# Machine Learning: Overview
# Part 1: Basics - selection, shrinkage, dimension reduction

A. Colin Cameron
U.C.-Davis

.

presented at CINCH Academy 2019
The Essen Summer School in Health Economics
and at
Friedrich Alexander University, Erlangen-Nurnberg

April 2019

## Introduction

- Machine learning methods include **data-driven algorithms** to predict $y$ given $\mathbf{x}$.
  - there are **many** machine learning (ML) methods
  - the best ML methods vary with the particular data application
  - and guard against in-sample overfitting.
- The main goal of the machine learning literature is **prediction** of $y$ and not estimation of $\beta$.
- For economics prediction of $y$ is sometimes a goal
  - e.g. do not provide hip transplant to individuals with low predicted one-year survival probability
  - then main issue is what is the best standard ML method.
- But often economics is interested in estimation of $\beta$ or estimation of a partial effect such as average treatment effects
  - then new methods using ML are being developed by econometricians.

## Econometrics for Machine Learning

- The separate slides on ML Methods in Economics consider the following microeconomics examples.
- Treatment effects under an unconfoundedness assumption
  - Estimate $\beta$ in the model $y = \beta x_1 + g(\mathbf{x}_2) + u$
  - the assumption that $x_1$ is exogenous is more plausible with better $g(\mathbf{x}_2)$
  - machine learning methods can lead to a good choice of $g(\mathbf{x}_2)$.
- Treatment effects under endogeneity using instrumental variables
  - Now $x_1$ is endogenous in the model $y = \beta_1 x_1 + g(\mathbf{x}_2) + u$
  - given instruments $\mathbf{x}_3$ and $\mathbf{x}_2$ there is a potential many instruments problem
  - machine learning methods can lead to a good choice of instruments.
- Average treatment effects for heterogeneous treatment
  - ML methods may lead to better regression imputation and better propensity score matching.

## Econometrics for Machine Learning (continued)

- ML methods involve data mining
  - using traditional methods data mining leads to the complications of pre-test bias and multiple testing.
- In the preceding econometrics examples, the ML methods are used in such a way that these complications do not arise
  - an asymptotic distribution for the estimates of $\beta_1$ or ATE is obtained
  - furthermore this is Gaussian.
- These methods can be viewed as semiparametric methods
  - without the curse of dimensionality!
- However, the underlying theory relies on difficult to understand and evaluate assumptions
  - such as "sparsity" - that few of the potential variables matter.
- Whether these assumptions are reasonable in practice is an open question.

- The course is broken into three sets of slides.
- **Part 1: Basics**
  - variable selection, shrinkage and dimension reduction
  - focuses on linear regression model but generalizes.
- Part 2: Flexible methods
  - nonparametric and semiparametric regression
  - flexible models including splines, generalized additive models, neural networks
  - regression trees, random forests, bagging, boosting
  - classification (categorical $y$) and unsupervised learning (no $y$).
- Part 3: Microeconometrics
  - OLS with many controls, IV with many instruments, ATE with heterogeneous effects and many controls.
- Parts 1 and 2 are based on the two books given in the references
  - *Introduction to Statistical Learning*
  - *Elements of Statistical Learning.*
- While most ML code is in R, these slides use Stata.

## Overview

1. Terminology
2. Model Selection
   1. Forwards selection, backwards selection and best subsets
   2. Goodness-of-fit measures
   3. Penalized goodness-of-fit measures
   4. Cross-validation
3. Shrinkage methods
   1. Variance-bias trade-off
   2. Ridge regression, LASSO, elastic net
4. Dimension reduction
   1. Principal components
   2. Partial LS
5. High-dimensional data

## 1. Terminology

- The 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 (or **features**) $\mathbf{x}$
  - 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.

## 2.1 Model selection: Forwards, backwards and best subsets

- Forwards selection (or specific to general)
  - start with simplest model (intercept-only) and in turn include the variable that is most statistically significant or most improves fit.
  - requires up to $p + (p-1) + \cdots + 1 = p(p+1)/2$ regressions where $p$ is number of regressors
- Backwards selection (or general to specific)
  - start with most general model and in drop the variable that is least statistically significant or least improves fit.
  - requires up to $p(p+1)/2$ regressions
- Best subsets
  - for $k = 1, ..., p$ find the best fitting model with $k$ regressors
  - in theory requires $\binom{p}{0} + \binom{p}{1} + \cdots + \binom{p}{p} = 2^p$ regressions
  - but leaps and bounds procedure makes this much quicker
  - $p < 40$ manageable though recent work suggests $p$ in thousands.
- Hybrid
  - forward selection but after new model found drop variables that do not improve fit.

## Stata Example

- These slides use Stata
  - most machine learning code is initially done in R.
- Generated data: $n = 40$
- Three correlated regressors.
  - $\begin{bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \end{bmatrix} \sim N\left( \begin{bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \end{bmatrix}, \quad \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix} \right)$
- But only $x_1$ determines $y$
  - $y = 2 + x_i + u_i$ where $u_i \sim N(0, 3^2)$.

## Fitted OLS regression

- As expected only $x_1$ is statistically significant at 5%
  - though due to randomness this is not guaranteed.

```
. * OLS regression of y on x1-x3
. regress y x1 x2 x3, vce(robust)

Linear regression                               Number of obs     =         40
                                                F(3, 36)          =       4.91
                                                Prob > F          =     0.0058
                                                R-squared         =     0.2373
                                                Root MSE          =     3.0907

------------------------------------------------------------------------------
             |               Robust
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   1.555582   .5006152     3.11   0.004     .5402873    2.570877
          x2 |   .4707111   .5251826     0.90   0.376    -.5944086    1.535831
          x3 |  -.0256025   .6009393    -0.04   0.966    -1.244364    1.193159
       _cons |   2.531396   .5377607     4.71   0.000     1.440766    3.622025
------------------------------------------------------------------------------
```

