STATISTICS 452/652: Statistical Learning and Prediction

September 9, 2020

Lecture 5: Variable Selection: Classical Approaches

(Reading: ISLR 6.1)

1 Goals of lecture

- We will address the following questions:
  - When we measure p variables in a regression problem, are they all important?
    - Do they all relate to $g(\mathbb{X})$?
  - If they don't, does it matter?
  - If it matters, how do we determine *which* variables we should use in a model?
- In this lecture, we will focus on well-established, older methods of variable selection
  - They are still in common use today
  - There are better methods (Lecture 6!)

2 Regression Reality: Does Size Matter?

- Just because we measure an explanatory variable, doesn't mean that it is related to the response
- It may seem that including all variables is the "safe" thing to do
  - In many applications, people actually want to know which variables are important
  - Furthermore, it turns out that there is a cost to carrying useless variables in a model
- Think about this in the context of the bias-variance tradeoff.

  - If model is not flexible/complex enough, this causes excess bias
  - If model is excessively flexible/complex, this causes excess variance

- We examined this in terms of polynomials, but the same is true with explanatory variables

- The important principles are:

  1. If a variable *is not* included in a model, it is the same as including it but forcing the coefficient to be 0
     (a) *Potentially* ***adds bias*** to predicted values if actual coefficient is not 0
     (b) *Definitely* ***does not affect variance***, because it ignores data: coefficient would be the same with another data set
  2. If a variable *is* included in a model, then its parameter is estimated using the data, including the errors in the data
     (a) *Potentially* ***removes bias*** from predicted values if actual coefficient is not 0
     (b) *Definitely* ***adds variance***, because coefficient would be different with another data set

- Read that last part again, because it's important to remember

  - Every variable you include in a model create a coefficient to be estimated
  - Each estimated coefficient adds variance to the predictions.
  - If you estimate more coefficients than you need, the model is overly complex and excessively variable

- In most cases, it pays to produce models that are SPARSE (or PARSIMONIOUS)

  - Getting rid of useless variables does not affect bias, reduces variance
    * Lots of reduction if n is not large compared to p
    * Smaller reduction if n is huge relative to p (idk...like 100:1? more?)
    * Model is ALWAYS better
  - Getting rid of useful variables affects both bias and variance
    * If variable effect is small, may not be worth the variance it costs to include it!
    * If variable effect is large, get big reductions in bias, so worth keeping

- Aside from prediction quality, smaller models are easier to use

  - More interpretable for domain specialist
  - Easier for future use, because requires fewer measurements
    * Some may be expensive or difficult to get

- Three main approaches to using data to reduce model complexity

1. Subset Selection, the oldest, most-used approach
   - Data identifies which variables seem to "explain" variability in Y
   - Returns a subset of $p' < p$ variables that are deemed worthy of inclusion into the model
   - Reduces variability by estimating fewer coefficients
   - MAY add bias by omitting variables with non-zero coefficients
   - Algorithms try to select combinations of variables that balance between increased bias and decreased variance
2. Shrinkage
   - Use coefficient estimates that are smaller than LS estimates
   - Returns reduced parameter estimates for all variables in the model
   - Reduces variability by preventing each coefficient from overreacting to δ errors in data
   - Adds bias by using slightly wrong estimates of regression coefficients
   - Algorithms try to apply amounts of shrinkage that balance between increased bias and decreased variance
3. Dimension Reduction
   - Exploit correlations within X to engineer features that combine mutual information into smaller number of features
   - Returns $p' < p$ new features that contain "most" of information in X
   - Reduces variability by estimating fewer coefficients
   - MAY add bias if unaccounted information from X helps to predict Y
   - Algorithms try to select appropriate number p' that balances between increased bias and decreased variance

3 Classical ("old") Approaches to Subset Selection

- Here we introduce two main classical approached to variable selection that are still in common use today.
- Neither is perfect, but
  - They are easy to understand
  - There are functions to do them in R and other packages.
- IDEALLY, the domain expert helps to identify sensible models.
  - These techniques are used when data owner can't do this with certainty or if they want to check their beliefs on the data

Table 1: Demonstration of all subsets for $p = 3$ variables.

