MA478 - Lesson 7

Clark

Up to this point in time we've talked about what a GLM is, what are allowable distributions to use for the random component, how random components can help determine our link function (though this isn't necessary), talked about how to fit GLMs and how to conduct inference. Today we're going to round out our GLM discussion with talking about how to determine the linear predictor. That is, we want to discuss how to structure:

Now, one thing to note as we start this discussion is that the meaning of β_j might vary depending on our choice of link function. That is if we consider:

We end up with differing ways that β_j impacts μ or $E[Y]$. The big goals for today are:

- Understand the basic concepts of variable selection for GLMs, including best subset selection and stepwise selection methods.
- Understand model selection metrics, to include Adjusted R2, AIC, BIC and Mallow's Cp.
- Know the concepts of validation and cross-validation procedures.
- Implement variable selection methods for GLMs in R.

Why do we want to conduct variable selection though in the first place? To see this, let's consider the linear model. We know that the Variance of $\hat{\beta}$ can be found through:

Let's consider two models, one where we noted collinearity and removed one variable, the other where we left two collinear variables in our model.

```
APFT <- read.csv("APFT.csv")

model_1 <- lm(APFT_Score~height,data=APFT)

model_2 <- lm(APFT_Score~height+weight,data=APFT)
```

```r
vcov(model_1)
```

```
##             (Intercept)      height
## (Intercept)  1162.40612 -16.7840878
## height        -16.78409   0.2429817
```

```r
vcov(model_2)
```

```
##             (Intercept)      height       weight
## (Intercept) 1492.894073 -25.8634394  1.761005502
## height       -25.863439   0.4902032 -0.047478196
## weight         1.761006  -0.0474782  0.009015105
```

```r
coef(model_1)
```

```
## (Intercept)      height
## 342.0188268  -0.9999333
```

```r
coef(model_2)
```

```
## (Intercept)      height      weight
## 298.5593006   0.1717723  -0.2224821
```

Here we see two things occurring. Our model coefficients change (as to be expected, why?) and our variance for height goes from 0.24 to 0.49. Meaning our certainty of our β_{height} value goes down by adding a redundant predictor to the model. This should be expected though as we add more collinear columns to our $\boldsymbol{X}$ matrix we get:

```r
det(solve(t(model.matrix(model_2))%*%model.matrix(model_2)))
```

```
## [1] 4.244968e-12
```

```r
det(solve(t(model.matrix(model_1))%*%model.matrix(model_1)))
```

```
## [1] 5.426693e-07
```

So we're much less stable in our estimates. Therefore, variable selection may help our accuracy in both our estimation AND our prediction. Though we haven't talked much about prediction intervals recall that for a linear regression model our prediction intervals are found through:

Just as an aside, though this discussion has revolved around linear regression, this is easily extended to GLMs by noting that the the prediction interval for a generalized linear model is (back of the envelope) calculated

by calculating the confidence interval on the scale of η and then feeding the upper and lower bounds of this interval through the inverse link function and the variance of β for a GLM is found through $(X^TWX)^{-1}\hat{\phi}$.

Regardless, the invertability of X^TX (or X^TWX) is of upmost important for both inference and predictions.

In general, for the linear model our estimates of β are well behaved when we have many more observations than predictors $n >> p$. If n is close to p our models may be over-fit and we especially run into issues when our columns of X are colinear. If $p > n$ all hope is lost and the variance of our $\hat{\beta}$ cannot be estimated.

In terms of model interpretability if we have a large number of predictors we likely will have collinear predictors and we will see many that have little to no effect on the response.

Typically there are three categories of variable selection. There's subset selection where we identify a subset of predictors that we believed to be realted to the response; then, fit a model on the reduced set.

There's regularization techniques that shrink the estimated coefficients toward zero relative to the model estimates (ridge regression/lasso - will save for another day).

And there's dimension reduction techniques such as principal components or partial least squares (again will save for another day).

Of the typical subset selections, the ones that are usually consider are forward stepwise selection, backward stepwise selection, and best subset selection. Many statisticians (including myself) abhor forward and backward stepwise selection procedures. From a recent article called "Step away from stepwise" Gary Smith found the following:

Background Stepwise regression is a popular data-mining tool that uses statistical significance to select the explanatory variables to be used in a multiple-regression model.

Findings A fundamental problem with stepwise regression is that some real explanatory variables that have causal effects on the dependent variable may happen to not be statistically significant, while nuisance variables may be coincidentally significant. As a result, the model may fit the data well in-sample, but do poorly out-of-sample.

