Statistical Learning

A. Colin Cameron
Univ. of Calif.- Davis

Based on James, Witten, Hastie and Tibsharani "An Introduction to Statistical Learning" (2013)
and *Hastie, Tibsharani and Friedman (2009) "The Elements of Statistical Learning"*

April 25, 2016

Introduction

- Problem: We want data-driven determination of a regression model that fits the data well but guards against in-sample overfitting.
- Solution:
  - Use one of several methods to choose the optimal model for a given value of a "tuning parameter" that defines the level of model complexity / size
    - ★ e.g. Forward stepwise selection for a given number of model parameters
    - ★ e.g. Ridge regression or lasso with a given value of the penalty parameter.
  - Then use cross-validation to choose the value of the tuning parameter
    - ★ this trades off variance and bias.
- Complications: nonlinear models, categorical data, identifying clusters.

Overview

1. Terminology, Statistical Learning (ISL chs.1-2)
2. Linear Regression (ISL ch.3)
3. Cross-Validation (ISL ch.5, ESL 219-235)
4. Subset Selection of Regressors (ISL ch.6.)
5. Shrinkage Methods: ridge, lasso, LAR (ISL ch.6.2 + ESL 73-79,86-93)
6. Dimension Reduction: PCA and partial LS (ISL ch.6.3)
7. High-dimensional data (ISL ch.6.4)
8. Nonlinear models: splines, local regression (ISL ch.7)
9. Tree-based methods, bagging, boosting (ISL ch.8)
10. Classification (ISL chs.4, 9): logit, k-nn, LDA, SVM
11. Unsupervised learning: PCA, clustering (ISL ch.10)
12. Introduction to R (ISL end each chapter)

1. General Framework: Terminology

- **Supervised learning**
  - We have both outcome y and regressors $\mathbf{x}$
  - 1. **Regression**: y is continuous
  - 2. **Classification**: y is categorical and we want to predict y
- **Unsupervised learning**
  - We have no outcome y - only several $\mathbf{x}$
  - 1. **Clustering**: e.g. principal components analysis or factor analysis.
- Two types of data sets
  - 1. **training data set** is used to fit a model
  - 2. **test data set** is additional data used to determine how good the model fit is
    - ★ use to guard against overfitting the training data.

Statistical Decision Theory

- From ESL pages 18-19.
- We wish to predict Y given X.
- We specify a loss function $L(Y, f(X))$ for penalizing prediction error.
- For regression use squared error loss $L(Y, f(X)) = (Y - f(X))^2$.
- Then minimize the expected prediction error

$$\begin{aligned} EPE(f) &= E_{Y,X}[(Y - f(X))^2] \\ &= E_X[E_{Y|X}[(Y - f(X))^2|X]] \end{aligned}$$

- Minimize EPE(f) pointwise

$$\begin{aligned} f(x) &= \arg\min_c [E_{Y|X}[(Y - c)^2|X = x]] \\ \partial/\partial c &= E_{Y|X}[-2(Y - c)|X = x] \\ &= 0 \text{ implies } c = E_{Y|X}[Y|X = x] \end{aligned}$$

- $f(x) = E[Y|X = x]$ minimizes expected squared error loss.

Statistical Learning

- **Statistical learning** for regression is estimating $f(\cdot)$ in

$$\begin{aligned} Y &= f(\mathbf{X}) + \varepsilon \\ Y &= \text{ scalar response} \\ \mathbf{X} &= (X_1, ..., X_p) \\ E[\varepsilon] &= 0 \text{ and } \varepsilon \text{ independent of } \mathbf{X} \end{aligned}$$

- **Prediction**: predict Y using $\widehat{Y} = \widehat{f}(X)$

$$\begin{aligned} E[(Y - \widehat{Y})^2] &= E[(f(\mathbf{X}) + \varepsilon - \widehat{f}(X))^2] \\ &= E[(f(\mathbf{X}) - \widehat{f}(X))^2] + E[\varepsilon^2] \text{ since } \varepsilon \perp X \ \& \ E[\varepsilon] = 0 \\ &= \text{Reducible error } + \text{ Irreducible error} \end{aligned}$$

- **Inference**: how does Y change as $\mathbf{X}$ changes
  - which predictors matter?
  - how do they affect Y?
  - is a linear model sufficient?

Types of models

- Methods to estimate $f(\cdot)$
  - **parametric** e.g. linear model
    - $f(X) = \beta_0 + \mathbf{X}'\boldsymbol{\beta} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$
    - estimate by least squares
  - **nonparametric** e.g. nearest-neighbors, kernel, splines
    - have a smoothness parameter.
- Very flexible models may not be the best
  - flexible models are generally more difficult to interpret
    - ISL Figure 2.7 shows trade-offs across different methods
  - and even if interested in just prediction can overfit.

Mean-squared error

- Recall we use squared error loss.
- For regression in-sample use **mean-squared error**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(\mathbf{x}_i))^2.$$

- But the goal is out-of-sample performance
  - test observation $(\mathbf{x}_0, y_0)$ is a previously unseen **test observation**
  - we want to obtain the lowest **test MSE** (not training MSE)

$$\text{Test MSE} = Ave(y_0 - \widehat{f}(\mathbf{x}_0))^2 = \frac{1}{n_0}\sum_{i=1}^{n_0}(y_i - \widehat{f}(\mathbf{x}_{0i}))^2.$$

- Often test MSE $>$ training MSE
  - since estimators aim to minimize training MSE
  - called **overfitting** the data.

Variance–Bias Trade-off

- Key result: **Expected test MSE**

$$E[(y_0 - \widehat{f}(\mathbf{x}_0))^2] = Var[\widehat{f}(\mathbf{x}_0)] + \{Bias(\widehat{f}(\mathbf{x}_0))\}^2 + Var(\varepsilon)$$

- Need to minimize both variance and bias!
- In general there is a trade-off with more flexible models having
  - less bias and more variance.
- Note: MSE (squared error loss is used)
  - for tractability
  - but many methods such as cross validation extend to other loss functions
    - ★ e.g. absolute error loss $E[|y_0 - \widehat{f}(\mathbf{x}_0)|]$
    - ★ e.g. $1[y_0 = \widehat{y}_0]$ for classification of categorical data such as $y = 0$ or 1.
- ESL chapter 7.2-7.3 provides much more detail on MSE.

Aside: Proof

- Proof: Let $\widehat{f}_0$ denote $\widehat{f}(\mathbf{x}_0)$

$$
\begin{aligned}
& E[(y_0 - \widehat{f}_0)^2] \\
= \; & E[(\widehat{f}_0 - y_0)^2] \\
= \; & E[(\widehat{f}_0 - f_0 - \varepsilon)^2] \text{ as } y_0 = f_0 + \varepsilon \\
= \; & E[\{\widehat{f}_0 - f_0\}^2] + E[\varepsilon^2] \text{ as } \varepsilon \perp X \text{ and } E[\varepsilon] = 0 \\
= \; & E[\{(\widehat{f}_0 - E[\widehat{f}_0]) + (E[\widehat{f}_0] - f_0)\}^2] + E[\varepsilon^2] \\
= \; & E[(\widehat{f}_0 - E[\widehat{f}_0])^2] + (E[\widehat{f}_0] - f_0)^2 + E[\varepsilon^2] \text{ as cross term } = 0 \\
= \; & Var[\widehat{f}_0] + \{Bias(\widehat{f}_0)\}^2 + Var(\varepsilon).
\end{aligned}
$$

2. Linear Regression

- Standard material.
- Many methods are based on the linear model and one can make it quite flexible with polynomials, splines, interactions, ...
- And many methods for linear models extend to nonlinear models.

