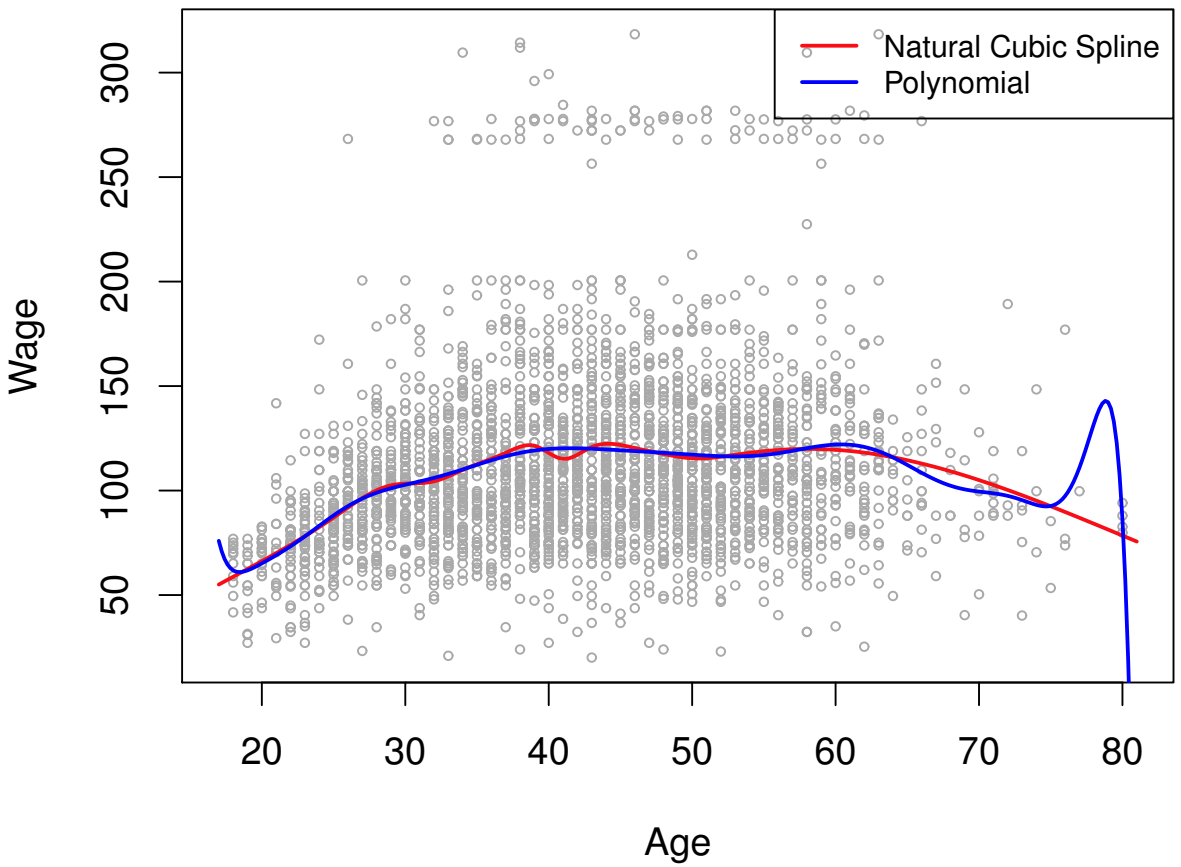
- If we have no pre-assumptions about the data, normally we place the knots on a uniform grid
- Otherwise we can place them at locations were we expect more variations of the fit
- The number of knots can be either determined visually, or more carefully via cross validation



This example: $K = 3$ for natural spline (NS) and $K = 4$ for cubic seem adequate (NS has 4 more conditions so in principal has 4 fewer degrees of freedom for a fixed K)

Why Splines over Polynomial Regression

- With splines we do not observe the high variations caused by fitting high order polynomials



Smoothing Splines

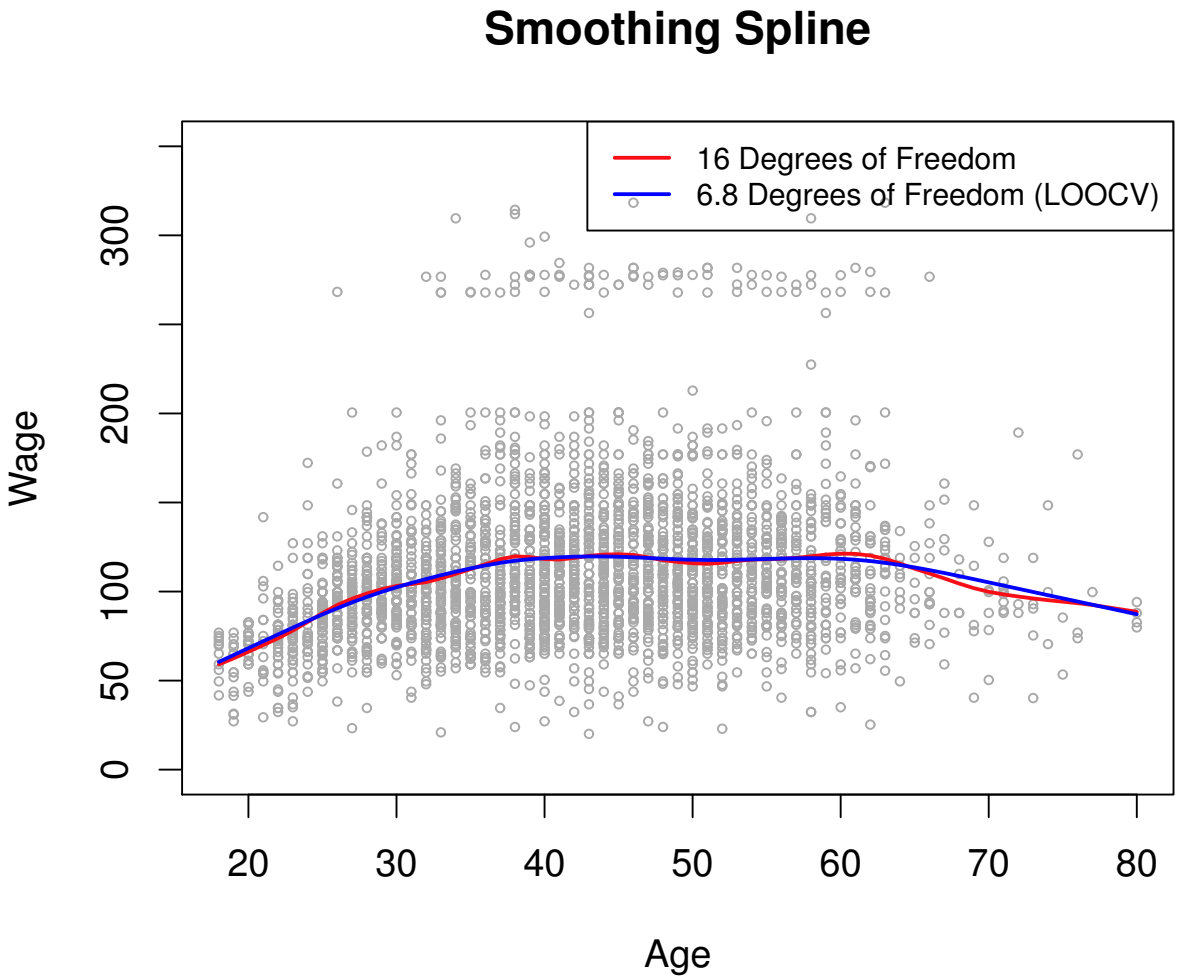
- Smoothing splines are technically nonparametric representations (no explicit equation)
- They are the solutions to the functional objective

$$\min_g \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int (g''(t))^2 \, dt$$

- This type of optimization is the mainly discussed in the context of calculus of variations (e.g., shortest path on a manifold, Brachistochrone problem)
- $\lambda = 0$ yields a curve trying to pass through all data points, $\lambda = \infty$ yields a linear regression fit
- There is only one free parameter to decide, which we can determine using cross validation
- Turns out that the LOOCV RSS can be calculated in closed form via a single fit

Example: Smoothing Splines

- Smoothing spline is simply a natural cubic spline with knots at every sample point (proof requires differential analysis)



GAMs: Generalized Additive Models

- Recall in linear regression we wanted to perform a fit by evaluating the coefficients β_j in the following model:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \epsilon_i$$

  where i corresponds to the i-th sample

- In GAMs we try to find **functions** f_j such that a good fit can be generated via

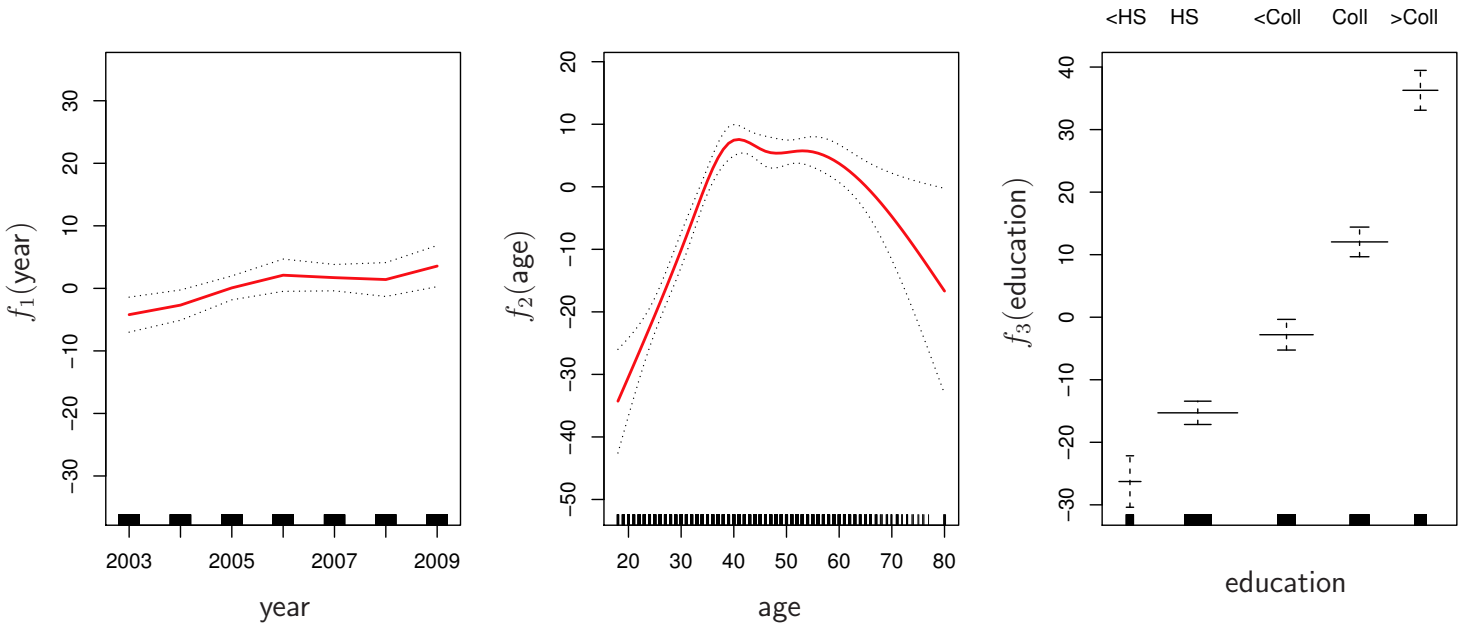
$$y_i = \beta_0 + f_1(x_{i,1}) + f_2(x_{i,2}) + \cdots + f_p(x_{i,p}) + \epsilon_i$$

- Here the unknowns are the functions f_j and still we have an additive model
- They can also be used for classification purposes

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + f_1(x_{i,1}) + f_2(x_{i,2}) + \cdots + f_p(x_{i,p})$$

GAMs: Generalized Additive Models

- We can either consider smoothness criteria or a priori make assumptions about the nature of each f_j (e.g., being a spline with certain degrees of freedom)
- Example: $wage = \beta_0 + f_1(year) + f_2(age) + f_3(education) + \epsilon$



year: NS with 4 degrees of freedom (DF), **age**: NS with 5 DF, **education**: step function fit with qualitative values

Some Coding Examples

Lets look into some quick examples!

Midterm Review

Scope and Details

- Chapters 1 through 6 (today's "beyond linearity" is chapter 7 and not included, however, PCR is included)
- Questions are mainly conceptual, few technical (may contain derivations as HW and quiz)
- Since HW and project contain extensive programmings, exam does not contain any
- Chapters 2, 3, 4, 5: technical & conceptual
- Chapter 6: only conceptual but you should know the LASSO and Ridge formulations
- Only bring a pen! No cheat sheets, no calculators, etc
- If you need an equation, it will be listed in the question
- Please carefully review all the material covered in the slides to fully understand the concepts
- HW 3 will be posted over the weekend, but you would have $\sim$17 days to work on it instead of 10 days (spring break)

Lecture 1 Review

Difference between supervised and unsupervised learning

Supervised Learning

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

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

– Normally from the available examples $(\boldsymbol{X}_1, Y_1), ..., (\boldsymbol{X}_N, Y_N)$ we choose N_1 of them (around 80%) for training and $N - N_1$ of them for the later evaluation of the model (testing)

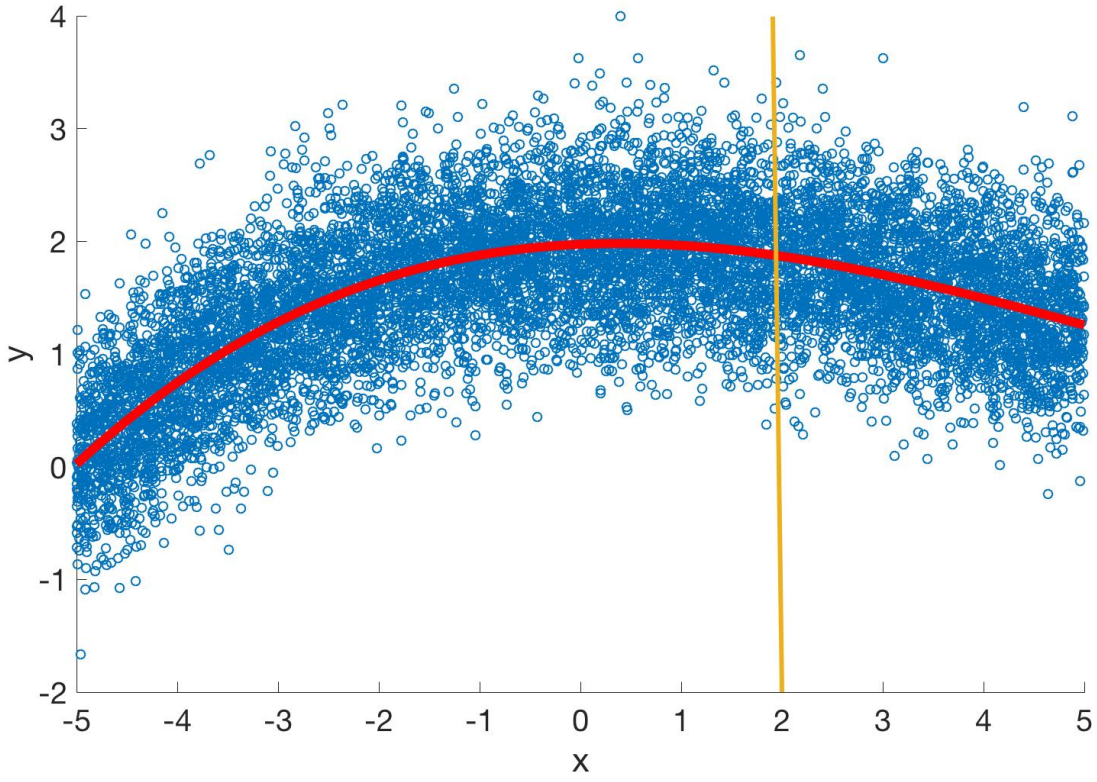
Unsupervised Learning

- No outcome variable, just a set of predictors (features) measured on a set of samples.
- objective is more fuzzy find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation.
- difficult to know how well your are doing.
- different from supervised learning, but can be useful as a pre-processing step for supervised learning.
- Examples
  - Cocktail Party problem [e.g., see this video]
  - [Just like the Facebook tagging system] you have collection of photos of multiple people, without information who is on which. You want to divide this dataset into multiple piles, each with photos of one individual

Meaning of Regression Function

Best Predictive Model if We Had Enough Data

- Suppose we had many data samples (X, Y) as below
- What would have been a good estimate of Y for $X = 2$



- In mathematical terms what we are after is $E(Y|X = 2)$

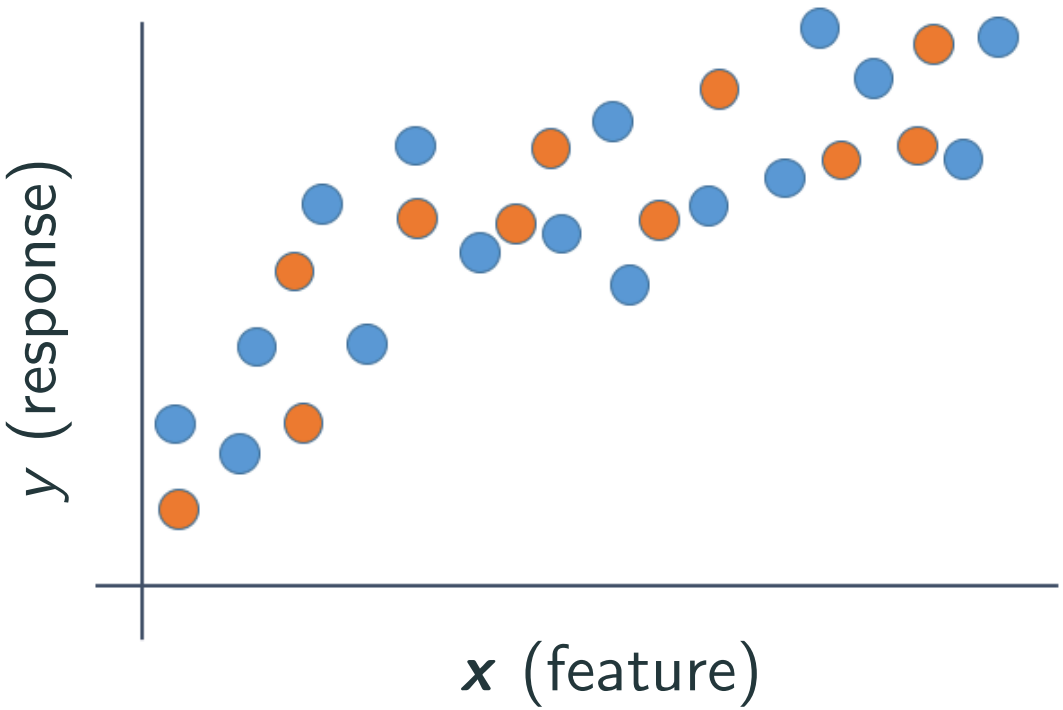
Best Predictive Model if We Had Enough Data

- In an ideal setting of having many data, the reasonable value of Y corresponding to $X = \boldsymbol{x}$ is the average of all responses at $\boldsymbol{X} = \boldsymbol{x}$
- It can be mathematically shown that $f(\boldsymbol{x}) = E(Y|\boldsymbol{X} = \boldsymbol{x})$ is the function that minimizes $E[(y - g(\boldsymbol{X}))^2|\boldsymbol{X} = \boldsymbol{x}]$ over all functions g at all points $\boldsymbol{X} = \boldsymbol{x}$.
- This ideal $f(\boldsymbol{x}) = E(Y|\boldsymbol{X} = \boldsymbol{x})$ is called the **regression function**

Training and Test Sets

How to Assess Model Accuracy in Practice

- As we said before, we don't normally have access to $f(\boldsymbol{x})$, so how can we assess the accuracy of an estimated model $\hat{f}(\boldsymbol{x})$?
- Usually we split the data into a training set and a test set to later evaluate the accuracy



Parametric vs Nonparametric Models

Parametric vs Nonparametric Models

- In parametric models, we use some predefined forms for the model $\hat{f}$ that we would like to fit
- For instance in linear regression we use the following general form and the fitting aims to find the coefficients