## 2.2 Selection using Statistical Significance

- **Not recommended** as pre-testing changes the distribution of $\widehat{\beta}$ but included for completeness
  - instead ML uses predictive ability.
- Stepwise forward based on $p < 0.05$
  - Stata add-on command `stepwise, pe(.05)`
  - chooses model with only intercept and $x_1$

```
. * Stepwise forward using statistical significance at five percent
. stepwise, pe(.05): regress y x1 x2 x3
                      begin with empty model
p = 0.0020 <  0.0500  adding  x1

      Source |       SS           df       MS      Number of obs   =        40
-------------+----------------------------------   F(1, 38)        =     11.01
       Model |  101.318018         1  101.318018   Prob > F        =    0.0020
    Residual |  349.556297        38  9.19884993   R-squared       =    0.2247
-------------+----------------------------------   Adj R-squared   =    0.2043
       Total |  450.874315        39  11.5608799   Root MSE        =     3.033

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   1.793535   .5404224     3.32   0.002     .6995073    2.887563
       _cons |   2.509313   .5123592     4.90   0.000     1.472097     3.54653
------------------------------------------------------------------------------
```

- Stepwise backward based on $p < 0.05$
  - Stata add-on command `stepwise, pr(.05)`
  - chooses model with only intercept and $x_1$

```
. * Stepwise backward using statistical significance at five percent
. stepwise, pr(.05): regress y x1 x2 x3
                      begin with full model
p = 0.9618 >= 0.0500  removing x3
p = 0.4410 >= 0.0500  removing x2

      Source |       SS           df       MS      Number of obs   =        40
-------------+----------------------------------   F(1, 38)        =     11.01
       Model |  101.318018         1  101.318018   Prob > F        =    0.0020
    Residual |  349.556297        38  9.19884993   R-squared       =    0.2247
-------------+----------------------------------   Adj R-squared   =    0.2043
       Total |  450.874315        39  11.5608799   Root MSE        =     3.033

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   1.793535   .5404224     3.32   0.002     .6995073    2.887563
       _cons |   2.509313   .5123592     4.90   0.000     1.472097     3.54653
------------------------------------------------------------------------------
```

- Option `hierarchical` allows selection in order of the specified regressors.

## 2.3 Goodness-of-fit measures

- 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 $\widehat{y}_0 = \widehat{f}(\mathbf{x}_0)$.
  - e.g. for OLS $\widehat{y}_0 = \mathbf{x}_0(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$.
- For regression consider **squared error loss** $(y - \widehat{y})^2$
  - some methods adapt to other loss functions
    - ★ e.g. absolute error loss and log-likelihood loss
  - and loss function for classification is $\mathbf{1}(y \neq \widehat{y})$.
- We wish to estimate the **true prediction error**
  - $\text{Err}_d = E_F[(y_0 - \widehat{y}_0)^2]$
  - for **test data set** point $(\mathbf{x}_0, y_0) \sim F$.

## Models overfit in sample

- We want to estimate the **true prediction error**
  - $E_F[(y_0 - \widehat{y}_0)^2]$ for **test data set** point $(\mathbf{x}_0, y_0) \sim F$.
- The obvious criterion is in-sample **mean squared error**
  - $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ where MSE = mean squared error.
- Problem: in-sample MSE **under-estimates** the true prediction error
  - Intuitively models "overfit" within sample.
- Example: suppose $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$
  - then $\widehat{\mathbf{u}} = (\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}_{OLS}) = (\mathbf{I} - \mathbf{M})\mathbf{u}$ where $\mathbf{M} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}$
    - ★ so $|\widehat{u}_i| < |u_i|$ (OLS residual is less than the true unknown error)
  - and use $\widehat{\sigma}^2 = s^2 = \frac{1}{n-k}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ and not $\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$
- Two solutions:
  - penalize for overfitting e.g. $\bar{R}^2$, AIC, BIC, Cp
  - use out-of-estimation sample prediction (cross-validation).

## 2.4 Penalized Goodness-of-fit Measures

- Two standard measures for general parametric model are
  - Akaike's information criterion
    - AIC$= -2\ln L + 2k$
  - BIC: Bayesian information criterion
    - BIC$= -2\ln L + (\ln n) \times k$
- Models with smaller AIC and BIC are preferred.
- AIC has a small penalty for larger model size
  - for nested models selects larger model if $-\Delta 2\ln L > 2\Delta k$
    - whereas $LR$ test$= -2\Delta\ln L$ of size $\alpha$ requires $-\Delta 2\ln L > \chi^2_\alpha(k)$.
- BIC has a larger penalty.

## AIC and BIC for OLS

- For classical regression with i.i.d. normal errors
  - $\ln L = -\frac{n}{2}\ln 2\pi - \frac{n}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^n(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2$
- Different programs then get different AIC and BIC.
- Econometricians use $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$ =MSE= $\frac{1}{n}\sum_{i=1}^n(y_i - \mathbf{x}_i'\widehat{\boldsymbol{\beta}})^2$
  - then AIC= $-\frac{n}{2}\ln 2\pi - \frac{n}{2}\ln\widehat{\sigma}^2 - \frac{n}{2} + 2k$.
- Machine learners use $\widehat{\boldsymbol{\beta}}$ and $\widetilde{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^n(y_i - \mathbf{x}_i'\widetilde{\boldsymbol{\beta}}_p)^2$
  - where $\widetilde{\boldsymbol{\beta}}_p$ is obtained from OLS in the largest model under consideration that has $p$ regressors including intercept
- Furthermore, constants such as $-\frac{n}{2}\ln 2\pi$ are often dropped.
- Also a finite sample correction is
  - AICC=AIC+$2(K+1)(K+2)/(N-K-2)$.