3. Cross-Validation

- Randomly divide available data into two parts
  - 1. training set
    - ★ model is fit on training set
  - 2. validation set or hold-out set
    - ★ MSE is computed for consequent predictions in validation set.

Single-split Validation

- E.g. Random half of sample is training and remaining is test data.
- Simple example is to choose the degree k of a polynomial in scalar regressor X

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_k X^k + \varepsilon.$$

- Then
  - 1. For each degree $k = 0, ..., p$
    - ★ estimate on the training set to get $\widehat{\beta}_k's$
    - ★ predict on the validation set to get $\widehat{Y}_k's$ and MSE_k
  - 2. Choose the degree k with lowest MSE_k.
- Problems with this single-split validation
  - 1. Lose precision due to smaller training set, so may actually overestimate the test error rate (MSE) of the model.
  - 2. And answers depend a lot on the particular single split.

Leave-one-out Cross Validation (LOOCV)

- Use a single observation for validation and $(n-1)$ for training
  - $\widehat{y}_{(-i)}$ is $\widehat{y}_i$ prediction after OLS on observations $1, .., i-1, i+1, ..., n$.
  - Cycle through all n observations doing this.
- Then LOOCV measure is

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_{(-i)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_{(-i)})^2$$

- Requires n regressions in general, except for OLS can show

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{y}_i}{1-h_{ii}}\right)^2$$

  where $\widehat{y}_i$ is fitted value from OLS on the full sample
  and h_{ii} is i^{th} diagonal entry in the hat matrix $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}$.
- Use for local regression such as k-NN and kernel but not global regression.

k-fold Cross Validation

- Randomly divide data into K groups or folds of approx. equal size
  - First fold is the validation set
  - Method is fit in the remaining $K-1$ folds
  - Compute MSE on the first fold
  - Repeat K times (drop second fold, third fold, ..) yields

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{k=1}^{K} MSE_{(j)}.$$

- Typically $K = 5$ or $k = 10$.
- LOOCV is case $k = n$.
  - LOOCV is not as good as the n folds are highly correlated with each other leading to higher variance
  - $k = 5$ or $k = 10$ has lower variance with bias still reasonable
  - LOOCV used for nonparametric regression where want good **local** fit.

k-fold Cross-Validation: one standard error rule

- k folds gives k estimates $MSE_{(1)}, ..., MSE_{(k)}$
  - this yields standard error of $\text{CV}_{(k)}$

$$\text{se}(CV_{(k)}) = \sqrt{\frac{1}{k=1}\sum_{j=1}^{k}(MSE_{(j)} - CV_{(k)})^2}$$

.

- Consider polynomial model of degree p.
  - one standard error rule computes CV and se(CV) for $p = 1, 2, ...$
  - then choose the lowest p for which CV is within one se(CV) of minimum CV.
- ESL Chs.7.4-7.10 has much more detail on cross-validation and on estimating training error and test error for MSE loss and more general loss functions.
- ESL Chs.7.11 presents the ".632 estimator" that is an adaptation of the usual bootstrap to correctly estimate test data MSE.

4. Subset Selection of Regressors

- General idea is to
  - 1. For $k = 1, 2, ..., p$ choose a "best" model with k regressors
  - 2. Choose among these p models based on model fit with penalty for larger models.
- Methods include
  - best subset
  - forwards stepwise
  - backwards stepwise
  - hybrid.

Goodness of fit criteria

- Define residual sum of squares and estimated error variance

$$RSS = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \text{ and } \widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2.$$

- Model selection criteria for model with k regressors.

Mallows C_p	C_p	$= \frac{1}{n}(RSS + 2k\widehat{\sigma}_p^2)$
Akaike information criteria	AIC	$= n\ln\widehat{\sigma}^2 + 2k + n(1 + \ln 2\pi)$
Bayesian information criteria	BIC	$= n\ln\widehat{\sigma}^2 + k\ln n + n(1 + \ln 2\pi)$
Adjusted R-squared	$\overline{R}^2$	$= 1 - \frac{RSS/(n-k-1)}{TSS/(n-1)}$

  - **IMPORTANT:** Here $\widehat{\sigma}_p^2$ is for the full model with p regressors.
- Note: Econometrics books use a different formula for AIC and BIC, using $\widehat{\sigma}^2$ in the fitted model; not $\widehat{\sigma}_p^2$ for the full model with $k = p$.
- Note: k is the effective degrees of freedom which may differ from the number of regressors e.g. ridge, lasso, PCA, ... See ESL 3.4, 5.4.
  - and instead of LOOCV use generalized cross validation (ESL p.244).

Subset Selection Procedures

- Best subset
  - For each $k = 1, ..., p$ find the model with lowest RSS (highest R^2)
  - Then use AIC etc. or CV to choose among the p models (want lowest test MSE)
  - Problem: 2^p total models to estimate.
- Stepwise forwards
  - Start with 0 predictors and add the regressor with lowest RSS
  - Start with this new model and add the regressor with lowest RSS
  - etc.
  - Requires $p + (p-1) + \cdots 1 = p(p+1)/2$ regressions.
- Stepwise backwards
  - similar but start with p regressors and drop weakest regressor, etc.
  - requires $n < p$.
- Hybrid
  - forward selection but after new model found drop variables that do not improve fit.

Subset Selection Procedures (continued)

- There are algorithms to speed these methods up
  - e.g. leaps and bounds procedure.
- Near enough may be good enough
  - best subsets gives the best model for the training data
  - but stepwise methods will get close and are much faster.

Subset Selection and Cross Validation

- Need to correctly combine cross validation and subset selection
  - 1. Divide sample data into K folds at random
  - 2. For each fold find best model with 0, 1, ..., p regressors and compute test error using the left out fold
  - 3. For each model size compute average test error over the K folds
  - 4. Choose model size with smallest average test error (or use one standard error rule)
  - 5. Using all the data find and fit the best model of this size.

5. Shrinkage Methods

- Shrinkage estimators minimize RSS with a penalty
  - this shrinks parameter estimates towards zero
- The extent of shrinkage is determined by a **tuning parameter**
  - this is determined by cross-validation.
- Ridge and lasso are not invariant to rescaling of regressors, so first standardize
  - so x_{ij} below is actually $(x_{ij} - \bar{x}_j)/s_j$
  - $\mathbf{x}_i$ does not include an intercept nor does data matrix $\mathbf{X}$
  - we can recover intercept β_0 as $\widehat{\beta}_0 = \bar{y}$.
- So work with $Y = \mathbf{X}'\boldsymbol{\beta} + \varepsilon = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$
  - instead of $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$.

Ridge Regression

- The **ridge estimator** $\widehat{\boldsymbol{\beta}}_\lambda$ of $\boldsymbol{\beta}$ minimizes

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda(||\boldsymbol{\beta}||_2)^2$$

  - where $\lambda \geq 0$ is a tuning parameter
  - $||\boldsymbol{\beta}||_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$ is L2 norm.
- Equivalently the ridge estimator minimizes RSS subject to $\sum_{j=1}^{p} \beta_j^2 \leq s$.
- The ridge estimator is

$$\widehat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}'\mathbf{y}.$$

- Features
  - $\widehat{\boldsymbol{\beta}}_\lambda \rightarrow \widehat{\boldsymbol{\beta}}_{OLS}$ as $\lambda \rightarrow 0$ and $\widehat{\boldsymbol{\beta}}_\lambda \rightarrow \mathbf{0}$ as $\lambda \rightarrow \infty$.
  - best when many predictors important with coeffs of similar size
  - best when LS has high variance
  - algorithms exist to quickly compute $\widehat{\boldsymbol{\beta}}_\lambda$ for many values of λ
  - then choose λ by cross validation.

