Machine Learning for Microeconometrics

A. Colin Cameron
Univ. of California- Davis

Abstract: These slides attempt to explain machine learning to empirical economists familiar with regression methods. The slides cover standard machine learning methods such as k-fold cross-validation, lasso, regression trees and random forests. The slides conclude with some recent econometrics research that incorporates machine learning methods in causal models estimated using observational data, specifically (1) IV with many instruments, (2) OLS in the partial linear model with many controls, and (3) ATE in heterogeneous effects model with many controls.

Presented at School of Economics, University of Sydney, and based on 2016 course ECN240F at Department of Economics, University of California - Davis.

April 12, 2017

Introduction

- Machine learning methods include **data-driven algorithms** to predict y given **x.**
  - there are **many** machine learning methods
  - the best methods vary with the particular data application
  - and guard against in-sample overfitting.
- The main goal is **prediction**
  - this is useful in some microeconometrics applications
  - e.g. predict one-year survival probability after hip transplant.

Introduction (continued)

- Current microeconometric applications instead focus on **causal estimation** of a key parameter, such as an average marginal effect, after controlling for confounding factors.
- Applications to date are to quasi-experimental methods (rather than fully structural models)
  - apply to models with selection on observables only
    - ★ good controls makes this assumption more reasonable
  - and to IV with available instruments
    - ★ good few instruments avoids many instruments problem.
- Machine learning methods determine good controls (or instruments)
  - but valid statistical inference needs to control for this data mining
  - currently active area of econometrics research.

- The machine learnings summary is based on the masters level book
  - ISL: Gareth James, Daniela Witten, Trevor Hastie and Robert Tibsharani (2013), *An Introduction to Statistical Learning: with Applications in R*, Springer.
  - A free legal pdf is at http://www-bcf.usc.edu/~gareth/ISL/
- Supplementary material is in the Ph.D. level book
  - ESL: Trevor Hastie, Robert Tibsharani and Jerome Friedman (2009), *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer.
  - A free legal pdf is at http://statweb.stanford.edu/~tibs/ElemStatLearn/index.html
- A recent book is
  - EH: Bradley Efron and Trevor Hastie (2016), *Computer Age Statistical Inference: Algorithms, Evidence and Data Science*, Cambridge University Press.
- My website has some material
  - http://cameron.econ.ucdavis.edu/e240f/machinelearning.html

- In this talk I first consider estimating prediction error
  - relevant regardless of how the prediction is obtained.
- Then I present various methods for obtaining predictions
  - some familiar such as OLS, k-nearest neighbors
  - others less familiar such as lasso, neural networks, regression trees, random forests.
- Finally I consider using these methods in causal microeconometric studies
  - IV with many instruments
  - OLS in partial linear model with many controls
  - ATE with heterogeneous effects and many controls.

Overview

1. Terminology
2. Estimating Prediction Error
    1. Cross-validation
    2. Goodness-of-fit measures
3. Regression (Supervised learning for continuous y)
    1. Subset selection of regressors
    2. Shrinkage methods: ridge, lasso, LAR
    3. Dimension reduction: PCA and partial LS
    4. High-dimensional data
4. Nonlinear models
    1. splines
    2. local regression
    3. neural networks
5. Regression trees
    1. Regression trees
    2. Bagging, random forests and boosting

Overview (continued)

6. Classification (categorical y): logit, k-nn, LDA, SVM
7. Unsupervised learning (no y): PCA, cluster analysis
8. Causal inference with machine learning
    1. IV estimation with many instruments
    2. Partial linear model with many controls
    3. ATE with heterogeneous effects and many controls
9. Big data
10. Some R commands for machine learning

1. Terminology

- Topic is called machine learning or statistical learning or data learning or data analytics where data may be big or small.
- **Supervised learning = Regression**
  - We have both outcome y and regressors $\mathbf{x}$
  - 1. **Regression**: y is continuous
  - 2. **Classification**: y is categorical
- **Unsupervised learning**
  - We have no outcome y - only several $\mathbf{x}$
  - 3. **Cluster Analysis**: e.g. determine five types of individuals given many psychometric measures.
- These slides
  - focus on 1.
  - briefly mention 2.
  - even more briefly mention 3.

Terminology (continued)

- Consider two types of data sets
  - 1. **training data set** (or **estimation sample)**
    - ★ used to fit a model
  - 2. **test data set** (or **hold-out sample** or **validation set**)
    - ★ additional data used to determine how good is the model fit
    - ★ a test observation $(\mathbf{x}_0, y_0)$ is a previously unseen observation.

2. Estimating Prediction Error

- We wish to predict y given $\mathbf{x} = (x_1, ..., x_p)$.
- A **training data set** d yields prediction rule $\widehat{f}(\mathbf{x})$
  - we predict y at point $\mathbf{x}_0$ using $\widehat{y} = \widehat{f}(\mathbf{x})$.
  - e.g. for OLS $\widehat{y} = \mathbf{x}_0(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$.
- Consider **squared error loss** $(y - \widehat{y})^2$
  - some methods adapt to other loss functions
- We wish to estimate the **true prediction error**
  - $\text{Err}_d = E_F[(y_0 - \widehat{y}_0)^2]$
  - for **test data set** point $(\mathbf{x}_0, y_0) \sim F$.
- This is **under-estimated** by the estimation sample **apparent error**
  - $\text{err} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ or mean squared error (MSE)
  - e.g. $E[\text{err}] = \frac{n}{n-k}\sigma^2 > \sigma^2$ in the classical linear regression model.

Two ways to estimate prediction error

- 1. **Cross validation**
  - e.g. k-fold cross validation, leave-one-out cross validation
  - use both a training set (estimation) and a validation set (evaluate prediction)
  - a nonparametric method
  - can be applied to a wide range of settings and loss functions
  - computationally more expensive.
- 2. **Penalty measures**
  - Akaike's information criterion (AIC), Mallows CP, $\overline{R}^2$
  - use only the training set
  - so estimation can use all the available data
  - a more parametric method as a model is specified
  - the penalty (for overfitting) varies with the setting and loss function
  - computationally quicker.

Single-split Validation

- Randomly **divide available data into two parts**
  - 1. model is fit on training set
  - 2. MSE is computed for predictions in validation set.
- Example: choose degree k of a polynomial in $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k + \varepsilon$.
  - for each degree $k = 0, ..., p$ estimate on the training set to get $\widehat{\beta}'s$, predict on the validation set to get $\widehat{y}_k's$ and MSE_k
  - 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. Results 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 nonparametric regression such as k-nearest neighbors, kernel and local linear regression but not for global regression.

k-fold Cross Validation

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

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

- 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.

One standard error rule for k-fold cross-validation:

- A variation where don't simply choose model with minimum $CV_{(K)}$
  - a further guard against overfitting.
- K folds gives K estimates $MSE_{(1)}, ..., MSE_{(K)}$
  - this yields standard error of $CV_{(k)}$

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

- 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.

Penalty measures

- A general result for regression model leading to prediction $\widehat{\mu}_i$ of $\mu_i = E[y_i]$ is that for observation i (see EH p.219)
  - $E[(y_i - \widehat{\mu}_i)^2] = E[\text{true prediction error}] - 2 \times Cov(\widehat{\mu}_i, y_i)$.
- So $(y_i - \widehat{\mu}_i)^2$ **underestimates** true prediction error
  - by more the greater the correlation between y_i and its prediction $\widehat{\mu}_i$.
- With structure can derive $Cov(\widehat{\mu}_i, y_i)$
  - For linear regression with i.i.d. errors and p regressors
    - ★ Mallows $C_p = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 + \frac{2}{n}\sigma^2 p$
    - ★ Using some other machine learning methods replace p with "effective degrees of freedom $p = \frac{1}{\sigma^2}\sum_{i=1}^{n} \widehat{Cov}(\widehat{\mu}_i, y_i)$.
- With normally distributed errors
  - AIC $= C_p +$ constant (Akaike's information criterion)
  - LOOCV asymptotically equivalent to AIC.