## More measures for OLS

- For OLS a standard measure is $\bar{R}^2$ (adjusted $R^2$)
  - $\bar{R}^2 = 1 - \frac{\frac{1}{n-k}\sum_{i=1}^n (y_i - \widehat{y}_i)^2}{\frac{1}{n-1}\sum_{i=1}^n (y_i - \bar{y})^2}$ (whereas $R^2 = 1 - \frac{\frac{1}{n-1}\sum_{i=1}^n (y_i - \widehat{y}_i)^2}{\frac{1}{n-1}\sum_{i=1}^n (y_i - \bar{y})^2}$)
  - $\bar{R}^2$ has a small penalty for model complexity
    - ★ $\bar{R}^2$ favors the larger nested model if the subset test $F > 1$.
- Machine learners also use Mallows $C_p$ measure
  - $C_p = (n \times \text{MSE}/\widetilde{\sigma}^2) - n + 2k$
    - ★ MSE$=\frac{1}{n}\sum_{i=1}^n (y_i - \mathbf{x}_i'\widehat{\boldsymbol{\beta}})^2$ and $\widetilde{\sigma}^2 = \frac{1}{N-p}\sum_{i=1}^n (y_i - \widetilde{y}_i)^2$
  - and some replace $p$ with "effective degrees of freedom" $p = \frac{1}{\sigma^2}\sum_{i=1}^n \widehat{Cov}(\widehat{\mu}_i, y_i)$.
- Note that for linear regression AIC, BIC, AICC and Cp are designed for models with homoskedastic errors.

## Example of penalty measures

- We will consider all 8 possible models based on $x_1$, $x_2$ and $x_3$.

```
. * Regressor lists for all possible models
. global xlist1

. global xlist2 x1

. global xlist3 x2

. global xlist4 x3

. global xlist5 x1 x2

. global xlist6 x2 x3

. global xlist7 x1 x3

. global xlist8 x1 x2 x3

.
. * Full sample estimates with AIC, BIC, Cp, R2adj penalties
. quietly regress y $xlist8

. scalar s2full = e(rmse)^2  // Needed for Mallows Cp
```

## Example of penalty measures (continued)

- Manually get various measures. All (but MSE) favor model with just $x_1$.

```
.   forvalues k = 1/8 {
  2.   quietly regress y ${xlist`k'}
  3.   scalar mse`k' = e(rss)/e(N)
  4.   scalar r2adj`k' = e(r2_a)
  5.   scalar aic`k' = -2*e(ll) + 2*e(rank)
  6.   scalar bic`k' = -2*e(ll) + e(rank)*ln(e(N))
  7.   scalar cp`k' =  e(rss)/s2full - e(N) + 2*e(rank)
  8.   display "Model " "${xlist`k'}" _col(15) " MSE=" %6.3f mse`k'  ///
>     " R2adj=" %6.3f r2adj`k' "  AIC=" %7.2f aic`k'  ///
>     " BIC=" %7.2f bic`k' " Cp=" %6.3f cp`k'
  9. }
Model          MSE=11.272 R2adj= 0.000  AIC= 212.41 BIC= 214.10 Cp= 9.199
Model x1       MSE= 8.739 R2adj= 0.204  AIC= 204.23 BIC= 207.60 Cp= 0.593
Model x2       MSE= 9.992 R2adj= 0.090  AIC= 209.58 BIC= 212.96 Cp= 5.838
Model x3       MSE=10.800 R2adj= 0.017  AIC= 212.70 BIC= 216.08 Cp= 9.224
Model x1 x2    MSE= 8.598 R2adj= 0.196  AIC= 205.58 BIC= 210.64 Cp= 2.002
Model x2 x3    MSE= 9.842 R2adj= 0.080  AIC= 210.98 BIC= 216.05 Cp= 7.211
Model x1 x3    MSE= 8.739 R2adj= 0.183  AIC= 206.23 BIC= 211.29 Cp= 2.592
Model x1 x2 x3 MSE= 8.597 R2adj= 0.174  AIC= 207.57 BIC= 214.33 Cp= 4.000
```

## Example of penalty measures (continued)

- User-written `vselect` command (Lindsey and Sheather 2010)
  - best subsets gives best fitting model (lowest MSE) with one, two and three regressors
  - and for each of these best fitting models gives various penalty measures
  - all measures favor model with just $x_1$.

```
. * Best subset selection with user-written add-on vselect
. vselect y x1 x2 x3, best

Response :              y
Selected predictors:   x1 x2 x3

Optimal models:

   # Preds     R2ADJ          C        AIC       AICC        BIC
         1  .2043123   .5925225   204.2265   204.8932   207.6042
         2  .1959877   2.002325   205.5761   206.7189   210.6427
         3  .1737073          4   207.5735   209.3382    214.329

predictors for each model:

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

## Example of penalty measures (continued)

- `vselect` also does forward selection and backward selection
  - then need to specify whether use R2adj, AIC, BIC or AICC
  - e.g. `vselect y x1 c2 c3, forward aic`
  - e.g. `vselect y x1 c2 c3, backward bic`
- And can specify that some regressors always be included
  - e.g. `vselect y x2 x3, fix(x1) best`
- User-written `gvselect` command (Lindsey and Sheather 2015) implements best subsets selection for any Stata command that reports $\ln L$
  - then best model of any size has highest $\ln L$
  - and best model size has lowest AIC or BIC.

## 2.5 Cross-validation

- Begin with single-split validation
  - for pedagogical reasons.
- Then present K-fold cross-validation
  - used extensively in machine learning
  - generalizes to loss functions other than MSE such as $\frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i|$
  - though more computation than e.g. BIC.
- And present leave-one-out cross validation
  - widely used for local fit in nonparametric regression.
- Given a selected model the final estimation is on the full dataset
  - usual inference ignores the data-mining.

## 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: estimate all 8 possible models with $x_1$, $x_2$ and $x_3$
  - for each model estimate on the training set to get $\widehat{\beta}' s$, predict on the validation set and compute MSE in the validation set.
  - choose the model with the lowest validation set MSE.
- 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.

## Single split validation example

- Randomly form
  - training sample $(n = 14)$
  - test sample $(n = 26)$

```
. * Form indicator that determines training and test datasets
. set seed 10101

. gen dtrain = runiform() > 0.5  // 1 for training set and 0 for test data

. count if dtrain == 1
  14
```

## Single split validation example (continued)

- In-sample (training sample) MSE minimized with $x_1$, $x_2$, $x_3$.
- Out-of-sample (test sample) MSE minimized with only $x_1$.