Ridge Derivation

- 1. Objective function includes penalty
  - $Q(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}'\boldsymbol{\beta}$
  - $\partial Q(\boldsymbol{\beta})/\partial\boldsymbol{\beta} = -2\mathbf{X}'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta} = \mathbf{0}$
  - $\Rightarrow \mathbf{X}'\mathbf{X}\boldsymbol{\beta} + \lambda\mathbf{I}\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}$
  - $\Rightarrow \widehat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$.
- 2. Form Lagrangian (multiplier is λ) from objective function and constraint
  - $Q(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$ and constraint $\boldsymbol{\beta}'\boldsymbol{\beta} \leq s$
  - $L(\boldsymbol{\beta}, \lambda) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda(\boldsymbol{\beta}'\boldsymbol{\beta} - s)$
  - $\partial L(\boldsymbol{\beta}, \lambda)/\partial\boldsymbol{\beta} = -2\mathbf{X}'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta} = \mathbf{0}$
  - $\Rightarrow \widehat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$
  - Here $\lambda = \partial L_{opt}(\boldsymbol{\beta}, \lambda, s)/\partial s$.

Lasso (Least Absolute Shrinkage And Selection)

- The **lasso estimator** $\widehat{\boldsymbol{\beta}}_\lambda$ of $\boldsymbol{\beta}$ minimizes

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j| = RSS + \lambda||\boldsymbol{\beta}||_1$$

  - where $\lambda \geq 0$ is a tuning parameter
  - $||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$ is L1 norm.
- Equivalently the lasso estimator minimizes RSS subject to $\sum_{j=1}^{p} |\beta_j| \leq s$.
- Features
  - best when a few regressors have $\beta_j \neq 0$ and most $\beta_j = 0$
  - leads to a more interpretable model than ridge.
- Lasso and ridge are special cases of bridge
  - minimize $\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|^\gamma$ for specified $\gamma > 0$.

Lasso versus Ridge

- Consider simple case where $n = p$ and $\mathbf{X} = \mathbf{I}$.
- OLS: $\widehat{\boldsymbol{\beta}}^{OLS} = (\mathbf{I}'\mathbf{I})^{-1}\mathbf{I}'\mathbf{y} = \mathbf{y}$
  - so $\widehat{\beta}_j^{OLS} = y_j$
- Ridge: $\widehat{\boldsymbol{\beta}}^{R} = (\mathbf{I}'\mathbf{I} + \lambda\mathbf{I})^{-1}\mathbf{I}'\mathbf{y} = \mathbf{y}/(1+\lambda)$
  - so $\widehat{\beta}_j^{R} = y_j/(1+\lambda)$
  - shrink towards zero
- Lasso shrinks some a bit towards 0 and sets others $= 0$

$$\widehat{\beta}_j^{L} = \begin{cases} y_j - \lambda/2 & \text{if } y_j > \lambda/2 \\ y_j + \lambda/2 & \text{if } y_j < -\lambda/2 \\ 0 & \text{if } |y_j| > \lambda/2 \end{cases}$$

- Best subset of size M in this example

$$\widehat{\beta}_j^{BS} = \widehat{\beta}_j \times \mathbf{1}[|\widehat{\beta}_j| \geq |\widehat{\beta}_{(M)}|]$$

where $\widehat{\beta}_{(M)}$ is the M^{th} largest OLS coefficient.

Lasso versus Ridge

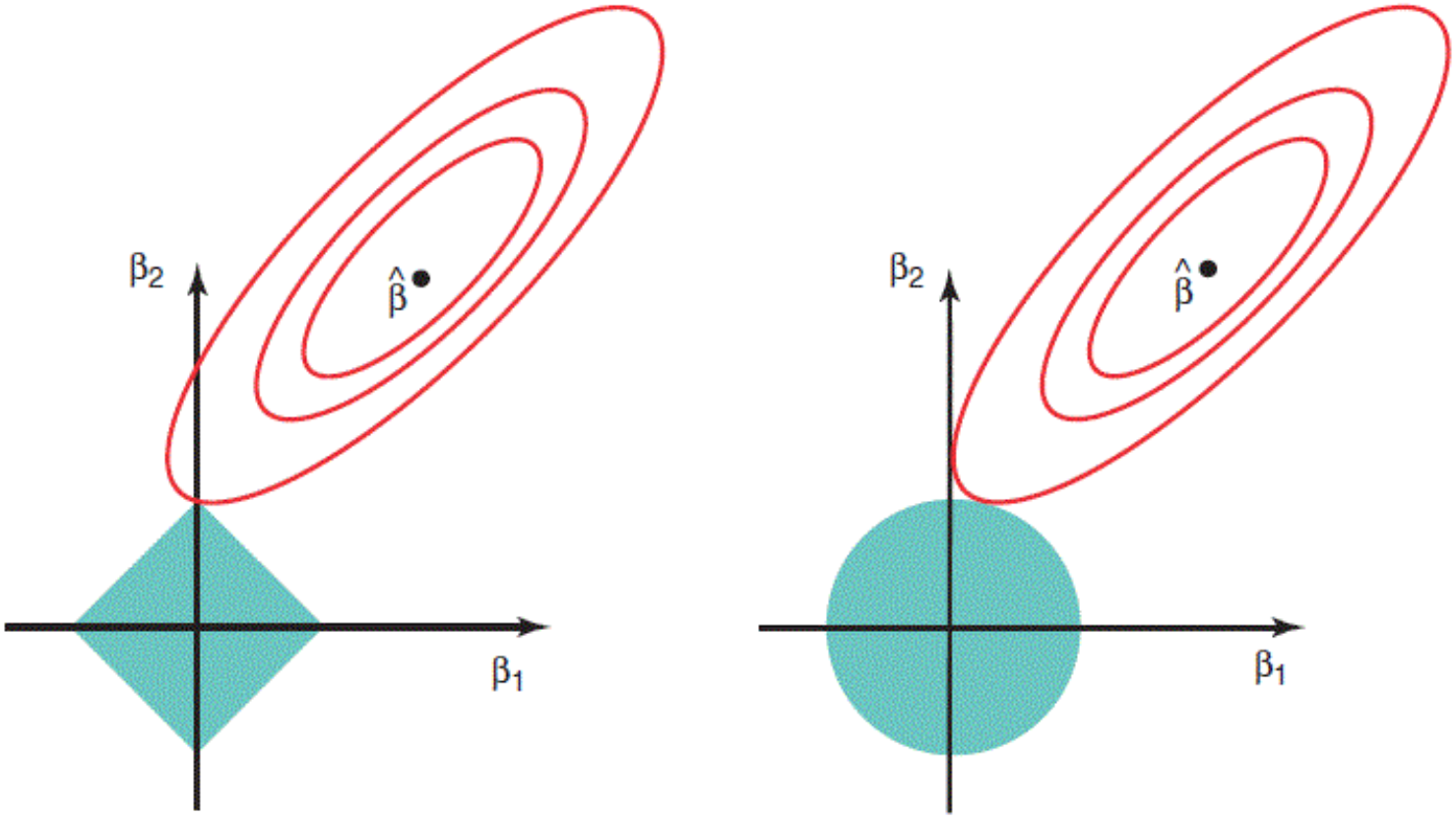


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Least Angle Regression (LAR)

- See ESL p.73-79, 86-93
- Lasso is a minor adaptation of LAR
  - Lasso is usually estimated using a LAR procedure.