Model	$f(X)$	Binary		
		X_1	X_2	X_3
1	β_0	0	0	0
2	$\beta_0 + \beta_1 X_1$	1	0	0
3	$\beta_0 + \beta_2 X_2$	0	1	0
4	$\beta_0 + \beta_3 X_3$	0	0	1
5	$\beta_0 + \beta_1 X_1 + \beta_2 X_2$	1	1	0
6	$\beta_0 + \beta_1 X_1 + \beta_3 X_3$	1	0	1
7	$\beta_0 + \beta_2 X_2 + \beta_3 X_3$	0	1	1
8	$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$	1	1	1

3.1 All Subsets Regression

- As the name suggests, do regression on all possible subsets out of the p available variables and pick the best one.
  - Not as easy as it sounds
  - With p variables in a pool, there are 2^p possible models to consider. See Table 1 for example with $p = 3$.
    - For example, with $p = 10$, this is over 1000.
    - For $p = 20$, it is over 1 million models.
- There are computational algorithms that shortcut the number of calculations needed
  - Identify early which models seem to be "lost causes"
  - Focus on better models
- Use algorithm to find best model of each size, $k = 0, 1, \ldots, p$
  - For each fixed size k, minimize sum of squared errors
    - Each size is a separate optimization
  - Let $\mathcal{M}_k$ denote the best model of size k.
    - Sample MSE for model is $sMSE_k$
    - Remember that $sMSE_0 \geq sMSE_1 \geq \ldots \geq sMSE_p$
    - So we can't just pick the model with the best $sMSE$
  - Using a criterion that balances bias and variance to choose best model (see Section 3.3)
  - Pick model with smallest criterion value
    - Model is $\mathcal{M}_{p',\text{AllSub}}$

3.2 Stepwise Selection

- Reduces computation by assuming that models of neighbouring size differ by only one variable
  - $\mathcal{M}_{k+1}$ must contain $\mathcal{M}_k$ plus one new variable
- Allows an algorithm to build models in sequence
  - Compare fewer models of a given size than all-subsets.
- Different versions of algorithm use assumption in different ways

FORWARD selection

1. Start with only an intercept, $\mathcal{M}_0$
2. Repeat these steps for $k = 0, 1, \ldots$ until no more variables can be added
   (a) Find the variable that reduces sMSE most compared to model $\mathcal{M}_k$
      - [Optionally: if sMSE is not reduced by "enough", then stop algorithm and report model $\mathcal{M}_k$ as "best" model.]
   (b) Add it to model to create $\mathcal{M}_{k+1}$
3. Now have a sequence of "best" models of each size
   - May be a different sequence
4. Using a criterion that balances bias and variance to choose best model (see Section 3.3)
5. Pick model with smallest criterion value
   - Model is $\mathcal{M}_{p',\text{Forward}}$

BACKWARD selection

1. Assuming $p < n$, start with all variables in model, $\mathcal{M}_p$
2. Repeat these steps for $k = p, p-1, \ldots, 1$
   (a) Find the variable in the model whose removal causes sMSE to increase the least compared to model $\mathcal{M}_k$
      - [Optionally: if sMSE is increased by "too much", then stop algorithm and report model $\mathcal{M}_k$ as "best" model.]
   (b) Remove it from the model to create $\mathcal{M}_{k-1}$
3. Now have a sequence of "best" models of each size

- May be a different sequence

4. Using a criterion that balances bias and variance to choose best model (see Section 3.3)
5. Pick model with smallest criterion value
    - Model is $\mathcal{M}_{p',\text{Backward}}$

HYBRID versions of algorithms

- There are a variety of algorithms that alternate forward and backward steps
- For example, any time a variable is added, see if a different variable would be removed by a backward step
- Or use criteria below at each step to compare models of size $k-1, k, k+1$ to decide direction of next step or termination

3.3 Criteria used for selection

- At some point, each algorithm returns a set of "best" models of each size, $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$
- Need to choose the best size, $\mathcal{M}_{p'}$
- ISLR describes several methods; I show best/most popular ones

MSPE: Surprisingly complicated!

- We already know we can do this using MSPE if we want
    - Single split:
        - Apply algorithms in training data to find $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$
        - Choose model size with best MSPE on test data
    - Multiple splits or CV, $r = 1, \ldots, R$
        - Repeat single split process *entirely*
            - *Apply algorithms in each training set*
            - Compute MSPEs on best model of each size from that training set
            - Determine best size p'_r *for that split*
        - Optimal sizes and variables in models may vary across splits!
        - Choose mode or average size among p'_r as "best" p'
        - Rerun same algorithm on full data and use model p' variables

Information Criteria (very popular, I recommend)

- Math-stat research has developed measures that balance bias-variance tradeoff in different ways
- Information criteria (IC's) assume that a model is fit using maximum likelihood (ML) estimation
  - LS=ML in linear regression model with normal errors.
- General form of an IC combines closeness of fit with model complexity in opposing ways:

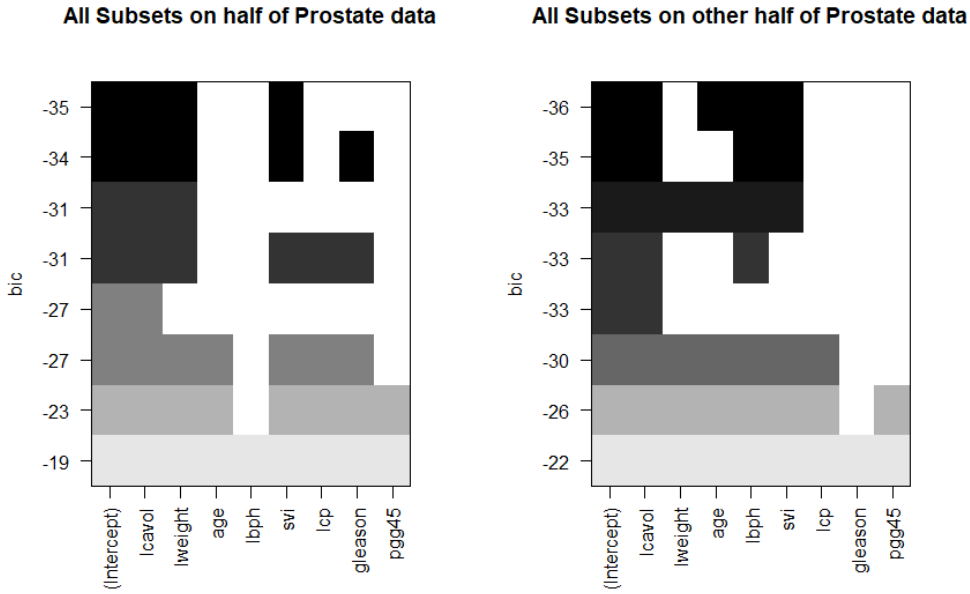
$$IC(w) = n\log(sMSE) + wk$$

  where w is a penalty coefficient and k is the number of parameters in the model
  - First term gets smaller as model fits data more closely
  - Second term gets larger as model gets more complex
  - Minimizing the IC balances bias-variance tradeoff
  - w may vary depending on theory of what IC is optimizing
    - $w = 2$ is asymptotically optimal for prediction
      - $IC(2)$ is Akaike Information Criterion, AIC
    - $w = \log(n)$ is asymptotically optimal for model selection
      - $IC(\log(n))$ is Bayesian Information Criterion, BIC
    - In finite samples, "optimality" may not be achieved perfectly with either one.
- For a given w the model with the smallest value of $IC(w)$ is "best"
  - It does not make sense to try to compare $IC(w)$ values for different w
- Different w may cause different models to be selected
  - The larger w is, the bigger the penalty for adding variables, so this will favour smaller models
  - Specifically, $\log n > 2$ as long as $n \geq 8$, in which case models chosen by BIC are never larger than those chosen by AIC

3.4 Important Notes

- All variable selection algorithms are subject to variability inherent in data
  - Anything optimized to a given data set will chase its unique errors to some degree
  - Different splits *OFTEN* give different "best" models
- *Treat chosen model as an estimate*
  - Think of it as $\hat{\mathcal{M}}_{\hat{p}'}$
- Don't think of it as the "right" model

Figure 1: Best models of each size chosen by all subsets regression applied to two random halves of the prostate data. A *row* represents a model containing variables whose columns are shaded. Note that the BIC values on the axis are not scaled properly, which allows each model row to have the same height on the plot.