```
. * Split sample validation - training and test MSE for the 8 possible models
. forvalues k = 1/8 {
  2.   quietly reg y ${xlist`k'} if dtrain==1
  3.   qui predict y`k'hat
  4.   qui gen y`k'errorsq = (y`k'hat - y)^2
  5.   qui sum y`k'errorsq if dtrain == 1
  6.   scalar mse`k'train = r(mean)
  7.   qui sum y`k'errorsq if dtrain == 0
  8.   qui scalar mse`k'test = r(mean)
  9.   display "Model " "${xlist`k'}" _col(16)  ///
>      " Training MSE = " %7.3f mse`k'train " Test MSE = " %7.3f mse`k'test
 10. }
Model          Training MSE =  10.496 Test MSE =  11.852
Model x1       Training MSE =   7.625 Test MSE =   9.568
Model x2       Training MSE =  10.326 Test MSE =  10.490
Model x3       Training MSE =   7.943 Test MSE =  13.052
Model x1 x2    Training MSE =   7.520 Test MSE =  10.350
Model x2 x3    Training MSE =   7.798 Test MSE =  15.270
Model x1 x3    Training MSE =   7.138 Test MSE =  10.468
Model x1 x2 x3 Training MSE =   6.830 Test MSE =  12.708
```

## K-fold cross-validation

- $K$-fold cross-validation
  - splits data into $K$ mutually exclusive folds of roughly equal size
  - for $j = 1, ..., K$ fit using all folds but fold $j$ and predict on fold $j$
  - standard choices are $K = 5$ and $K = 10$.
- The following shows case $K = 5$

| | Fit on folds | Test on fold |
|---|---|---|
| $j = 1$ | 2,3,4,5 | 1 |
| $j = 2$ | 1,3,4,5 | 2 |
| $j = 3$ | 1,2,4,5 | 3 |
| $j = 4$ | 1,2,3,5 | 4 |
| $j = 5$ | 1,2,3,4 | 5 |

- The $K$-fold CV estimate is

$$\text{CV}_K = \tfrac{1}{K}\textstyle\sum_{j=1}^{K} \text{MSE}_{(j)}, \text{ where } \text{MSE}_{(j)} \text{ is the MSE for fold } j.$$

## K-fold cross validation example

- User-written `crossfold` command (Daniels 2012) implements this
  - do so for the model with all three regressors and $K = 5$
  - `set seed` for replicability.

```
. * Five-fold cross validation example for model with all regressors
. set seed 10101

. crossfold regress y x1 x2 x3, k(5)

             |       RMSE
-------------+-----------
        est1 |   3.739027
        est2 |   2.549458
        est3 |   3.059801
        est4 |   2.532469
        est5 |   3.498511

. matrix RMSEs  = r(est)

. svmat RMSEs, names(rmse)

. quietly generate mse = rmse^2

. quietly sum mse

. display _n "CV5 (average MSE in 5 folds) = " r(mean) " with st. dev. = " r(sd)

CV5 (average MSE in 5 folds) = 9.6990836 with st. dev. = 3.3885108
```

## K-fold Cross Validation example (continued)

- Now do so for all eight models with $K = 5$
  - model with only $x_1$ has lowest CV$(5)$

```
. * Five-fold cross validation measure for all possible models
. drop rm*      // Drop variables created by previous crossfold

. drop _est*    // Drop variables created by previous crossfold

. forvalues k = 1/8 {
  2.    set seed 10101
  3.    quietly crossfold regress y ${xlist`k'}, k(5)
  4.    matrix RMSEs`k' = r(est)
  5.    svmat RMSEs`k', names(rmse`k')
  6.    quietly generate mse`k' = rmse`k'^2
  7.    quietly sum mse`k'
  8.    scalar cv`k' = r(mean)
  9.    scalar sdcv`k' = r(sd)
 10.    display "Model " "${xlist`k'}" _col(16) "  CV5 = " %7.3f cv`k' ///
>       " with st. dev. = " %7.3f sdcv`k'
 11. }
Model            CV5 =  11.960 with st. dev. =   3.561
Model x1         CV5 =   9.138 with st. dev. =   3.069
Model x2         CV5 =  10.407 with st. dev. =   4.139
Model x3         CV5 =  11.776 with st. dev. =   3.272
Model x1 x2      CV5 =   9.173 with st. dev. =   3.367
Model x2 x3      CV5 =  10.872 with st. dev. =   4.221
Model x1 x3      CV5 =   9.639 with st. dev. =   2.985
Model x1 x2 x3   CV5 =   9.699 with st. dev. =   3.389
```

## 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

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

- Requires $n$ regressions in general
  - except for OLS can show $\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i-\widehat{y}_i}{1-h_{ii}}\right)^2$
    - where $\widehat{y}_i$ is fitted value from OLS on the full training sample
    - and $h_{ii}$ is $i^{th}$ diagonal entry in the hat matrix $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}$.
- Used for bandwidth choice in local nonparametric regression
  - such as k-nearest neighbors, kernel and local linear regression
  - but not used for machine learning (see below).

## Leave-one-out cross validation example

- User-written command `loocv` (Barron 2014)
  - slow as written for any command, not just OLS.

