

STAT350 Tutorial 9

Dylan Maciel

3/11/2020

In this week's tutorial we'll be covering **model selection**. Until now, our approach has been to choose our model and run with it; the regressors were chosen at the beginning of the process and in the end we checked if our model was adequate with respect to prediction accuracy, model assumptions, etc. Sometimes the best model only includes a subset of the available predictors. There are many ways to approach finding this model (approximately). So far you've been introduced to **forward selection, backward elimination, and stepwise regression** through a few types of selection **criteria**: the coefficient of multiple determination, residual mean square, Mallows's C_p Statistic, Akaike information criteria (AIC), and Bayesian information criteria (BIC).

By choosing a "best" subset of predictors we come up to significant topic in statistics: **the bias-variance trade-off**. Recall what happens to a sampling distribution when we increase the sample size – the variance decreases. So, for a given sample of data, a model with only a few parameters to estimate will have low variance and as we increase the number of parameters the variance of their estimates will increase.

But how does this affect the accuracy of the model? If there are too few (or non-ideal) predictors in our model it will have high bias and as we include more predictors, we lower the bias by capturing relevant relationships in the data.

We care about bias and variance as they contribute the error in the model. This is summed up in the figure below, where we see there is an ideal balance between the bias and the variance present in the model which will lead to the minimum total error in the model. **We want to find the simplest model that fits the data well.**

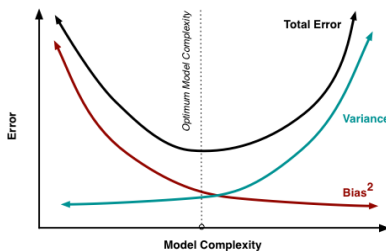


Figure 1: Bias-variance trade-off and the effect on error. (Taken from <https://stats.stackexchange.com/questions/336433/bias-variance-tradeoff-math>)

Stepwise Regression using AIC

I'll be going through a variable selection example with the U.S. census data found in the faraway package. We'll be predicting life expectancy (`Life.Exp`) given some numerical descriptors of the state, there are 7.


```
library(faraway)
data(state)
state_data <- data.frame(state.x77, row.names = state.abb)
head(state_data)
```

##	Population	Income	Illiteracy	Life.Exp	Murder	HS.Grad	Frost	Area
## AL	3615	3624	2.1	69.05	15.1	41.3	20	50708
## AK	365	6315	1.5	69.31	11.3	66.7	152	566432
## AZ	2212	4530	1.8	70.55	7.8	58.1	15	113417
## AR	2110	3378	1.9	70.66	10.1	39.9	65	51945
## CA	21198	5114	1.1	71.71	10.3	62.6	20	156361
## CO	2541	4884	0.7	72.06	6.8	63.9	166	103766

Before going through the procedure I'll introduce the criterion we'll be using:

$$AIC = -2\ln(\mathcal{L}) + 2p.$$

Conveniently, in the OLS setting this simplifies to


$$AIC = -2\ln\left(\frac{SS_{RES}}{n}\right) + 2p.$$

This criterion comes from information theory and tells us how well the model fits the data while trying to prevent over-fitting; we always compare AIC values between models. In the second equation above, we see that AIC does this by balancing the amount of residual error (the first term) with the amount of predictors included in the model (the second term); remember adding a predictor will never increase the residual error.

We want to find the model with minimum AIC.

With the stepwise procedure we can start with the full model or a model with a single predictor. At each step, the AIC of models that are one included or excluded predictor away are computed and compared to the current model. If one or more of the proposed models reduces AIC, we proceed with the one that has minimum AIC and take another step. Otherwise, the current model is the final model and we stop.

To do this in R we'll first fit the full model with all predictors.

```
state_md1 <- lm(Life.Exp ~ ., data = state_data)
```