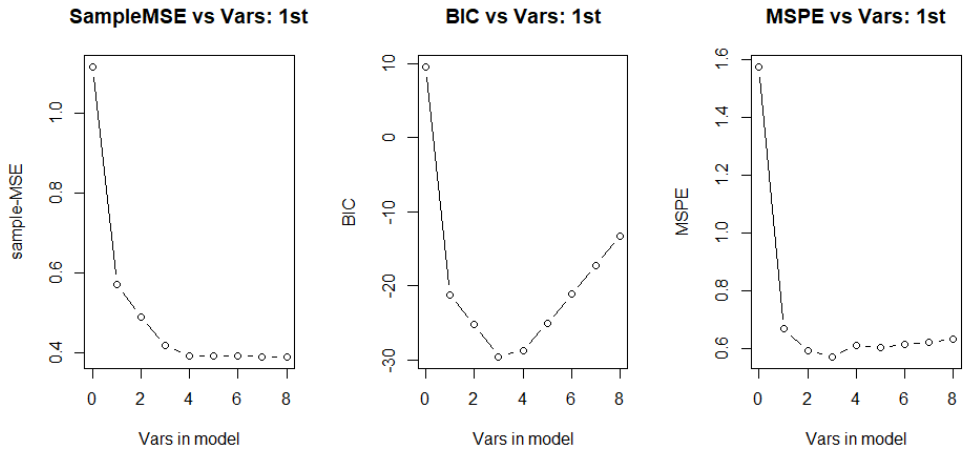


Example: Variable selection on Prostate data (L5 - Variable Selection Algorithms.R) We return to the prostate cancer example to demonstrate all-subsets and stepwise regression algorithms. Normally I would run these procedures on the full data set. However, to show how variable the process of variable selection is, I split the data in half and ran the algorithms on each half. See the program for full details.

ALL SUBSETS I used the `regsubsets()` function from the `leaps` library, although there are other functions in other libraries that do this also. I used the function to record the best subset of each size. The results are in Figure 1. Notice that there is one variable, `lcavol`, that is clearly important in all models for both halves. Also, `svi` is pretty important in most models on both halves. In each half, both large models and small models are not as good as models with an intermediate number of variables. The big difference is `lbph`. In the second half, it is present in all top models that have 2 or more variables, including the top 4 models. In the first half, it appears only in the model with all variables (where it *has* to appear!). The situation would be even muddier if we allowed more than one model of each size to be displayed (tinker with the `nbest=` option in the `regsubsets()` function). *Here is where a subject matter expert would be useful!*

Next, the program re-fits each model and computes several variable-selection criteria on the top model of each size:

Figure 2: Best models of each size chosen by all subsets regression applied to two random halves of the prostate data. A *row* represents a model containing variables whose columns are shaded. Note that the BIC values on the axis are not scaled properly, which allows each model row to have the same height on the plot.



- Training error: the sample-MSE from that model fit
- BIC
- Test error: the mean squared prediction error (MSPE) from predicting the data from one half using the model fit to the other.

I plotted these three criteria against the model size so that you can see how the criteria change with increasing model complexity. See Figure 2.

Notice that

1. All three measures decrease rapidly when the model moves from 0 to 1 variable. That first variable (`lcavol`) seems to be very important.
2. sMSE keeps decreasing as the model size increases, although after a while, the decreases are negligible. In other words, adding more variables hardly improves the model's closeness to the data at all. This would suggest that these later variables are not useful.
3. MSPE and BIC both decrease for a while, then start to increase. There is a minimum in the middle, and both suggest the same value $\hat{p}' = 3$
4. The increase in MSPE after the minimum is gradual. This is typical.
    - Very often, a few variables eat up a lot of bias values, so MSPE improves a lot as the complexity grows from something that is too simple.
    - Variables that truly have $\beta_j = 0$ remove no bias and an amount of variance that depends on sample size, but usually adding just one variable makes only a small difference

- Variables that truly have small, but nonzero parameters reduce bias by very small amounts, and may or may not overcome the variance increase.
- The overall result is a shape like what we see in the third plot

5. BIC puts a very high penalty on each added variable, so it increases faster when marginal or useless variables are added.

STEPWISE I continue with the two halves of the data and run a hybrid form of stepwise regression on each half. This particular algorithm relies on $IC(w)$ for a w that you specify (as `k=`) to tell it what steps to take:

1. Regardless of what $\mathcal{M}_k$ currently is, there is an action that can be taken with each variable
   (a) Any variable in the model could be removed
   (b) Any variable not in the model could be added
2. So for each variable, fit the regression for the model that adds/subtracts it and compute $IC(w)$ for the model fit
3. If no action can make $IC(w)$ smaller than the current model, stop.
4. Otherwise take whatever action leads to the smallest $IC(w)$ and repeat the steps on the new model