```
. * Leave-one-out cross validation
. loocv regress y x1


 Leave-One-Out Cross-Validation Results
 ─────────────────────────────────────────
           Method         │     Value
 ─────────────────────────┼───────────────
 Root Mean Squared Errors │   3.0989007
 Mean Absolute Errors     │   2.5242994
 Pseudo-R2                │   .15585569
 ─────────────────────────┴───────────────

. display "LOOCV MSE = " r(rmse)^2
LOOCV MSE = 9.6031853
```

## How many folds?

- LOOCV is the special case of $K$-fold CV with $K = N$
  - it has little bias
    - as all but one observation is used to fit.
  - but large variance
    - as the $n$ predicted $\widehat{y}_{(-i)}$ are based on very similar samples
    - so subsequent averaging does not reduce variance much.
- The choice $K = 5$ or $K = 10$ is found to be a good compromise
  - neither high bias nor high variance.
- Remember: For replicability set the seed as this determines the folds.

## One standard error rule for K-fold cross-validation

- $K$ folds gives $K$ estimates $MSE_{(1)}, ..., MSE_{(K)}$
  - so we can obtain a standard error of $\mathrm{CV}_{(K)}$

$$\mathrm{se}(\mathrm{CV}_{(K)}) = \sqrt{\frac{1}{K-1} \sum_{j=1}^{K} (MSE_{(j)} - CV_{(K)})^2}.$$

- A further guard against overfitting that is sometimes used
  - don't simply choose model with minimum $\mathrm{CV}_{(K)}$
  - instead choose the smallest model for which CV is within one se(CV) of minimum CV
  - clearly could instead use e.g. a 0.5 standard error rule.
- Example is determining degree $p$ of a high order polynomial in $x$
  - if $\mathrm{CV}_{(K)}$ is minimized at $p = 7$ but is only slightly higher for $p = 3$ we would favor $p = 3$.

## 3. Shrinkage Estimation

- 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, elastic net
  - reduce the dimension of the regressors
    - principal components analysis.
- Linear regression may predict well if include interactions and powers as potential regressors.
- And methods can be adapted to alternative loss functions for estimation.

## 3.1 Variance-bias trade-off

- Consider regression model

$$y = f(\mathbf{x}) + u \text{ with } E[u] = 0 \text{ and } u \perp \mathbf{x}.$$

- For out-of-estimation-sample point $(y_0, \mathbf{x}_0)$ the true prediction error

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

- The last term $Var(u)$ is called irreducible error
  - we can do nothing about this.
- So need to **minimize sum of variance and bias-squared**!
  - more flexible models have less bias (good) and more variance (bad).
  - this trade-off is fundamental to machine learning.

## Variance-bias trade-off and shrinkage

- Shrinkage is one method that is biased but the bias may lead to lower squared error loss
  - first show this for estimation of a parameter $\beta$
  - then show this for prediction of $y$.
- The mean squared error of a scalar estimator $\widetilde{\beta}$ is

$$\begin{aligned}
\text{MSE}(\widetilde{\beta}) &= E[(\widetilde{\beta}-\beta)^2] \\
&= E[\{(\widetilde{\beta}-E[\widetilde{\beta}])+(E[\widetilde{\beta}]-\beta)\}^2] \\
&= E[(\widetilde{\beta}-E[\widetilde{\beta}])^2]+(E[\widetilde{\beta}]-\beta)^2+2\times 0 \\
&= Var(\widetilde{\beta})+Bias^2(\widetilde{\beta})
\end{aligned}$$

  - as the cross product term $2\times E[(\widetilde{\beta}-E[\widetilde{\beta}])(E[\widetilde{\beta}]-\beta)] =$ constant$\times E[(\widetilde{\beta}-E[\widetilde{\beta}])]=0$.

## Bias can reduce estimator MSE: a shrinkage example

- Suppose scalar estimator $\widehat{\beta}$ is unbiased for $\beta$ with
  - $E[\widehat{\beta}] = \beta$ and $Var[\widehat{\beta}] = v$ so $\text{MSE}(\widehat{\beta}) = v$.
- Consider the shrinkage estimator
  - $\widetilde{\beta} = a\widehat{\beta}$ where $0 \leq a \leq 1$.
- Bias: $Bias(\widetilde{\beta}) = E[\widetilde{\beta}] - \beta = a\beta - \beta = (a-1)\beta$.
- Variance: $Var[\widetilde{\beta}] = Var[a\widehat{\beta}] = a^2 Var(\widehat{\beta}) = a^2 v$.

$$\begin{aligned} \text{MSE}(\widetilde{\beta}) &= Var[\widetilde{\beta}] + Bias^2(\widetilde{\beta}) = a^2 v + (a-1)^2\beta^2 \\ \text{MSE}(\widetilde{\beta}) &< \text{MSE}[\widehat{\beta}] \text{ if } \beta^2 < \frac{1+a}{1-a}v. \end{aligned}$$

- So $\text{MSE}(\widetilde{\beta})$ $<$MSE$[\widehat{\beta}]$ for $a=0$ if $\beta^2 < v$ (and for $a = 0.9$ if $\beta^2 < 19v$).
- The ridge estimator shrinks towards zero.
- The LASSO estimator selects and shrinks towards zero.

## James-Stein estimator

- This remarkable 1950's/1960's result was a big surprise
  - an estimator has lower MSE than the maximum likelihood estimator.
- Suppose $y_i \sim N(\mu_i, 1)$, $i = 1, ..., n$.
- The MLE is $\widehat{\mu}_i = y_i$ with $\text{MSE}(\widehat{\mu}_i) = 1$.
- The James-Stein estimator is $\widetilde{\mu}_i = (1-c)y_i + c\overline{y}$
  - where $c = \frac{1}{n-3}\sum_{i=1}^n (y_i - \overline{y})^2$ and $n \geq 4$
  - this has $\text{MSE}(\widetilde{\mu}_i) < MSE(\widehat{\mu}_i)$ for $n \geq 4$!
- The estimator can be given an empirical Bayes interpretation.

## Bias can therefore reduce predictor MSE

- Now consider prediction of $y_0 = \beta x_0 + u$ where $E[u] = 0$
  - using $\widetilde{y}_0 = \widetilde{\beta} x_0$ where treat scalar $x_0$ as fixed.
- Bias: $Bias(\widetilde{y}_0) = E[x_0\widetilde{\beta}] - \beta x_0 = x_0(E[\widetilde{\beta}] - \beta) = x_0 Bias(\widetilde{\beta})$.
- Variance: $Var[\widetilde{y}_0] = Var[x_0\widetilde{\beta}] = x_0^2 Var(\widetilde{\beta})$.
- The mean squared error of a scalar estimator $\widetilde{\beta}$ is

$$\begin{aligned} \text{MSE}(\widetilde{y}_0) &= Var(\widetilde{y}_0) + Bias^2(\widetilde{y}_0) + Var(u) \\ &= x_0^2 Var(\widetilde{\beta}) + (x_0 Bias(\widetilde{\beta}))^2 + Var(u) \\ &= x_0^2\{Var(\widetilde{\beta}) + Bias^2(\widetilde{\beta})\} + Var(u) \\ &= x_0^2 Bias^2(\widetilde{\beta}) + Var(u). \end{aligned}$$

- So bias in $\widetilde{\beta}$ that reduces $\text{MSE}(\widetilde{\beta})$ also reduces $\text{MSE}(\widetilde{y}_0)$.

## 3.2 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, LASSO and elastic net are not invariant to rescaling of regressors, so first standardize
  - so $x_{ij}$ below is actually $(x_{ij} - \bar{x}_j)/s_j$
  - and demean $y_i$ so below $y_i$ is actually $y_i - \bar{y}$
  - $\mathbf{x}_i$ does not include an intercept nor does data matrix $\mathbf{X}$
  - we can recover intercept $\beta_0$ as $\widehat{\beta}_0 = \bar{y}$.
- So work with $y = \mathbf{x}'\boldsymbol{\beta} + \varepsilon = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$

## Demeaning data

- The commands below do this automatically
  - but for completeness following code demeans.