6. Dimension Reduction

- Reduce from p regressors to $M < p$ linear combinations of regressors
  - Form $\mathbf{X}^* = \mathbf{XA}$ where $\mathbf{A}$ is $p \times M$ and $M < p$
  - $\mathbf{Y} = \beta_0 + \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$ reduced to
  - $\mathbf{Y} = \beta_0 + \mathbf{X}^*\boldsymbol{\beta} + \mathbf{v}$
    $= \beta_0 + \mathbf{X}\boldsymbol{\beta}^* + \mathbf{v}$ where $\boldsymbol{\beta}^* = \mathbf{A}\boldsymbol{\beta}$.
- Two methods
  - 1. Principal components
    - ★ use only $\mathbf{X}$ to form $\mathbf{A}$ (unsupervised)
  - 2. Partial least squares
    - ★ also use relationship between $\mathbf{y}$ and $\mathbf{X}$ to form $\mathbf{A}$ (supervised).
- For both should standardize regressors as not scale invariant.
- And often use cross-validation to determine M.

Principal Components Analysis (PCA)

- Eigenvalues and eigenvectors of $\mathbf{X}'\mathbf{X}$
  - Let $\tilde{} = \text{Diag}[\lambda_j]$ to be $p \times p$ vector of eigenvalues of $\mathbf{X}'\mathbf{X}$
  - Order so $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_1$
  - Let $\mathbf{H} = [\mathbf{h}_1 \ \cdots \ \mathbf{h}_p]$ be $p \times p$ vector of corresponding eigenvectors
  - $\mathbf{X}'\mathbf{X}\mathbf{h}_1 = \lambda_1\mathbf{h}_1$ and $\mathbf{X}'\mathbf{X}\mathbf{H} = \tilde{}\mathbf{H}$ and $\mathbf{H}'\mathbf{H}$
- Then
  - the j^{th} principal component is $\mathbf{X}\mathbf{h}_j$
  - $M-$principal components regression uses $\mathbf{X}^* = \mathbf{X}\mathbf{A}$ where $\mathbf{A} = [\mathbf{h}_1 \ \cdots \ \mathbf{h}_M]$.

Principal Components Analysis

- The first principal component has the largest sample variance among all normalized linear combinations of the columns of $\mathbf{X}$.
- The second principal component has the largest variance subject to being orthogonal to the first, and so on.
- PCA is unsupervised so seems unrelated to $\mathbf{Y}$ but
  - ESL says does well in practice.
  - PCA has the smallest variance of any estimator that estimates the model $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$ with i.i.d. errors subject to constraint $\mathbf{C}\boldsymbol{\beta} = \mathbf{c}$ where $\dim[\mathbf{C}] \leq \dim[\mathbf{X}]$.
  - PCA discards the $p - M$ smallest eigenvalue components whereas ridge does not, though ridge does shrink towards zero the most for the smallest eigenvalue components (ESL p.79).

Partial Least Squares

- Partial least squares produces a sequence of orthogonal linear combinations of the regressors.
- 1. Standardize each regressor to have mean 0 and variance 1.
- 2. Regress y individually on each $\mathbf{x}_j$ and let $\mathbf{z}_1 = \sum_{j=1}^{p} \widehat{\theta}_{1j}\mathbf{x}_j$
- 3. Regress y on $\mathbf{z}_1$ and let $\widehat{\mathbf{y}}^{(1)}$ be prediction of $\mathbf{y}$.
- 4. Orthogonalize each $\mathbf{x}_j$ by regress on $\mathbf{z}_1$ to give $\mathbf{x}_j^{(1)} = \mathbf{x}_j - \mathbf{z}_1\widehat{\tau}_j$ where $\widehat{\tau}_j = (\mathbf{z}_1'\mathbf{z}_1)^{-1}\mathbf{z}_1'\mathbf{x}_j^{(1)}$.
- 5. Go back to step 1 with $\mathbf{x}_j$ now $\mathbf{x}_j^{(1)}$, etc.
  - When done $\widehat{\mathbf{y}} = \widehat{\mathbf{y}}^{(1)} + \widehat{\mathbf{y}}^{(2)} + \cdots$
- Partial least squares turns out to be similar to PCA
  - especially if R^2 is low.

7. High-Dimensional Models

- High dimensional simply means p is large relative to n
  - in particular $p > n$
  - n could be large or small.
- Problems with $p > n$:
  - C_p, AIC, BIC and $\overline{R}^2$ cannot be used.
  - due to multicollinearity cannot identify best model, just one of many good models.
  - cannot use regular statistical inference on training set
- Solutions
  - Forward stepwise, ridge, lasso, PCA are useful in training
  - Evaluate models using cross-validation or independent test data
    - ★ using e.g. R^2 or MSE.

8. Nonlinear Models

- Models with single regressor
  - 1. polynomial regression
  - 2. step functions
  - 3. regression splines
  - 4. smoothing splines
  - 5. local regression
  - polynomial is global while the others break range of x into pieces.
- Model with multiple regressors
  - generalized additive models.

Basis Functions

- General approach (scalar X for simplicity)

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \cdots + \beta_K(x_i) + \varepsilon_i$$

  - where $b_1, ..., b_K$ are basis functions that are fixed and known.
- Polynomial regression sets $b_j(x_i) = x_i^j$
  - typically $K \leq 3$ or 4.
  - fits globally and can overfit at boundaries.
- Step functions: separate fits in each interval (c_j, c_{j+1})
  - piecewise constant $b_j(x_i) = 1[c_j \leq x_i < c_{j+1}]$
  - piecewise linear use $1[c_j \leq x_i < c_{j+1}]$ and $x_i \times 1[c_j \leq x_i < c_{j+1}]$
  - problem is discontinuous at the cut points (does not connect)
  - solution is splines.

Splines

- Begin with piecewise linear with two knots at c and d

$$\begin{aligned} f(x) &= \alpha_1 1[x < c] + \alpha_2 x 1[x < c] + \alpha_4 1[c \leq x < d] \\ &\quad + \alpha_4 x 1[c \leq x < d] + \alpha_5 1[x \geq d] + \alpha_6 x 1[x \geq d]. \end{aligned}$$

- To make continuous at c (so $f(c-) = f(c)$) and d we need two constraints

$$\begin{aligned} \text{at } c: &\quad \alpha_1 + \alpha_2 c = \alpha_3 + \alpha_4 c \\ \text{at } d: &\quad \alpha_3 + \alpha_4 d = \alpha_5 + \alpha_6 d. \end{aligned}$$

- Alternatively introduce truncated power basis functions

$$h_+(x) = x_+ = \begin{cases} x & x > 0 \\ 0 & \textit{otherwise}. \end{cases}$$

- Then the following imposes the two constraints (so have $6 - 2 = 4$ regressors)

$$f(x) = \beta_0 + \beta_1 x + \beta_2 (x - c)_+ + \beta_2 (x - c)_+$$

Cubic Regression Splines

- This is the standard.
- Piecewise cubic model with K knots
  - require $f(x)$, $f'(x)$ and $f''(x)$ to be continuous at the K knots
- Then can do OLS with

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - c_1)_+^3 + \cdots + \beta_{(3+K)} (x - c_K)_+^3$$

  - for proof when $K = 1$ see ISL exercise 7.1.
- This is the lowest degree regression spline where the graph of $\widehat{f}(x)$ on x seems smooth and continuous to the naked eye.
  - There is no real benefit to a higher order spline.