See the example below, using BIC on the first half of the data:

```
> # Procedure:
> # Fit minimum and maximum model first ("initial" and "final" below)
> # In step(), object=SMALLEST MODEL, scope=list(upper=LARGEST MODEL)
> #   argument to k gives it the penalty coefficient for the IC to us
> #   Here we use BIC via log(n).
>
> # First half of data
>
> initial.1 <- lm(data=prostate[prostate$set==1,],
+                 formula=lpsa~ 1)
> final.1 <- lm(data=prostate[prostate$set==1,],
+               formula=lpsa~lcavol + lweight + age + lbph + svi + lc
>
> step1 <- step(object=initial.1, scope=list(upper=final.1),
+               k = log(n1))
Start:  AIC=9.52
lpsa ~ 1

           Df Sum of Sq    RSS      AIC
```

```
+ lcavol   1   28.1756 29.702 -21.2194
+ svi      1   18.7494 39.128  -6.8866
+ lcp      1   14.5390 43.338  -1.5722
+ pgg45    1   11.1107 46.767   2.3867
+ gleason  1    9.0726 48.805   4.6049
+ lweight  1    8.3021 49.575   5.4194
+ age      1    7.9472 49.930   5.7903
<none>                57.877   9.5195
+ lbph     1    1.4111 56.466  12.1872

Step:  AIC=-21.22
lpsa ~ lcavol

          Df Sum of Sq    RSS      AIC
+ lweight  1    4.2347 25.467 -25.2668
+ svi      1    2.8306 26.871 -22.4761
+ pgg45    1    2.2629 27.439 -21.3890
<none>                29.702 -21.2194
+ gleason  1    1.6002 28.102 -20.1480
+ age      1    1.2326 28.469 -19.4722
+ lbph     1    0.9737 28.728 -19.0014
+ lcp      1    0.6174 29.084 -18.3605
- lcavol   1   28.1756 57.877   9.5195

Step:  AIC=-25.27
lpsa ~ lcavol + lweight

          Df Sum of Sq    RSS      AIC
+ svi      1    3.7350 21.732 -29.5627
+ gleason  1    2.5687 22.898 -26.8441
+ pgg45    1    2.3137 23.153 -26.2683
<none>                25.467 -25.2668
+ lcp      1    1.0477 24.419 -23.5000
+ age      1    0.0365 25.431 -21.3901
+ lbph     1    0.0010 25.466 -21.3176
- lweight  1    4.2347 29.702 -21.2194
- lcavol   1   24.1081 49.575   5.4194

Step:  AIC=-29.56
lpsa ~ lcavol + lweight + svi

          Df Sum of Sq    RSS     AIC
<none>                21.732 -29.563
+ gleason  1    1.2848 20.447 -28.780
+ pgg45    1    0.5283 21.204 -26.891
```

```
+ age       1     0.0418 21.690 -25.712
+ lcp       1     0.0328 21.699 -25.690
+ lbph      1     0.0235 21.709 -25.668
- svi       1     3.7350 25.467 -25.267
- lweight   1     5.1391 26.871 -22.476
- lcavol    1     8.9261 30.658 -15.620
>
```

The "best" model by this method has `lcavol`, `lweight`, and `svi`. In this example, the results match what all-subsets gave us, but this doesn't always happen. Also, If you run the program, you see that the final model chosen in the second half does not match the one in this half.

---

4 What to take away from this

- Variable selection is a ***very difficult task***. It is loaded with uncertainty
  - Two data sets from the same population need not yield the same recommended model, and often don't.
- All subsets uses brute force to search through all possible combinations of variables for a "best" one
- Stepwise algorithms make assumptions to reduce the computational burden and focus computing on potentially good models
- Information criteria are valid and valuable measures that balance the bias-variance tradeoff too allow comparisons of models with different complexities.
  - Sadly, they require working from a complete statistical model for a problem and will not be available for more general SL algorithms[1]
- The algorithms can identify different "best" models

[1] Part of the research done in my lab is to develop approximate IC's that *can* be used with SL methods that are not fundamentally based in statistical models.

5 Exercises

Application

Refer to the Air Quality data described previously. With `Ozone` as the response variable, we have three potential explanatory variables, `Temp`, `Wind`, and `Solar.R`. We added two more variables,

- `AQ$TWcp = AQ$Temp*Wind`
- `AQ$TWrat = AQ$Temp/Wind`

We will now do variable selection with these five variables

1. Use all-subsets regression.
   (a) **Report the variables in the best model of each size.**
   (b) Compute BIC on each of these models and **report the BIC values for the models.**
   (c) Identify the best model. **What variables are in it?**
2. Use the hybrid stepwise algorithm that is the default in the step() function. **Report the model that it chooses as "best."**
3. Use 10-fold CV to estimate the MSPE for the stepwise model selection process. That is,
   (a) Set the seed to 2928893 before running the `sample.int()` function.
   (b) Create 10 folds
   (c) Run `step()` on each training set
   (d) Find the best model, and compute the prediction error on it
   (e) **Report the separate MSPEs from each fold,** $MSPE_v$, $v = 1, \ldots, 10$ **and the MSPE for the full data.**