Class 6: Computational Topics

MFE 402

Dan Yavorsky

Last Class

We thought carefully about select variable types...

- Categorical X variables
- Logged X or Y variables

...about relationships between variables

- Multicollinearity

...and about endogeneity

- Errors in Variables
- Omitted Variables

We also thought about individual data points

- Influential observations that have high leverage and large standardized residuals
- Forecasting or predicting new observations and the two sources of error: inherent error and sampling error

Topics for Today

- Midterm
- Stepwise and All Subset Regressions – to optimize in-sample fit
- Cross Validation – to optimize out-of-sample prediction
- Bootstrapping – to estimate standard errors

Best-Subsets and Stepwise Regressions

Best-Subsets Regressions

Suppose you have:

- Many candidate X variables
- Limited experience or theory to guide model development
- Constraints on the number of variables you can include in your model
- Possibly a limited number of observations

One approach: Fit **all** possible models and choose the **best** one, where "best" is:

- almost-always a penalized measure (e.g., $\bar{R}^2$, an IC, Mallows' C_p)
- usually related to in-sample fit

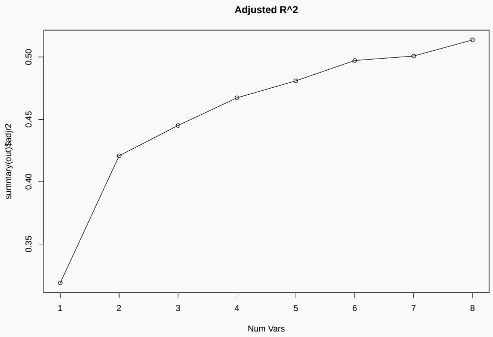
The number of models increases exponentially with the number of candidate variables (p), at a rate of 2^p, so this is computationally intensive:

- if $p = 10$, there are 1,024 possible models
- if $p = 20$, there are 1,048,576 possible models
- if $p = 30$, there are 1,073,741,824 possible models

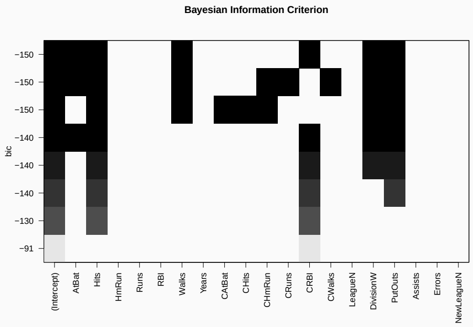
Computation

```r
data(Hitters, package="ISLR")
Hitters <- Hitters[complete.cases(Hitters),]
out <- leaps::regsubsets(Salary ~ ., data=Hitters, nbest=1)
```

```r
plot(summary(out)$adjr2, type="o",
     xlab="Num Vars", main="Adjusted R^2")
```



```r
plot(out, scale="bic",
     main="Bayesian Information Criterion")
```



Stepwise Regressions

For some p, exhaustively searching all possible models can be infeasible or impracticable.

One option is a stepwise approach:

1. Decide on a significance threshold (say, $\alpha = 0.10$)
2. Start with a model with no variables
3. Fit K models with one variable, keep the model with the lowest p-value
4. Fit K - 1 models each with one additional variable, keep the model with the lowest p-value
    - possibly also drop any variables with a p-value above α
5. Repeat step 4 until you can't add any more variables

Alternatively, start with a model with all K variables, and drop variables one at a time