```
. * Standardize regressors and demean y
. foreach var of varlist x1 x2 x3 {
  2.     qui egen z`var' = std(`var')
  3.     }

. quietly summarize y

. quietly generate ydemeaned = y - r(mean)

. summarize ydemeaned z*

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   ydemeaned |         40   -1.71e-08    3.400129  -6.650633   7.501798
         zx1 |         40    2.05e-09           1  -1.594598   2.693921
         zx2 |         40    2.79e-10           1   -2.34211    2.80662
         zx3 |         40    2.79e-09           1  -1.688912   2.764129
```

- The original variables $x_1$ to $x_3$ had standard deviations 0.89867, 0.94222 and 1.03462
  - means differ from zero due to single precision rounding error.

## 3.3 Ridge Regression

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

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

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

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 \text{ subject to} \sum_{j=1}^{p} \beta_j^2 \leq s.$$

- 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 $\lambda$
  - then choose $\lambda$ 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 $\lambda$) 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 $\lambda = \partial L_{opt}(\boldsymbol{\beta}, \lambda, s)/\partial s$.

## More on Ridge

- Hoerl and Kennard (1970) proposed ridge as a way to reduce MSE of $\widetilde{\boldsymbol{\beta}}$.
- We can write ridge as $\widehat{\boldsymbol{\beta}}_{\lambda} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{X} \times \widehat{\boldsymbol{\beta}}_{OLS}$
  - so shrinkage of OLS
- For scalar regressor and no intercept $\widehat{\beta}_{\lambda} = a\widehat{\beta}_{OLS}$ where $a = \frac{\sum_i x_i^2}{\sum_i x_i^2 + \lambda}$
  - like earlier example of $\widetilde{\beta} = a\widehat{\beta}$.
- Ridge is the posterior mean for $\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I})$ with prior $\boldsymbol{\beta} \sim N(0, \gamma^2\mathbf{I})$
  - though $\gamma$ is a specified prior parameter whereas $\lambda$ is data-determined.
- Ridge is estimator in model $\mathbf{y} \sim (\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I})$ with stochastic constraints $\boldsymbol{\beta} \sim (0, \gamma^2\mathbf{I})$.

## 3.4 LASSO (Least Absolute Shrinkage And Selection)

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

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

  - where $\lambda \geq 0$ is a tuning parameter to be determined
  - $||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$ is L1 norm.
- Equivalently the LASSO estimator minimizes

$$\sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 \text{ subject to} \sum_{j=1}^{p} |\beta_j| \leq s.$$

- 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 versus Ridge (key figure from ISL)

- LASSO is likely to set some coefficients to zero.

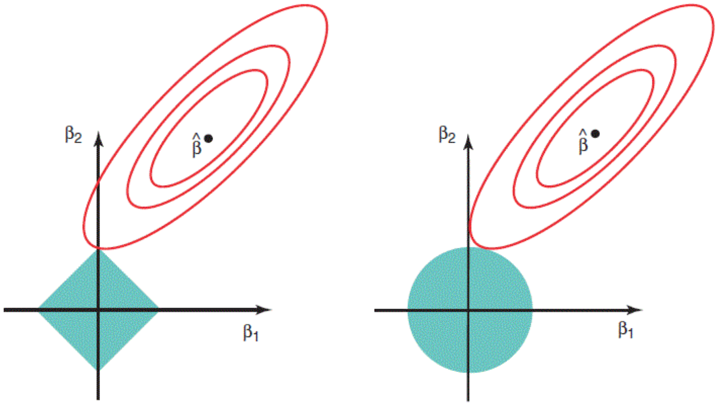

**FIGURE 6.7.** *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.*

## 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 shrinks all $\beta's$ towards zero

$$\begin{aligned}\widehat{\boldsymbol{\beta}}^{R} &= (\mathbf{I}'\mathbf{I} + \lambda\mathbf{I})^{-1}\mathbf{I}'\mathbf{y} = \mathbf{y}/(1+\lambda) \\ \widehat{\beta}_j^{R} &= y_j/(1+\lambda)\end{aligned}$$

- LASSO shrinks some a bit towards 0 and sets others $= 0$

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

- Aside: 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.

## Computation of LASSO estimator

- Most common is a coordinate wise descent algorithm
  - also called a shooting algorithm due to Fu (1998)
  - exploits the special structure in the nondifferentiable part of the LASS objective function that makes convergence possible.
- The algorithm for given $\lambda$ ($\lambda$ is later chosen by CV)
  - denote $\boldsymbol{\beta} = (\beta_j, \boldsymbol{\beta}^{-j})$ and define $S_j(\beta_j, \boldsymbol{\beta}^{-j}) = \partial RSS/\partial\beta_j$
  - start with $\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}_{OLS}$
  - at step $m$ for each $j = 1, ..., p$ let $S_0 = S_j(0, \widehat{\boldsymbol{\beta}}^{-j})$ and set

$$\widehat{\beta}_j = \begin{cases} \frac{\lambda - S_0}{2\mathbf{x}_j'\mathbf{x}_j} & \text{if } S_0 > \lambda \\ \frac{-\lambda - S_0}{2\mathbf{x}_j'\mathbf{x}_j} & \text{if } S_0 > \lambda \\ 0 & \text{if } S_0 > \lambda \end{cases}$$

  - form new $\widehat{\boldsymbol{\beta}}_m = [\widehat{\beta}_1 \cdots \widehat{\beta}_p]$ after updating all $\widehat{\beta}_j$.
- Alternatively LASSO is a minor adaptation of least angle regression
  - so estimate using the forward-stagewise algorithm for LAR.

## LASSO extensions

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

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

- There are other extensions - LASSO is popular.

## 3.5 Elastic net

- Elastic net combines ridge regression and LASSO with objective function

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

  - ridge penalty $\lambda$ averages correlated variables
  - LASSO penalty $\alpha$leads to sparsity.
- Here I use the `elasticregress` package (Townsend 2018)
  - `ridgeregress (alpha=0)`
  - `lassoregress (alpha=1)`
  - `elasticregress.`
- K-fold classification is used with default $K = 10$
  - `set seed` for replicability.
- In part 3 I instead use the `lassopack` package.

## 3.6 Examples: Ridge

- For ridge regression I needed to set `epsilon` to a low value to avoid warning message.
- OLS was $\widehat{y} = 1.556x_1 + 0.471x_2 - 0.026x_3 + 2.532$