Aside: Penalty measures

- Consider
  - $y_i \sim (\mu_i, \sigma^2)$, $i = 1, ..., n$
  - μ_i varies with $\mathbf{x}_i$ but view $\mathbf{x}_i$ as fixed so suppress $\mathbf{x}_i$
  - stacked as $\mathbf{y} \sim (\boldsymbol{\mu}, \sigma^2 \mathbf{I})$.
- Regression gives prediction $\widehat{\boldsymbol{\mu}} = \mathbf{r}(\mathbf{y})$.
- Define
  - Prediction error: $\text{Err}_i = E[(y_{0i} - \widehat{\mu}_i)^2]$ for new observation y_{0i}
  - Apparent error: $\text{err}_i = (y_i - \widehat{\mu}_i)^2$
- Then taking expectations w.r.t. $\mathbf{y}$ and y_0
  - $E[\text{Err}_i] = E[\text{err}_i] + 2Cov(\widehat{\mu}_i, y_i)$
  - where $Cov(\widehat{\mu}_i, y_i) = E[(\widehat{\mu}_i - \mu_i)(y_i - \mu_i)$

Variance-bias trade-off

- Consider regression model

$$\begin{aligned} y &= f(\mathbf{x}) + \varepsilon \\ E[\varepsilon] &= 0 \text{ and } \varepsilon \text{ independent of } \mathbf{x} \end{aligned}$$

- For out-of-estimation-sample point $(y_0, \mathbf{x}_0)$ the 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 sum of variance and bias-squared**!
- Trade-off: more flexible models have less bias and more variance.
- And bias can be good if MSE is our goal
  - e.g. shrinkage estimators
  - e.g. In model $y_i \sim N(\mu_i, 1)$ the MLE is $\widehat{\mu}_i = y_i$, $i = 1, ..., n$
    - ★ better is the James-Stein estimator $\widetilde{\mu}_i = \overline{y} + b(y_i - \overline{y})$ where $b = 1 - \frac{1}{n-3}\sum_{i=1}^{n}(y_i - \overline{y})^2$ and $n \geq 4$.

Stata Example

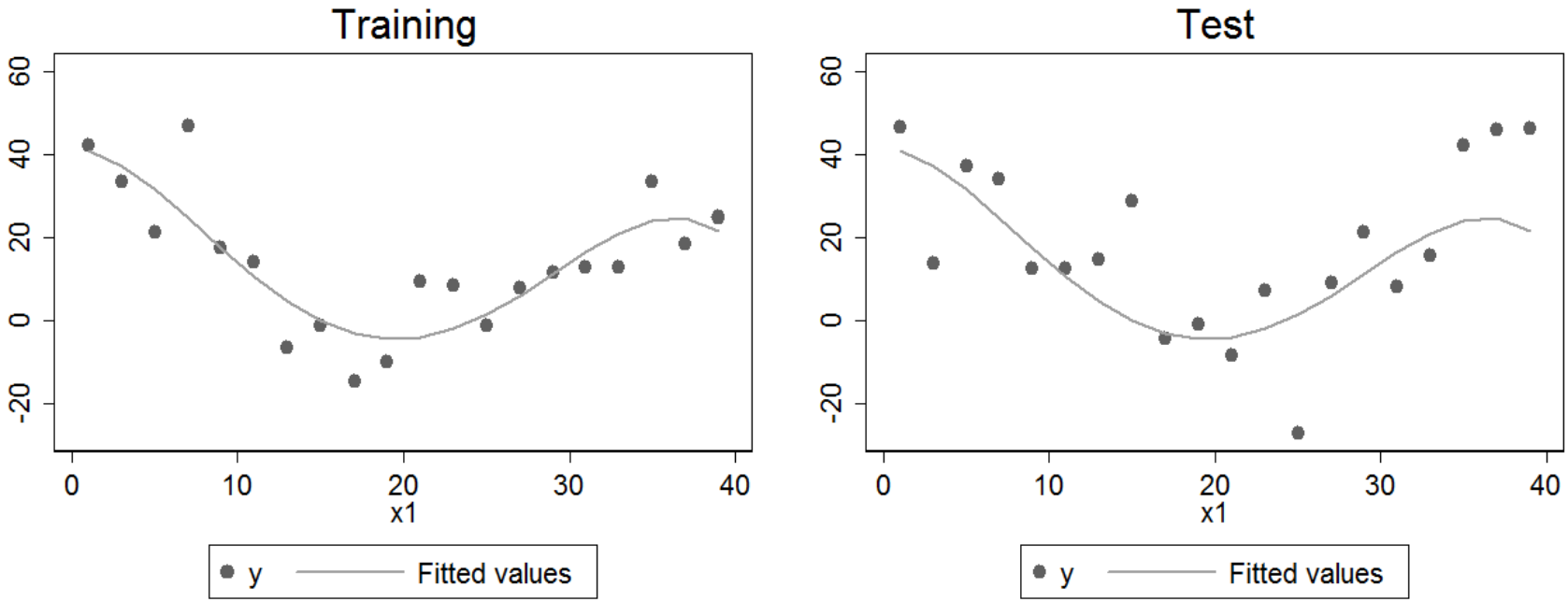
- D.g.p. is quadratic with $n = 40$. Fit OLS polynomial of degree 4.

```
. * Generate data - y = 42 - 4*x + 0.1*x^2 + e
. qui set obs 40

. set seed 10101

. gen x1 = _n - mod(_n+1,2)  // x1 = 1 1 3 3 5 5 .... 39 39

. gen x2 = x1^2

. gen x3 = x1^3

. gen x4 = x1^4

. gen dtrain = mod(_n,2)==1  // dtrain = 1 0 1 0 .... 1 0

. gen y = 2 + 0.1*(x1-20)^2 + rnormal(0,10)

. reg y x1-x4, noheader
```

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
x1	.4540487	3.347179	0.14	0.893	-6.341085	7.249183
x2	-.437711	.3399652	-1.29	0.206	-1.127877	.2524551
x3	.020571	.0127659	1.61	0.116	-.0053452	.0464871
x4	-.0002477	.0001584	-1.56	0.127	-.0005692	.0000738
_cons	37.91263	9.619719	3.94	0.000	18.38357	57.4417

Predictions in training and test data sets

- Left panel: Training data scatterplot and fitted curve.
- Right panel: Test data scatter plot (different y, same x) and predictions.
- Clearly predicts much worse in test data set.