Computation

```r
out_lm <- lm(Salary ~ ., data=Hitters)

out_f <- step(out_lm, direction="forward", trace=0)
out_b <- step(out_lm, direction="backward", trace=0)
out_2 <- step(out_lm, direction="both", trace=0)

lmtest::coeftest(out_2); summary(out_2)$adj.r.squared
```

```

t test of coefficients:

               Estimate  Std. Error t value  Pr(>|t|)    
(Intercept)  162.535442   66.907843  2.4292 0.0158301 *  
AtBat         -2.168650    0.536299 -4.0437 6.996e-05 ***
Hits           6.918017    1.646648  4.2013 3.686e-05 ***
Walks          5.773225    1.584826  3.6428 0.0003274 ***
CAtBat        -0.130080    0.055496 -2.3439 0.0198584 *  
CRuns          1.408249    0.390397  3.6072 0.0003731 ***
CRBI           0.774312    0.209606  3.6941 0.0002706 ***
CWalks        -0.830826    0.263594 -3.1519 0.0018183 ** 
DivisionW   -112.380057   39.214381 -2.8658 0.0045109 ** 
PutOuts        0.297373    0.074440  3.9948 8.504e-05 ***
Assists        0.283168    0.157655  1.7961 0.0736726 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] 0.5222606
```

```r
lmtest::coeftest(out_f); summary(out_f)$adj.r.squared
```

```

t test of coefficients:

               Estimate  Std. Error t value  Pr(>|t|)    
(Intercept)  163.103588   90.778536  1.7967 0.0736220 .  
AtBat         -1.979873    0.633978 -3.1229 0.0020077 ** 
Hits           7.500768    2.377534  3.1549 0.0018082 ** 
HmRun          4.330883    6.201447  0.6984 0.4856158    
Runs          -2.376210    2.980755 -0.7972 0.4261225    
RBI           -1.044962    2.600876 -0.4018 0.6882042    
Walks          6.231286    1.828504  3.4079 0.0007662 ***
Years         -3.489054   12.412186 -0.2811 0.7788736    
CAtBat        -0.171340    0.135237 -1.2670 0.2063804    
CHits          0.133991    0.674552  0.1986 0.8427129    
CHmRun        -0.172861    1.617237 -0.1069 0.9149671    
CRuns          1.454305    0.750458  1.9379 0.0537951 .  
CRBI           0.807709    0.692619  1.1662 0.2446905    
CWalks        -0.811571    0.328083 -2.4737 0.0140574 *  
LeagueN       62.599423   79.261401  0.7898 0.4304236    
DivisionW   -116.849246   40.366952 -2.8947 0.0041408 ** 
PutOuts        0.281893    0.077441  3.6401 0.0003329 ***
Assists        0.371069    0.221199  1.6775 0.0947232 .  
Errors        -3.360760    4.391632 -0.7653 0.4448566    
NewLeagueN   -24.762325   79.002629 -0.3134 0.7542178    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] 0.510627
```

Comments on Best-Subsets and Stepwise Approaches

There are basically no guarantees:

- With a best-subsets approach, there is no guarantee that you find the best model for your needs (be them causal, predictive, interpretable, etc.), only the model that (out of the available data) is optimal based on the selected in-sample criterion
- With a stepwise approach, there is no guarantee that you find even the same model as the best-subsets model, let alone the best model for your needs

We don't rely on stepwise regression or any other automated statistical pattern recognition to pull understanding from our data sets because there is currently no way of providing the critical contextual inputs into these algorithms and because an understanding of the ***context is absolutely critical to making sense of our noisy non-experimental data.*** *The last person you want to analyze an economic data set is a statistician, which is what you get when you run stepwise regression. – Ed Leamer (2007)*

Cross Validation

Cross Validation

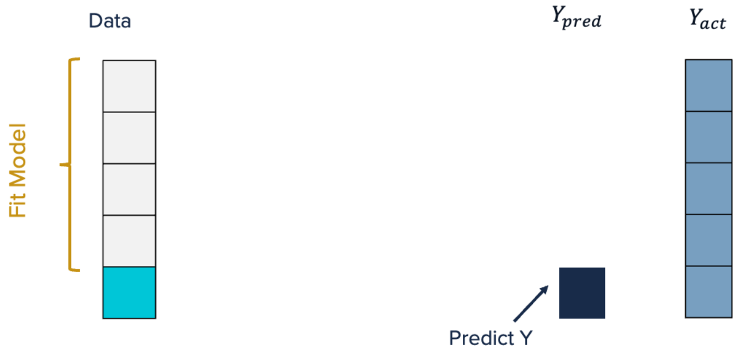
One way to assess **prediction error** is to set aside some of your data (the “test data”), fit your model on the rest (“training data”). Then assess predictions on the training data.