Next, we apply the `step()` function to the model object. An important argument is `direction` which can be set to 'forward', 'backward', or 'both' (stepwise). The default criteria for this function is `AIC`, so this combined with the direction specifies the variable selection procedure.

```
step(state_md1, direction = 'both')
```

```
## Start:  AIC=-22.18
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##      Frost + Area
##
##           Df Sum of Sq   RSS   AIC
## - Area      1    0.0011 23.298 -24.182
## - Income     1    0.0044 23.302 -24.175
## - Illiteracy  1    0.0047 23.302 -24.174
## <none>                23.297 -22.185
## - Population  1    1.7472 25.044 -20.569
## - Frost       1    1.8466 25.144 -20.371
## - HS.Grad     1    2.4413 25.738 -19.202
## - Murder      1   23.1411 46.438  10.305
##
## Step:  AIC=-24.18
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##      Frost
##
##           Df Sum of Sq   RSS   AIC
## - Illiteracy  1    0.0038 23.302 -26.174
## - Income      1    0.0059 23.304 -26.170
## <none>                23.298 -24.182
## - Population  1    1.7599 25.058 -22.541
## + Area        1    0.0011 23.297 -22.185
## - Frost       1    2.0488 25.347 -21.968
## - HS.Grad     1    2.9804 26.279 -20.163
## - Murder      1   26.2721 49.570  11.569
```

```
##
## Step:  AIC=-26.17
## Life.Exp ~ Population + Income + Murder + HS.Grad + Frost
##
##           Df Sum of Sq   RSS   AIC
## - Income      1      0.006 23.308 -28.161
## <none>                23.302 -26.174
## - Population  1      1.887 25.189 -24.280
## + Illiteracy  1      0.004 23.298 -24.182
## + Area        1      0.000 23.302 -24.174
## - Frost       1      3.037 26.339 -22.048
## - HS.Grad     1      3.495 26.797 -21.187
## - Murder      1     34.739 58.041  17.456
##
## Step:  AIC=-28.16
## Life.Exp ~ Population + Murder + HS.Grad + Frost
##
##           Df Sum of Sq   RSS   AIC
## <none>                23.308 -28.161
## + Income      1      0.006 23.302 -26.174
## + Illiteracy  1      0.004 23.304 -26.170
## + Area        1      0.001 23.307 -26.163
## - Population  1      2.064 25.372 -25.920
## - Frost       1      3.122 26.430 -23.877
## - HS.Grad     1      5.112 28.420 -20.246
## - Murder      1     34.816 58.124  15.528
##
## Call:
## lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
##     data = state_data)
##
## Coefficients:
## (Intercept)  Population      Murder    HS.Grad      Frost
##  7.103e+01   5.014e-05  -3.001e-01   4.658e-02  -5.943e-03
```

The `step()` function returns all the intermediate models. Each step begins with the AIC of the current

model and shows the residual error and AIC of the possible models for the next step. If a variable has a - the AIC reported is for the model with that variable removed. Similarly, a + indicates a model with that variable added. The `<none>` row is the row that corresponds to the current model. Another thing you'll notice in the output; **models with minimum RSS don't necessarily have minimum AIC.**

For our example we get a final model that has four of the seven possible predictors. Interestingly, we never took a forward step to get to this model; only backward eliminations. A couple comments about the path; (1) if we were to transform the variables before doing variable selection it's likely the path would change (you should check for this with the full model beforehand), (2) different criteria can lead to different paths. These methods are more guides rather than set in stone conclusions; knowledge of the data and it's topic should always be taken into consideration. Figure 10.11 in your text provides a flowchart to guide you in the process.

If we wanted to use the BIC, given by

$$BIC = -2\log(\mathcal{L}) + p\ln(n),$$

we could simply use the AIC values given, subtract $2p$, then add $p\ln(n)$. With the `step()` function we can set $k = \log(n)$ to use BIC values in selection. BIC places more weight on the second term (if $n > 7$), and therefore prefers models with less parameters when compared to AIC. If you want to use other criteria I recommend you have a look at the `regsubsets()` function found in the `leaps` package.

To perform the variable selection procedures with hypothesis tests you'll be looking at the summary of the `lm` objects. These contain the p-values associated with the conditional t-tests for each predictor. `update()` is a useful function which allows you to **change a `lm()` object without creating a new one.** So, if we start with our full model:

```
##
## Call:
## lm(formula = Life.Exp ~ ., data = state_data)
##
## Coefficients:
## (Intercept)  Population      Income  Illiteracy      Murder      HS.Grad
##   7.094e+01   5.180e-05  -2.180e-05   3.382e-02  -3.011e-01   4.893e-02
##      Frost      Area
##  -5.735e-03  -7.383e-08
```

And, now I'll remove the predictor `Area` with the following code:

```
update(state_mdl, . ~ . - Area)
```

```
##
```

```
## Call:
```

```
## lm(formula = Life.Exp ~ Population + Income + Illiteracy + Murder +
```

```
##    HS.Grad + Frost, data = state_data)
```

```
##
```

```
## Coefficients:
```

```
## (Intercept)  Population      Income  Illiteracy      Murder      HS.Grad
```

```
##  7.099e+01   5.188e-05  -2.444e-05   2.846e-02  -3.018e-01   4.847e-02
```

```
##      Frost
```

```
## -5.776e-03
```

This way you don't have to define new linear model at each step of the variable selection process.