```
. * Ridge regression with lambda determined by cross validation
. set seed 10101

. ridgeregress y x1 x2 x3, epsilon(0.00001) numfolds(5)

Ridge regression                       Number of observations    =        40
                                       R-squared                 =    0.2283
                                       alpha                     =    0.0000
                                       lambda                    =    0.2914
                                       Cross-validation MSE      =    9.5405
                                       Number of folds           =         5
                                       Number of lambda tested   =       100
----------------------------------------------------------------------------
           y |      Coef.
-------------+--------------------------------------------------------------
          x1 |   1.152362
          x2 |   .4876821
          x3 |   .0929504
       _cons |    2.65541
----------------------------------------------------------------------------
```

## Ridge example (continued)

- If instead regress on the standardized coefficients
  - `set seed 10101`
  - `ridgeregress ydemeaned zx1 zx2 zx3, /// epsilon(0.00001) numfolds(5)`
- Then find
  - same R-squared and lambda and Cross-validation MSE
  - _cons is zero
  - `b_zx1` = `b_x1 * st.dev.(x1)`
  - similar for $x_2$ and $x/3$

## LASSO example

- OLS was $\widehat{y} = 1.556x_1 + 0.471x_2 - 0.026x_3 + 2.532$
- OLS on $x_1$ and $x_2$ is $\widehat{y} = 1.554x_1 + 0.468x_2 + 2.534$.

```
. * LASSO with lambda determined by cross validation
. set seed 10101

. lassoregress y x1 x2 x3, numfolds(5)

LASSO regression                     Number of observations     =         40
                                     R-squared                  =     0.2293
                                     alpha                      =     1.0000
                                     lambda                     =     0.2594
                                     Cross-validation MSE       =     9.3871
                                     Number of folds            =          5
                                     Number of lambda tested    =        100
------------------------------------------------------------------------------
           y |      Coef.
-------------+----------------------------------------------------------------
          x1 |     1.3505
          x2 |   .2834002
          x3 |          0
       _cons |   2.621573
------------------------------------------------------------------------------
```

## Elastic net example

- OLS was $\widehat{y} = 1.556x_1 + 0.471x_2 - 0.026x_3 + 2.532$
- OLS on $x_1$ and $x_2$ is $\widehat{y} = 1.554x_1 + 0.468x_2 + 2.534$.

```
. * Elastic net with lambda and alpha determined by cross validation
. set seed 10101

. elasticregress y x1 x2 x3, numalpha(50) epsilon(0.00001)  numfolds(5)