- When data are scarce (as is often the case), this is not possible

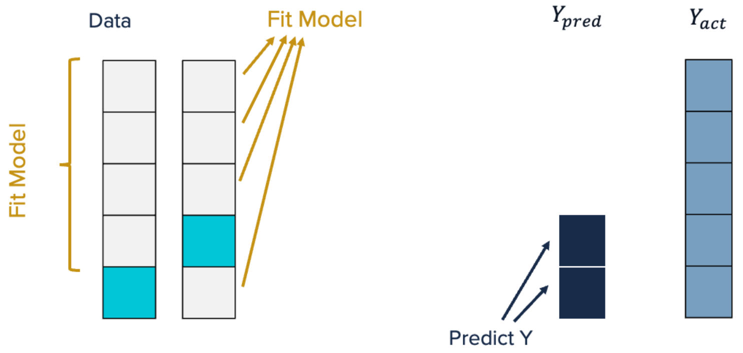
A general, simple, and widely-used method for estimating prediction error (in any model, not just linear regression) is K-fold cross validation:

- Split your data into K roughly-equal-sized parts
- For each of the K parts:
  - Fit the model on the other K - 1 parts
  - Assess your prediction errors on the K^{th} part
- Average all prediction errors to get a MSFE estimate

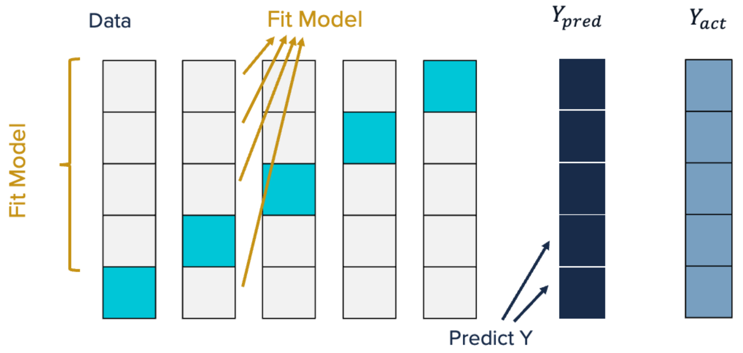
K-Fold Cross Validation



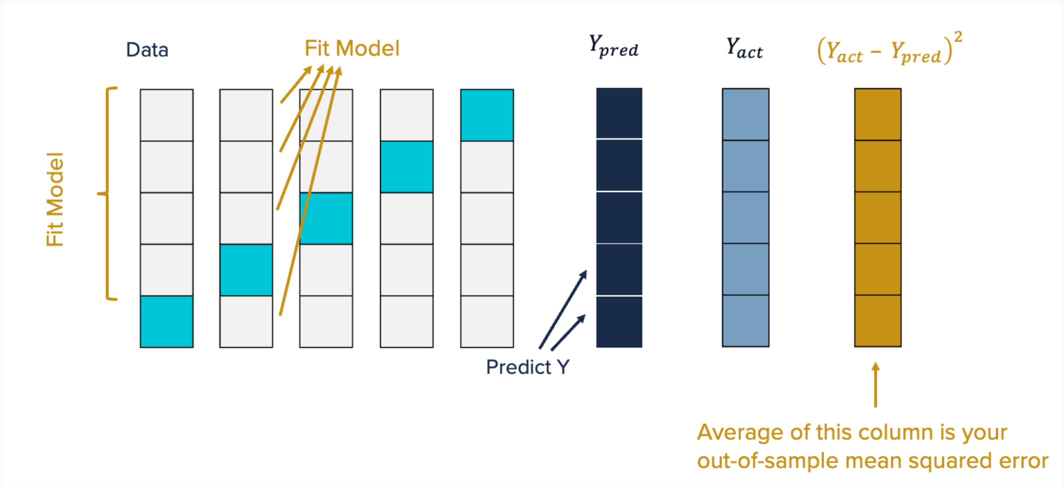
K-Fold Cross Validation



K-Fold Cross Validation



K-Fold Cross Validation



CV and a General MSFE Estimator

It turns out that a general (not just under an assumption of homoskedasticity) estimator for the MSFE of the Linear CEF Model is equivalent to n-fold (i.e., leave-one-out) cross validation:

The estimator is:

$$\hat{\text{MSFE}}_1 = \frac{1}{n}\sum_{i=1}^{n} \tilde{e}_i^2 \qquad \text{where} \quad \tilde{e}_i = \hat{e}_i/(1 - h_{ii})$$

We solved for $\tilde{e}_i$ analytically, but we could have done CV:

- For each observation i:
    - Fit the model on the other $n - 1$ observations
    - Calculate $\tilde{e}_i = Y_i - \tilde{Y}_i$
- Then compute the mean squared error of the n prediction errors $\tilde{e}_i$, $i = 1, \ldots, n$

Computation

```r
# get data
dat <- read.table("support/cps09mar.txt")
exper <- dat[,1] - dat[,4] - 6
lwage <- log( dat[,5]/(dat[,6]*dat[,7]) )
sam <- dat[,11]==4 & dat[,12]==7 & dat[,2]==0
dat <- data.frame(exper=exper[sam],
                  lwage=lwage[sam])

y <- matrix(lwage[sam], ncol=1)
x <- cbind(1, exper[sam])

xxi <- solve(crossprod(x))
xy <- crossprod(x,y)
betahat <- xxi %*% xy

yhat <- x %*% betahat