Split-sample validation

- Test MSE lowest for quadratic (Training MSE lowest for quartic).

```
. * Split sample validation - training and test MSE for polynomials up to deg 4
. forvalues k = 1/4 {
  2.    qui reg y x1-x`k' if dtrain==1
  3.    qui predict y`k'hat
  4.    qui gen y`k'errorsq = (y`k'hat - y)^2
  5.    qui sum y`k'errorsq if dtrain == 1
  6.    qui scalar mse`k'train = r(mean)
  7.    qui sum y`k'errorsq if dtrain == 0
  8.    qui scalar mse`k'test = r(mean)
  9. }

. di _n "MSE linear     Train = " mse1train "  Test = " mse1test _n ///
>    "MSE quadratic  Train = " mse2train "  Test = " mse2test _n ///
>    "MSE cubic      Train = " mse3train "  Test = " mse3test _n ///
>    "MSE quartic    Train = " mse4train "  Test = " mse4test _n

MSE linear     Train = 252.32258  Test = 412.98285
MSE quadratic  Train = 92.781786  Test = 184.43114
MSE cubic      Train = 87.577254  Test = 208.24569
MSE quartic    Train = 72.864095  Test = 207.78885
```

Five-fold cross validation for quartic on full sample

- Apply to OLS on quadratic regression with all 40 observations
  - Randomly form five folds, estimate on four, predict on fifth; repeat.
  - $CV_{(5)} = \frac{1}{5}(15.27994 + \cdots + 8.444316) = 12.39305.$

```
. * Five-fold cross validation example for quadratic
. set seed 10101

. crossfold regress y x1 x2
```

	RMSE
est1	15.27994
est2	16.77849
est3	11.15653
est4	10.30595
est5	8.444316

```
. * Compute five-fold cross validation measure - average of the above
. matrix RMSEs  = r(est)

. svmat RMSEs, names(rmses)

. sum rmses
```

Variable	Obs	Mean	Std. Dev.	Min	Max
rmses1	5	12.39305	3.501561	8.444316	16.77849

Five-fold cross validation for all models

- CV measure is lowest for quadratic

```
. * Five-fold cross validation measure for polynomials up to degree 4
. forvalues k = 1/4 {
  2.    qui set seed 10101
  3.    qui crossfold regress y x1-x`k'
  4.    qui matrix RMSEs`k' = r(est)
  5.    qui svmat RMSEs`k', names(rmses`k')
  6.    qui sum rmses`k'
  7.    qui scalar cv`k' = r(mean)
  8. }

. di _n "CV(5) for k = 1,..,4 = " cv1 ",  " cv2 ",  "cv3 ",  "cv4

CV(5) for k = 1,..,4 = 12.393046,  12.393046,  12.629339,  12.475117
```

AIC and BIC penalty measures for full-sample

- AIC $= -2 \ln L + 2p$, BIC $= -2 \ln L + p \ln N$
- Both favor quadratic.

```
. * Full sample estimates with AIC, BIC penalty - polynomials up to deg 4
.   forvalues k = 1/4 {
  2.    qui reg y x1-x`k'
  3.    qui scalar aic`k' = -2*e(ll) + 2*e(rank)
  4.    qui scalar bic`k' = -2*e(ll) + e(rank)*ln(e(N))
  5. }

. di _n "AIC for k = 1,..,4 = " aic1 ",  " aic2 ",  "aic3 ",  "aic4, ///
>    _n "BIC for k = 1,..,4 = " bic1 ",  " bic2 ",  "bic3 ",  "bic4

AIC for k = 1,..,4 = 348.99841,  314.26217,  316.01317,  315.3112
BIC for k = 1,..,4 = 352.37617,  319.32881,  322.76869,  323.7556
```

3. Regression Methods

1. Terminology
2. Estimating Prediction Error
3. **Regression (Supervised learning for continuous y)**
4. Nonlinear models
5. Regression trees
6. Classification (categorical y): logit, k-nn, LDA, SVM
7. Unsupervised learning (no y): PCA, cluster analysis
8. Causal inference with machine learning
9. Big data
10. Some R commands for machine learning

3. Regression Methods

- Consider linear regression model with p potential regressors where p is too large.
- Methods that **reduce the model complexity** are
  - choose a subset of regressors
  - shrink regression coefficients towards zero
    - ridge, lasso, LAR
  - reduce the dimension of the regressors
    - principal components analysis.
- Linear regression may predict well if include interactions and powers as potential regressors.

Subset Selection of Regressors

- General idea is for each model size choose best model and then chose between the different model sizes.
- So
  - 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.

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 for best subsets.
- 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.

Shrinkage Methods

- Shrinkage estimators minimize RSS (residual sum of squares) with a penalty for model size
  - this shrinks parameter estimates towards zero.
- The extent of shrinkage is determined by a **tuning parameter**
  - this is determined by cross-validation or e.g. AIC.
- 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 all towards zero
- Lasso shrinks some a bit towards 0 and sets others $= 0$

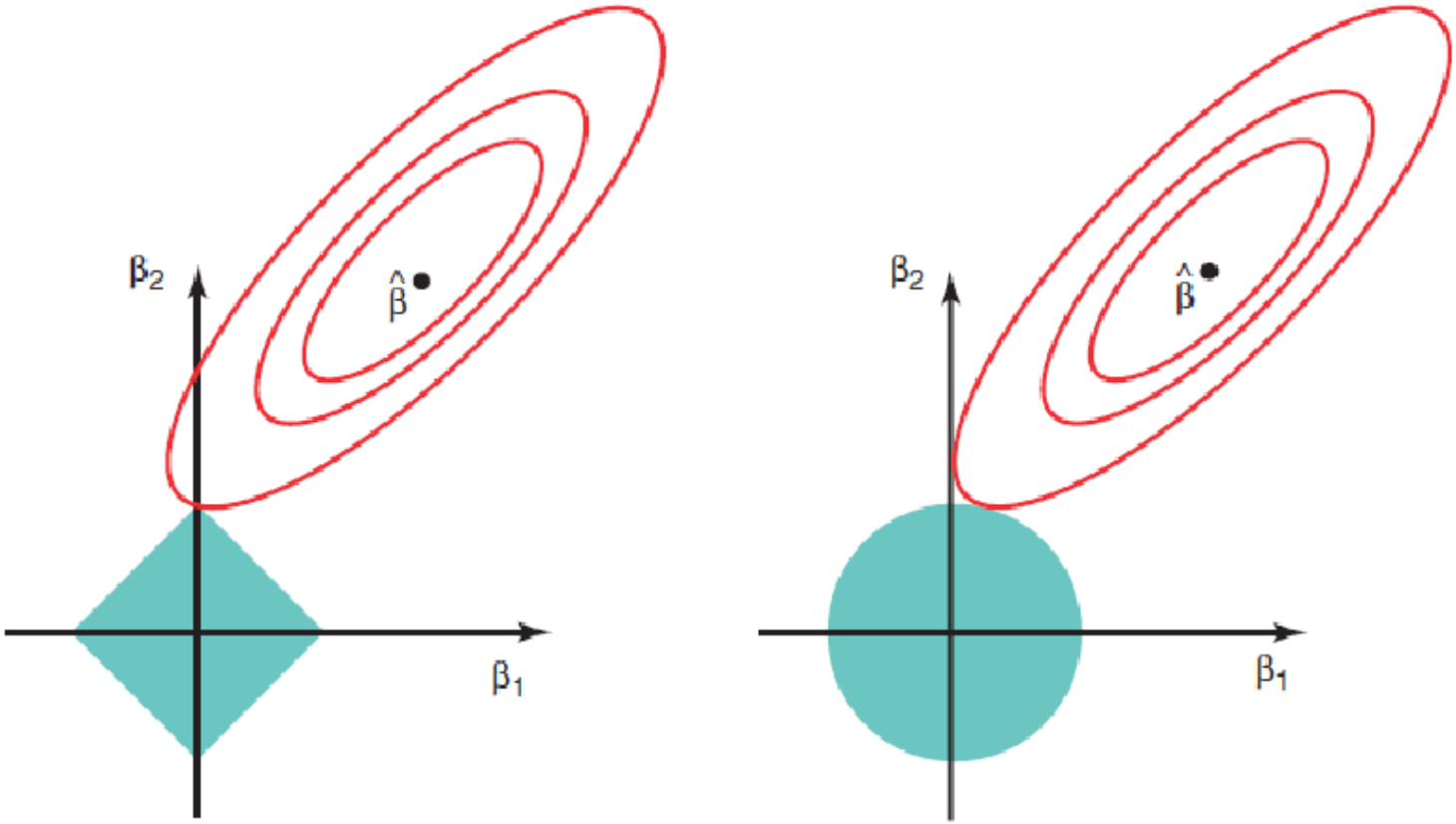
$$\widehat{\beta}_j^{L} = \begin{cases} y_j - \lambda/2 & \text{if } y_j > \lambda/2 \\ y_j + \lambda/2 & \text{if } y_j < -\lambda/2 \\ 0 & \text{if } |y_j| > \lambda/2 \end{cases}$$