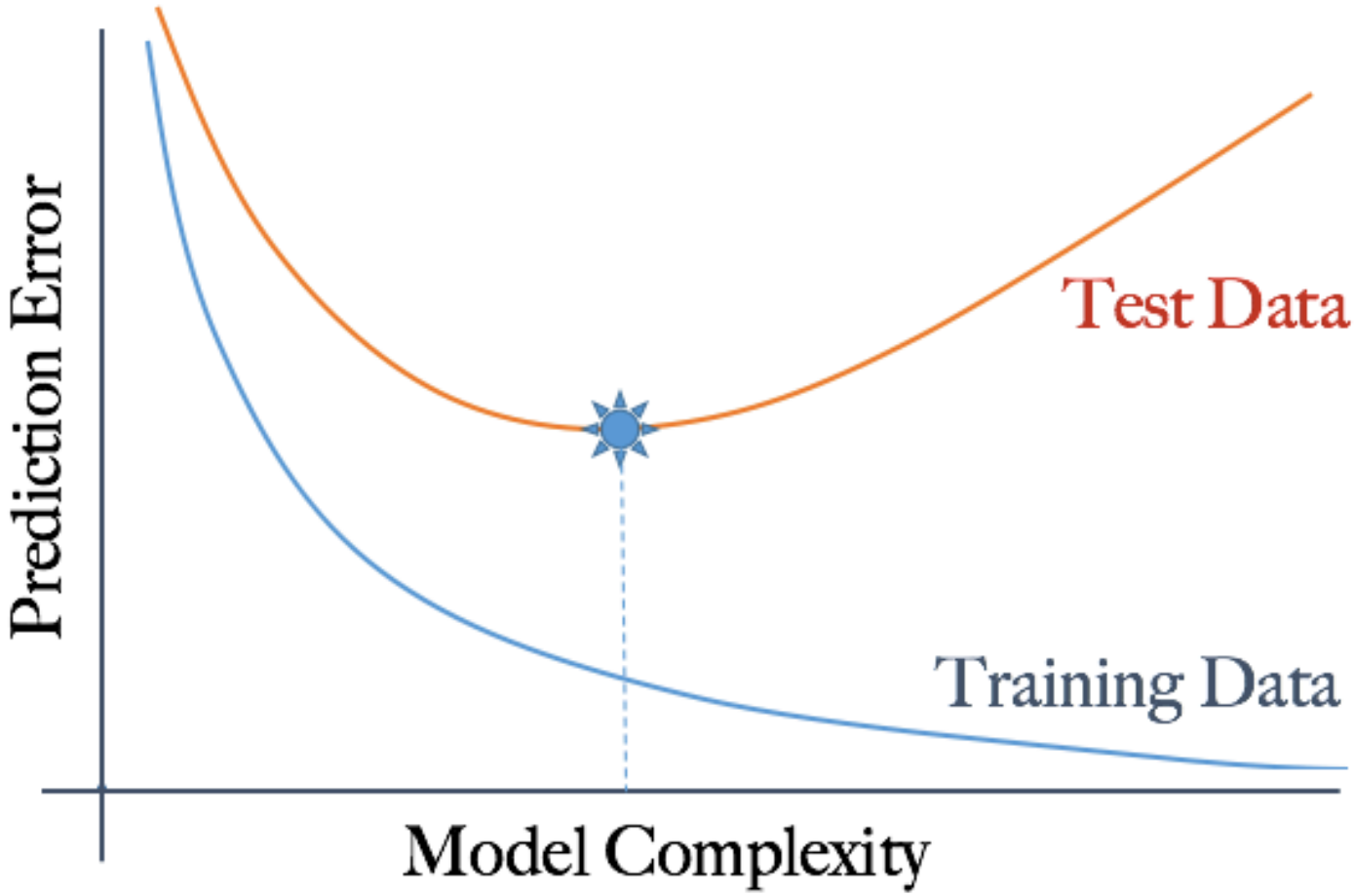
$$Y = f_L(\boldsymbol{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p$$

- Non-parametric methods do not make explicit assumptions about the functional form of the fit. Instead they seek an estimate of f that gets as close to the data points as possible without being too rough or wiggly
- Thin-plate spline is an example of nonparametric fit, where you can control the smoothness of the fit vs its flexibility to match the training data

Test/Training Curve

Typical Curve

– The balance between model flexibility and test error typically looks as below



Test Error in Terms of Bias, Variance and Noise

Bias vs Variance(Theory)

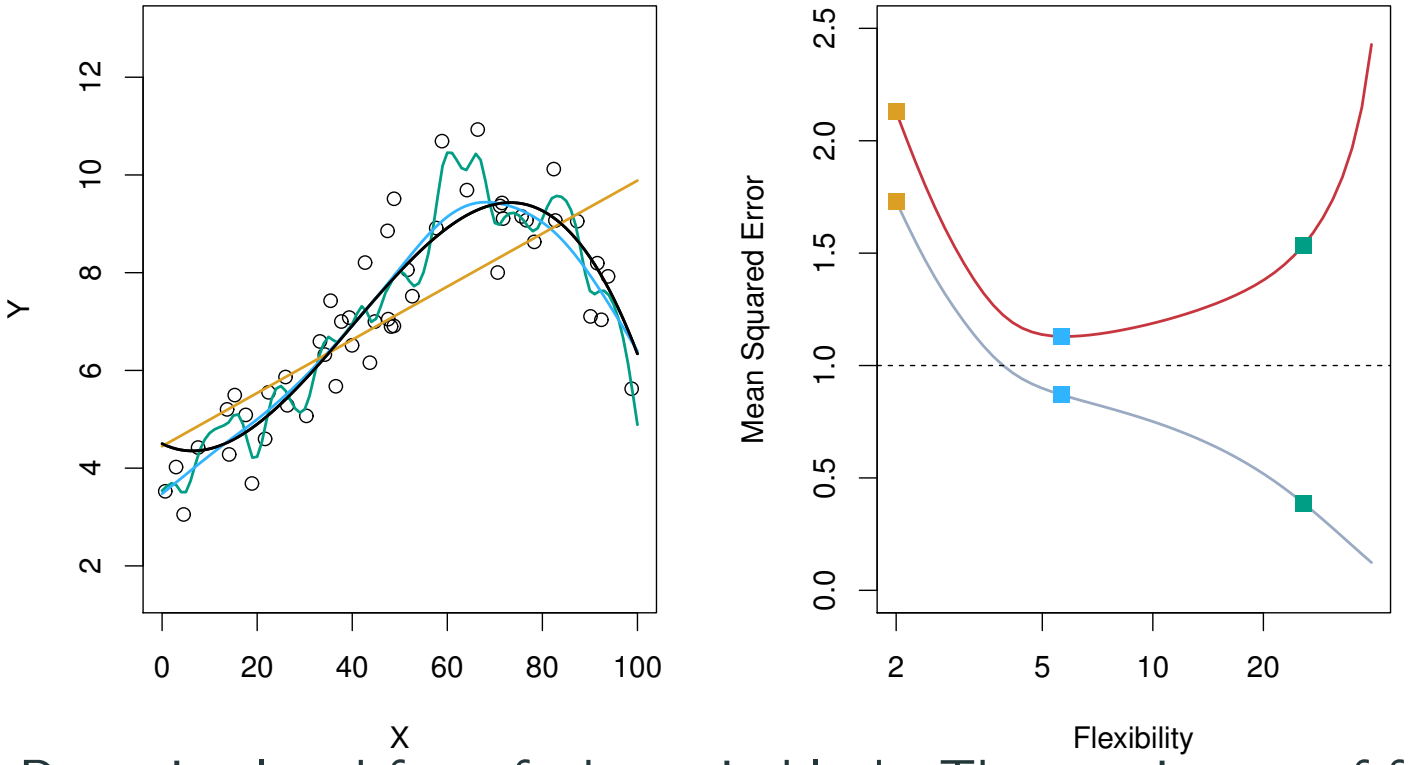
- Suppose we fit a model $\hat{f}$ and we have a large test set to accurately calculate the MSR with respect to the test data
- Assuming that $(\boldsymbol{x}_t, Y_t)$ is a point in the test set and T is the training set, we generally have

$$E_T\left(Y_t - \hat{f}(\boldsymbol{x}_t)\right)^2 = \text{var}(\hat{f}(\boldsymbol{x}_t)) + \left(\text{Bias}(\hat{f}(\boldsymbol{x}))\right)^2 + \text{var}(\epsilon)$$

$$\text{Bias}(\hat{f}(\boldsymbol{x})) = E_T(\hat{f}(\boldsymbol{x}_t) - f(\boldsymbol{x}_t))$$

- [Lets derive the equation above]
- Normally as we make $\hat{f}$ more flexible by making it complex (using more sophisticated formulations and more features) the **model variance** term $\text{var}(\hat{f}(\boldsymbol{x}_t))$ increases. This on the other hand decreases the bias (which we want to happen)
- So reducing the test error becomes a trade-off between the bias and the variance

Bias vs Variance



Left: Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves). Right: Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.

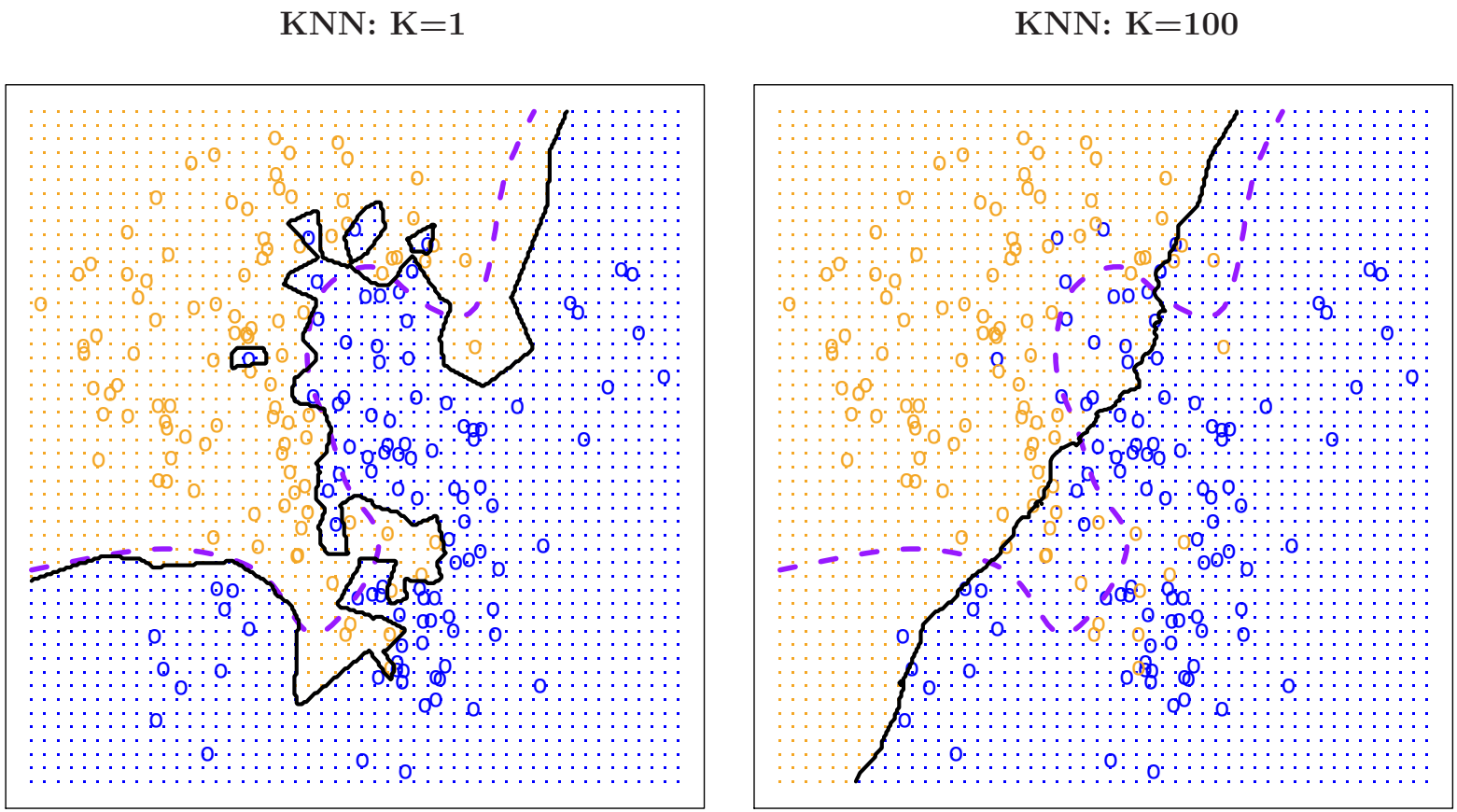
Trade Offs

We just discussed the bias vs variance trade off. In general we deal with various trade offs

- Prediction accuracy versus interpretability:
  e.g., 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? (just discussed on the figures above)
- Parsimony versus black-box:
  We often prefer a simpler model involving fewer variables over a black-box predictor involving them all

Classification

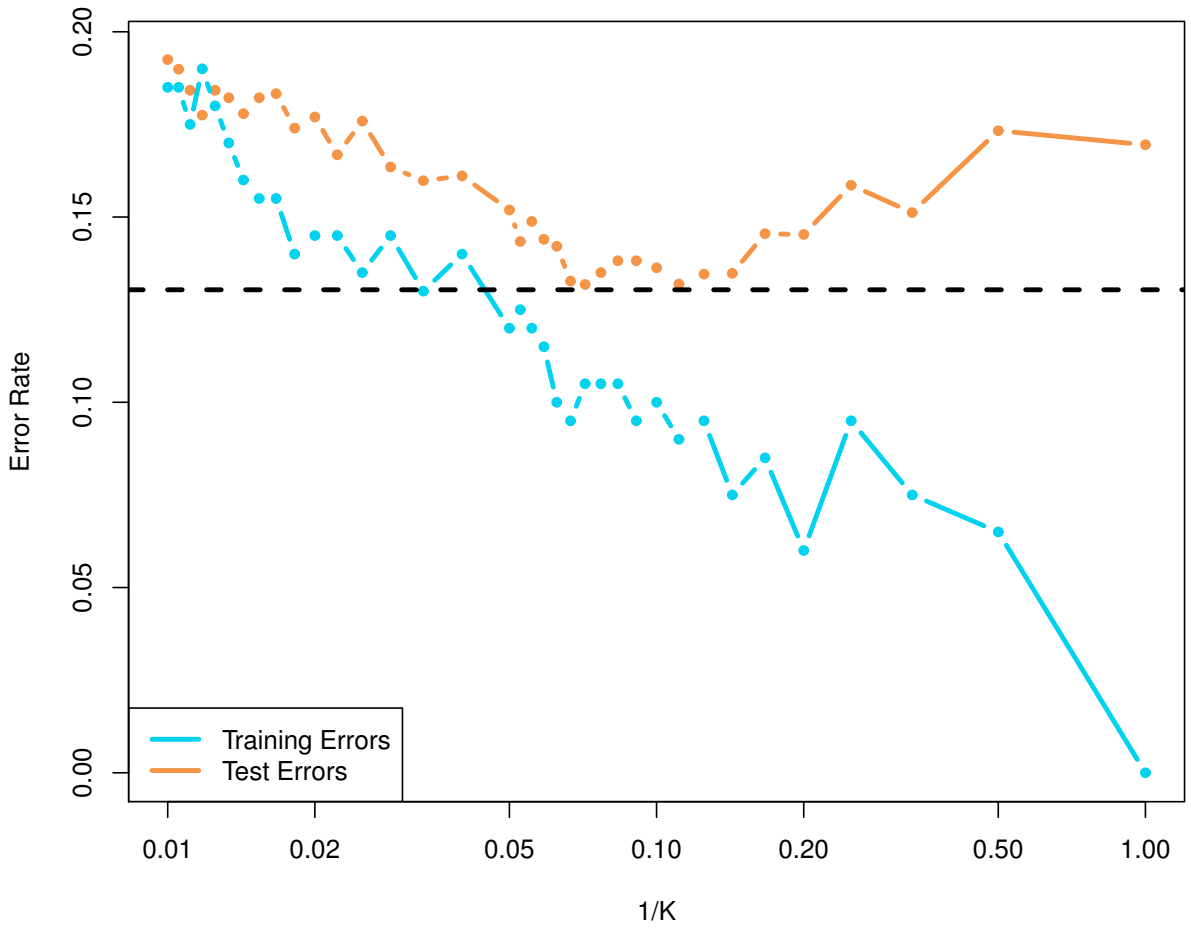
- How overfitting looks for classification problems
- K- Nearest Neighbors Approach (the dense grid is somehow our test)



(KNN Classifier)

Classification

- The Trade Off Curve
- K- Nearest Neighbors Approach



(Trade Off)

Lecture 2 Review

Linear Models

Some Basic Probability Overview

- For a continuous random variable X we often define a probability density distribution $f_X(x)$ where

$$\mathbb{P}(a \leq X \leq b) = \int_a^b f_X(x)\mathrm{d}x$$

- A random variable X is normally distributed with mean μ and variance σ^2 (denoted as $\mathcal{N}(\mu, \sigma^2)$), when

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Expectation of the weighted sum: if $x_1, \ldots, x_n$ are random variables with mean $\mathbb{E}(x_i) = \mu_i$, then for constants α_i:

$$\mathbb{E}(\alpha_1 x_1 + \ldots \alpha_n x_n) = \alpha_1\mu_1 + \ldots + \alpha_n\mu_n$$

- Variance of the weighted sum: if $x_1, \ldots, x_n$ are **independent** random variables with variance $var(x_i) = \sigma_i^2$, then

$$var(\alpha_1 x_1 + \ldots \alpha_n x_n) = \alpha_1^2\sigma_1^2 + \ldots + \alpha_n^2\sigma_n^2$$

Hypothesis Testing

Hypothesis Testing

In hypothesis testing only one of these two cases happens:

- You **reject** H_0 and **accept** H_1 since you have enough evidence in favor of H_1
- You **fail to reject** H_0, since you don not have enough evidence to support H_1

While you see a lot of documents talking about "*accepting* H_0", technically you should use the term *"fail to reject"*

- It might be the case that H_0 is false, but your data is not enough to reject it (does not mean you should accept it)

Example: H_0: Tim is innocent H_1: Tim is guilty

If you have enough to support Tim is guilty, you reject H_0. If you do not have enough to show that Tim is guilty (failure to reject H_0), that does not mean he is innocent (accepting H_0)

Hypothesis Testing: Types of Error

	Reject H_0 (accept H_1)	Do not reject H_0
H_0 true in reality (probability)	Type I error α	Correct decision $1-\alpha$
H_1 true in reality (probability)	Correct decision $1-\beta$	Type II error β

- α is a small number that we determine and is called the significance level (the probability of making type I error)
- We decide on how confident we want to make a claim in favor of H_0 and $1-\alpha$ is our confidence about this
- Normally people take α to be 0.05 or 0.01, giving you 95% or 99% chance of validity in making the argument in support of H_0
- We also have a type II error (calling $1-\beta$ the power of the test), but here we do not want to focus on that

Your Take Away from Hypothesis Testing

- We have some independent samples from a distribution and we guess something about our data
- We form a hypothesis test with some null and alternate hypotheses
- We fix some value for the significance α, meaning that $1 - \alpha$ is how confident we want to be in making our claim
- We form a test statistic (the resulting random variable can have a very complicated distribution)
- We calculate the p-value:
  - If p-value$\leq \alpha$: reject H_0 (accept H_1)
  - If p-value$> \alpha$: fail to reject H_0

Linear Regression!