Elastic-net regression                     Number of observations     =         40
                                           R-squared                  =     0.2293
                                           alpha                      =     0.9388
                                           lambda                     =     0.2637
                                           Cross-validation MSE       =     9.3929
                                           Number of folds            =          5
                                           Number of alpha tested     =         50
                                           Number of lambda tested    =        100
------------------------------------------------------------------------------
           y |      Coef.
-------------+----------------------------------------------------------------
          x1 |   1.333747
          x2 |   .2993508
          x3 |          0
       _cons |    2.62516
------------------------------------------------------------------------------
```

## 3.7 Other Stata commands for LASSO

- User-written command `lassoshooting` (Christian Hansen)
  - uses the coordinate descent (called lasso shooting) algorithm of Fu (1998)
  - with theoretical or user-choice of $\lambda$ (no cross validation)
  - now superseded by the `lassopack` package.
- `Lassopack` package of Ahrens, Hansen and Schaffer (2019) https://arxiv.org/abs/1901.05397
  - `cvlasso` for $\lambda$ chosen by K-fold cross-validation and h-step ahead rolling cross-validation for cross-section, panel and time-series data
  - `rlasso` for theory-driven ('rigorous') penalization for the lasso and square-root lasso for cross-section and panel data
  - `lasso2` for information criteria choice of $\lambda$
  - used in later set of slides.
- User-written command `lars` (Mander)
  - `lars ydemeaned zx1 zx2 zx3 zx4, a(lasso)`
  - at each step minimizes Mallows Cp

## 4. 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 $M < p$
  - $\mathbf{y} = \beta_0 + \mathbf{X}^*\delta + \mathbf{u}$ after dimension reduction
  - $\mathbf{y} = \beta_0 + \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$ where $\boldsymbol{\beta} = \mathbf{A}\delta$.
- Two methods mentioned in ISL
  - 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)
    - ★ I have not seen this used in practice.
- For both should standardize regressors as not scale invariant.
- And often use cross-validation to determine $M$.

## 4.1 Principal Components Analysis (PCA)

- Suppose $\mathbf{X}$ is normalized to have zero means so $ij^{th}$ entry is $x_{ji} - \bar{x}_j$.
- The first principal component has the largest sample variance among all normalized linear combinations of the columns of $n \times p$ matrix $\mathbf{X}$
  - the first component is $\mathbf{Xh}_1$ where $\mathbf{h}_1$ is $p \times 1$
  - normalize $\mathbf{h}_1$ so that $\mathbf{h}_1'\mathbf{h}_1 = 1$
  - then $\mathbf{h}_1$ max $Var(\mathbf{Xh}_1) = \mathbf{h}_1'\mathbf{X}'\mathbf{Xh}_1$ subject to $\mathbf{h}_1'\mathbf{h}_1 = 1$
  - the maximum is the largest eigenvalue of $\mathbf{X}'\mathbf{X}$ and $\mathbf{h}_1$ is the corresponding eigenvector.
- The second principal component has the largest variance subject to being orthogonal to the first, and so on.

## Formulas for PCA

- Eigenvalues and eigenvectors of $\mathbf{X}'\mathbf{X}$
  - Let $\Lambda = \mathrm{Diag}[\lambda_j]$ 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 Example

- Command pca default is to standardize the data.
- Given d.g.p. for $x_1, x_2, x_3$ we expect eigenvalues 2,0.5,0.5 as $n \longrightarrow \infty$

```
. * Principal components with default correlation option that standardizes data
. pca x1 x2 x3

Principal components/correlation                  Number of obs    =        40
                                                  Number of comp.  =         3
                                                  Trace            =         3
    Rotation: (unrotated = principal)             Rho              =    1.0000

    --------------------------------------------------------------------------
       Component |   Eigenvalue   Difference         Proportion   Cumulative
    -------------+------------------------------------------------------------
           Comp1 |      1.81668      1.08919             0.6056       0.6056
           Comp2 |      .727486       .27165             0.2425       0.8481
           Comp3 |      .455836            .             0.1519       1.0000
    --------------------------------------------------------------------------

Principal components (eigenvectors)

    ----------------------------------------------------------
        Variable |    Comp1     Comp2     Comp3 | Unexplained
    -------------+------------------------------+-------------
              x1 |   0.6306   -0.1063   -0.7688 |           0
              x2 |   0.5712   -0.6070    0.5525 |           0
              x3 |   0.5254    0.7876    0.3220 |           0
    ----------------------------------------------------------
```

## Principal Components Analysis Example (continued)

- First principal component is $0.6306zx_1 + 0.5712zx_2 + 0.5254zx_3$
  - where $zx_j$ are standardized
  - and has variance 1.8618 that explains $1.8618/3 = 0.6056$ of the variance.
- Generate all three principal components and summarize

```
. * Generate the 3 principal components and their means, st.devs., correlations
. quietly predict pc1 pc2 pc3

. summarize pc1 pc2 pc3

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         pc1 |         40   -3.35e-09    1.347842   -2.52927   2.925341
         pc2 |         40   -3.63e-09    .8529281  -1.854475    1.98207
         pc3 |         40    2.08e-09    .6751564  -1.504279   1.520466

. correlate pc1 pc2 pc3
(obs=40)

             |      pc1      pc2      pc3
-------------+---------------------------
         pc1 |   1.0000
         pc2 |   0.0000   1.0000
         pc3 |  -0.0000  -0.0000   1.0000
```

## Principal Components Analysis Example (continued)

- Compare correlation coefficient from OLS on first principal component ($r = 0.4444$) with OLS on all three regressors ($r = 0.4871$) and each single regressor.

```
. * Compare R from OLS on all three regressors, on pc1, on x1, on x2, on x3
. quietly regress y x1 x2 x3

. predict yhat
(option xb assumed; fitted values)

. correlate y yhat pc1 x1 x2 x3
(obs=40)

             |        y     yhat      pc1       x1       x2       x3
-------------+------------------------------------------------------
           y |   1.0000
        yhat |   0.4871   1.0000
         pc1 |   0.4219   0.8661   1.0000
          x1 |   0.4740   0.9732   0.8086   1.0000
          x2 |   0.3370   0.6919   0.7322   0.5077   1.0000
          x3 |   0.2046   0.4200   0.7824   0.4281   0.2786   1.0000
```

## Principal Components Analysis (continued)

- PCA is unsupervised so seems unrelated to $\mathbf{y}$ but
  - *Elements of Statistical Learning* says does well in practice.
  - PCA has the smallest variance of any estimator that estimates the model $\mathbf{y} = \mathbf{X}\beta + \mathbf{u}$ with i.i.d. errors subject to constraint $\mathbf{C}\beta = \mathbf{c}$ 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).
- For completeness next give partial least squares which is supervised.

## 4.2 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.

## 5. 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 $R^2$.

## 6. Some R Commands

- These are from *An Introduction to Statistical Learning: with Applications in R*. **There may be better newer commands.**
- Basic regression
  - OLS is lm.fit
  - cross-validation for OLS uses cv.glm()
  - bootstrap uses boot() function in boot 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
- Dimension reduction
  - principal components: pcr() function in pls library
  - CV for PCA: pcr(,validation="CV")
  - partial least squares: plsr() function in pls library

## 7. 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

## References (continued)

- A recent book is
  - EH: Bradley Efron and Trevor Hastie (2016), *Computer Age Statistical Inference: Algorithms, Evidence and Data Science*, Cambridge University Press.
- Interesting book: Cathy O'Neil (2016), *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*.
- My website has some material including these slides
  - http://cameron.econ.ucdavis.edu/e240f/machinelearning.html