- 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



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.
- Forward-stagewise algorithm for LAR proceeds as follows
  - Choose a small ε.
  - 1. Start with initial residual $\mathbf{r} = \mathbf{y}$, and $\beta_1 = \beta_2 = \cdots = \beta_p = 0$.
  - 2. Find the predictor $\mathbf{z}_j$ $(j = 1, ..., p)$ most correlated with $\mathbf{r}$
  - 3. Update $\beta_j = \beta_j + \delta_j$, where $\delta_j = \varepsilon \times sign(\mathbf{z}_j'\mathbf{r})$.
  - 4. Set $\mathbf{r} = \mathbf{r} - \delta_j \mathbf{z}_j$, and repeat Steps 2 and 3 many times.
- Choose the step with minimum Mallows CP.

Lasso extensions

- Can weight each β differently
  - Belloni, Chernozhoukov et al. do this.
- The group lasso allows to include regressors as groups (e.g. race dummies as a group)
  - with L groups minimize over $\boldsymbol{\beta}$

$$\sum_{i=1}^{n}\left(y_i-\sum_{l=1}^{L}\mathbf{x}_i'\boldsymbol{\beta}_l\right)^2+\lambda\sum_{l=1}^{L}\sqrt{\rho_l}\left(\sum_{j=1}^{p_l}|\beta_{lj}|\right).$$

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 $\Lambda = \text{Diag}[\lambda_j]$ 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 $\mathbf{X}'\mathbf{X}\mathbf{H} = \Lambda\mathbf{H}$ 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.

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. MSE or R^2.

Stata Example: best subset selection using penalty

- Section 2 quadratic d.g.p. $n = 40$ and polynomials to degree 4.
- A model with three regressors has lowest $\overline{R}^2$, AIC and Mallows CP
  - With regressors x^3, x^4 and x^2 that minimize MSE with three regressors.

```
. * Subset selection with add-on vselect
. vselect y x1 x2 x3 x4, best

Response :              y
Selected predictors:    x3 x4 x2 x1

Optimal models:

   # Preds     R2ADJ          C        AIC       AICC        BIC
         1   .0724281   49.77402   345.1659   345.8326   348.5437
         2   .5815134   3.679737   314.2622    315.405   319.3288
         3   .6002677   3.018401   313.3322   315.0969   320.0877
         4   .5890628          5   315.3112   317.8567   323.7556

predictors for each model:

1   :   x4
2   :   x2 x1
3   :   x3 x4 x2
4   :   x3 x4 x2 x1
```

Subset selection using penalty

- User add-on command `vselect` also allows forwards and backwards selection and specifying regressor(s) that should always be included.
  - `* Stepwise forwards using AIC`
  - `vselect y x1 x2 x3 x4, forward aic`
  - `* Stepwise backwards using AIC`
  - `vselect y x1 x2 x3 x4, backward aic`
  - `* Best subsets with x1 always included`
  - `vselect y x2 x3 x4, fix(x1) best.`
- User add-in command `gvselect` can be used for other estimation commands, not just OLS
  - `* Add-on command gvselect for OLS regression with x1 always included`
  - `gvselect <xlist> x2 x3 x4:  regress y <xlist> x1`

Subset selection using statistical significance

- This uses Stata built-in command stepwise
  - Backward selection best model has three regressors x^2, x^3 and x^4.

```
. * Stepwise backward using statistical significance at five percent
. stepwise, pr(.05): regress y x1 x2 x3 x4
                      begin with full model
p = 0.8929 >= 0.0500  removing x1
```

Source	SS	df	MS		
				Number of obs	= 40
				F(3, 36)	= 20.52
Model	8274.11622	3	2758.03874	Prob > F	= 0.0000
Residual	4838.24796	36	134.395777	R-squared	= 0.6310
				Adj R-squared	= 0.6003
Total	13112.3642	39	336.214466	Root MSE	= 11.593

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
x4	-.0002289	.0000764	-3.00	0.005	-.0003839	-.000074
x2	-.3929775	.0815041	-4.82	0.000	-.5582754	-.2276796
x3	.0189766	.0049143	3.86	0.000	.0090099	.0289433
_cons	39.05448	4.592698	8.50	0.000	29.74006	48.36891

Subset selection using statistical significance (continued)

- Forward selection in order x, x^2, x^3 and x^4 chooses no regressors
  - as regress y x1 leads to $t = -0.45$ for regressor x!

```
. * Stepwise forward in specified order (hierarchical) testing at five percent
. stepwise, pe(.05) hierarchical: regress y x1 x2 x3 x4
                      begin with empty model
p = 0.6543 >= 0.0500  testing x1
p >= 0.0500           for all terms in model
```

Source	SS	df	MS		
Model	0	0	.	Prob > F	.
Residual	13112.3642	39	336.214466	R-squared	0.0000
				Adj R-squared	0.0000
Total	13112.3642	39	336.214466	Root MSE	18.336

	Number of obs	=	40
	F(0, 39)	=	0.00

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
_cons	15.86777	2.8992	5.47	0.000	10.00359	21.73196

Standardize variables

- Standardize regressors to $(0, 1)$ and demean dependent variable.

```
. * Standardize variables
. foreach var of varlist x1 x2 x3 x4 {
  2.     qui egen z`var' = std(`var')
  3.     }

. qui sum y

. qui gen ydemeaned = y - r(mean)
```

Lasso computed using Lasso shooting algorithm

- Stata user-written command `lassoshooting.ado` due to Christian Hansen
  - uses the coordinate descent (called lasso shooting) algorithm of Fu (1998) Penalized Regressions: The Bridge Versus the Lasso. Journal of Computational and Graphical Statistics, 7:397-416
  - this converts a nonsmooth convex problem to a smooth convex problem by reducing to a sequence of one-dimensional optimizations made smooth by introducing an additional parameter.
- The command has an algorithm for initial computation of lambda
  - then vary this to get a reasonable number of variables selected
  - CV is not used and output does not give MSE.
  - better to use R function cv.glmnet(,alpha=1) in glmnet library

Compute lambda

- Formula gives $\lambda = 38.6$

```
. * Compute lamda for lassoShooting command
. qui reg y x1 x2 x3 x4

. scalar nobs = e(N)

. scalar c = 1.1

. scalar p = e(rank)    // number of variables

. scalar gamma = 0.1/ln(nobs)

. scalar ke = 1

. di "lambda from theory: " 2*c*sqrt(nobs)*invnormal(1 - (gamma/(2*ke*p)))
lambda from theory: 38.692789

. di "alternative:" 2*c*sqrt(nobs)*invnormal(1 - (0.05/(2*ke*p)))
alternative:35.840145

. di "alternative:" 2*c*sqrt(2*nobs*log(2*p)/gamma)
alternative:181.35217
```

Lasso computed using Lasso shooting algorithm

- Result below for $\lambda = 30$ chooses x^4 (regressor named $zx4$)
  - No regressors chosen when $\lambda = 38.6$
  - $x1$ and x^4 chosen when $\lambda = 20$

```
. * LASSO using lassoShooting.do
. * Verbose 0, 1, 2 gives increasing output as does fdisplay 0, 1
. lassoShooting ydemeaned zx1 zx2 zx3 zx4, ///
>   lambda(30) lasiter(100) verbose(1) fdisplay(1)
Number of iterations: 6
Total Shoots: 24
Number of iterations: 2
Total Shoots: 8
```

Selected	LASSO	Post-LASSO
zx4	.05425149	5.6875173

```
. * Lasso is penalized coefficient and post-Lasso is OLS coefficient
. reg ydemeaned zx4, noheader
```

ydemeaned	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
zx4	5.687517	2.827807	2.01	0.051	-.0370777	11.41211
_cons	-2.43e-07	2.792235	-0.00	1.000	-5.652585	5.652584

Lasso computed using LARS algorithm

- From output given in the next slides
  - first step has no regressors
  - second step includes x^4
  - third step additionally includes x (so x and x^4)
  - fourth step drops x^4 and adds x^3 (so x and x^3)
  - fifth step keeps the same variables but different coefficient values
  - etcetera up to eleven steps.
- At each step compute Mallows CP
  - fourth step has lowest CP so choose this (so x and x^3)
  - more precisely $zx1$ and $zx3$.