Other Splines

- Regression splines overfit at boundaries.
- A natural spline is an adaptation that restricts the relationship to be linear past the lower and upper boundaries of the data.
- Regression splines and natural splines require choosing the cut points (e.g. use quintiles of x)
- Smoothing splines use all distinct values of x as knots but then add a smoothness penalty that penalizes curvature.
  - The function $g(\cdot)$ minimizes

$$\sum_{i=1}^{n}(y_i - g(\mathbf{x}_i))^2 + \lambda \int_a^b g''(t)dt \text{ where } a \leq \text{all } x_i \leq b.$$

  - $\lambda = 0$ connects the data points and $\lambda \rightarrow \infty$ gives OLS.
- B splines are discussed in ESL ch.5 appendix.

Local Polynomial Regression

- Local polynomial at $x = x_0$ of degree d

$$\widehat{f}(x_0) = \sum_{j=0}^{d} \widehat{\beta}_j^0 x_i^j$$

  - where $\widehat{\beta}_0^0, ..., \widehat{\beta}_d^0$ minimize the locally weighted least squares

$$\sum_{i=1}^{n} K_\lambda(x_0, x_i) \left(y_i - \sum_{j=0}^{d} \beta_j^0 x_i^j\right)^2.$$

- The weights $K_\lambda(x_0, x_i)$ are given by a kernel function and are highest at $x_i = x_0$.
- The tuning parameter λ determines how far out to average.
- $d = 0$ is local constant (Nadaraya-Watson kernel regression).
- $d = 1$ is local linear.
- Can generalize to local ML max $\sum_{i=1}^{n} K_\lambda(x_0, x_i) \ln(f(y_i, x_i, \theta^0)$.

Flexible Models with Multiple Predictors

- For splines use multivariate adaptive regression splines (MARS) - see ESL ch.9.4.
- For fully nonparametric regression run into curse of dimensionality problems
  - so place some structure.
- Economists use single-index models with $f(\mathbf{x}) = g(\mathbf{x}'\boldsymbol{\beta})$ with $g(\cdot)$ unspecified.
  - advantage is interpretability
  - project pursuit regression (below) generalizes.
- Regression trees are used a lot (next topic).
- Here consider
  - generalized additive models
  - neural networks.

Generalized Additive Models (GAMs)

- A linear combination of scalar functions

$$y_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i,$$

  where x_j is the j^{th} regressor and $f_j(\cdot)$ is (usually) determined by the data.
- Advantage is interpretability (due to each regressor appearing additively).
- Can make more nonlinear by including interactions such as $x_{i1} \times x_{i2}$ as a separate regressor.
- For $f_j(\cdot)$ unspecified reduces $p-$dimensional problem to sequence of one-dimensional problems.
- ESL ch.9.1.1 presents the backfitting algorithm when smoothing splines are used that minimize the penalized RSS

$$\text{PRSS}(\alpha, f_1, ..., f_p) = \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)$$

- Problems implementing if many possible regressors.

Project Pursuit Regression

- See ESL chapter 11.2.
- The GAM is additive in functions $f_j(x_j)$, $j = 1, ..., p$, that are distinct for each regressor.
- Instead be additive in functions of $x_1, ..., x_p$, $m = 1, ..., M$.
- Project pursuit regression minimizes $\sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2$ where

$$f(\mathbf{x}_i) = \sum_{m=1}^{M} g_m(\mathbf{x}_i'\boldsymbol{\omega}_m)$$

  - additive in derived features $\mathbf{x}'\boldsymbol{\omega}_m$ rather than in the $x_j's$.
- Here the $g_m(\cdot)$ functions are unspecified.
- This is a multi-index model with case $M = 1$ being a single-index model.

Neural Networks

- See ESL chapter 11.2-11.10.
- Neural network is a richer model for $f(\mathbf{x}_i)$ than project pursuit, but unlike project pursuit all functions are specified. Only parameters need to be estimated.
- Consider a neural network with two layers: Y depends on $\mathbf{Z}'s$ (a hidden layer) that depend on $\mathbf{X}'s$.

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \mathbf{X}'\alpha_m) \qquad\qquad m = 1, ..., M \\ &\text{usually } \sigma(v) = 1/(1+e^{-v}) \\ T &= \beta_0 + \mathbf{Z}'\boldsymbol{\beta} \\ f(\mathbf{X}) &= g(T) \\ &\text{usually } g(T) = T \end{aligned}$$

- So $f(\mathbf{x}_i) = \sum_{m=1}^{M} \sigma(\alpha_{0m} + \mathbf{x}_i'\alpha_m)$ where $\sigma(v) = 1/(1+e^{-v})$.
- We need to find the number M of hidden units and estimate the $\alpha's$.

Neural Networks (continued)

- Minimize the sum of squared residuals but need a penalty on $\alpha's$ to avoid overfitting.
  - Since penalty is introduced standardize $x's$ to (0,1).
  - Best to have too many hidden units and then avoid overfit using penalty.
- Neural nets are good for prediction
  - especially in speech recognition, image recognition, ...
  - but very difficult (impossible) to interpret.
- Estimate iteratively using iterative gradient methods
  - initially people used back propagation
  - faster is to use variable metric methods (such as BFGS) that avoid using the Hessian or use conjugate gradient methods
  - different starting values lead to different estimates (nonconvex objective function) so use several starting values and average results or use bagging.
- Deep learning uses nonlinear transformations such as neural networks
  - deep nets are an improvement on original neural networks.