More on Simple Linear Regression

- We consider the model

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

- β_0 and β_1 are two unknown constants that represent the intercept and slope, also known as coefficients and ϵ is the error term.
- We are given samples of the form $(x_1, y_1), \ldots, (x_n, y_n)$, using which we try to fit some values $\hat{\beta}_0$ and $\hat{\beta}_1$ for the model coefficients
- After this fit we can predict future responses to a test sample x_t, using

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

Determining the Model Coefficients

– Remember that we had samples $(x_1, y_1), \ldots (x_n, y_n)$ and we would like to determine $\hat{f}(x)$ by

$$\min \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

– Using our simple model we would like to decide β_0 and β_1 such that the **Residual Sum of Squares** (RSS)

$$RSS = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

is minimized

Determining the Model Coefficients

– Taking the derivative with respect to β_0 and β_1 and setting it to zero, for $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ we get

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \qquad \text{and} \qquad \hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

– We may use the simple equalities:

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

to get the final equations

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

What is the Confidence Interval for the Coefficients Obtained?

- We can define confidence intervals that the true β_1 and β_0 are in it with 95% confidence
- For $\sigma^2 = \text{var}(\epsilon)$, we define

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

- The 95% confidence intervals for β_0 and β_1 are

$$\left[\beta_0 - 2\text{SE}(\hat{\beta}_0), \beta_0 + 2\text{SE}(\hat{\beta}_0) \right], \quad \left[\beta_0 - 2\text{SE}(\hat{\beta}_0), \beta_0 + 2\text{SE}(\hat{\beta}_0) \right]$$

(see the code)

Is There a Relationship Between x and y?

- We would like to know if there is really a relationship between x and y or if the fit is useless?
- We form a hypothesis testing for β_1 (if it is zero, then x and y are not related):
  - $H_0 : \beta_1 = 0$
  - $H_1 : \beta_1 \neq 0$
- Our test statistic is chosen to be $t = \frac{\hat{\beta}_1 - \beta_1}{SE(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$
- We look this up in the t-distribution table and find the p-value

(see the code)

If p-value$\leq \alpha$: reject H_0 (accept H_1)
If p-value$> \alpha$: fail to reject H_0

- For the example provided p-value $= 2 \times 10^{-16}$ and we reject H_0 (meaning that x and y are related)

How Well does the Model Explain the Data?

- We can also answer the question of how well our fitted model explains the data by defining another statistic:

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

where TSS is the Total Sum of Squares

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2, \quad TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

- For general regression problems (not only the simple one) R^2 statistic measures the proportion of variability in y that can be explained by x
- R^2 close to 1 indicates that our model explains a large proportion of the response variability, and R^2 close to zero indicates that our model cannot explain much of the variability in response

`(see the code)`

Multiple Linear Regression

Multiple Linear Regression

– We would like to minimize the following squared error

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_1^{(i)} - \beta_2 x_2^{(i)} \ldots - \beta_p x_p^{(i)} \right)^2$$

– Consider using the following matrices/vectors

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \ \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \ \boldsymbol{X}_{n \times (p+1)} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{pmatrix}$$

note that matrix $\boldsymbol{X}$ has the samples along the rows

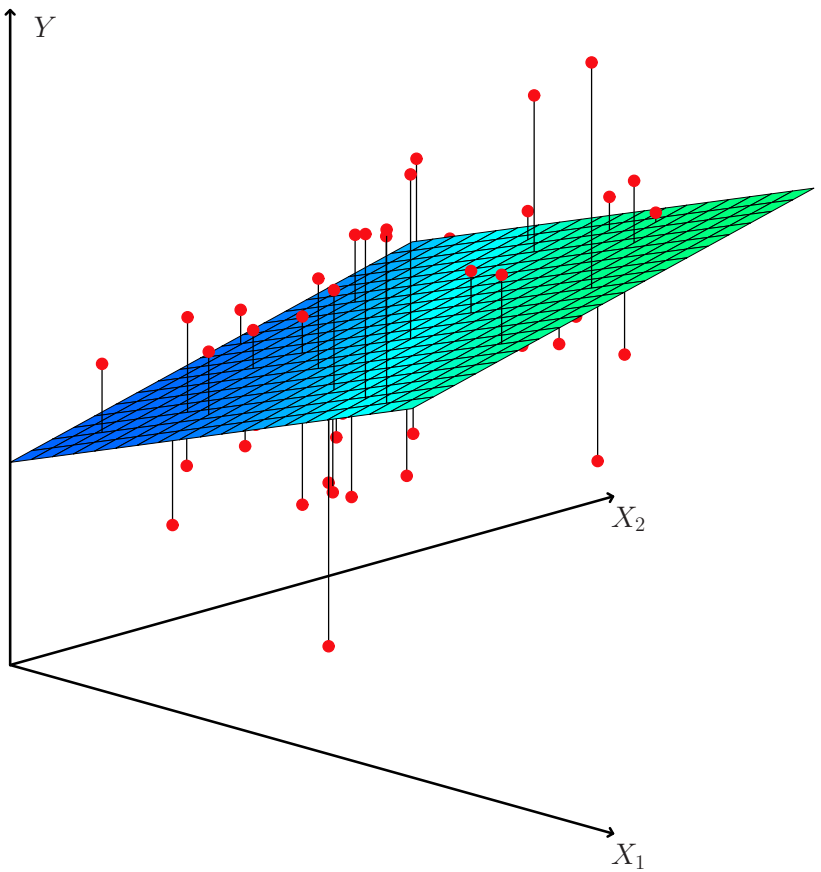
– Then, it is straightforward to see that

$$RSS = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$$

Multiple Linear Regression

– Similar to what we did before we can set $\partial RSS/\partial \boldsymbol{\beta} = 0$ (requires little bit of knowing how to do vector/matrix derivatives) and get

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



Are the Features and Response Related?

- We would like to know if at least one of the features $x_1, \ldots, x_p$ is useful in predicting the response.
- We form a hypothesis testing as follows:
  - $H_0 : \beta_1 = \beta_2 = \ldots = 0$
  - H_1 : at least one β_i is non-zero
- For $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$, our test statistic is chosen to be

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

which turns to have an F distribution

If p-value$\leq \alpha$: reject H_0, If p-value$> \alpha$: fail to reject H_0

- We can either use F-distribution tables and find the p-value; or use this rule:*if F is much larger than 1, we reject H_0; if F is very close to 1, we fail to reject H_0*

(see the code)

What are the Best Selection of Features? Forward/ Backward

- As noted sometimes some features can be redundant and we would like to find the best subset of features that predicts well and is not redundant
- In general this problem is "NP-hard" (computationally very hard) and we need to assess 2^p models
- There are some heuristics to do this that we will see later:
  - Forward selection: model p regressions each with only one feature, pick the one with least RSS, repeat it with selected feature and combination of others, ...
  - Backward selection: Start with all features and remove variable with largest p-value, run a new regression, remove variable of largest p-value, ...

How Can We Handle Categorical Features?

- Sometimes our features do not take numerical values, instead they take categorical values
- **Example**: In a regression problem we have a feature called `ethnicity`, which takes possible values of `Asian, Caucasian, African-American`
- We can introduce 2 dummy variables (features) e_A, e_C
  - $e_A = 1,\ e_C = 0$ if Asian
  - $e_A = 0,\ e_C = 1$ if Caucasian
  - $e_A = 0,\ e_C = 0$ if African-American
- Basically, for every categorical feature that has L levels, we need to define $L - 1$ dummy variables

Can We Only Fit Flat Curves with Linear Regression?

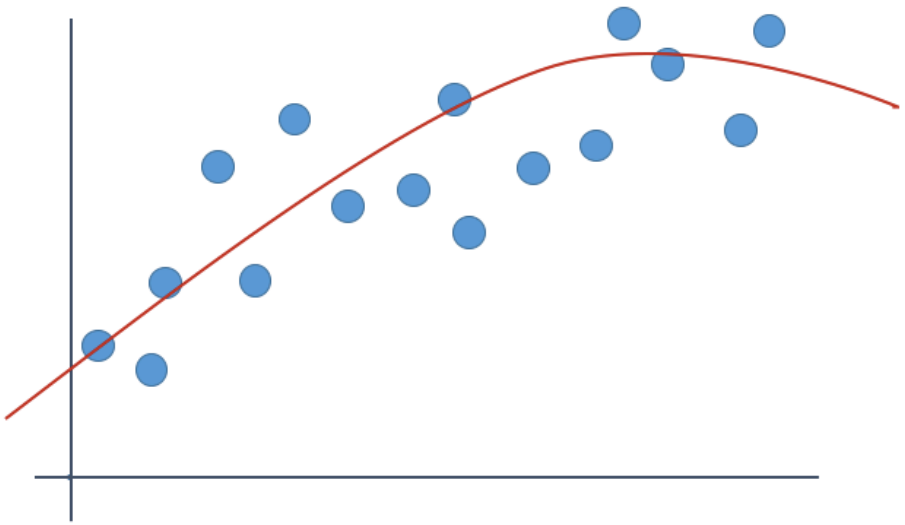
- Based on the equation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_p x_p + \epsilon$$

  one might get the impression that linear regression is only good for fitting flat surfaces (linear manifolds)
- If we include powers of a feature, e.g., $x_1, x_1^2, \ldots$ or cross terms between the features, e.g., $x_1 x_2, x_2 x_3 x_5$, etc, then we can also fit nonlinear surfaces
- Of course knowing what powers or what cross terms to include in the feature list is not always clear

Can We Only Fit Flat Curves with Linear Regression?

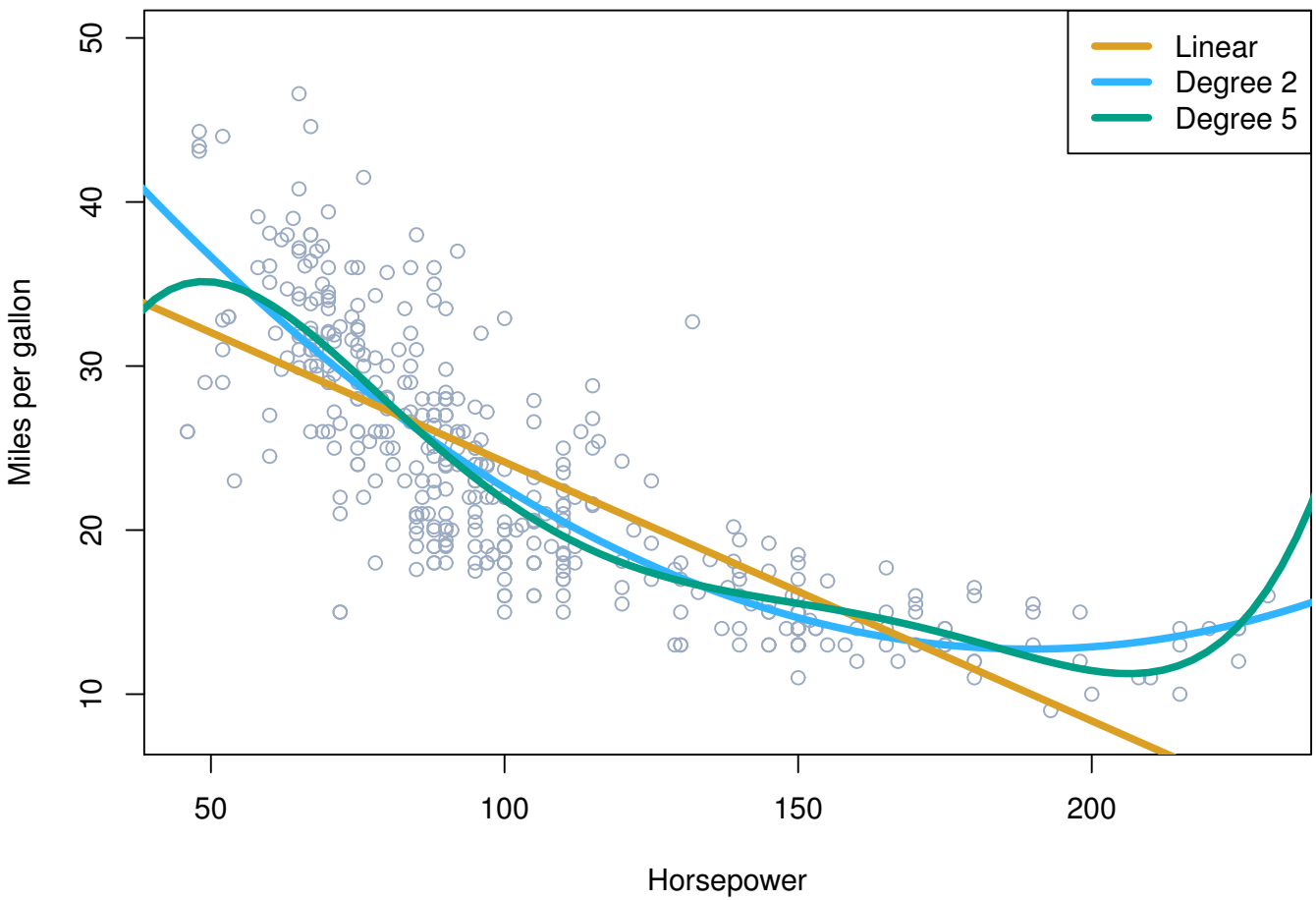
– **Example**: For a problem wtih only one feature, we have a set of point that look lying on a parabola, we use the regression $y = \beta_0 + \beta_1 x + \beta_2 x^2$



Can We Only Fit Flat Curves with Linear Regression?

– **Example**: Regressing Mile per Gallon in terms of the Horse Power

$$\texttt{mpg} = \beta_0 + \sum_{i=1}^{p} \beta_i (\texttt{horsepower})^i$$



Classification

Brief Overview of Maximum Likelihood

- Maximum likelihood (ML) is a statistical estimation technique
- The main goal in ML is estimating the parameters of a statistical model given some sample observations
- Let $x_1, x_2, \cdots x_n$ be samples from a distribution with some unknown parameter θ and joint distribution

$$f(x_1, x_2, \cdots, x_n|\theta)$$

- The maximum likelihood estimate of θ based on the observations $\tilde{x}_1, \tilde{x}_2, \cdots \tilde{x}_n$ is

$$\theta_{ML} = \underset{\theta}{\text{argmax}} \; f(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n|\theta)$$

- When $x_1, x_2, \cdots x_n$ are i.i.d samples from a distribution $f(\cdot)$, then

$$f(x_1, x_2, \cdots, x_n|\theta) = f(x_1|\theta)f(x_2|\theta)\cdots f(x_n|\theta)$$

Brief Overview of Maximum Likelihood

Example. We have a normal distribution $\mathcal{N}(\mu, 1)$ and we do not know μ. We take 5 independent samples from this distribution and the values turn out to be

$$\tilde{x}_1 = 2.5377,\ \tilde{x}_2 = 3.8339,\ \tilde{x}_3 = -0.2588,\ \tilde{x}_4 = 2.8622,\ \tilde{x}_5 = 2.3188,$$

what is the ML estimate of μ.

Solution. If we take 5 independent samples $x_1, x_2, \cdots x_5$ from a normal distribution $\mathcal{N}(\mu, 1)$, their joint distribution is

$$f(x_1, x_2, x_3, x_4, x5|\mu) = \prod_{i=1}^{5} \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x_i - \mu)^2}{2}),$$

some basic calculus yields $\mu_{ML} = \frac{\tilde{x}_1 + \tilde{x}_2 + \cdots + \tilde{x}_5}{5} = 2.2587$ (why?)

Binary Classification

- In simple regression for a single feature x we fitted a line $y = \beta_0 + \beta_1 x$ to the data
- In binary classification with only one feature, we don't have values any more, but two classes (say class 0 and class 1)
- Can we do the fit in a way that the sign of $\beta_0 + \beta_1 x$ becomes an indicator of the class for us?
- In other words, for a given feature x_t, we make a decision based on the following:

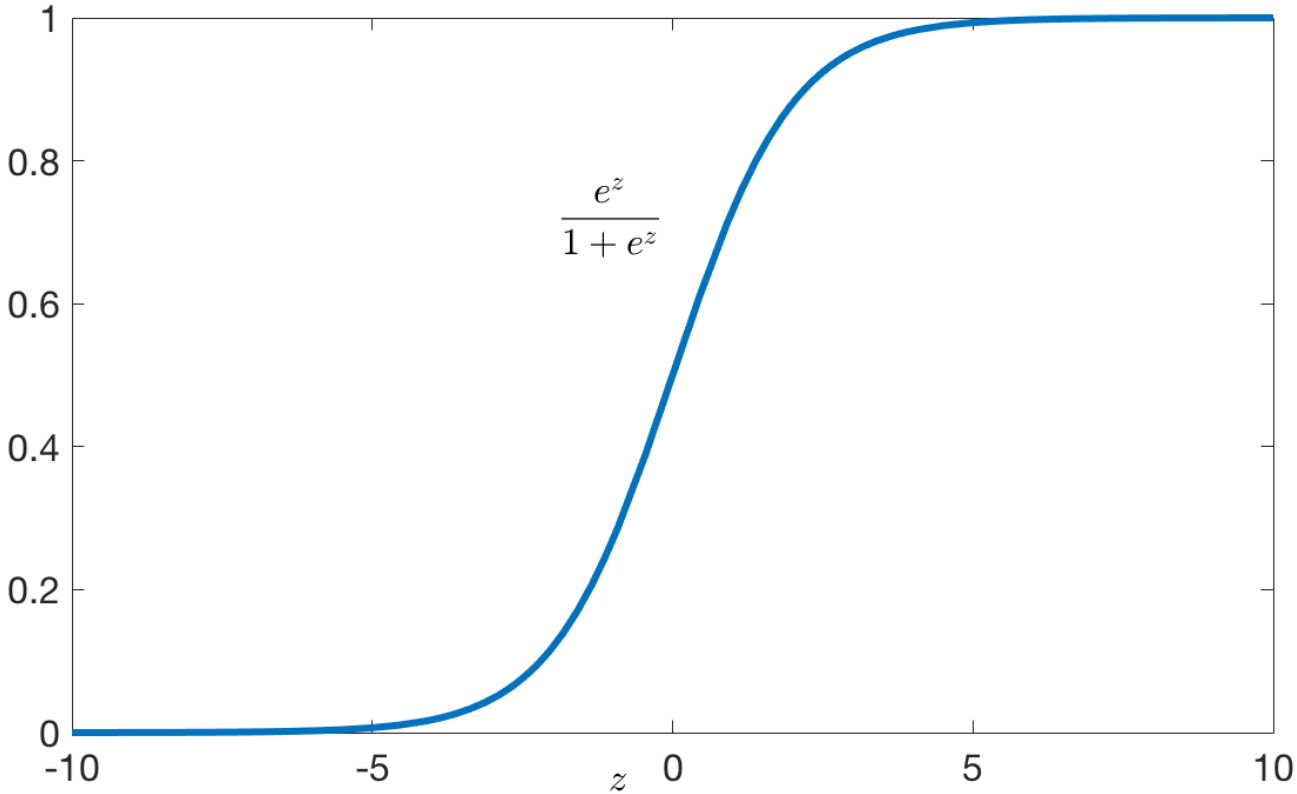
$$y_t = \begin{cases} 1 & \beta_0 + \beta_1 x_t > 0 \\ 0 & \beta_0 + \beta_1 x_t < 0 \end{cases},$$

- A smooth function that takes almost binary values $0, 1$ based on the sign of the input z is

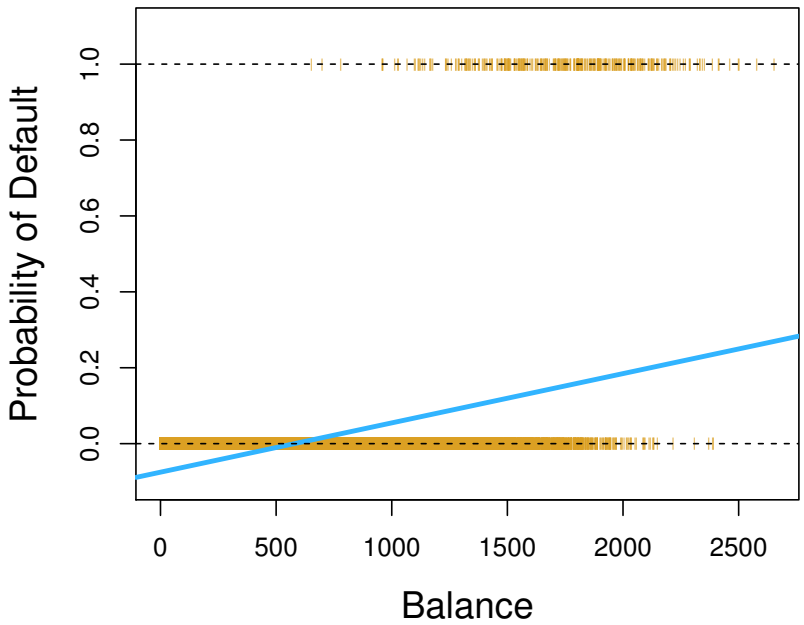
$$\frac{e^z}{1+e^z} \approx \begin{cases} 1 & z >> 0 \\ 0 & z << 0 \end{cases}$$

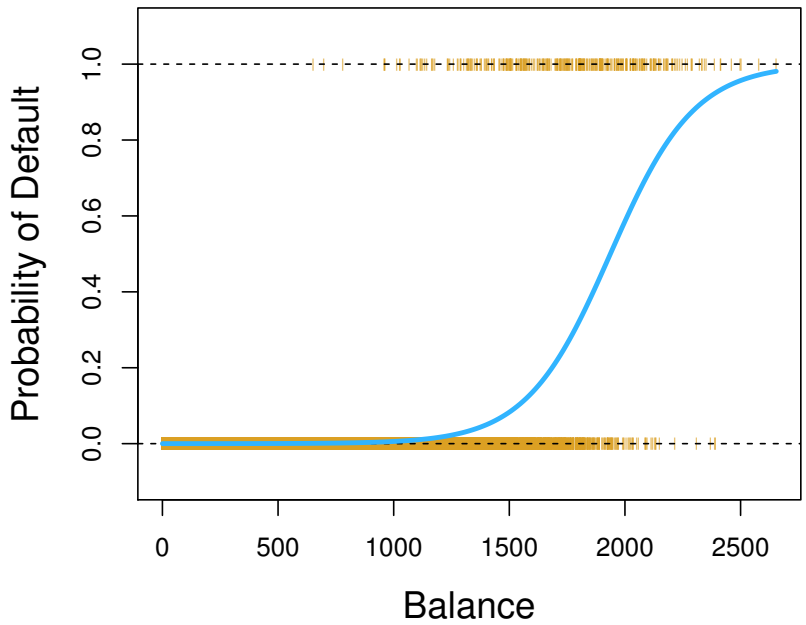
Binary Classification

– When we have a smooth approximation of the sign function, learning the parameters β_0 and β_1 is numerically easier



Binary Classification





Trying to treat the classification problem as a regression problem does not produce reasonable results!