Lasso using LARS (continued)

- `lars ydemeaned zx1 zx2 zx3 zx4, a(lasso)`
  - yields coefficients at each of 11 steps

```
sbeta[11,4]
```

	c1	c2	c3	c4
r1	0	0	0	0
r2	0	0	0	14.544161
r3	-136.81263	0	0	151.35679
r4	-191.36341	0	201.34569	0
r5	-205.80797	0	215.79025	0
r6	-253.36327	110.71544	150.46381	0
r7	-231.17085	-1.421e-14	321.12102	-82.453021
r8	-260.03834	-1.421e-14	446.96952	-182.44786
r9	0	-1169.5549	2138.0951	-964.75152
r10	0	-1171.7384	2143.1645	-967.6894
r11	33.119025	-1318.5119	2353.4804	-1064.3873

Lasso using LARS (continued)

Cp, R-squared and Actions along the sequence of models

Step	Cp	R-square	Action
1	40.6356	0.0000	
2	37.7082	0.0627	+zz4
3	0.7066	0.5586	+zz1
4	-0.0881 *	0.5942	+zz3 -zz4
5	1.5425	0.5989	
6	3.1528	0.6038	+zz2
7	4.8322	0.6079	+zz4
8	6.4119	0.6133	
9	7.0180	0.6310	-zz1
10	9.0172	0.6310	
11	11.0000	0.6312	+zz1

* indicates the smallest value for Cp

The coefficient values for the minimum Cp

Variable	Coefficient
zz1	-30.6427
zz3	32.2411

Ridge Regression

```
. * Ridge regression with lambda = 0.01
. ridgereg ydemeaned xx1 xx2 xx3 xx4, kr(0.01) model(orr)

* (OLS) Ridge Regression - Ordinary Ridge Regression

  ydemeaned = xx1 + xx2 + xx3 + xx4

  Ridge k Value     =    0.01000     |   Ordinary Ridge Regression
------------------------------------------------------------------------------
  Sample Size       =         40
  Wald Test         =    43.2729     |   P-Value > Chi2(4)        =     0.0000
  F-Test            =    10.8182     |   P-Value > F(4 , 35)      =     0.0000
 (Buse 1973) R2     =     0.5876     |   Raw Moments R2           =     0.5876
 (Buse 1973) R2 Adj =     0.5405     |   Raw Moments R2 Adj       =     0.5405
  Root MSE (Sigma)  =    12.4293     |   Log Likelihood Function  =   -154.8891
------------------------------------------------------------------------------
- R2b= 0.5906   R2b Adj= 0.5438  F-Test =   12.62 P-Value > F(4 , 35)  0.0000
- R2v= 0.5098   R2v Adj= 0.4538  F-Test =    9.10 P-Value > F(4 , 35)  0.0000
------------------------------------------------------------------------------
   ydemeaned |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         xx1 |  -30.16199   41.33836    -0.73   0.470    -114.0833     53.75934
         xx2 |    5.23065   173.3994     0.03   0.976    -346.7888     357.2501
         xx3 |   15.73436   247.2998     0.06   0.950     -486.311     517.7797
         xx4 |   11.10298   115.2382     0.10   0.924     -222.843     245.0489
       _cons |  -2.52e-07   1.965243    -0.00   1.000    -3.989655     3.989654
------------------------------------------------------------------------------
```

Ridge Regression

- Coefficients are generally smaller than OLS
  - and in-sample RMSE is 11.754 for OLS and 12.4295 for ridge.

```
. * Versus OLS
. regress ydemeaned zx1 zx2 zx3 zx4
```

Source	SS	df	MS		
Model	8276.6599	4	2069.16497	Number of obs	= 40
Residual	4835.70386	35	138.162967	F(4, 35)	= 14.98
				Prob > F	= 0.0000
				R-squared	= 0.6312
				Adj R-squared	= 0.5891
Total	13112.3638	39	336.214455	Root MSE	= 11.754

ydemeaned	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
zx1	5.303288	39.09335	0.14	0.893	-74.06043	84.66701
zx2	-211.1309	163.9824	-1.29	0.206	-544.0328	121.7711
zx3	376.8585	233.8694	1.61	0.116	-97.92175	851.6387
zx4	-170.4384	108.9798	-1.56	0.127	-391.6791	50.80238
_cons	-2.39e-07	1.858514	-0.00	1.000	-3.772984	3.772984

1. Terminology
2. Estimating Prediction Error
3. Regression (Supervised learning for continuous y)
4. **Nonlinear models**
5. Regression trees
6. Classification (categorical y): logit, k-nn, LDA, SVM
7. Unsupervised learning (no y): PCA, cluster analysis
8. Causal inference with machine learning
9. Big data
10. Some R commands for machine learning

4. Nonlinear Models

- Basis function models
  - 1. polynomial regression
  - 2. step functions
  - 3. regression splines
  - 4. smoothing splines
  - 5. wavelets
  - polynomial is global while the others break range of x into pieces.
- Other methods
  - local polynomial regression
  - generalized additive models
  - neural networks.

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)

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

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 first 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 $\sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2$ 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 $\mathbf{X}'s$.

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

- So $f(\mathbf{x}_i) = \sum_{m=1}^{M} g(\alpha_{0m} + \mathbf{x}_i'\alpha_m)$ where $g(v) = 1/(1+e^{-v})$.
- 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.

5. Regression Trees

1. Terminology
2. Estimating Prediction Error
3. Regression (Supervised learning for continuous y)
4. Nonlinear models
5. **Regression trees**
6. Classification (categorical y): logit, k-nn, LDA, SVM
7. Unsupervised learning (no y): PCA, cluster analysis
8. Causal inference with machine learning
9. Big data
10. Some R commands for machine learning

5. Regression Trees: Overview

- Regression Trees sequentially split regressors **x** into regions that best predict y
  - e.g., first split is income $<$ or $>$ \$12,000
    and second split is on gender
    and third split is income $<$ or $>$ \$30,000 (if income $>$ \$12,000).
- Trees do not predict well
  - due to high variance
  - e.g. split data in two then can get quite different trees
  - e.g. first split determines future splits.
- Better methods are then given next
  - bagging (bootstrap averaging) computes regression trees for different samples obtained by bootstrap and averages the predictions.
  - random forests use only a subset of the predictors in each bootstrap sample
  - boosting grows trees based on residuals from previous stage
  - bagging and boosting are general methods (not just for trees).