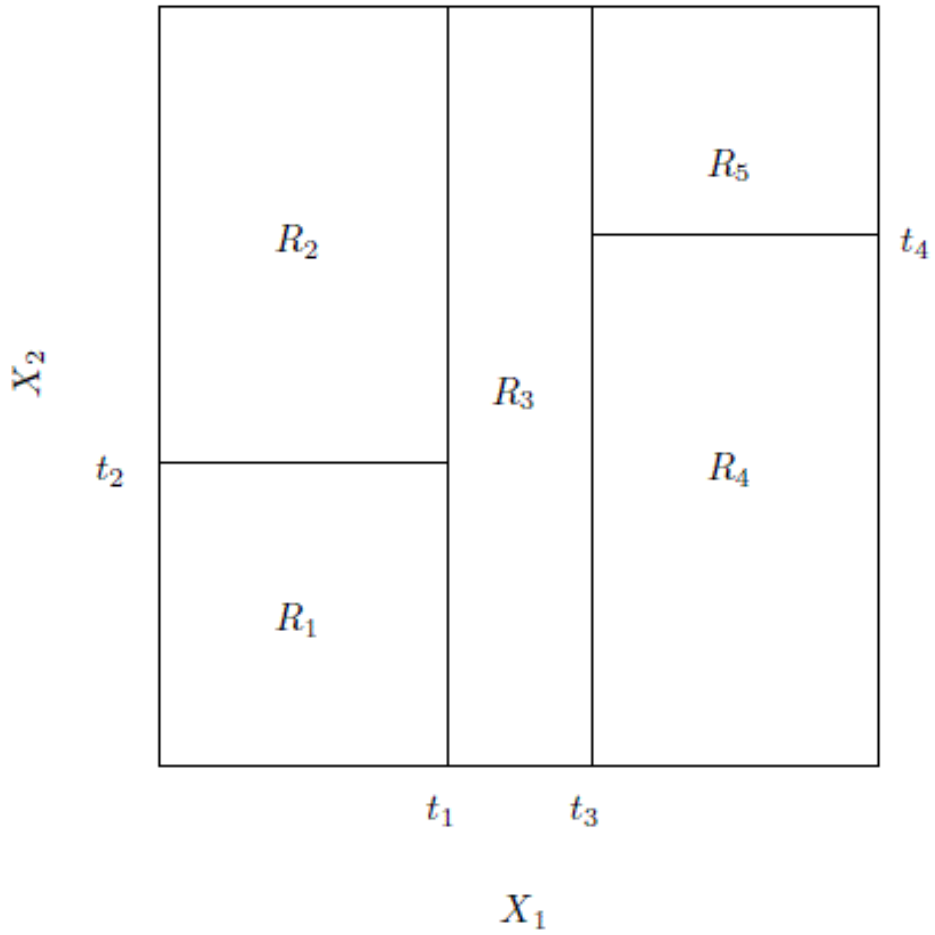
9. Tree Based Methods

- Regression Trees
  - sequentially split $\mathbf{x}'s$ into rectangular regions in way that reduces RSS
  - then $\widehat{y}_i$ is the average of $y's$ in the region that $\mathbf{x}_i$ falls in
  - with J blocks RSS$= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2$.
- Need to determine both the regressor j to split and the split point s.
  - For any regressor j and s, define the pair of half-planes $R1(j,s) = \{X | X_j < s\}$ and $R2(j,s) = \{X | X_j \geq s\}$
  - Find the value of j and s that minimize

$$\sum_{i:\mathbf{x}_i \in R1(j,s)} (y_i - \bar{y}_{R1})^2 + \sum_{i:\mathbf{x}_i \in R1(j,s)} (y_i - \bar{y}_{R1})^2$$

    where $\bar{y}_{R1}$ is the mean of y in region $R1$ (and similar for $R2$).
  - Once this first split is found, split both $R1$ and $R2$ and repeat
  - Each split is the one that reduces RSS the most.
  - Stop when e.g. less than five observations in each region.

- The following diagram arises if (1) split X1 in two; (2) split the lowest X1 values on the basis of X2 into R1 and R2; (3) split the highest X1 values into two regions (R3 and R4/R5); (4) split the highest X1 values on the basis of X2 into R4 and R5.



- The model is of form $f(X) = \sum_{j=1}^{J} c_m \times \mathbf{1}[X \in R_j]$.
- The approach is a topdown greedy approach
  - top down as start with top of the tree
  - greedy as at each step the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.
- This leads to overfitting, so prune
  - use cost complexity pruning (or weakest link pruning)
  - this penalizes for having too many terminal nodes
  - see ISL equation (8.4).
- Regression trees are easy to understand if there are few regressors
- But they do not predict as well as chapter 6-7 methods
  - due to high variance (e.g. split data in two then can get quite different trees).
- Better methods (bagging, random forests and boosting) are given next.

Bagging (Bootstrap Aggregating)

- This method is a general method for improving prediction that works especially well for regression trees.
- Idea is that averaging reduces variance.
- So average regression trees over many samples
  - where different samples are obtained by bootstrap (so not completely independent of each other)
  - For each sample obtain a large tree and prediction $\widehat{f}_b(x)$.
  - Average all these predictions: $\widehat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(x)$.
- Get test error by using out-of-bag (OOB) observations not in the bootstrap sample
  - $\Pr[j^{th} \text{ obs not in resample}] = (1-\frac{1}{n})^n \rightarrow e^{-1} = 0.368 \simeq 1/3$.
  - this replaces cross validation.
- Interpretation of trees is now difficult so
  - record the total amount that RSS is decreased due to splits over a given predictor, averaged over all B trees.
  - A large value indicates an important predictor.

Random Forests

- The B bagging estimates are correlated in part because if a regressor is important it will appear near the top of the tree in each bootstrap sample.
  - The trees look similar from one resample to the next.
- As for boosting get bootstrap samples.
- But within each bootstrap sample each time a split in a tree is considered, use only a random sample of $m < p$ predictors in deciding the next split.
  - usually $m \simeq \sqrt{p}$.
- This reduces correlation across bootstrap resamples.
- Simple bagging is random forest with $m = p$.

Boosting

- This method is also a general method for improving prediction.
- Regression trees use a greedy algorithm.
- Boosting uses a slower algorithm to generate a sequence of trees
  - each tree is grown using information from previously grown trees
  - and is fit on a modified version of the original data set
  - boosting does not involve bootstrap sampling.
- Specifically (with λ a penalty parameter)
  - given current model b fit a decision tree to model $b's$ residuals (rather than the outcome Y)
  - then update $\widehat{f}(x) =$ previous $\widehat{f}(x) + \lambda \widehat{f}^b(x)$
  - then update the residuals $r_i =$ previous $r_i - \lambda \widehat{f}^b(x_i)$
  - the boosted model is $\widehat{f}(x) = \sum_{b=1}^{B} \lambda \widehat{f}^b(x_i)$.

10. Classification: Loss Function

- $y's$ are now categorical (e.g. binary if two categories).
- Use (0,1) loss function (ESL pp.20-21).
  - 0 if correct classification and 1 if misclassified.
- $L(G, \widehat{G}(X))$ is 0 on diagonal of $K \times K$ table and 1 elsewhere
  - where G is actual categories and $\widehat{G}$ is predicted categories.
- Then minimize the expected prediction error

$$\begin{aligned} EPE &= E_{G,X}[L(G, \widehat{G}(X))] \\ &= E_X\left[\sum_{k=1}^{K} L(G, \widehat{G}(X)) \times \Pr[G_k|X]\right] \end{aligned}$$

- Minimize EPE pointwise

$$\begin{aligned} f(x) &= \arg\min_{g \in G}\left[\sum_{k=1}^{K} L(G_k, g) \times \Pr[G_k|X = x]\right] \\ \partial/\partial c &= \arg\min_{g \in G}[1 - \Pr[g|X = x]] \\ &= \max_{g \in G} \Pr[g|X = x] \end{aligned}$$

- Called Bayes classifier. Classify the most probable class.

Test Error Rate

- Instead of MSE we use the **error rate**

$$\text{Error rate} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[y_i \neq \widehat{y}_i],$$

  where indicator $\mathbf{1}[A] = 1$ if event A happens and $= 0$ otherwise.
- The **test error rate** is for the n_0 observations in the test sample

$$\text{Ave}(\mathbf{1}[y_0 \neq \widehat{y}_0]) = \frac{1}{n_0}\sum_{i=1}^{n_0} \mathbf{1}[y_{0i} \neq \widehat{y}_{0i}].$$

- Cross validation uses number of misclassified observations. e.g. LOOCV is

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Err_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[y_i \neq \widehat{y}_{(-i)}].$$

- Some terminology
  - A confusion matrix is a $K \times K$ table of counts of $(y, \widehat{y})$
  - In 2×2 case with $y = 1$ or 0
    - ★ sensitivity is % of $y = 1$ with prediction $\widehat{y} = 1$
    - ★ specificity is % of $y = 0$ with prediction $\widehat{y} = 0$
    - ★ ROC curve plots sensitivity against 1−sensitivity as threshold for $\widehat{y} = 1$

Classification Methods

- Regression methods predict probabilities and then use Bayes classifier.
  - logistic regression, multinomial regression, k nearest neighbors.
- Discriminant analysis additionally assumes a distribution for the x's.
- Support vector classifiers and support vector machines use separating hyperplanes of X and extensions.