ehat <- y - yhat
```

```r
# select example observation
j <- 56

# calculate e_tilde from formula
hii <- t(x[j,]) %*% xxi %*% x[j,]
ehat[j] / (1 - hii)
```

```
          [,1]
[1,] 0.4442777
```

```r
# calculate e_tilde from loo regression
reg <- lm(lwage ~ exper, data=dat[-j,])
y[j] - x[j,] %*% coef(reg)
```

```
          [,1]
[1,] 0.4442777
```

Computation

```r
# analytic formula
P <- x %*% xxi %*% t(x)
etilde <- ehat / (1-diag(P))
mean(etilde^2)
```

```
[1] 0.511118
```

```r
# loo regression
n <- length(y)
etilde_loo <- vector(length=n)
for(i in 1:n) {
    coef_est <- coef(lm(lwage ~ exper, data=dat[-i,]))
    etilde_loo[i] <- y[i] - x[i,] %*% coef_est
}

mean(etilde_loo^2)
```

```
[1] 0.511118
```

Bootstrap

Bootstrap Procedure

Two general purpose methods:

1. cross validation measures out-of-sample **prediction errors**
2. bootstrap measures **standard errors** for CI's and HT's

The bootstrap procedure:

for b in $1, \dots, B$ (where B is often 1,000 or 10,000):

- Sample (*with replacement*) n obs from the dataset, call this bootstrap sample b
- Calculate the quantity of interest (QOI) from the bootstrap sample, call this $\hat{\theta}^{(b)}$

Then assess this distribution of the B values of the QOI $\hat{\theta}^{(b)}$, as explained next

Note that for the Linear CEF Model, no assumption on the distribution of the errors (ie, homoskedasticity or normality) is needed

Bootstrap Standard Errors

The bootstrap estimator of $\text{Var}(\hat{\theta})$ is the sample variance across the B values of the QOI:

$$\hat{V}_{\hat{\theta}}^{\text{boot}} = \frac{1}{B - 1} \sum_{b=1}^{B} \left(\hat{\theta}^{(b)} - \bar{\theta} \right)^2 \qquad \text{where} \quad \bar{\theta} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^{(b)}$$