Conclusion Many Big-Data researchers believe that, the larger the number of possible explanatory variables, the more useful is stepwise regression for selecting explanatory variables. The reality is that stepwise regression is less effective the larger the number of potential explanatory variables. Stepwise regression does not solve the Big-Data problem of too many explanatory variables. Big Data exacerbates the failings of stepwise regression.

In other words, it is well known that we are not guaranteed to get anything of value from doing forward or backward stepwise selection procedures though they remain popular due to ease of implementation.

Best subset is ok, but tends to be computationally expensive. The algorithm is:

```
library(ISLR)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

```r
dim(Hitters)
```

```
## [1] 322  20
```

```r
sum(is.na(Hitters$Salary)) # Salary is missing 59 players
```

```
## [1] 59
```

```r
Hitters <- na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

```r
sum(is.na(Hitters$Salary))
```

```
## [1] 0
```

Let's say we want to fit a model to determine salary for MLB players. Why would we not want to include all the predictors?

```r
#library(leaps)
#regfit.full <- regsubsets(Salary ~ ., data = Hitters)
#reg.summary <- summary(regfit.full)
```

This still doesn't answer the question as to what of these 8 choices are best. To get this we need to compute a statistic for each model that quantifies whether one model is better than the other.

In courses like MA103 we talk about $R^2 = 1 - \frac{SSE}{SST}$. Let's look at this a bit. If I have two models:

How would I compute SSE for both of these?

Why MUST it be that model 2 has a lower SSE than model 1?

```r
#reg.summary$rsq
```

If our models were nested we could compute an F-statistic or a χ^2 statistic for GLMs. However our models aren't. If we look at model 6 it contains AtBat but model 7 does not. Therefore not nested.

So, what should we do. Well, I recommend AIC or BIC. These statistics are defined by computing the log likelihood of the model and then penalizing them for complexity. The formulas are:

There's also DIC or Mallows CP or Adjusted R^2 which all do something similar, but I find AIC and BIC to be sufficient (and mainstream enough that other statisticians won't question it). For our example we have:

```
#plot(reg.summary$bic, xlab = "Number of Predictors", ylab = "BIC", type = "l")
```

So here we see the model with six predictors would be the best, which is:

```
#coef(regfit.full, 6)
```

So, great, we have a way to do this for linear models, but what about GLMs? Well, there are packages in R such as `bestglm` that purport to do this, but I don't like using packages that I have no idea what they're doing.

So, what do I do? Well, I think a little bit and build models deliberately. I want to predict or explain salary I'm probably not going to put in put walks for career and walks in 1986. I build out a few models that are under consideration, then I calculate AIC or BIC for them. Perhaps I try to compare models for career vs models for just 1986

```
model1 <- glm(Salary~Hits+HmRun+RBI+Years,data=Hitters,family=Gamma)
model2 <- glm(Salary~CHits+CHmRun+CRBI,data=Hitters,family=Gamma)
AIC(model1)
```

```
## [1] 3702.367
```

```
AIC(model2)
```

```
## [1] 3746.577
```

```
BIC(model1)
```

```
## [1] 3723.8
```

```
BIC(model2)
```

```
## [1] 3764.438
```

```
model3 <- glm(Salary~Hits+HmRun+RBI+Years,data=Hitters)

AIC(model3)
```

```
## [1] 3856.428
```

```
BIC(model3)
```

```
## [1] 3877.861
```

Here by both statistics model 1 would be preferable to model 2 and both are preferable to model 3 that assumes a linear model.

Both AIC and BIC are meant to estimate the predictive power of a statistical model. An alternative you may be familiar with is cross-validation. In many ways cross-validation has supplanted AIC and BIC, especially in the machine learning literature, but it's not a panacea by any stretch of imagination. If you can compute a likelihood and you have a model with a clearly defined number of parameters, I would much rather compute AIC or BIC for model selection; however, as we will see in this course, this isn't always the case

Typically, cross validation meant to determine how a model performs at prediction. You've likely heard the term *out of sample prediction*, which means we want to predict on data that have not been used to fit the model. Typically, a data set is split into pieces, one or more are used for estimation, and the other piece or pieces are used for validation via prediction.

There are a number of ways to split a data set into parts. Perhaps the most common is k-fold cross validation. Which does:

Really any set of criteria could be used to determine the *best* model. Typically if the response is continuous then squared error is typically used, if the response is binary then misclassification is used. I think for count data it is perhaps less clear, though I feel often squared error loss gets used here though I'm not sure it's entirely appropriate.

If you are choosing between competing models, meaning you are doing model selection instead of model assessment (we talked about this last week) I would encourage you first to consider if your models are nested and a statistical test can be performed. If not, then I would encourage you to look at AIC/BIC or cross-validation.