Logit and k-NN

- Directly model $p(\mathbf{X}) = \Pr[y|\mathbf{X}]$.
- Logistic (logit) regression for binary case obtains MLE for

$$\ln\left(\frac{p(\mathbf{X})}{1-p(\mathbf{X})}\right) = \beta_0 + \mathbf{X}'\boldsymbol{\beta}.$$

- Statisticians implement using a statistical package for the class of generalized linear models (GLM)
  - logit is in the Bernoulli (or binomial) family with logistic link
  - logit is often the default.
- k-nearest neighbors KNN for many classes
  - $\Pr[Y = j|\mathbf{X} = \mathbf{X}_0] = \frac{1}{K}\sum_{i\in N_0} \mathbf{1}[y_i = j]$
  - where N_0 is the K observations on $\mathbf{X}$ closest to $\mathbf{X}_0$
- In both cases we obtain predicted probabilities
  - then assign to the class with highest predicted probability.

Linear Discriminant Analysis

- Discriminant analysis specifies a joint distribution for $(Y, \mathbf{X})$.
- Linear discriminant analysis with K categories
  - assume $\mathbf{X}|Y = k$ is $N(\boldsymbol{\mu}_k, \circ)$ with density $f_k(\mathbf{X}) = \Pr[\mathbf{X} = \mathbf{x}|Y = k]$
  - and let $\pi_k = \Pr[Y = k]$
- The desired $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$ is obtained using Bayes theorem

$$\Pr[Y = k|\mathbf{X} = \mathbf{x}] = \frac{\pi_k f_k(\mathbf{X})}{\sum_{j=1}^{K} \pi_j f_j(\mathbf{X})}.$$

- Assign observation $\mathbf{X} = \mathbf{x}$ to class k with largest $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$.
  - Upon simplification this is equivalent to choosing model with largest **discriminant function**

$$\delta_k(\mathbf{x}) = \mathbf{x}' \circ^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k' \circ^{-1} \boldsymbol{\mu}_k + \ln \pi_k$$

  - use $\widehat{\boldsymbol{\mu}}_k = \bar{\mathbf{x}}_k$, $\widehat{\circ} = \widehat{\text{Var}}[\mathbf{x}_k]$ and $\widehat{\pi}_k = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[y_i = k]$.
- Called linear discriminant analysis as linear in $\mathbf{x}$.

Quadratic Discriminant Analysis

- Quadratic discriminant analysis
  - allow different variances so $\mathbf{X}|Y = k$ is $N(\boldsymbol{\mu}_k, \circ_k)$
- Upon simplification, the Bayes classifier assigns observation $\mathbf{X} = \mathbf{x}$ to class k which has largest

$$\delta_k(\mathbf{x}) = -\frac{1}{2}\mathbf{x}' \circ_k^{-1} \mathbf{x} + \mathbf{x}' \circ_k^{-1} \boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k{}' \circ_k^{-1} \boldsymbol{\mu}_k - \frac{1}{2}\ln|\circ_k| + \ln \pi_k$$

  - called quadratic discriminant analysis as linear in $\mathbf{x}$
- Use rather than LDA only if have a lot of data as requires estimating many parameters.

LDA versus Logit

- ESL ch.4.4.5 compares linear discriminant analysis and logit
  - Both have log odds ratio linear in X
  - LDA is joint model if Y and X versus logit is model of Y conditional on X.
  - In the worst case logit ignoring marginal distribution of X has a loss of efficiency of about 30% asymptotically in the error rate.
  - If $X's$ are nonnormal (e.g. categorical) then LDA still doesn't do too bad.

Linear and Quadratic Boundaries

- LDA uses a linear boundary to classify and QDA a quadratic

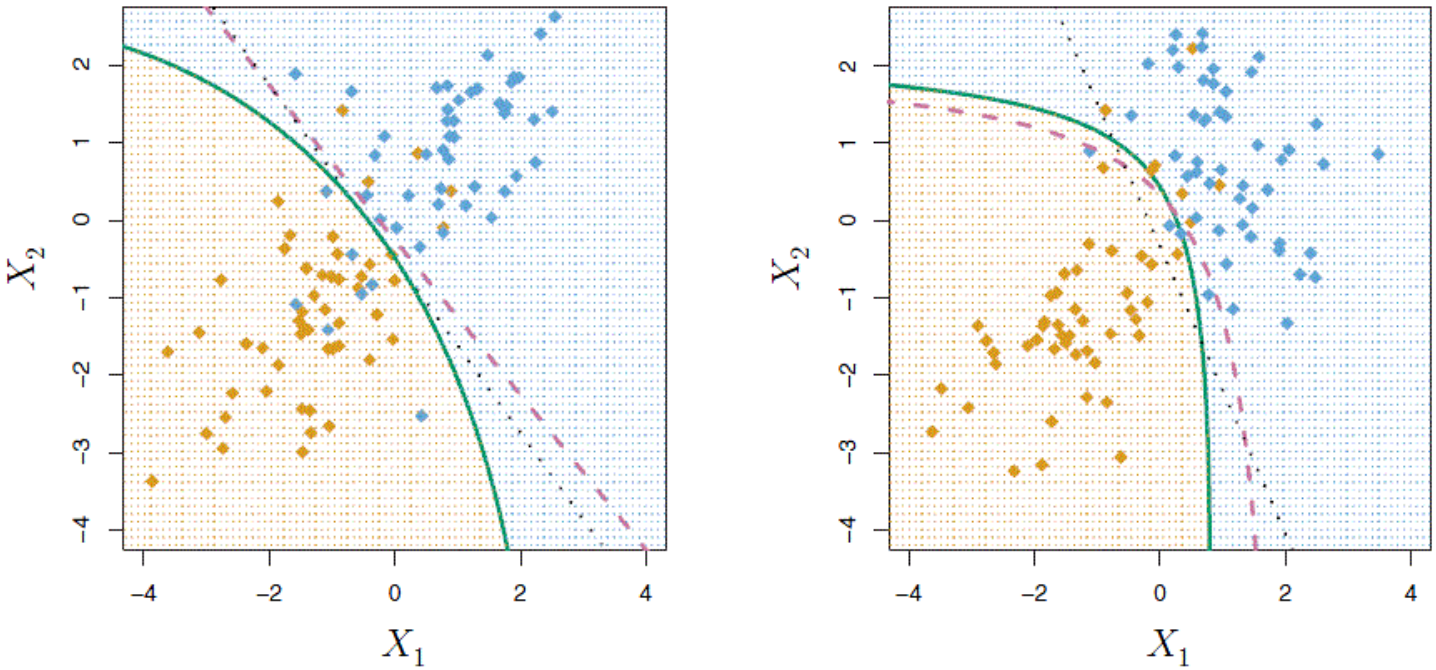


FIGURE 4.9. Left: *The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with* $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$. *The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA.* Right: *Details are as given in the left-hand panel, except that* $\mathbf{\Sigma}_1 \neq \mathbf{\Sigma}_2$. *Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.*

Support Vector Classifier

- Build on LDA idea of linear boundary to classify when $K = 2$.
- Maximal margin classifier
  - classify using a separating hyperplane (linear combination of X)
  - if perfect classification is possible then there are an infinite number of such hyperplanes
  - so use the separating hyperplane that is furthest from the training observations
  - this distance is called the maximal margin.
- Support vector classifier
  - generalize maximal margin classifier to the nonseparable case
  - this adds slack variables to allow some y's to be on the wrong side of the margin
  - $\text{Max}_{\boldsymbol{\beta},\varepsilon} M$ (the margin - distance from separator to training X's) subject to $\boldsymbol{\beta}'\boldsymbol{\beta} \neq \mathbf{1}$, $y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq M(1 - \varepsilon_i)$, $\varepsilon_i \geq 0$ and $\sum_{i=1}^{n} \varepsilon_i \leq C$.