The standard errors are the square-roots of the diagonal elements:

$$s\left(\hat{\theta}_j^{\text{boot}} \right) = \sqrt{\left[\hat{V}_{\hat{\theta}}^{\text{boot}} \right]_{jj}}$$

We can use these standard errors to construct normal-approximation bootstrap confidence intervals (and test hypotheses):

$$\text{CI}^{\text{se boot}} = \left[\hat{\theta} - c \times s\left(\hat{\theta}_j^{\text{boot}} \right),\ \hat{\theta} + c \times s\left(\hat{\theta}_j^{\text{boot}} \right) \right] \qquad \text{where} \quad c = z^*_{1-\alpha/2}$$

Bootstrap Percentile Intervals

Given that we have an empirical distribution, we can simply take the empirical quantiles as the confidence interval boundary values.

- For example, if you have 10,000 bootstrap values of $\hat{\theta}^{(b)}$, sort those values (call the sorted values $\hat{\theta}_*^{(b)}$) and take $\hat{\theta}_*^{(250)}$ and $\hat{\theta}_*^{(9750)}$ as your 95% confidence interval:

$$\mathrm{CI}^{\text{pi boot}} = \left(\hat{\theta}_*^{(250)}, \ \hat{\theta}_*^{(9750)}\right)$$

The most useful feature of this approach is that it is transformation-respecting. If you want a confidence interval for $m(\hat{\theta})$ where $m(\cdot)$ is some function, you simply calculate $(m(\hat{\theta})_*^{(250)}, \ m(\hat{\theta})_*^{(9750)})$

Example Computations

```
# get data
dat <- read.table("support/cps09mar.txt")
exper <- dat[,1] - dat[,4] - 6
lwage <- log( dat[,5]/(dat[,6]*dat[,7]) )
sam <- dat[,11]==4 & dat[,12]==7 & dat[,2]==0
dat <- data.frame(exper=exper[sam], lwage=lwage[sam])

# run regression
out <- lm(lwage ~ exper, data=dat)
tt <- summary(out)$coefficients[1:2,1:2]
tt
```

```
               Estimate  Std. Error
(Intercept) 2.876515044 0.067631401
exper       0.004776039 0.004335196
```

```
# calculate CIs
cbind(low  = tt[,1] - 2*tt[,2],
      high = tt[,1] + 2*tt[,2])
```

```
                     low       high
(Intercept)  2.741252241 3.01177785
exper       -0.003894353 0.01344643
```

```
# bootstrap
set.seed(1234); B <- 1000; n <- nrow(dat)

res <- matrix(NA_real_, nrow=B, ncol=2)
for(b in 1:B) {
    draws <- sample(1:n, size=n, replace=T)
    res[b,] <- lm(lwage ~ exper, data=dat[draws, ])$coef
}

# CIs from stderr
serr <- apply(res, 2, function(x) sqrt(var(x))); serr
```

```
[1] 0.06939970 0.00418621
```

```
cbind(low  = out$coef - 2*serr,
      high = out$coef + 2*serr)
```

```
                     low       high
(Intercept)  2.737715643 3.01531444
exper       -0.003596382 0.01314846
```

```
# CIs from percentiles
rbind(sort(res[,1])[c(25, 975)],
      sort(res[,2])[c(25, 975)])
```

```
             [,1]       [,2]
[1,]  2.742938763 3.01277981
[2,] -0.003159473 0.01333674
```

Next Time

Causal inference

- When can we make *causal* statements rather than correlational ones?
- Introduce a model to think about causality and see what makes determining causality hard
- Demonstrate 3 methods (based on OLS regressions) for estimating causal effects:
    - Conditioning on covariates (AB tests and covariate adjustments)
    - Difference-in-Differences (Diff in Diff)
    - Regression Discontinuity Designs (RDD)