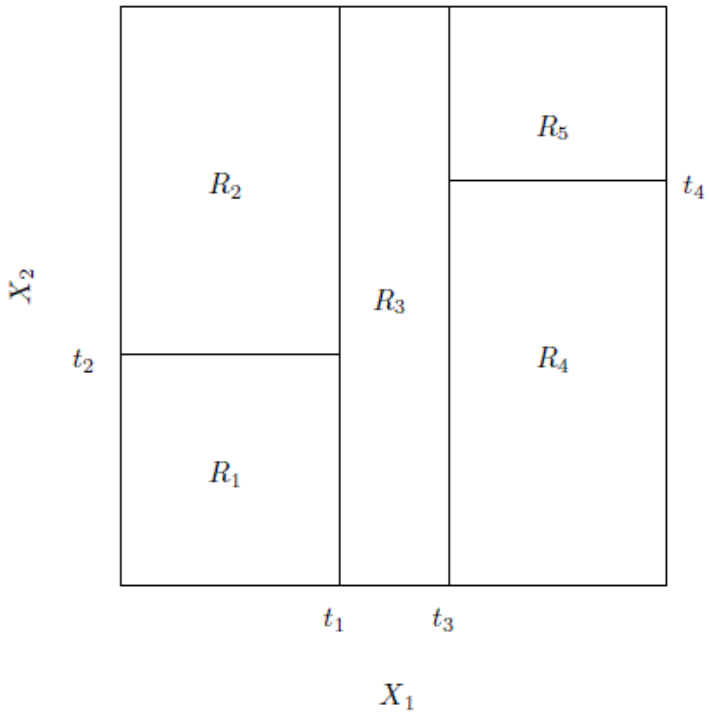
Regression Trees

- 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 split point 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 methods given so far
  - 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 $\widehat{f}(x) = \sum_{b=1}^{B} \lambda \widehat{f}^b(x_i)$.

6. Classification

1. Terminology
2. Estimating Prediction Error
3. Regression (Supervised learning for continuous y)
4. Nonlinear models
5. Regression trees
6. **Classification (categorical y): logit, k-nn, LDA, SVM**
7. Unsupervised learning (no y): PCA, cluster analysis
8. Causal inference with machine learning
9. Big data
10. Some R commands for machine learning

6. Classification Methods

- $y's$ are now categorical (e.g. binary if two categories).
- Use (0,1) loss function
  - 0 if correct classification and 1 if misclassified.
- Regression methods predict probabilities and then use Bayes classifier.
  - logistic regression, multinomial regression, k nearest neighbors
  - assign to class with the highest predicted probability
  - in binary case $\widehat{y} = 1$ if $\widehat{p} \geq 0.5$ and $\widehat{y} = 0$ if $\widehat{p} < 0.5$.
- Discriminant analysis additionally assumes a normal distribution for the x's
  - use Bayes theorem to get $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$.
- Support vector classifiers and support vector machines use separating hyperplanes of X and extensions.

A Different 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 $\widehat{y} = 1$

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 $N(\boldsymbol{\mu}_k, \Sigma)$ 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**

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

  - use $\widehat{\boldsymbol{\mu}}_k = \bar{\mathbf{x}}_k$, $\widehat{\Sigma} = \widehat{\text{Var}}[\mathbf{x}_k]$ 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 $N(\boldsymbol{\mu}_k, \Sigma_k)$
- Upon simplification, the Bayes classifier assigns observation $\mathbf{X} = \mathbf{x}$ to class k which has largest

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

  - 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

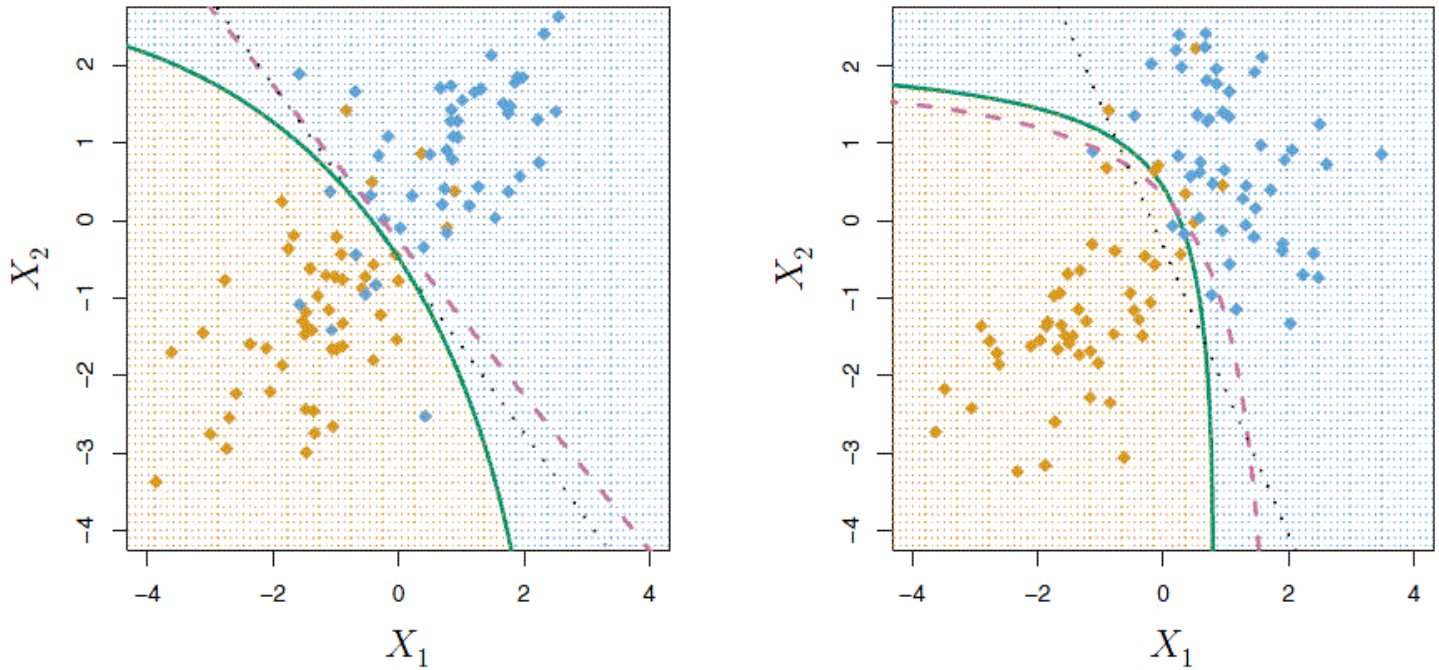


FIGURE 4.9. Left: *The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with* $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$. *The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated 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.

7. Unsupervised Learning

1. Terminology
2. Estimating Prediction Error
3. Regression (Supervised learning for continuous y)
4. Nonlinear models
5. Regression trees
6. Classification (categorical y): logit, k-nn, LDA, SVM
7. **Unsupervised learning (no y): PCA, cluster analysis**
8. Causal inference with machine learning
9. Big data
10. Some R commands for machine learning

7. 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 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.

8. Causal Inference with Machine Learning

1. Terminology
2. Estimating Prediction Error
3. Regression (Supervised learning for continuous y)
4. Nonlinear models
5. Regression trees
6. Classification (categorical y): logit, k-nn, LDA, SVM
7. Unsupervised learning (no y): PCA, cluster analysis
8. **Causal inference with machine learning**
9. Big data
10. Some R commands for machine learning

8. Causal Inference with Machine Learning

- Current microeconometric applications instead focus on **causal estimation** of a key parameter, such as an average marginal effect, after controlling for confounding factors
  - apply to models with selection on observables only
    - ★ good controls makes this assumption more reasonable
  - and to IV with available instruments
    - ★ good few instruments avoids many instruments problem.
- Machine learning methods determine good controls (or instruments)
  - but valid statistical inference needs to control for this data mining
  - currently active area of econometrics research.