How Does Binary Classification Work?

- We somehow learn β_0 and β_1 from the training data (will be explained soon)
- We are given a test point x_t, for which we evaluate $\beta_0 + \beta_1 x_t$
- We pass this quantity to our smooth sign approximation

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

- If $p(x_t)$ was closer to 1 our prediction of the class for x_t is class one (e.g., $p(x_t) = 0.7$) and if $p(x_t)$ was closer to 0 our prediction of the class for x_t is class zero (e.g., $p(x_t) = 0.3$)
- Now that $p(\cdot)$ generates some value between zero and one for us, one immediate interpretation for it is being the probability of label 1

$$p(x_t) = \mathbb{P}(y = 1 | x_t) = 1 - \mathbb{P}(y = 0 | x_t)$$

so if $p(x_t) = 0.7$, then the test label is 1 with probability 0.7, and 0 with probability 0.3

How to Do the Training for the Simple Logistic Regression?

– We do not have a closed form solution like the linear regression case, and each fit requires solving an optimization (max likelihood) problem

What Happens for More than One Feature?

- In case of multiple features, only minor modification is required
- We still try to maximize $\prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i}$, but now we have

$$p(x_t) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}$$

- We run the maximization to estimate $\beta_0, \beta_1, \cdots, \beta_p$
- In practice you never have to do the maximization and most software such as R, Python and Matlab have packages to do that numerically

What Happens for More than Two Classes?

- Example, based on some features such as city, year of education and number of publications, classify the students of a class into undergrads, Masters, and PhDs
- Recall our method of classification in the binary case, we evaluated $p(x_t)$ which was technically $\mathbb{P}(Y = 1|x_t)$ and if it was closer to 1 then our class prediction was 1, if it was small, then $\mathbb{P}(Y = 0|x_t) = 1 - \mathbb{P}(Y = 1|x_t)$ would be large and our prediction is class zero
- One way of interpreting this is evaluating $\mathbb{P}(Y = k|x_t)$ for $k = 0, 1$ and the k that produces the largest value for $\mathbb{P}(Y = k|x_t)$ is our predicted label
- Now for K labels, we evaluate $\mathbb{P}(Y = k|x_t)$ for $k = 1, 2, \cdots, K$ and the k that produces the largest value for $\mathbb{P}(Y = k|x_t)$ is our predicted label

What Happens for More than Two Classes?

- For K labels, we evaluate $\mathbb{P}(Y = k|x_t)$ for $k = 1, 2, \cdots, K$ and the k that produces the largest value for $\mathbb{P}(Y = k|x_t)$ is our predicted label
- When we have $K > 2$ labels (e.g., $y \in$ {while, yellow, green}) and p features $x_1, x_2, \cdots, x_p$, we fit K models parametrized by

$$\begin{aligned} \text{Label 1:} \quad & \{\beta_0^{(1)}, \beta_1^{(1)}, \cdots, \beta_p^{(1)}\} \\ \text{Label 2:} \quad & \{\beta_0^{(2)}, \beta_1^{(2)}, \cdots, \beta_p^{(2)}\} \\ & \vdots \\ \text{Label K:} \quad & \{\beta_0^{(K)}, \beta_1^{(K)}, \cdots, \beta_p^{(K)}\} \end{aligned}$$

- For this problem we consider the following form:

$$p_k(\boldsymbol{x}) = \mathbb{P}(Y = k|\boldsymbol{x}) = \frac{e^{\beta_0^{(k)} + \cdots + \beta_p^{(k)} x_p}}{e^{\beta_0^{(1)} + \cdots + \beta_p^{(1)} x_p} + \ldots + e^{\beta_0^{(K)} + \cdots + \beta_p^{(K)} x_p}}$$

- What is the sum of all $\mathbb{P}(Y = k|\boldsymbol{x})$ for a fixed $\boldsymbol{x}$?

Linear and Quadratic Discriminant Analysis

Linear Discriminant Analysis (LDA)

- Recall

$$\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x}) = \frac{\mathbb{P}(Y = \ell)\mathbb{P}(\boldsymbol{X} = \boldsymbol{x} | Y = \ell)}{\sum_{k=1}^{K} \mathbb{P}(Y = k)\mathbb{P}(\boldsymbol{X} = \boldsymbol{x} | Y = k)} = \frac{\pi_\ell f_\ell(\boldsymbol{x})}{\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x})}$$

- The purpose of LDA is learning a model for $\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x})$
- In the formulation above, $f_\ell(\boldsymbol{x})$ is in a sense the distribution we consider for the data points in class ℓ, and π_ℓ is the probability that we pick some random sample and it belongs to class ℓ
- In LDA, we assume that all $f_\ell(\boldsymbol{x})$ have a multivariate normal distribution with similar covariances and different means, i.e.

$$f_\ell(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_\ell)\right)$$

- Unlike logistic regression, which involved a rather complicated maximization for learning, in LDA we have closed form expressions for π_ℓ, $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}$ and classifying new test points becomes very easy

Linear Discriminant Analysis (LDA)

– Given a training set $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)$ where the responses y can take K distinct class values $1, 2, \cdots, K$, we can easily learn the LDA model by calculating π_ℓ, $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}$ via (considering c_ℓ to be the index of samples in class ℓ)

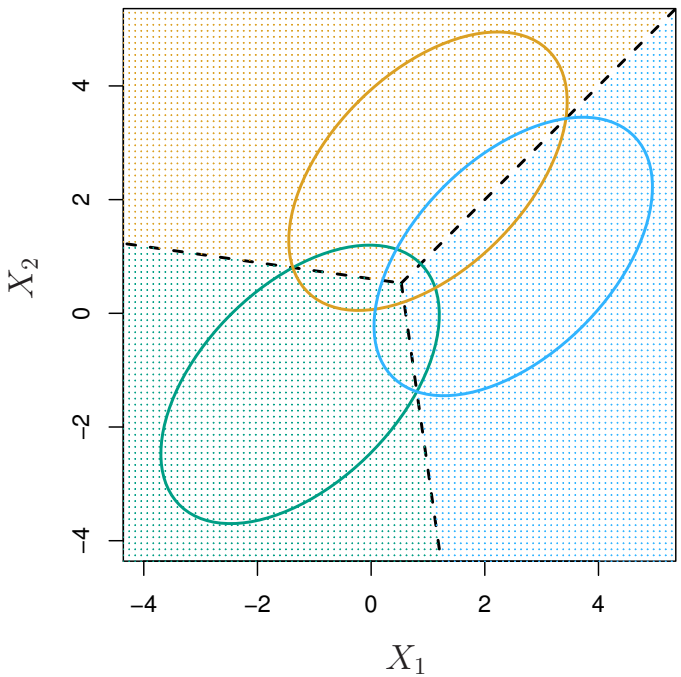
$$\hat{\pi}_\ell = \frac{\#\text{ of elements in } c_\ell}{n}$$

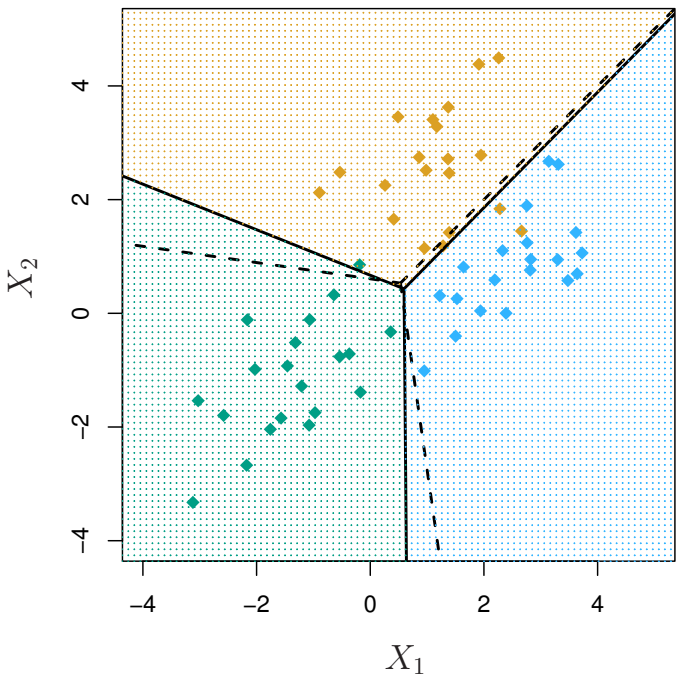
$$\hat{\boldsymbol{\mu}}_\ell = \frac{1}{\#\text{ of elements in } c_\ell} \sum_{i \in c_\ell} \boldsymbol{x}_i$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i \in c_\ell} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\ell)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\ell)^\top$$

– After this point for a new test point $\boldsymbol{x}_t$ we have all that is needed to calculate $\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x}_t)$ for $\ell = 1, \cdots, K$ and pick as the label the one that is largest

Linear Discriminant Analysis (LDA)





Linear Discriminant Analysis (LDA)

- In practice to assign a label to a given test point $\boldsymbol{x}_t$ we do not need to calculate

$$\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x}_t) = \frac{\pi_\ell f_\ell(\boldsymbol{x}_t)}{\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_t)}$$

  and only comparing $\pi_\ell f_\ell(\boldsymbol{x}_t)$ is enough

- This reduces to evaluate

$$\delta_\ell = \boldsymbol{x}_t^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_\ell - \frac{1}{2} \boldsymbol{\mu}_\ell^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_\ell + \log \pi_\ell$$

  and pick as the class ℓ corresponding to the largest δ_ℓ

- You can find the decision boundary between class i and j by finding the points for which $\delta_i = \delta_j$

- [see the sample Matlab code]

Quadratic Discriminant Analysis (QDA)

- Recall

$$\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x}_t) = \frac{\pi_\ell f_\ell(\boldsymbol{x}_t)}{\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_t)}$$

- The purpose of QDA is learning a model for $\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x})$ in a more flexible way compared to LDA
- In QDA, we assume that all $f_\ell(\boldsymbol{x})$ have a multivariate normal distribution with similar covariances and different means, i.e.

$$f_\ell(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_\ell|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_\ell) \right)$$

- The main difference between LDA and QDA is in LDA we consider a single $\boldsymbol{\Sigma}$ for all classes, but in QDA we allow more flexibility by having a different covariance matrix for each class
- Similar to LDA, QDA can be learned easily and we can obtain closed form expressions for π_ℓ, $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}_\ell$

Quadratic Discriminant Analysis (QDA)

– Given a training set $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)$ where the responses y can take K distinct class values $1, 2, \cdots, K$, we can easily learn the QDA model by calculating π_ℓ, $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}_\ell$ via (considering c_ℓ to be the index of samples in class ℓ)

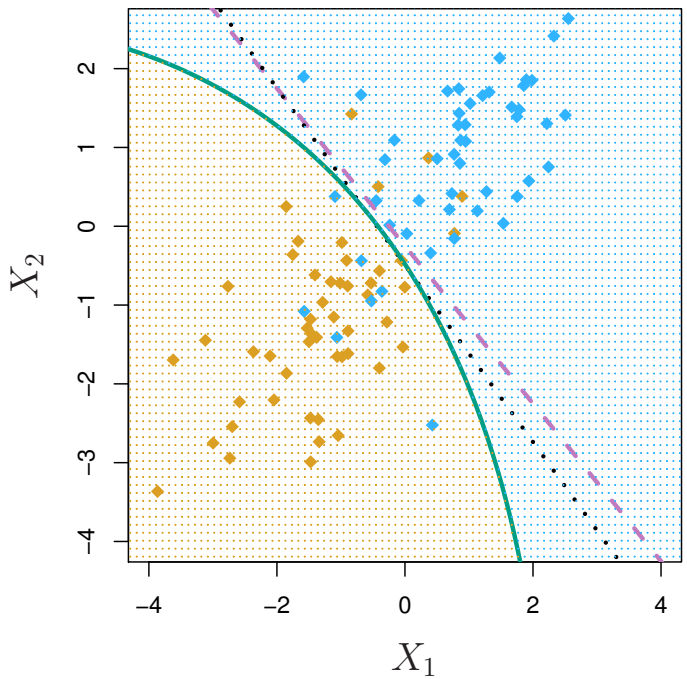
$$\hat{\pi}_\ell = \frac{\#\text{ of elements in } c_\ell}{n}$$

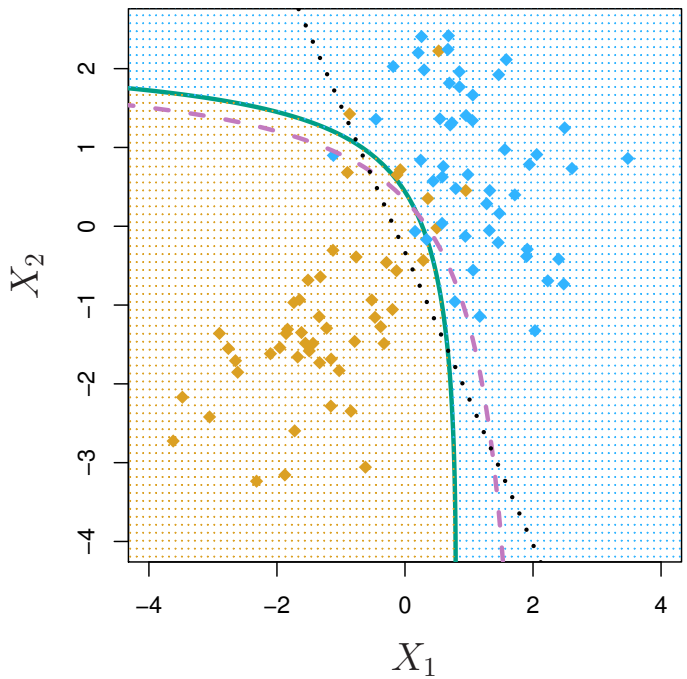
$$\hat{\boldsymbol{\mu}}_\ell = \frac{1}{\#\text{ of elements in } c_\ell} \sum_{i \in c_\ell} \boldsymbol{x}_i$$

$$\hat{\boldsymbol{\Sigma}}_\ell = \frac{1}{\#\text{ of elements in } c_\ell - 1} \sum_{i \in c_\ell} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\ell)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\ell)^\top$$

– After this point for a new test point $\boldsymbol{x}_t$ we have all that is needed to calculate $\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x}_t)$ for $\ell = 1, \cdots, K$ and pick as the label the one that is largest

Quadratic Discriminant Analysis (QDA)





Quadratic Discriminant Analysis (QDA)

- In practice to assign a label to a given test point $\boldsymbol{x}_t$ we do not need to calculate

$$\mathbb{P}(Y = \ell | \boldsymbol{X} = \boldsymbol{x}_t) = \frac{\pi_\ell f_\ell(\boldsymbol{x}_t)}{\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_t)}$$

  and only comparing $\pi_\ell f_\ell(\boldsymbol{x}_t)$ is enough

- This reduces to evaluate

$$\delta_\ell = -\frac{1}{2} \log \det(\boldsymbol{\Sigma}_\ell) - \frac{1}{2} (\boldsymbol{x}_t - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu}_\ell) + \log \pi_\ell$$

  and pick as the class the ℓ corresponding to the largest δ_ℓ

- [see the sample Matlab code]

Cross Validation

Introduction

- Recall that we fitted out models using training data and were interested in evaluating the performance with respect to independent test data

Introduction

- Recall that we fitted out models using training data and were interested in evaluating the performance with respect to independent test data
- To produce justifiable model reliability arguments, the test data should not be used in the training

Introduction

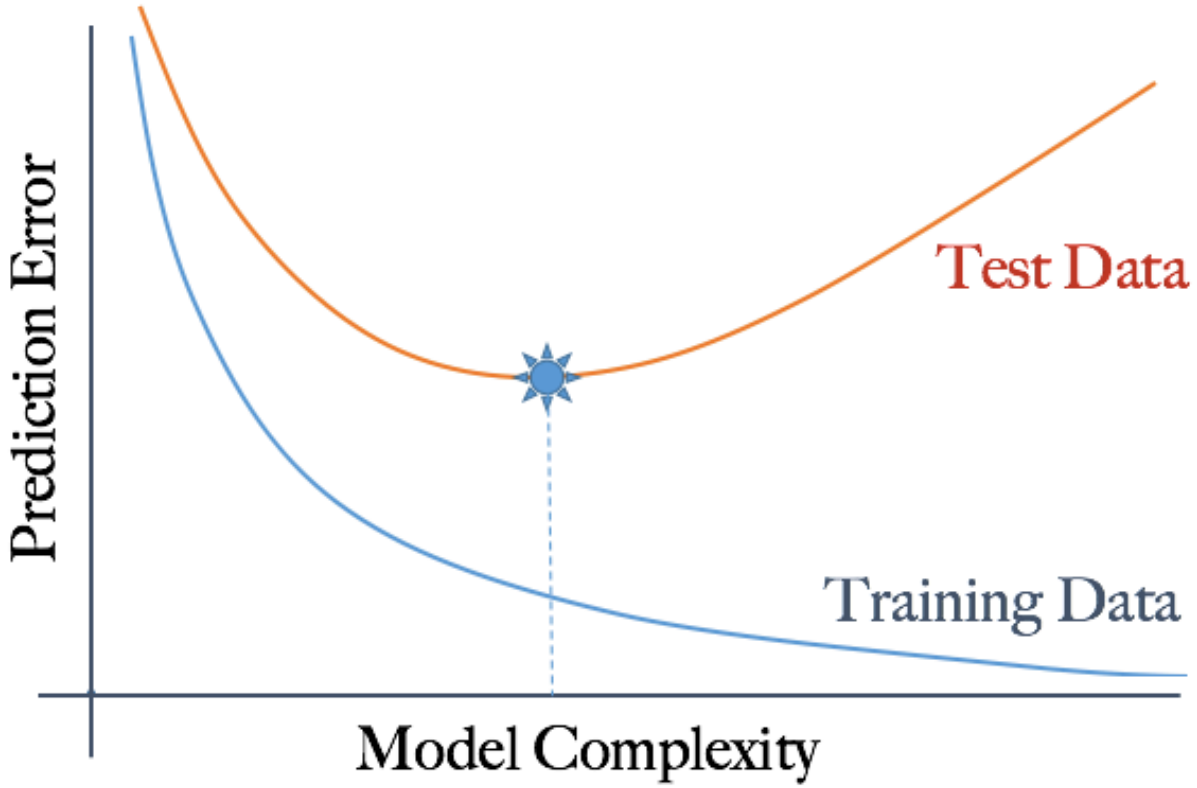
- Recall that we fitted out models using training data and were interested in evaluating the performance with respect to independent test data
- To produce justifiable model reliability arguments, the test data should not be used in the training
- If the model is evaluated against the training data the results can be very distracting