Support Vector Machines

- The support vector classifier has linear boundary
  - $f(\mathbf{x}_0) = \beta_0 + \sum_{i=1}^{n} \alpha_i \mathbf{x}_0' \mathbf{x}_i$, where $\mathbf{x}_0' \mathbf{x}_i = \sum_{j=1}^{p} x_{0j} x_{ij}$.
- The support vector machine has nonlinear boundaries
  - $f(\mathbf{x}_0) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_0, \mathbf{x}_i)$ where $K(\cdot)$ is a kernel
  - polynomial kernel $K(\mathbf{x}_0, \mathbf{x}_i) = (1 + \sum_{j=1}^{p} x_{0j} x_{ij})^d$
  - radial kernel $K(\mathbf{x}_0, \mathbf{x}_i) = \exp(-\gamma \sum_{j=1}^{p} (x_{0j} - x_{ij})^2)$
- Now extend to $K > 2$ classes (see ISL ch. 9.4).
  - one-versus-one or all-pairs approach
  - one-versus-all approach.

11. Unsupervised Learning

- Challenging area: no y, only $\mathbf{X}$.
- Principal components analysis.
- Clustering Methods
  - k means clustering.
  - hierarchical clustering.

Principal Components

- Initially discussed in section 6 on dimension reduction.
- Goal is to find a few linear combinations of X that explain a good fraction of the total variance $\sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$ for mean 0 X's.
- $Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$ where $\sum_{j=1}^{p} \phi_{jm}^2 = 1$ and ϕ_{jm} are called factor loadings.
- A useful statistic is the proportion of variance explained (PVE)
  - a scree plot is a plot of PVE_m against m
  - and a plot of the cumulative PVE by m components against m.
  - choose m that explains a "sizable" amount of variance
  - ideally find interesting patterns with first few components.
- Easier when used PCA earlier in supervised learning as then observe Y and can treat m as a tuning parameter.

K-Means Clustering

- Goal is to find homogeneous subgroups among the X.
- K-Means splits into K distinct clusters where within cluster variation is minimized.
- Let $W(C_k)$ be measure of variation
  - Minimize$_{C_1,...,C_k} \sum_{k=1}^{K} W(C_k)$
  - Euclidean distance $W(C_k) = \frac{1}{n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$
- Global maximum requires K^n partitions.
- Instead use algorithm 10.1 (ISL p.388) which finds a local optimum
  - run algorithm multiple times with different seeds
  - choose the optimum with smallest $\sum_{k=1}^{K} W(C_k)$.

Hierarchical Clustering

- Do not specify K.
- Instead begin with n clusters (leaves) and combine clusters into branches up towards trunk
  - represented by a dendrogram
  - eyeball to decide number of clusters.
- Need a dissimilarity measure between clusters
  - four types of linkage: complete, average, single and centroid.
- For any clustering method
  - it is a difficult problem to do unsupervised learning
  - results can change a lot with small changes in method
  - clustering on subsets of the data can provide a sense of robustness.

12. Introduction to R

- Chapter 2 (Statistical Learning)
  - define: A=matrix(data=c(1,2,3,4), nrow=2, ncol=2)
  - list subcomponent: A[1,2]
  - remove: rm()
  - import: read.table() or read.csv
  - set the dataset for analysis: fix()
  - graphics: plot(x,y,xlab="x-axis",ylab-"y-axis",main="plot y xs x")
  - draw line: abline()
  - summary statistics: summary()
- Chapter 3 (Regression)
  - install package on computer: install.packages("package")
  - call package for this run: library(package)
  - OLS: lm.fit = lm(y~x,data)
  - se results: summary(lm.fit)
  - confidence interval: confint(lm.fit)
  - predict: predict()
  - write functions: Loadlibraries=function()

Introduction to R (continued)

- Chapter 4 (Classification)
  - logistic: glm(...,family=binomial)
  - LDA: lda() function in MASS library
  - QDA: qda() function in MASS library
  - kNN: knn() function in class library
- Chapter 5 (Cross-Validation and Bootstrap)
  - set seed: set.seed()
  - training set: sample(n,m) where n=#totalobs and m<n is #training
  - LOOCV: glm() and cv.glm() for and GLM
  - loops: for (in in 1:10){ + ... + ... + }
  - bootstrap: boot() function in boot library

Introduction to R (continued)

- Chapter 6 (Lienar Selection and Regularization)
  - best subset: regsubsets() in leaps library
  - forward stepwise: regsubsets(,method="forward")
  - backward stepwise: regsubsets(,method="backward")
  - ridge: glmnet(,alpha=0) function in glmnet library
  - lasso: glmnet(,alpha=1) function in glmnet library
  - CV for ridge/lasso: cv.glmnet()
  - principal components: pcr() function in pls library
  - CV for PCA: pcr(,validation="CV")
  - partial least squares: plsr() function in pls library

Introduction to R (continued)

- Chapter 7 (Nonlinear)
  - regression splines: bs(x,knots=c()) in lm() function
  - natural spline: ns(x,knots=c()) in lm() function
  - smoothing spline: function smooth.spline() in spline library (This does not use data frames. It needs data matrices.)
  - loess: function loess
  - generalized additive models: function gam() in gam library
- Chapter 8 (Tree-Based methods)
  - classification tree: function tree() in tree library
  - cross-validation: cv.tree() function
  - pruning: function prune.tree()
  - random forest: randomForest() in randomForest library
  - bagging: function randomForest()
  - boosting: gbm() function in library gbm

Introduction to R (continued)

- Chapter 9 (Support Vector Machines)
  - support vector classifier: svm(... kernel="linear") in e1071 library
  - support vector machine: svm(... kernel="polynomial") or svm(... kernel="radial") in e1071 library
  - receiver operator characteristic curve: rocplot in ROCR library.
- Chapter 10 (Unsupervised Learning)
  - principal components analysis: function prcomp()
  - k-means clusterning: function kmeans()
  - hierarchical clustering: function hclust()

References

- Undergraduate / Masters level book
  - **ISL:** Gareth James, Daniela Witten, Trevor Hastie and Robert Tibsharani (2013), An Introduction to Statistical Learning: with Applications in R, Springer.
  - free legal pdf at http://www-bcf.usc.edu/~gareth/ISL/
  - $25 hardcopy via http://www.springer.com/gp/products/books/mycopy
- Masters / PhD level book
  - **ESL:** Trevor Hastie, Robert Tibsharani and Jerome Friedman (2009), The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer.
  - free legal pdf at http://statweb.stanford.edu/~tibs/ElemStatLearn/index.html
  - $25 hardcopy via http://www.springer.com/gp/products/books/mycopy