IV with Many Instruments

- Instrumental variables model with **many valid instruments**
  - Model: $y_i = \mathbf{d}_i'\boldsymbol{\alpha} + \varepsilon_i$
  - Complication: At least one component of $\mathbf{d}_i$ is endogenous
  - Instruments: $\mathbf{x}_i$ that satisfy $E[\mathbf{d}_i|\mathbf{x}_i] = \mathbf{0}$.
- Many instruments can arise naturally or because we additionally consider powers and interactions of a smaller number of instruments
  - Let the p instruments be $\mathbf{f}_i = (f_{i1}, ..., f_{ip}) = (f_1(\mathbf{x}_i), ..., f(\mathbf{x}_{ip}))$.
- There is no problem in getting consistent IV estimates.
- In theory efficiency improves with more instruments
  - but **asymptotic theory works poorly** if include too many instruments.

- Belloni et al. (2012) propose using LASSO to pick just a few instruments
  - 2012 Econometrica paper is good for details
    - ★ A. Belloni, D. Chen, V. Chernozhukov and C. Hansen (2012), "Sparse Models and Methods for Optimal Instruments with an Application to Eminent Domain", Econometrica, Vol. 80, 2369-2429.
  - 2014 JEP paper is more accessible and has application
    - ★ A. Belloni, V. Chernozhukov and C. Hansen (2014), "High-dimensional methods and inference on structural and treatment effects," Journal of Economic Perspectives.
- Key assumption is that only s of p possible instruments are needed
  - an assumption of **sparsity**
  - use LASSO or other methods to choose the s instruments
  - then proceed with usual IV estimator and inference.

- For simplicity consider just one endogenous regressor
  - so $y_i = \mathbf{d}_i'\boldsymbol{\alpha} + \varepsilon_i = \alpha d_i + \mathbf{w}_i'\boldsymbol{\delta} + \varepsilon_i$.
  - then in i.i.d. case optimal instrument $D_i = E[d_i|\mathbf{x}_i]$
  - we model $d_i = \mathbf{f}_i'\boldsymbol{\beta} + v_i$
  - we do LASSO of d_i on $\mathbf{f}_i$ to give $\widehat{\boldsymbol{\beta}}$ (Lasso or post-lasso)
  - the instrument for d_i is $\widehat{d}_i = \mathbf{f}_i'\widehat{\boldsymbol{\beta}}$ and $\mathbf{w}_i$ as instrument for $\mathbf{w}_i$.
- Combining:
  - Model: $y_i = \mathbf{d}_i'\boldsymbol{\alpha} + \varepsilon_i = \alpha d_i + \mathbf{w}_i'\boldsymbol{\delta} + \varepsilon_i$
  - IV estimator: $\widehat{\boldsymbol{\alpha}} = (\sum_{i=1}^n \widehat{\mathbf{d}}_i'\mathbf{d}_i)^{-1}(\sum_{i=1}^n \widehat{\mathbf{d}}_i'\mathbf{y}_i)$
    - ★ where $\mathbf{d}_i = (d_i, \mathbf{w}_i)$ and $\widehat{\mathbf{d}}_i = (\widehat{d}_i, \mathbf{w}_i)$.
  - $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\alpha}}) = (\sum_{i=1}^n \widehat{\mathbf{d}}_i'\mathbf{d}_i)^{-1}(\sum_{i=1}^n \widehat{\varepsilon}_i^2\widehat{\mathbf{d}}_i'\widehat{\mathbf{d}}_i)(\sum_{i=1}^n \widehat{\mathbf{d}}_i'\mathbf{d}_i)^{-1}$.

- Further implementation details
  - we do LASSO of d_i on $\mathbf{f}_i$ to give $\widehat{\boldsymbol{\beta}} = \min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(d_i - \mathbf{f}_i'\boldsymbol{\beta})^2 + \frac{\lambda}{n}|Y_l(\boldsymbol{\beta})|$ where λ is the penalty level $\widehat{Y}_l(\boldsymbol{\beta}) = Diag(\widehat{\gamma}_1, ..., \widehat{\gamma}_p)$ are penalty loadings.
  - $\lambda = c2\sqrt{n}\Phi^{-1}(1 - \frac{\gamma}{2k_e p})$; $\gamma = 0.1/\ln(\max(p, n))$; c=1.1.
  - Ideally $\widehat{\gamma}_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n} f_{ij}^2 v_i^2}$, $j = 1, ..., p$.
  - Since don't know v_i^2 use initial conservative $\widehat{\gamma}_j$, use consequent $\widehat{v}_i^2$, and iterate.
  - Uses Stata add-on lassoShooting.ado.
- Further theory details on **approximate sparsity**
  - $E[d_i|\mathbf{x}_i] = \mathbf{f}_i'\boldsymbol{\beta} + a(\mathbf{x}_i)$
  - $\sum_{j=1}^{p} \mathbf{1}[\beta_j \neq 0] \leq s = o(n)$ # instruments grows at rate less than n
  - $a(\mathbf{x}_i) \leq O_p(\sqrt{s/n})$ approximation error at most of order $\sqrt{s/n}$ and hence $\leq o_p(1)|$

- Application: effect of endogenous court decisions on house prices
  - y_{ct} = home price index within court circuit c at time t
  - d_{ct} = # of takings appellate decisions that rule that a taking was unlawful
  - $y_{ct} = \alpha_c + \alpha_t + \alpha_c t + \beta d_{ct} + \mathbf{w}'_{ct}\delta + \varepsilon_{ct}$
  - Frisch-Waugh partial out fixed effects, time trends and $\mathbf{w}_{ct}$
  - $\widetilde{y}_{ct} = \alpha + \beta \widetilde{d}_{ct} + error$
  - find best instruments for $\widetilde{d}_{ct}$ using Lasso
  - $p = 183$ possible and find $s = 1$ (JEP) or $s = 2$ (Ecta)
  - the JEP instrument is the square of the number of panels with one or more members with JD from a public university.

Aside: Consistent Model Selection

- Consistent parameter estimation
  - means $\widehat{\theta}_j \xrightarrow{p} \theta_{j0}$ as $n \longrightarrow \infty$
  - no problem asymptotically if include x_j when irrelevant
  - since $\widehat{\theta}_j \xrightarrow{p} \theta_{j0} = 0$ if x_j irrelevant.
- Consistent model selection
  - means probability of choosing correct model $\longrightarrow 1$ as $n \longrightarrow \infty$
  - now we want to drop x_j if $\theta_{j0} = 0$
  - problem with testing at 5% as 5% of time include irrelevant regressors
  - solution 1: let test size$\longrightarrow 0$ as $n \longrightarrow \infty$
  - solution 2: use BIC (Bayesian information criteria).

Consistent model selection

- Biostatistics includes regressors (controls) if $p < 0.05$
  - may include irrelevant regressors and also leads to pre-test bias
- Economics instead uses economics theory and previous studies to include regressors regardless of their statistical significance,
- Data mining / machine learning chooses regressors from a much wider range of potential regressors
  - but need to make sure all relevant regressors are included
  - and that there is no pre-test bias
  - examples are given next.

Partial Linear Model with Many Controls

- Partial linear model with many controls
  - Model: $y_i = \alpha d_i + g(\mathbf{z}_i) + \varepsilon_i$
    - ★ where $E[\varepsilon_i | d_i, \mathbf{z}_i] = 0$
  - Estimator: OLS given knowledge of $g(\mathbf{z})$
  - Complication: $g(\mathbf{z})$ is unknown
    - ★ and $\mathbf{z}$ is high dimensional so can't use e.g. Robinson (1988) differencing estimator.
- Wrong Solution: have some method to pick subset of controls to include
  - will sometimes pick wrong controls (due to randomness)
    - ★ so OLS is inconsistent.