Introduction

- Recall that we fitted out models using training data and were interested in evaluating the performance with respect to independent test data
- To produce justifiable model reliability arguments, the test data should not be used in the training
- If the model is evaluated against the training data the results can be very distracting
- The training error rate is often quite different from the test error rate, and in particular the former can dramatically underestimate the latter (recall the accuracy vs complexity chart)

Training vs Test Model Evaluation

Recall this plot from the first session



Real-World Data Issues and Test Performance

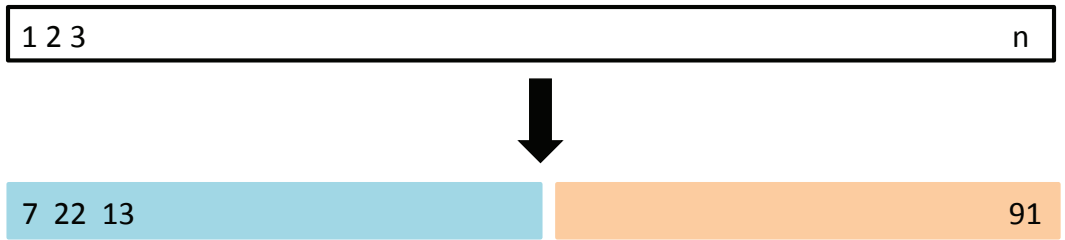
- A good evaluation is possible when a large test set is available

Real-World Data Issues and Test Performance

- A good evaluation is possible when a large test set is available
- Often such set is not available

Real-World Data Issues and Test Performance

- A good evaluation is possible when a large test set is available
- Often such set is not available
- We are interested in a class of methods that estimate the test error by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those (held out) observations



Validation Set Approach

- This is the standard approach we have been using so far

Validation Set Approach

- This is the standard approach we have been using so far
- We randomly divide the available set of samples into two parts: a training set and a validation or hold-out set (sometimes 50%-50% splitting, often 80%-20% splitting)

Validation Set Approach

- This is the standard approach we have been using so far
- We randomly divide the available set of samples into two parts: a training set and a validation or hold-out set (sometimes 50%-50% splitting, often 80%-20% splitting)
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set

Validation Set Approach

- This is the standard approach we have been using so far
- We randomly divide the available set of samples into two parts: a training set and a validation or hold-out set (sometimes 50%-50% splitting, often 80%-20% splitting)
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set
- The error with reference to the hold-out set is an approximation of the test error

example

- Recall the automobile data: Regressing Mile per Gallon in terms of the Horse Power

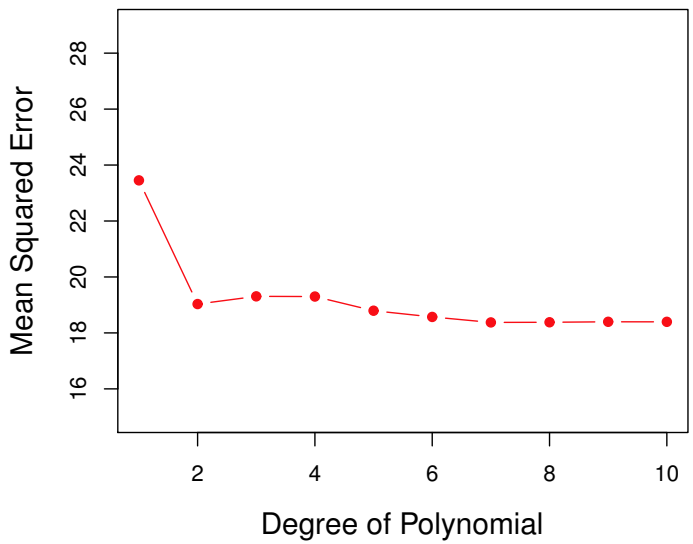
$$\texttt{mpg} = \beta_0 + \sum_{i=1}^{p} \beta_i \texttt{(horsepower)}^i$$

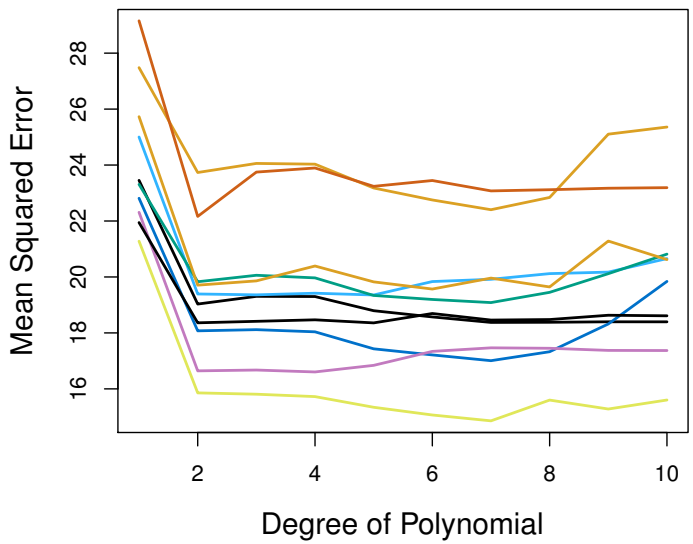
example

- Recall the automobile data: Regressing Mile per Gallon in terms of the Horse Power

$$\texttt{mpg} = \beta_0 + \sum_{i=1}^{p} \beta_i (\texttt{horsepower})^i$$

- We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.





Validation Set Cons & Pros

- The procedure is **simple to do** (as we have done so far) and only a subset of the observations those that are included in the training set rather than in the validation set are used to fit the model

Validation Set Cons & Pros

- The procedure is **simple to do** (as we have done so far) and only a subset of the observations those that are included in the training set rather than in the validation set are used to fit the model
- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set

Validation Set Cons & Pros

- The procedure is **simple to do** (as we have done so far) and only a subset of the observations those that are included in the training set rather than in the validation set are used to fit the model
- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set
- While the estimated test error vary a lot, **finding information such as model selection is still possible**

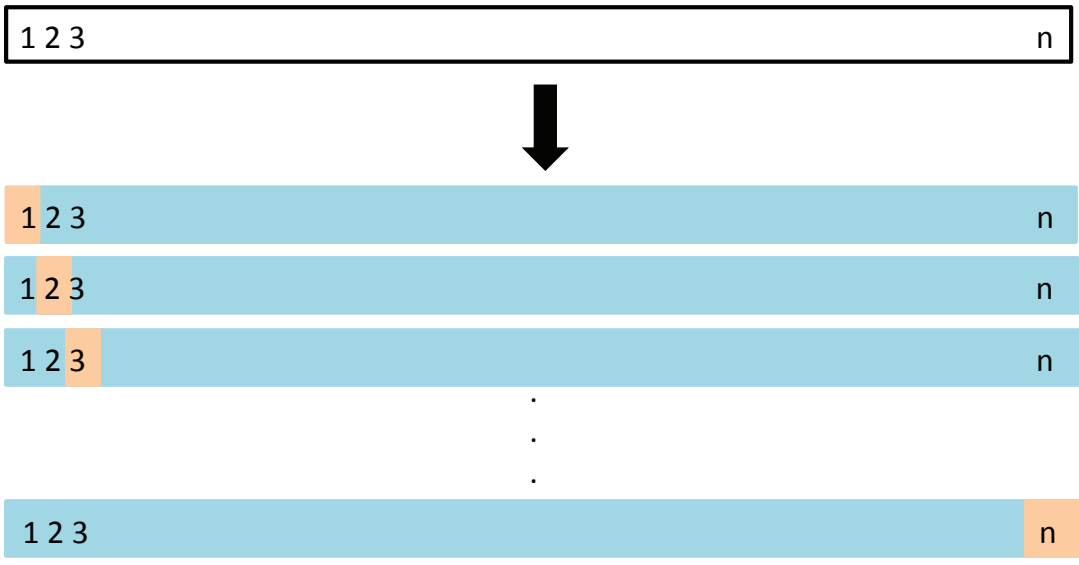
Validation Set Cons & Pros

- The procedure is **simple to do** (as we have done so far) and only a subset of the observations those that are included in the training set rather than in the validation set are used to fit the model
- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set
- While the estimated test error vary a lot, **finding information such as model selection is still possible**
- Since a large portion of the data need to be held aside, the model fits are not accurate enough

Leave-One Out Cross-Validation (LOOCV)

- We have n data points $(x_1, y_1), \cdots, (x_n, y_n)$, we use $n-1$ for the training and one instance for the test
- Of course a single test point is no where close to the true test error, but this process is repeated n times, every time $n-1$ points used for training and one point left out for the test
- Considering $MSE_1 = (y_1 - \hat{y}_1)^2, \cdots MSE_n = (y_n - \hat{y}_n)^2$, an approximation of the test error is

$$CV_n = \frac{1}{n} \sum_{i=1}^{n} MSE_i$$



Cons & Pros with LOOCV

- It has a very small bias compared to the validation set approach (it almost uses as much data as possible to fit the model)
- The test error overestimation is less than the validation set approach (because of what we mentioned above)
- Its results are reproducible unlike the validation set approach which uses a random subset of the data for test evaluation
- It can be computationally very expensive (requires running the algorithm n times)
- For linear models there is a shortcut to calculate CV_n that only requires fitting the model once with the entire data (but **this shortcut only applies to linear models**)

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

where $\hat{y}_i$ are the fitted values of the original least squares problem and h_i are only data dependent

K-Fold Cross Validation

- Widely used approach for estimating test error

K-Fold Cross Validation

- Widely used approach for estimating test error
- This approach involves **randomly dividing the set of observations into K groups**, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $K - 1$ folds

K-Fold Cross Validation

- Widely used approach for estimating test error
- This approach involves **randomly dividing the set of observations into K groups**, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $K - 1$ folds
- This is done in turn for each part $k = 1, 2, \cdots, K$, and then the results are combined

K-Fold Cross Validation

- Widely used approach for estimating test error
- This approach involves **randomly dividing the set of observations into K groups**, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $K - 1$ folds
- This is done in turn for each part $k = 1, 2, \cdots, K$, and then the results are combined
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model

$$CV_K = \frac{1}{K} \sum_{k=1}^{K} MSE_k$$

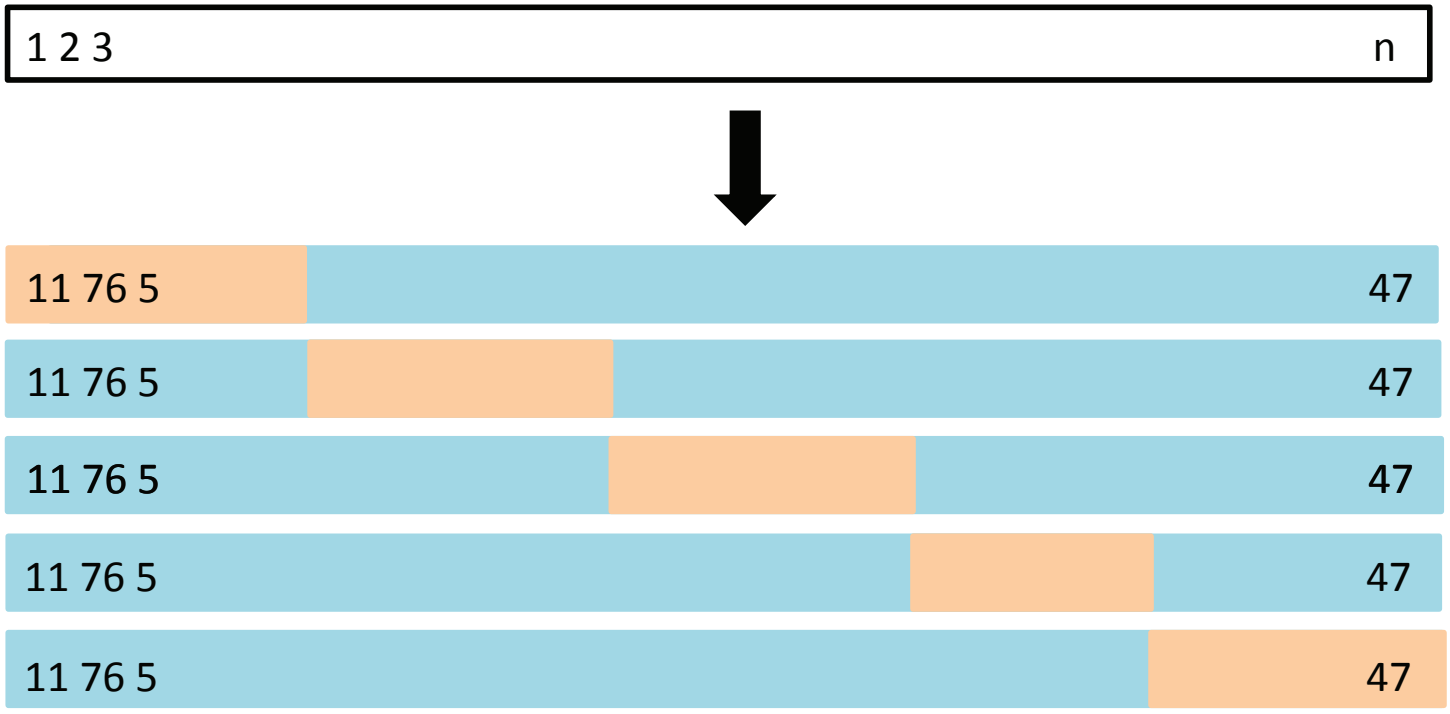
K-Fold Cross Validation

- Widely used approach for estimating test error
- This approach involves **randomly dividing the set of observations into K groups**, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $K - 1$ folds
- This is done in turn for each part $k = 1, 2, \cdots, K$, and then the results are combined
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model

$$CV_K = \frac{1}{K} \sum_{k=1}^{K} MSE_k$$

- Often $K = 5$ or $K = 10$ is what is considered in application

K-Fold Cross Validation



K-Fold Cross Validation and LOOCV

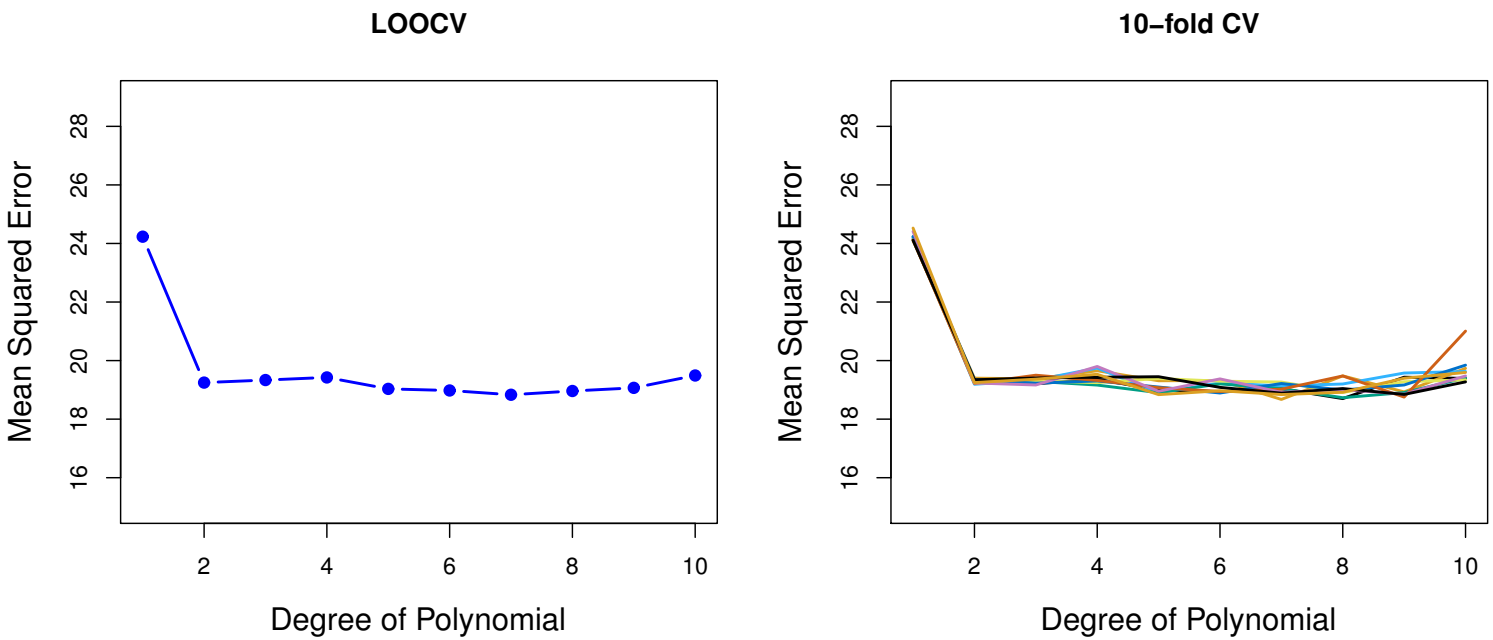
- LOOCV is a special case of K-fold CV for $K = n$

K-Fold Cross Validation and LOOCV

- LOOCV is a special case of K-fold CV for $K = n$
- In general K-fold CV is much cheaper than LOOCV because it only requires K model fits vs n model fits

K-Fold Cross Validation and LOOCV

- LOOCV is a special case of K-fold CV for $K = n$
- In general K-fold CV is much cheaper than LOOCV because it only requires K model fits vs n model fits
- For model selection, K-fold CV often gives us similar outcomes at a much lower computational cost



K-Fold Cross Validation and LOOCV

- Aside form the computational issues, even surprisingly K-Fold CV produces better test estimates than the LOOCV

K-Fold Cross Validation and LOOCV

- Aside form the computational issues, even surprisingly K-Fold CV produces better test estimates than the LOOCV
- LOOCV has a lower bias compared to the K-fold CV, since it uses more data to fit the model

K-Fold Cross Validation and LOOCV

- Aside form the computational issues, even surprisingly K-Fold CV produces better test estimates than the LOOCV
- LOOCV has a lower bias compared to the K-fold CV, since it uses more data to fit the model
- But K-fold CV has a lower variance compared to the LOOCV, since LOOCV is the sum of n highly correlated random variables while the correlation between the MSEs in K-fold CV is lower, recall

$$var(X + Y) = var(X) + var(Y) + 2cov(X, Y)$$

Bootstrap

Bootstrap

- The bootstrap is a flexible and very powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method

Bootstrap

- The bootstrap is a flexible and very powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method
- It can provide an estimate of the standard error of a coefficient, or a **confidence interval for that coefficient**, regardless of how complex the derivation of that coefficient is

Bootstrap via an Example

- Lets explain bootstrap via an example

Bootstrap via an Example

- Lets explain bootstrap via an example
- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y (random quantities)

Bootstrap via an Example

- Lets explain bootstrap via an example
- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y (random quantities)
- We will invest α shares in X, and will invest the remaining $1 - \alpha$ in Y

Bootstrap via an Example

- Lets explain bootstrap via an example
- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y (random quantities)
- We will invest α shares in X, and will invest the remaining $1 - \alpha$ in Y
- To minimize the risk, we want to minimize $var(\alpha X + (1 - \alpha)Y)$

Bootstrap via an Example

- Lets explain bootstrap via an example
- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y (random quantities)
- We will invest α shares in X, and will invest the remaining $1-\alpha$ in Y
- To minimize the risk, we want to minimize $var(\alpha X + (1-\alpha)Y)$
- We can show that (in the class we do it) that the minimizer is

$$\alpha = \frac{var(Y) - cov(X,Y)}{var(X) + var(Y) - 2cov(X,Y)}$$

Bootstrap via an Example

- In real-world we do not know $var(X), var(Y), cov(X, Y)$

Bootstrap via an Example

- In real-world we do not know $var(X), var(Y), cov(X, Y)$
- Suppose we are given a data set containing pairs of X and Y. We can estimate $var(X), var(Y), cov(X, Y)$ from the sample set and get an estimate $\hat{\alpha}$ for the optimal share

Bootstrap via an Example

- In real-world we do not know $var(X), var(Y), cov(X, Y)$
- Suppose we are given a data set containing pairs of X and Y. We can estimate $var(X), var(Y), cov(X, Y)$ from the sample set and get an estimate $\hat{\alpha}$ for the optimal share
- Ideally we can generate these sample sets many times, and estimate an $\hat{\alpha}$ for each and look into the histogram

Bootstrap via an Example

- In real-world we do not know $var(X), var(Y), cov(X, Y)$
- Suppose we are given a data set containing pairs of X and Y. We can estimate $var(X), var(Y), cov(X, Y)$ from the sample set and get an estimate $\hat{\alpha}$ for the optimal share
- Ideally we can generate these sample sets many times, and estimate an $\hat{\alpha}$ for each and look into the histogram
- However in a real-world we only have one sample set to use

Bootstrap via an Example

- In real-world we do not know $var(X), var(Y), cov(X, Y)$
- Suppose we are given a data set containing pairs of X and Y. We can estimate $var(X), var(Y), cov(X, Y)$ from the sample set and get an estimate $\hat{\alpha}$ for the optimal share
- Ideally we can generate these sample sets many times, and estimate an $\hat{\alpha}$ for each and look into the histogram
- However in a real-world we only have one sample set to use
- Bootstrap yet allows us to generate good estimates of α **using only one sample set!**

Bootstrap via an Example

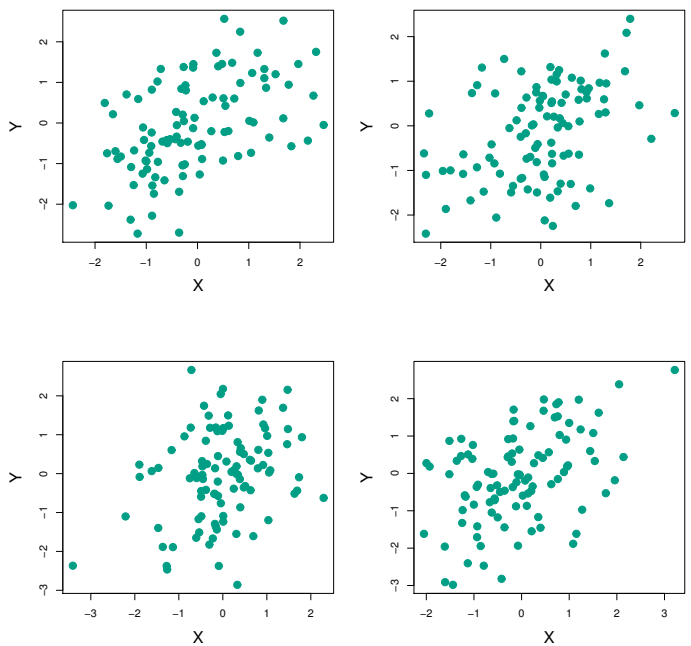
To see how nicely Bootstrap works, lets compare its outcome with the case that α is generated from many synthetic sample generations

- We generate 1000 sample sets each containing 100 pairs of X, Y

Bootstrap via an Example

To see how nicely Bootstrap works, lets compare its outcome with the case that α is generated from many synthetic sample generations

- We generate 1000 sample sets each containing 100 pairs of X, Y
- For the synthetic data generated $var(X) = 1, var(Y) = 1.25$ and $cov(X, Y) = 0.5$ which yield an optimal value of $\alpha = 0.6$



Bootstrap via an Example

- To get the left panel we generate 1000 synthetic sample sets, for each obtain $\hat{\alpha}$ and plot the histogram and calculate

$$\bar{\alpha} = \frac{1}{1000}\sum_{i=1}^{1000}\hat{\alpha}_i, \quad SE(\alpha) = \sqrt{\frac{1}{999}\sum_{i=1}^{1000}(\hat{\alpha}_i - \bar{\alpha})^2}$$

Bootstrap via an Example

- To get the left panel we generate 1000 synthetic sample sets, for each obtain $\hat{\alpha}$ and plot the histogram and calculate

$$\bar{\alpha} = \frac{1}{1000}\sum_{i=1}^{1000}\hat{\alpha}_i, \quad SE(\alpha) = \sqrt{\frac{1}{999}\sum_{i=1}^{1000}(\hat{\alpha}_i - \bar{\alpha})^2}$$

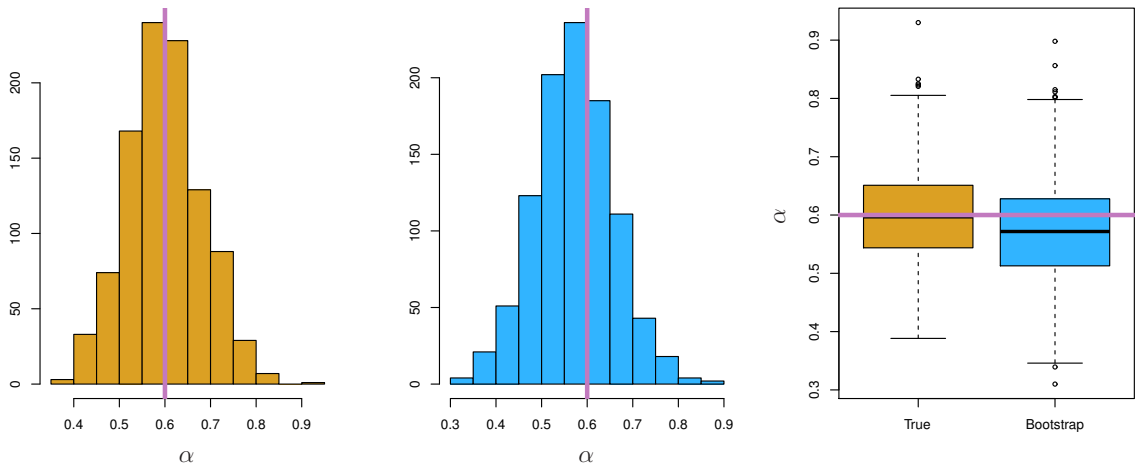
- For the bootstrap we only use one of the sample sets and regenerate new sample set by sampling with replacement

Bootstrap via an Example

- To get the left panel we generate 1000 synthetic sample sets, for each obtain $\hat{\alpha}$ and plot the histogram and calculate

$$\bar{\alpha} = \frac{1}{1000}\sum_{i=1}^{1000}\hat{\alpha}_i, \quad SE(\alpha) = \sqrt{\frac{1}{999}\sum_{i=1}^{1000}(\hat{\alpha}_i - \bar{\alpha})^2}$$

- For the bootstrap we only use one of the sample sets and regenerate new sample set by sampling with replacement
- Surprisingly, the results are very close



Bootstrap General Framework

- Suppose a black-box calculates $\hat{\alpha}$ from a sample set, e.g., a coefficient in linear or logistic regression

Bootstrap General Framework

- Suppose a black-box calculates $\hat{\alpha}$ from a sample set, e.g., a coefficient in linear or logistic regression
- We are interested in estimating the variability of $\hat{\alpha}$ without examining many new sample sets

Bootstrap General Framework

- Suppose a black-box calculates $\hat{\alpha}$ from a sample set, e.g., a coefficient in linear or logistic regression
- We are interested in estimating the variability of $\hat{\alpha}$ without examining many new sample sets
- Denoting the first bootstrap data set by Z^1, we use Z^1 to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^1$

Bootstrap General Framework

- Suppose a black-box calculates $\hat{\alpha}$ from a sample set, e.g., a coefficient in linear or logistic regression
- We are interested in estimating the variability of $\hat{\alpha}$ without examining many new sample sets
- Denoting the first bootstrap data set by Z^1, we use Z^1 to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^1$
- This procedure is repeated B (say 100 or 1000) times, in order to produce B different bootstrap data sets, $Z^1, Z^2, \cdots, Z^B$, and the corresponding α estimates, $\hat{\alpha}^1, \cdots, \hat{\alpha}^B$

Bootstrap General Framework

- Suppose a black-box calculates $\hat{\alpha}$ from a sample set, e.g., a coefficient in linear or logistic regression
- We are interested in estimating the variability of $\hat{\alpha}$ without examining many new sample sets
- Denoting the first bootstrap data set by Z^1, we use Z^1 to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^1$
- This procedure is repeated B (say 100 or 1000) times, in order to produce B different bootstrap data sets, $Z^1, Z^2, \cdots, Z^B$, and the corresponding α estimates, $\hat{\alpha}^1, \cdots, \hat{\alpha}^B$
- We estimate the standard error of these bootstrap estimates using the formula

$$SE_B(\alpha) = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{\alpha}^i - \bar{\hat{\alpha}})^2} \quad \text{where} \quad \bar{\hat{\alpha}} = \frac{1}{B}\sum_{i=1}^{B}\hat{\alpha}^i$$

Bootstrap General Framework

- Suppose a black-box calculates $\hat{\alpha}$ from a sample set, e.g., a coefficient in linear or logistic regression
- We are interested in estimating the variability of $\hat{\alpha}$ without examining many new sample sets
- Denoting the first bootstrap data set by Z^1, we use Z^1 to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^1$
- This procedure is repeated B (say 100 or 1000) times, in order to produce B different bootstrap data sets, $Z^1, Z^2, \cdots, Z^B$, and the corresponding α estimates, $\hat{\alpha}^1, \cdots, \hat{\alpha}^B$
- We estimate the standard error of these bootstrap estimates using the formula

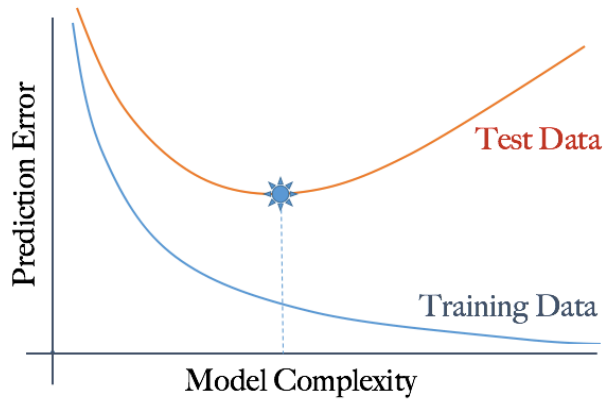
$$SE_B(\alpha) = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{\alpha}^i - \bar{\hat{\alpha}})^2} \quad \text{where} \quad \bar{\hat{\alpha}} = \frac{1}{B}\sum_{i=1}^{B}\hat{\alpha}^i$$

- This serves as an estimate of the standard error of α estimated from the original data set!

Model Selection

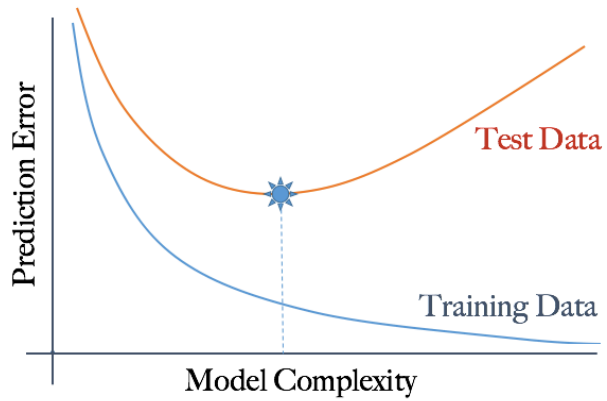
Why Cross Validation?

- As mentioned earlier, model selection based on the RSS or R^2 statistics can be misleading, since the training error is not a good representative of the actual test error



Why Cross Validation?

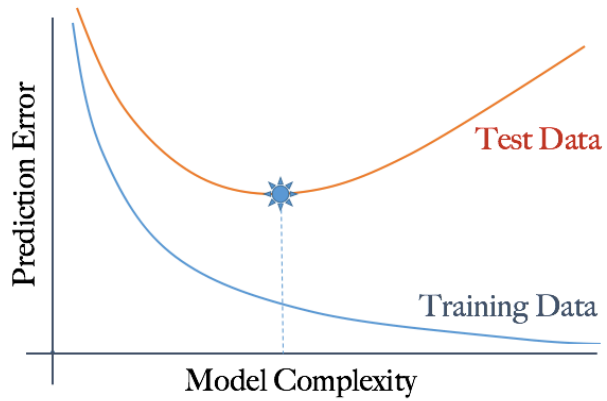
- As mentioned earlier, model selection based on the RSS or R^2 statistics can be misleading, since the training error is not a good representative of the actual test error



- Instead through a process of splitting the data into training and validations sets, we were able to use LOOCV or K-Fold CV as estimates of the test error

Why Cross Validation?

- As mentioned earlier, model selection based on the RSS or R^2 statistics can be misleading, since the training error is not a good representative of the actual test error



- Instead through a process of splitting the data into training and validations sets, we were able to use LOOCV or K-Fold CV as estimates of the test error
- We discussed why K-Fold CV is a more desirable estimate, computationally and statistically

Adjusting the Training Statistics for Test Error Approximation

Adjusting Techniques

- We introduce few other ways of adjusting the training error **to make it a better representative of the test error**

Adjusting Techniques

- We introduce few other ways of adjusting the training error **to make it a better representative of the test error**
- These adjustments are **not as reliable as Cross validation**, but they are easier to **calculate**

Adjusting Techniques

- We introduce few other ways of adjusting the training error **to make it a better representative of the test error**
- These adjustments are **not as reliable as Cross validation**, but they are easier to **calculate**
- These quantities were **more widely used before** the widespread use of computers for regression and machine learning

Adjusting Techniques

- We introduce few other ways of adjusting the training error **to make it a better representative of the test error**
- These adjustments are **not as reliable as Cross validation**, but they are easier to **calculate**
- These quantities were **more widely used before** the widespread use of computers for regression and machine learning
- Now that computers can help performing multiple fits computationally fast enough, often K-Fold CV is considered as the desirable test error approximation

List of Other Techniques

Methods to adjust the training error for the number of variables to estimate the test MSE:

- C_p statistic

List of Other Techniques

Methods to adjust the training error for the number of variables to estimate the test MSE:

- C_p statistic
- Akaike information criterion (AIC)

List of Other Techniques

Methods to adjust the training error for the number of variables to estimate the test MSE:

- C_p statistic
- Akaike information criterion (AIC)
- Bayesian information criterion (BIC)

List of Other Techniques

Methods to adjust the training error for the number of variables to estimate the test MSE:

- C_p statistic
- Akaike information criterion (AIC)
- Bayesian information criterion (BIC)
- Adjusted R^2

C_p Statistic

- For a fitted least squares model with d predictors

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

- $\hat{\sigma}^2$ is an estimate of the noise variance
- $\hat{\sigma}^2$ is normally estimated using all the predictors (full model)
- It is an unbiased estimate of the **test MSE**
- The smaller C_p, the better the model (we can pick models with the smallest C_p statistic)
- Becomes a better estimate of the test error as the sample size, n, increases

AIC: Akaike Information Criterion

- Defined for a large class of models based on the maximum likelihood criterion
- When we consider the noise ϵ be of i.i.d Gaussian, the MLE and MSE return identical results and in this case we have

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

  which is a multiple of C_p (no preference over using one vs the other)
- $\hat{\sigma}^2$ is an estimate of the noise variance
- $\hat{\sigma}^2$ is normally estimated using all the predictors (full model)
- The smaller AIC, the better the model (we can pick models with the smallest AIC statistic)

BIC: Bayesian Information Criterion

- Takes a Bayesian approach to estimate the test error
- In the case of least squares the formulation is

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

  which takes an *almost* similar form as the previous two statistics
- $\hat{\sigma}^2$ is an estimate of the noise variance
- $\hat{\sigma}^2$ is normally estimated using all the predictors (full model)
- The smaller BIC, the better the model (we can pick models with the smallest AIC statistic)
- When $n < 7$, BIC imposes a smaller penalty on the number of variables, but for $n > 7$ that $\log n > 2$ the penalty is larger
- In other words in standard observation regimes where n is sufficiently large, BIC tends to pick smaller models than AIC or C_p

Adjusted R^2

- Presents a way of making the R^2 statistic dependent on the number of predictors
- Recall the R^2 statistic:

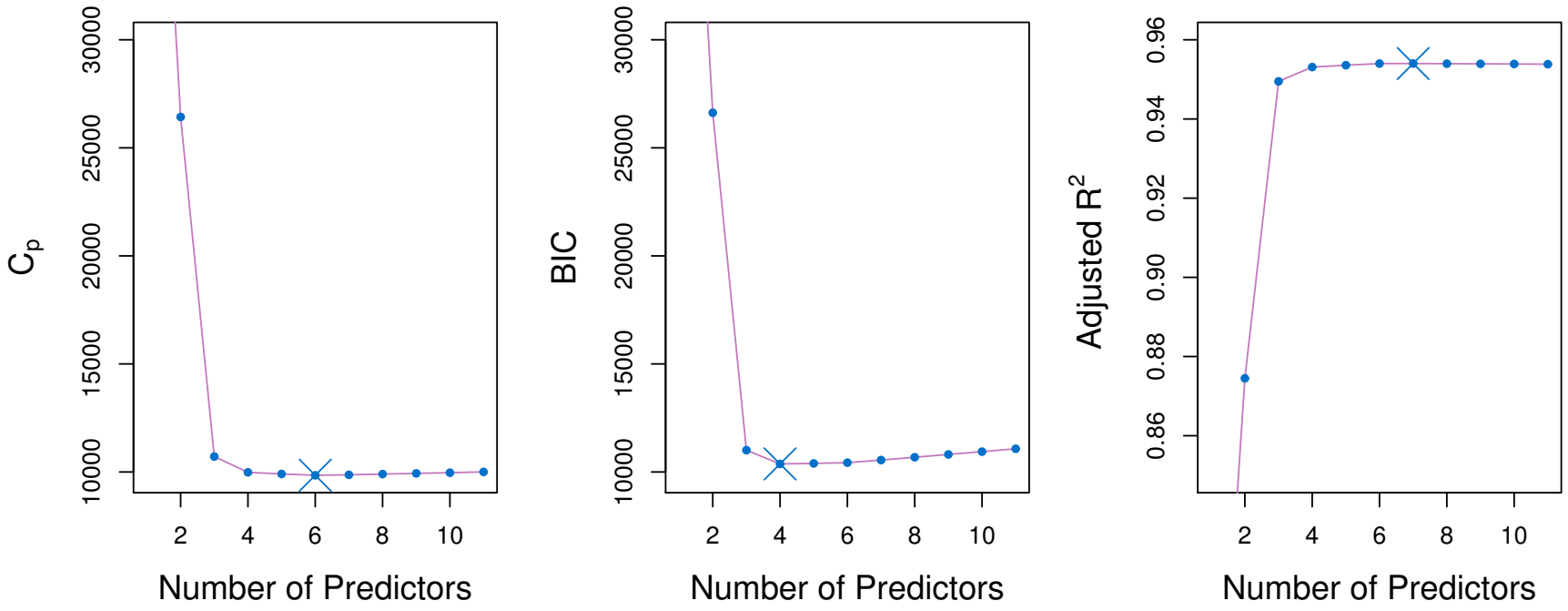
$$R^2 = 1 - \frac{RSS}{TSS}, \qquad \text{where} \quad TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

- The formulation for adjusted R^2 is

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

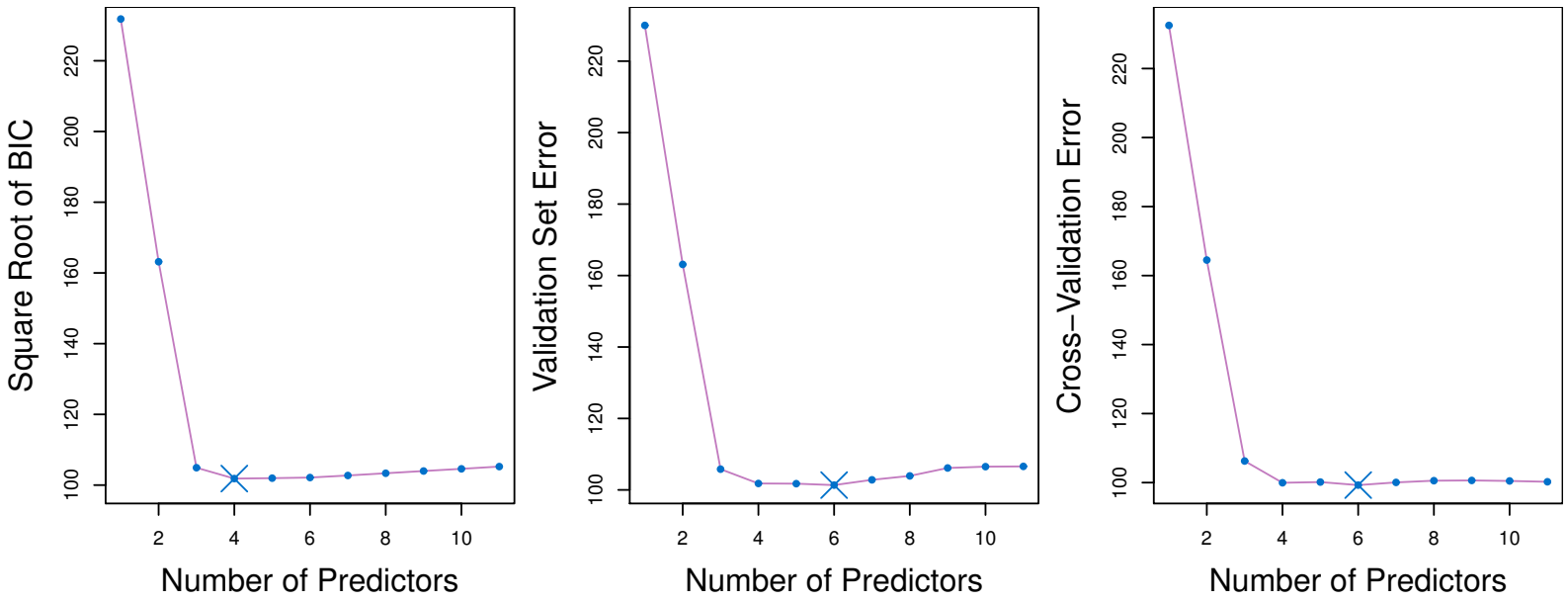
- Unlike the other three statistics that being small indicates a better model, for adjusted R^2 we are interested in models that tend to generate values closer to 1
- The use of C_p, AIC and BIC is more motivated in statistical learning theory than the adjusted R^2

Example Comparing the Performances



C_p, BIC, and adjusted R^2 for the best models of each size for the Credit data set

Comparison Against CV Techniques



- The results are not much different
- Note that nowadays CV methods are computationally fast to implement and regardless of the model can always be used as a reliable selection tool

How to Use These Statistics in Model Selection

- **Best subset selection** formal procedure (NP-hard and computationally not possible for large p)

Algorithm 6.1 *Best subset selection*

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

How to Use These Statistics in Model Selection

- **Forward stepwise selection** (computationally tractable)

Algorithm 6.2 *Forward stepwise selection*

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

- Forward selection can even be used when $n < p$

How to Use These Statistics in Model Selection

- **Backward stepwise selection** (computationally tractable)

Algorithm 6.3 *Backward stepwise selection*

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

- Backward selection requires $p < n$ (to allow the full model to be fit)

Shrinkage Methods

Ridge Regression

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

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

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

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

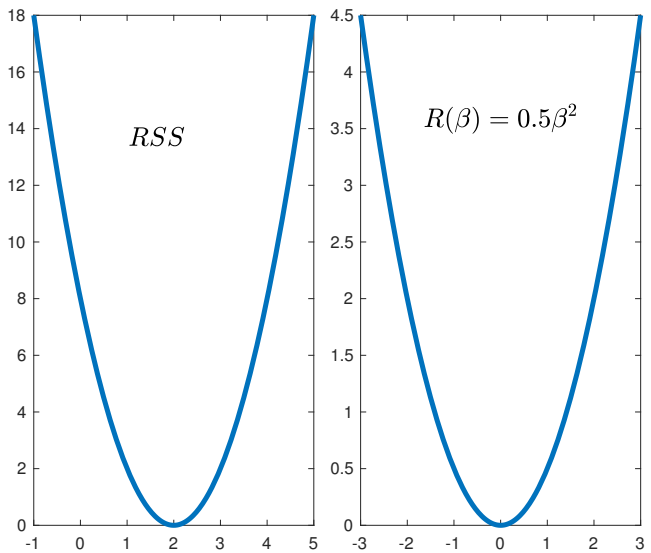
- Here, λ is a tuning parameter

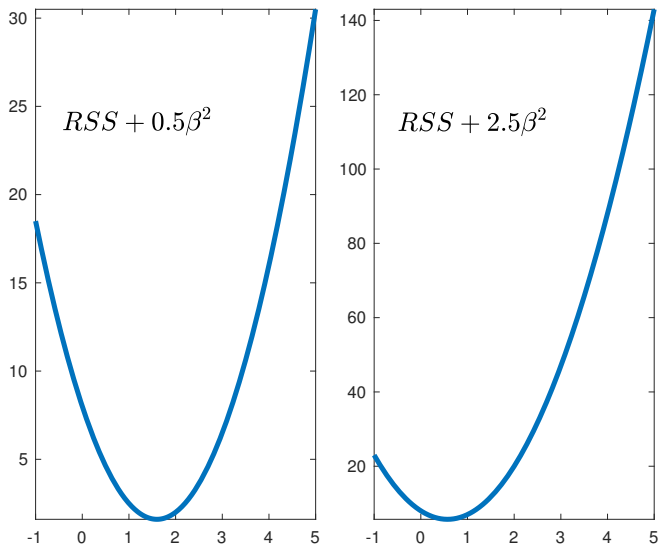
Ridge Regression

- As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small
- However, the second term, $\lambda \sum_{j=1}^{p} \beta_j^2$, called a shrinkage penalty, encourages solutions that are close to zero, and so it has the effect of shrinking the estimates of β_j towards zero
- The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates (trade off between bias and variance)
- Selecting a good value for λ is critical; often cross-validation is used for this

Effect of Increasing λ on the β

- The figure below shows how increasing the Ridge penalty pushes the minimizers of the mixed RSS objective to zero





Shrinkage Example

- Previously from the homework assignments you remember that the least squares solution to fit data point $(x_1, y_1), \cdots, (x_n, y_n)$ was obtained via the minimization:

$$\min_{\beta} \sum_{i=1}^{N} (y_i - \beta x_i)^2 \quad \therefore \quad \hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$$

- We can show that if we run the Ridge regression problem

$$\min_{\beta} \sum_{i=1}^{N} (y_i - \beta x_i)^2 + \lambda \beta^2$$

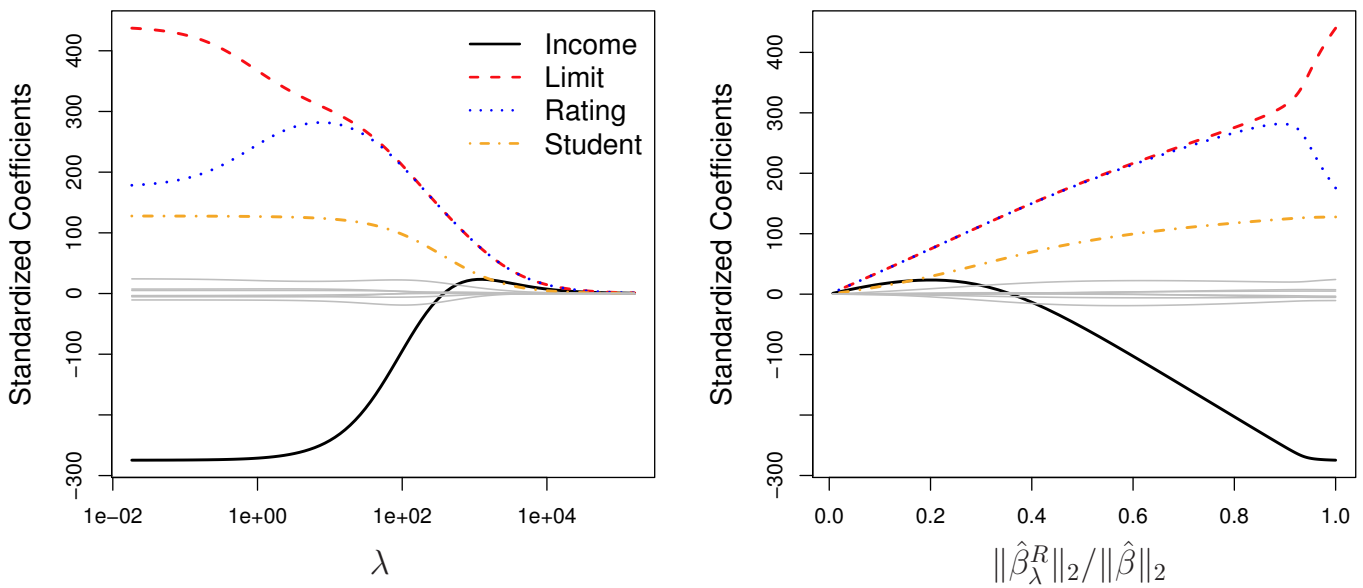
the new estimate becomes

$$\hat{\beta}^R = \frac{\sum_{i=1}^{n} x_i y_i}{\lambda + \sum_{i=1}^{n} x_i^2}$$

- Note how increasing λ pushes $\hat{\beta}^R$ towards zero

Credit Data Example

- Left: 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 λ on the x-axis, we display $\|\hat{\beta}^R\|/\|\hat{\beta}\|$ (**shrinkage factor**)



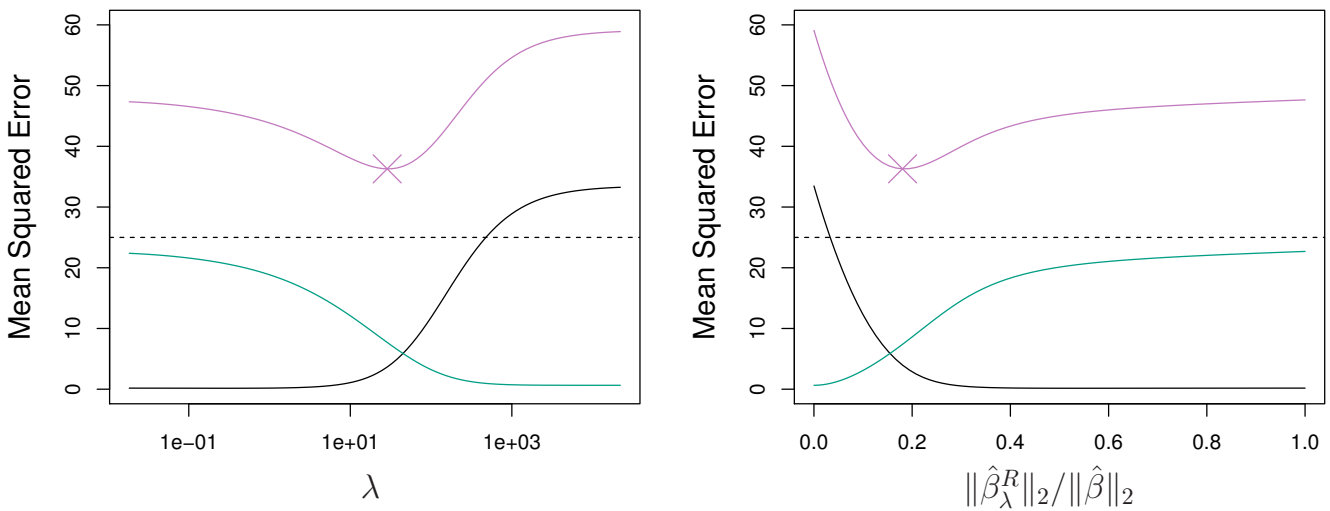
Scaling of the Predictors

- In the standard least-squares if we scale a feature value by c, the corresponding coefficient scales by c^{-1}
- However when we have the Ridge regularized objective, this is no more the case
- To see a consistent behavior, for the Ridge regularized problem we often work with standardized features:

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

Bias-Variance Trade-Off

- A toy example: squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of λ and $\|\hat{\beta}^R\|/\|\hat{\beta}\|$. The horizontal dashed lines indicate the minimum possible MSE (**the standard least squares,** $\lambda = 0$ **in nowhere close**). The purple crosses indicate smallest ridge regression model MSE values



- Remember (test error = bias + variance + noise variance)

LASSO (Least Absolute Selection and Shrinkage Operator)

- 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 (none of the model coefficients become explicitly zero)
- The Lasso is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients minimize the quantity

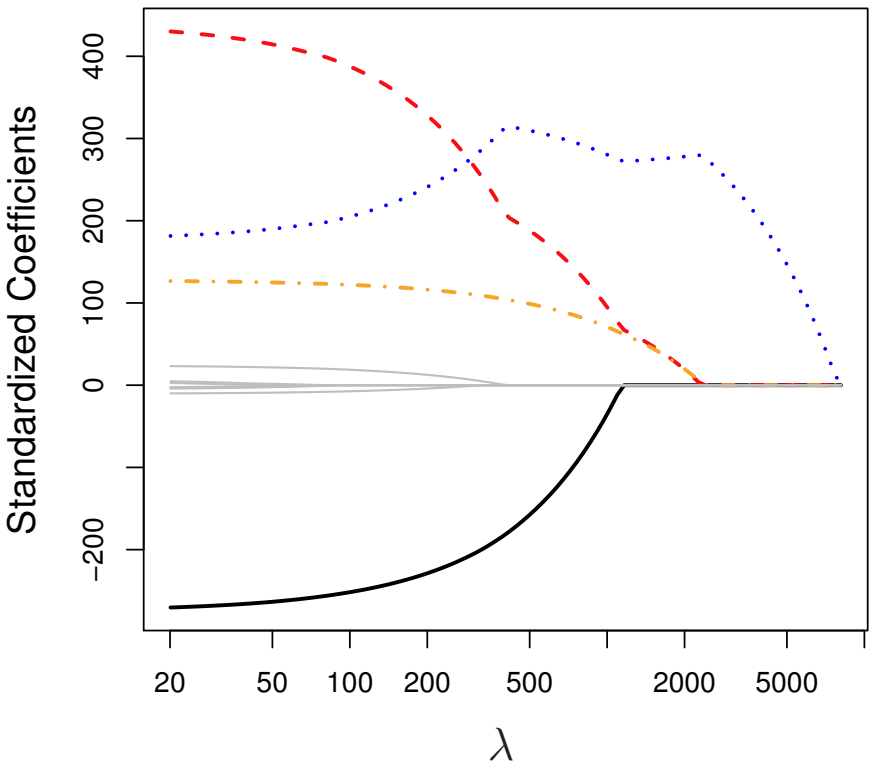
$$RSS_{LASSO} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = RSS + \lambda \|\boldsymbol{\beta}\|_1$$

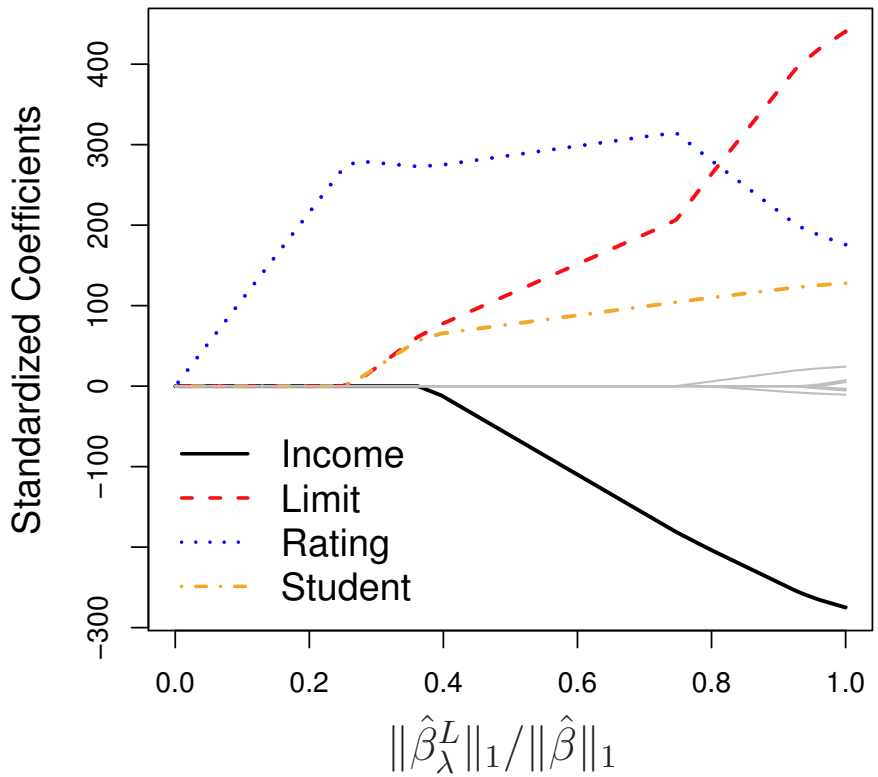
- We call $\|\boldsymbol{\beta}\|_1$ the L-1 norm of $\boldsymbol{\beta}$

LASSO

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero
- However, in the case of the lasso, the L1 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
- Technically the lasso performs the training and variable selection together
- We say that the lasso yields sparse models that is, models that involve only a subset of the variables
- As in ridge regression, selecting a good value of for the lasso is critical; cross-validation is again the method of choice

Example: Credit Data





Why LASSO promotes Sparsity?

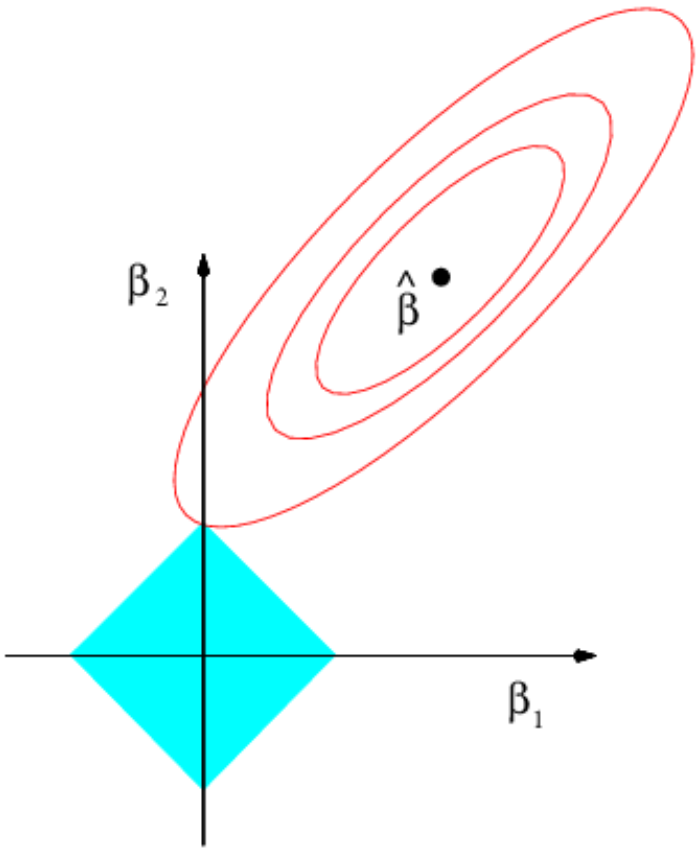
- From a convex optimization perspective, one can show that the lasso and ridge regression coefficient estimates solve the problems

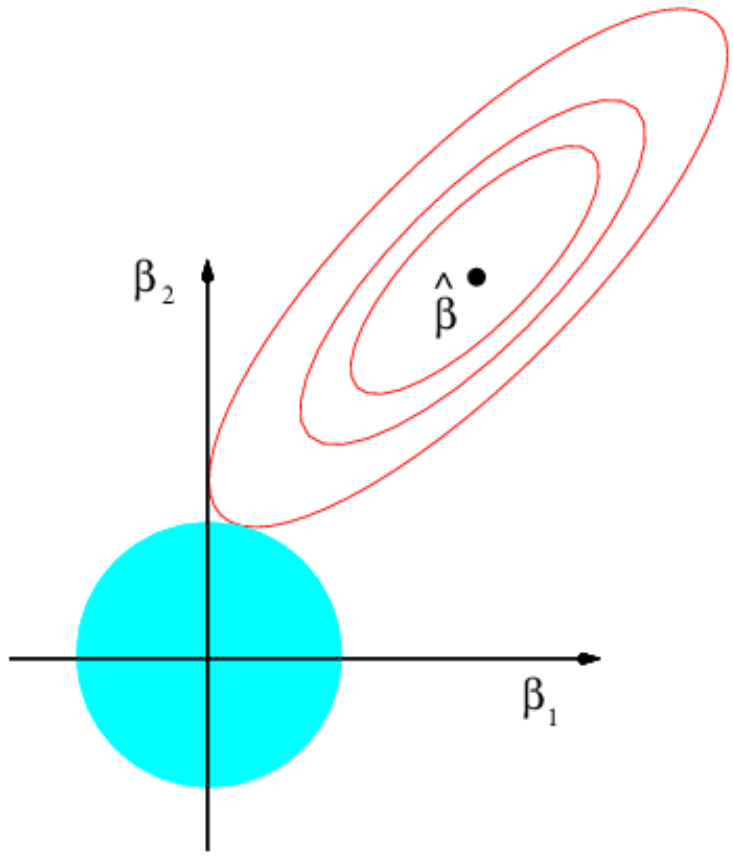
$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \quad \textit{subject to} \quad \|\boldsymbol{\beta}\|_1 \leq \tau$$

and

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \quad \textit{subject to} \quad \|\boldsymbol{\beta}\|^2 \leq \tau'$$

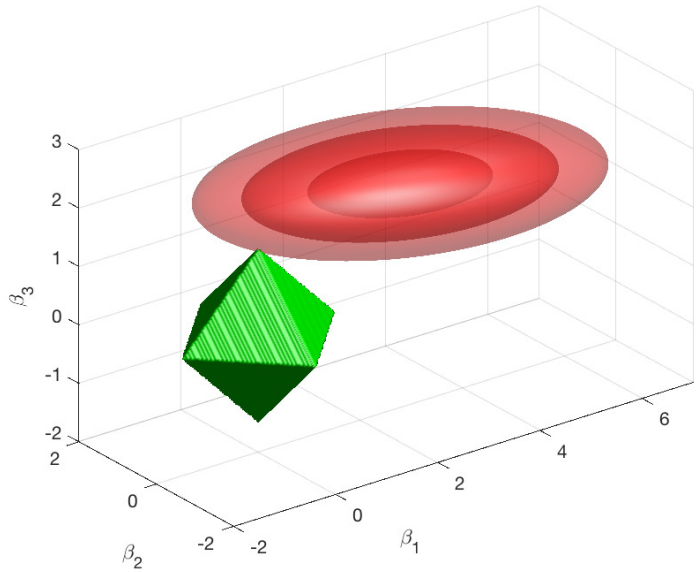
The Geometry of the Two Problems

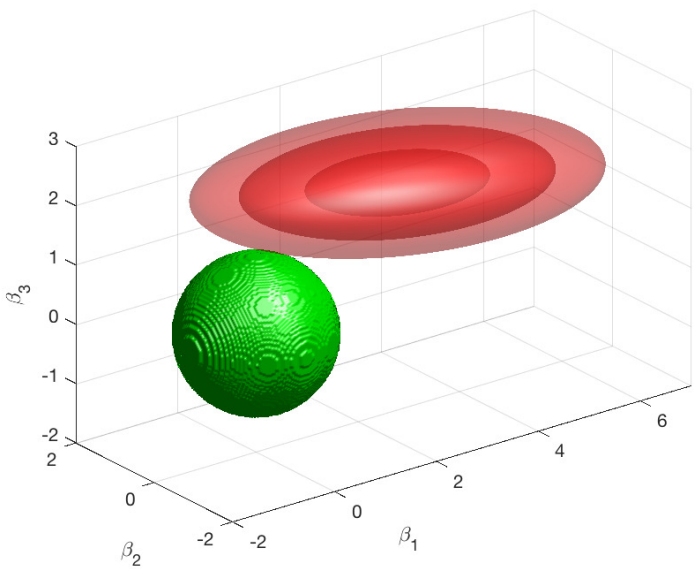




The Geometry of the Two Problems In Higher Dimension

See the MATLAB code attached:





LASSO or Ridge?

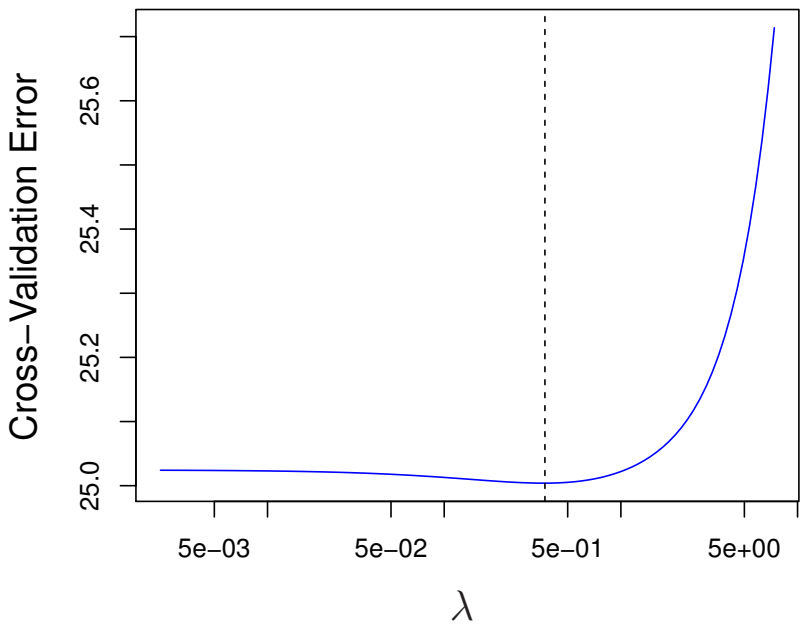
- Neither ridge regression nor the lasso will universally dominate the other
- In general, one might expect the lasso to perform better when the response is a function of only a relatively small number of predictors
- However, the number of predictors that is related to the response is never known a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

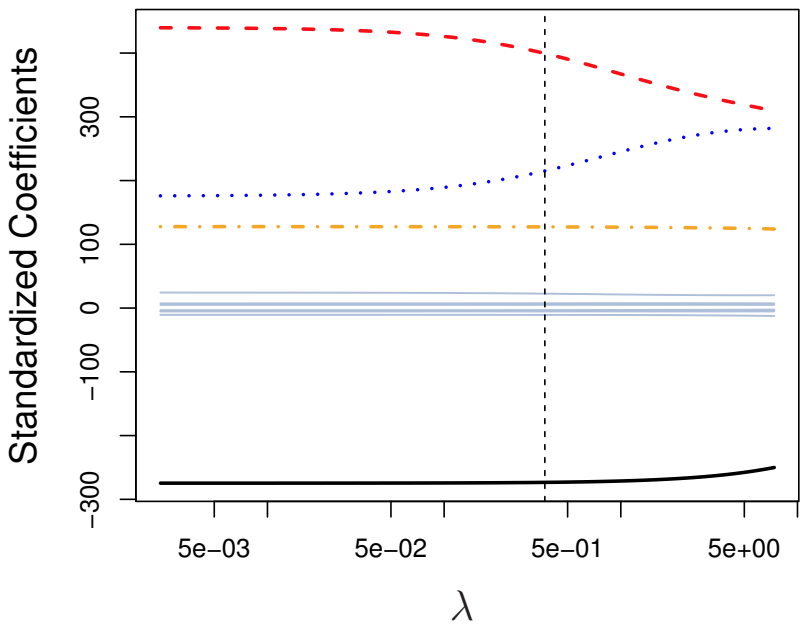
How to Determine λ?

- As for subset selection, for ridge regression and lasso we require a method to determine which of the models under consideration is the best
- That is, we require a method selecting a value for the tuning parameter λ
- 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 the smallest
- Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter

CV and Ridge Example

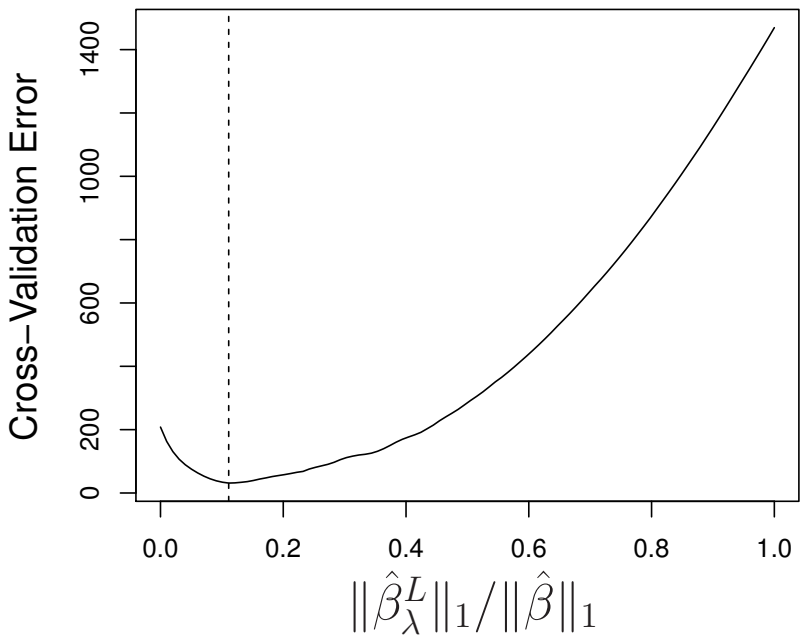
Determining λ via cross validation for the Ridge problem (credit data)

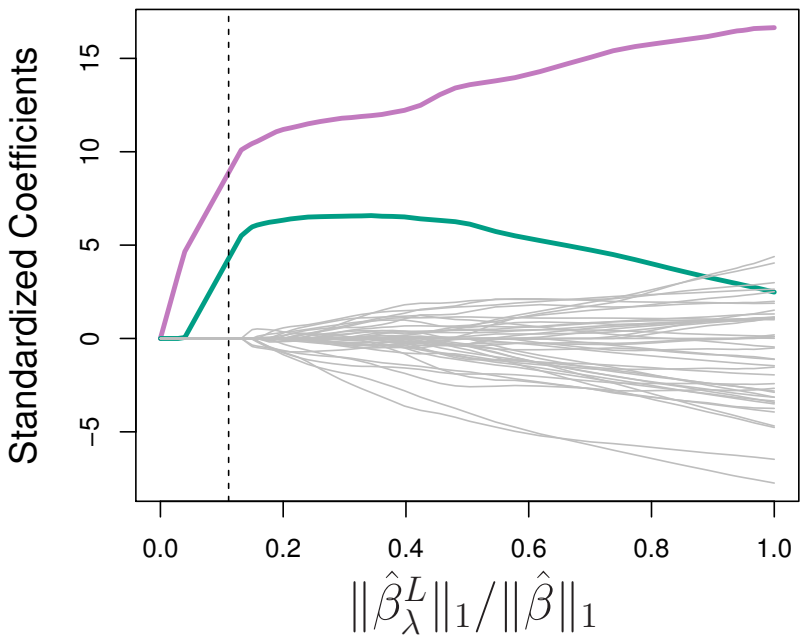




CV and LASSO Example

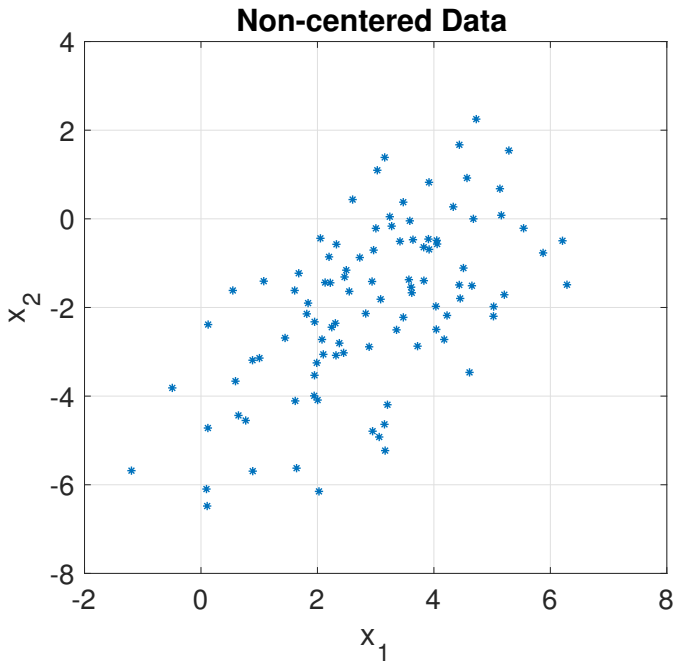
Determining λ via cross validation for the LASSO problem (simulated data)

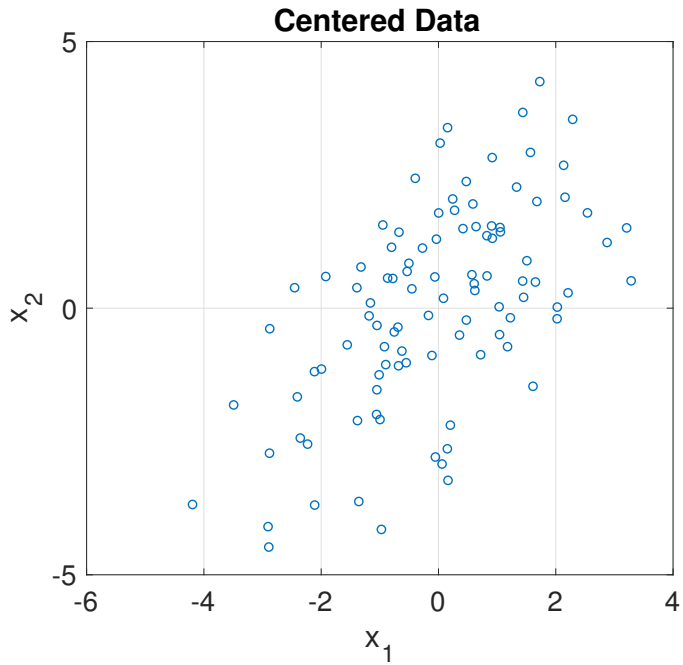




Principal Component Regression (PCR)

- Consider $\boldsymbol{X}$ to be an $n \times p$ matrix representing our feature data (each column corresponds to a feature and each row to a sample)
- We assume that $\boldsymbol{X}$ is centered, meaning that each column is zero mean
- This technically means that our data cloud is centered around the origin and not some other point in the space





Principal Component Regression (PCR)

- Recall the singular value decomposition (SVD)

$$\boldsymbol{X} = \boldsymbol{U}_{n\times p}\boldsymbol{\Sigma}_{p\times p}\boldsymbol{V}^{\top}_{p\times p}$$

  where

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{bmatrix}$$

- When the first r entries of $\boldsymbol{\Sigma}$ are the dominant terms, the first r columns in $\boldsymbol{U}$ form a subspace (coordinate system) that very well represent our data
- In PCR, we move from the original feature space to the new feature space represented by $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_r$
- Technically, instead of a fit like

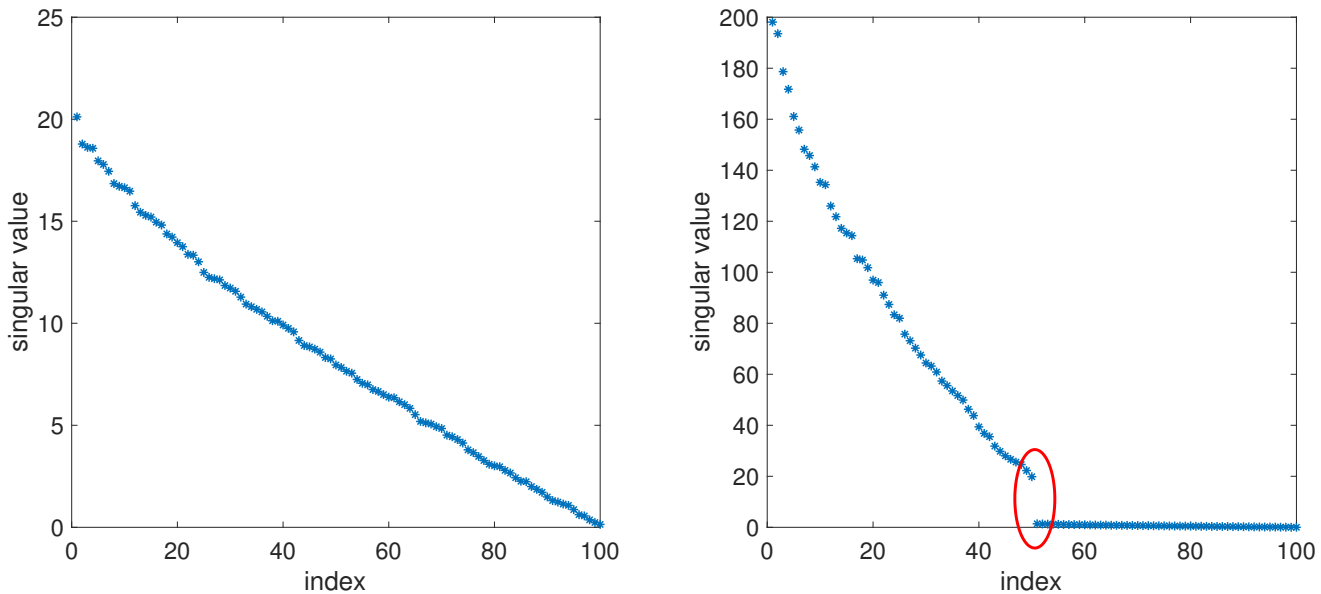
$$\boldsymbol{y} = \beta_0 + \beta_1\boldsymbol{x}_1 + \cdots \beta_p\boldsymbol{x}_p$$

  we fit the model

$$\boldsymbol{y} = \gamma_0 + \gamma_1\boldsymbol{z}_1 + \cdots \gamma_r\boldsymbol{z}_r$$

How to Determine r

- Normally, the number of principal components, r, can be determined by cross validation
- If we see a good contrast between two consecutive singular values, roughly choosing r (or at least an initial value) can become easy!



- Despite all this, a formal selection requires CV
- This is technically because such selection only relies on $\boldsymbol{X}$, and no guarantee that the selected features contribute to the response

More on PCR

- PCR identifies linear combinations, or directions, that best represent the predictors $x_1, \cdots, x_p$
- These directions are identified in an unsupervised way, since the response Y is not used to help determine the principal component directions
- That is, **the response does not supervise the identification of the principal components**
- Consequently, PCR suffers from a potentially serious drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response
- **Partial Least Squares** (PLS, details of which are skipped in this course) can be considered as a remedy to this

Questions?

References


https://www.alsharif.info/iom530, 2013.


J. Friedman, T. Hastie, and R. Tibshirani.
The elements of statistical learning.
Springer series in statistics, 2nd edition, 2009.


G. James, D. Witten, T. Hastie, and R. Tibshirani.
https://lagunita.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/nonlinear.pdf, 2013.


G. James, D. Witten, T. Hastie, and R. Tibshirani.
An introduction to statistical learning: with applications in R, volume 112.
Springer, 2013.