- Proposed solution: double selection
  - initial rich set of controls
    - ★ p potential controls $\mathbf{x}_i = (x_{i1}, ..., x_{ip}) = (f_1(\mathbf{z}_i), ..., f_p(\mathbf{z}_i))$
    - ★ e.g. series expansions
  - assume sparsity so that only $s < n$ need be included
  - use Lasso to choose controls that (1) predict y; and/or (2) predict d.
- Combining:
  - Structural model: $y_i = \alpha d_i + \mathbf{x}_i'\delta + \varepsilon_i$ and $d_i = \mathbf{x}_i'\delta + v_i$
  - Reduced form is
    - ★ (1) $y_i = \mathbf{x}_i'\boldsymbol{\beta} + u_i$
    - ★ (2) $d_i = \mathbf{x}_i'\delta + v_i$
  - Apply Lasso to (1) and (2) separately
  - Choose those $\mathbf{x}_i$ to be union of $\mathbf{x}'s$ from (1) and (2)
  - Do OLS of y on d and these $\mathbf{x}'s$.

- Application: Effect of abortion policy on crime (Donohue and Levitt)
  - y_{st} = crime rate (Violent, property or murder)
  - d_{st} = abortion rate for state i at time t ($n = 50$, $T = 12$)
  - $y_{st} = \alpha d_{st} + \mathbf{x}'_{st}\delta + \delta_s + \gamma_t + \varepsilon_{st}$
  - analyze first-differences with state FE's model
    $\Delta y_{st} = \alpha \Delta d_{st} + \Delta \mathbf{x}'_{st}\delta + \gamma_s + u_{st}$
    $\Delta d_{st} = \Delta \mathbf{x}'_{st}\boldsymbol{\beta} + \lambda_s + v_{st}$
  - $p = 284$ possible variables in $\mathbf{x}$ (see paper) and find $s = 10$
    - ★ lagged prisoners, lagged police$\times t$, initial income difference, initial income difference$\times t$, initial beer consumption difference$\times t$, average income, average income$\times t$, initial abortion rate.
  - similar $\widehat{\alpha}$ to Donohue-Levitt who have many more controls but standard errors one-third the size.

ATE with heterogeneous effects and many controls

- Max Farrell (2015), "Robust Estimation of Average Treatment Effect with Possibly more Covariates than Observations", Journal of Econometrics, 189. 1-23.
- Set up assumes unconfoundedness
  - Multivalued treatment $D \in \{0, 1, ..., J\}$
  - Generalized treatment score $p_j(x) = \Pr[D = j|X = x]$
  - Conditional outcome mean function $\mu_j(x) = E[Y|D = j, X = x]$
  - i.i.d. sample $\{y_i, d_i, x_i\}$, $i = 1, ..., N$
  - assume selection on observables.
- Method
  - fit $p_j(x)$ and $\mu_j(x)$ using group lasso applied to multinomial logit and least squares
  - refit $p_j(x)$ and $\mu_j(x)$ using penalized $\Pr[D = j|X = x]$
  - estimate $\mu_j = E[Y|D = j]$ by

    $$\widehat{\mu}_j = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1[d_i=j](y_i-\widehat{\mu}_j(x_i))}{\widehat{p}_j(x_i)} + \widehat{\mu}_j(x_i)\right].$$
  - simulation and apply to Dehejia-Wahba data.

LATE and ATE with many controls

- Belloni, Chernozhukov, Fernandez-Val and Hansen (2015), "Program Evaluation with High-Dimensional Data"
- Binary treatment and heterogeneous effects with endogenous treatment and valid instruments
  - use lasso along the way
  - allow for estimation of functions (such as local quantile treatment effects over a range of quantiles)
- Key is to use moment condition that allows inference to be unaffected by first-stage estimation
  - First stage: $\widehat{\boldsymbol{\alpha}}$ solves $\sum_{i=1}^{n} h(\mathbf{w}_i, \widehat{\boldsymbol{\alpha}}) = \mathbf{0}$
  - Second stage: $\widehat{\boldsymbol{\beta}}$ solves $\sum_{i=1}^{n} g(\mathbf{w}_i, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}) = \mathbf{0}$
    - ★ where $g(\cdot)$ chosen so that $E[\partial g(\mathbf{w}_i, \boldsymbol{\alpha}, \boldsymbol{\beta})/\partial \boldsymbol{\beta} = \mathbf{0}$.

ATE: Random Forests Example

- Random forests predict very well. Susan Athey big on this.
- Susan Athey and Stefan Wager (2017), "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests," JASA, forthcoming.
  - Standard binary treatment and heterogeneous effects with unconfoundness assumption
  - Use random forests to determine the controls
  - Proves asymptotic normality and gives point-wise confidence intervals
    - ★ This is a big theoretical contribution.

Susan Athey

- Susan Athey's website has several papers on machine learning in economics.
- Susan Athey (2017), "Beyond Prediction: Using Big Data for Policy Problems," Science 355, 483-485.
  - Off-the shelf prediction methods assume a stable environment
    - ★ includes Kleinberg et al (2015) AER hip replacement
  - This article considers causal prediction by
    - ★ adjust for confounders e.g. Belloni et al., Athey et al.
    - ★ designed experiments e.g. Blake et al.
    - ★ excellent references to the latest work.

9. Big Data

1. Terminology
2. Estimating Prediction Error
3. Regression (Supervised learning for continuous y)
4. Nonlinear models
5. Regression trees
6. Classification (categorical y): logit, k-nn, LDA, SVM
7. Unsupervised learning (no y): PCA, cluster analysis
8. Causal inference with machine learning
9. **Big data**
10. Some R commands for machine learning

9. Big Data

- Hal Varian (2014), "Big Data: New Tricks for Econometrics", JEP, Spring, 3-28.
- Tools for handling big data
  - file system for files split into large blocks across computers
    - ★ Google file system (Google), Hadoop file system
  - database management system to handle large amounts of data across many computers
    - ★ Bigtable (Google), Cassandra
  - accessing and manipulating big data sets across many computers
    - ★ MapReduce (Google), Hadoop.
  - language for Mapreduce / Hadoop
    - ★ Sawzall (Google), Pig
  - Computer language for parallel processing
    - ★ Go (Google - open source)
  - simplified structured query language (SQL) for data enquiries
    - ★ Dremel, Big Query (Google), Hive, Drill, Impala.

- Methods
  - article discusses k-fold CV, trees, lasso, ...
  - small discussion of causality and prediction
  - (note that a classic fail is Google flu trends)
  - for references mentions ESL and ISL.

10. Some R Commands used in ISL

- Basic regression
  - OLS is lm.fit
  - cross-Validation for OLS uses cv.glm()
  - bootstrap uses boot() function in boot library
- Basic classification
  - logistic: glm function
  - discriminant analysis: lda() and qda functions in MASS library
  - k nearest neighbors: knn() function in class library
- Variable selection
  - best subset, forward stepwise and backward stepwise: regsubsets() in leaps library
- Penalized regression
  - ridge regression: glmnet(,alpha=0) function in glmnet library
  - lasso: glmnet(,alpha=1) function in glmnet library
  - CV to get lambda for ridge/lasso: cv.glmnet() in glmnet library

Some R Commands (continued)

- Dimension reduction
  - principal components: pcr() function in pls library
  - CV for PCA: pcr(,validation="CV")
  - partial least squares: plsr() function in pls library
- Splines
  - 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
- Local regression
  - loess: function loess
  - generalized additive models: function gam() in gam library

Some R Commands (continued)

- 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
- 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.
- Unsupervised Learning
  - principal components analysis: function prcomp()
  - k-means clustering: function kmeans()
  - hierarchical clustering: function hclust()

11. 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
- Interesting book: Cathy O'Neil, Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy.