Set S04 - Model comparison

STAT 401 (Engineering) - Iowa State University

April 27, 2017

Comparing nested models

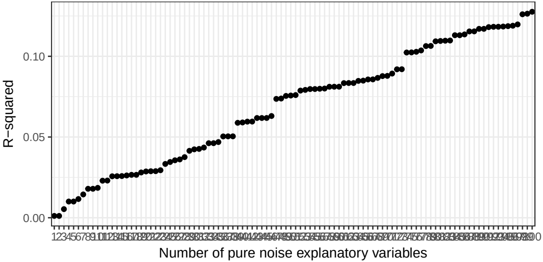
Recall that we have discussed how to compare nested regression models:

- linear regression: F-tests
- generalized linear regression models: likelihood ratio (drop-in-deviance) tests

How do we compare non-nested models?

R^2 always increases as explanatory variables are added

Since the coefficient of determination (R^2) explains "the proportion of variation explained by the model", it seems you would want to choose the model with the highest R^2. But R^2 always increases as explanatory variables are added to the model and thus cannot be used to compare models with different numbers of explanatory variables.



For this reason, sometimes R^2 is reported with a subscript that indicates the number of βs in the model.

Adjusted R-squared

One way to remedy this is to use "adjusted R^2" which can be calculated using the formula

$$\overline{R}^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$$

where

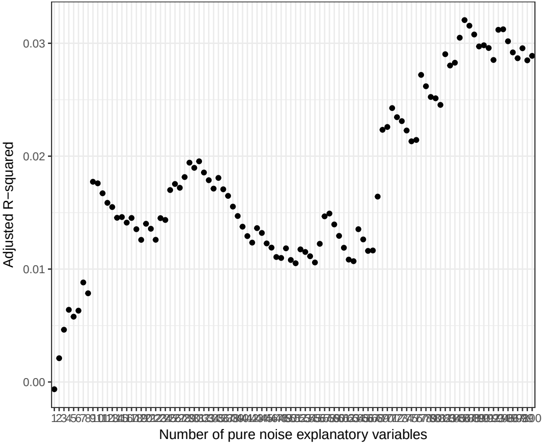
- R^2 is the unadjusted R^2
- n is the number of observations
- p is the number of βs

This formula is equivalent to

$$\overline{R}^2 = 1 - \frac{SSE/df_e}{SST/df_t}.$$

The idea with adjusted R^2 is that it only increases if the inclusion of a new explanatory variable is more than one would expect to see by chance.

Adjusted R^2 can decrease



Model criterion

An alternative to the use of adjusted R^2 is model criterion. Recall that the deviance is $-2\log L(\hat{\theta}_{MLE})$ and is a measure of how well the model fits the data (smaller values indicate better fit). Information criterion attempt to balance this fit with a penalty for too many parameters. They have the form

$$IC = -2\log L(\hat{\theta}_{MLE}) + \text{penalty}$$

where the specific model criterion determines the penalty. For example, here are some criterion

- AIC: $-2\log L(\hat{\theta}_{MLE}) + 2p$
- BIC: $-2\log L(\hat{\theta}_{MLE}) + p\log(n)$
- AICc: $-2\log L(\hat{\theta}_{MLE}) + 2p(p+1)/(n-p-1)$

There are a number of alternatives that also modify the "fit" component:

- DIC: $-2\log L(E[\theta|y]) + 2p_D$ (effective number of parameters)
- BPIC: $-2\log p(\tilde{y}|E[\theta|y]) + 2p_D$
- $\vdots$

Obtaining AIC/BIC in R

```
lm1 <- lm(Fertility ~ . , data = swiss)
AIC(lm1)
```

```
[1] 326.0716
```

```
BIC(lm1)
```

```
[1] 339.0226
```

```
lm2 <- update(lm1, . ~ . -Examination)
AIC(lm1, lm2)
```

```
    df      AIC
lm1  7 326.0716
lm2  6 325.2408
```

```
BIC(lm1, lm2)
```

```
    df      BIC
lm1  7 339.0226
lm2  6 336.3417
```

Obtaining AICc in R

```
AICcmodavg::AICc(lm1)

[1] 328.9434

AICcmodavg::AICc(lm2)

[1] 327.3408

sme::AICc(lm1)

[1] 328.9434

sme::AICc(lm2)

[1] 327.3408
```

Posterior model probability

Another option is a Bayesian posterior model probability. Recall that

$$p(M_j|y) = \frac{p(y|M_j)p(M_j)}{\sum_i p(y|M_i)p(M_i)} = \frac{1}{1 + \sum_{i \neq j} \frac{p(y|M_i)}{p(y|M_j)} \frac{p(M_i)}{p(M_j)}}$$

where $p(y|M_i)/p(y|M_j)$ is the Bayes Factor, $p(M_i)/p(M_j)$ is the prior odds,

$$p(y|M_j) = \int p(y|\theta)p(\theta|M_j)d\theta,$$

- $p(y|\theta)$ is the statistical model (or likelihood) and
- $p(\theta|M_j)$ is the prior over model parameters.

This integral provides a natural penalty for increased number of parameters. BIC is an asymptotic approximation to $p(y|M_j)$.

Bayes Factors in R

```r
library("bayess")
m <- BayesReg(swiss$Fertility, swiss[,-1])
```

```
          PostMean PostStError Log10bf EvidAgaH0
Intercept  70.1426      1.0339
x1         -3.7865      1.5463  0.4357       (*)
x2         -1.9939      1.9614 -0.6087
x3         -8.1123      1.7043  3.3083    (****)
x4          4.2062      1.4240  0.9664      (**)
x5          3.0389      1.0767  0.8215      (**)


Posterior Mean of Sigma2: 50.2438
Posterior StError of Sigma2: 71.877
```

Model selection

Information criterion (and posterior model probabilities) are often used to select a model. Once the model is selected, inference, e.g. parameter estimation and interpretation, is (typically) performed as usual. That is, the process of selecting the model is completely neglected. To account for our uncertainty in model selection, we can perform model averaging.

Bayesian model averaging

Bayesian model averaging can be performed by finding the posterior distribution for any desired quantity. For example, model averaging for an unknown mean μ is

$$p(\mu|y) = \sum_{j=1}^{J} p(\mu|M_j, y)p(M_j|y)$$

and model averaging for an unknown predictive value $\tilde{y}$ is

$$p(\tilde{y}|y) = \sum_{j=1}^{J} p(\tilde{y}|M_j, y)p(M_j|y).$$

AIC model averaging

You can also perform AIC model averaging using Akaike weights:

$$w_j = \frac{e^{-\Delta_j}/2}{\sum_{i=1}^{J} e^{-\Delta_i}/2}$$

where

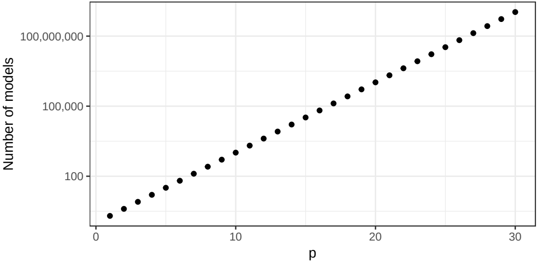
$$\Delta_j = AIC_{min} - AIC_j.$$

Since there is no notion of a posterior distribution, we calculate an estimate and standard error of quantities of interest. For example, a model averaged mean estimate is

$$\hat{\mu} = \sum_{j=1}^{J} \hat{\mu}_j w_j.$$

Step-wise model selection

Once you have chosen a criteria, a common approach is to choose the model that optimizes that criteria. But in regression problems there are 2^p models to consider which is often too many to enumerate. An alternative is to use a stepwise selection procedure that compares all *neighboring* models.



Stepwise selection in R

```
d <- data.frame(X = X) %>% mutate(y = 10 * X.1 + 10 * X.2 + 10 * X.3 +
                                          X.4 +      X.5 +      X.6 +
                                       .1 * X.7 + .1 * X.8 + .1 * X.9 +
                                     rnorm(n()))
m <- step(lm(y ~ ., data = d), k = 2, trace=FALSE) # AIC
summary(m)


Call:
lm(formula = y ~ X.1 + X.2 + X.3 + X.4 + X.5 + X.6 + X.7 + X.8 +
    X.9 + X.10 + X.17 + X.19 + X.27 + X.37 + X.53 + X.65 + X.68 +
    X.74 + X.76 + X.91, data = d)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5111 -0.6685  0.0019  0.6684  3.1214

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.001448   0.031547  -0.046  0.96339
X.1          9.972749   0.031948 312.154  < 2e-16 ***
X.2          9.991404   0.030897 323.374  < 2e-16 ***
X.3          9.979478   0.031495 316.856  < 2e-16 ***
X.4          1.057033   0.032263  32.763  < 2e-16 ***
X.5          0.979328   0.030921  31.672  < 2e-16 ***
X.6          0.988577   0.031698  31.187  < 2e-16 ***
X.7          0.091666   0.031979   2.866  0.00424 **
X.8          0.080109   0.031920   2.510  0.01225 *
X.9          0.146014   0.031284   4.667 3.48e-06 ***
X.10         0.050789   0.031908   1.592  0.11178
X.17         0.048445   0.031588   1.534  0.12544
```

How to select a criterion

All the criterion (and posterior probabilities) are based on some theoretical basis. So you can choose a criterion based on your preferred theoretical basis or based on what your field uses. Often times a criterion is justified based on predictive performance. But then why not just use predictive ability as your criterion?

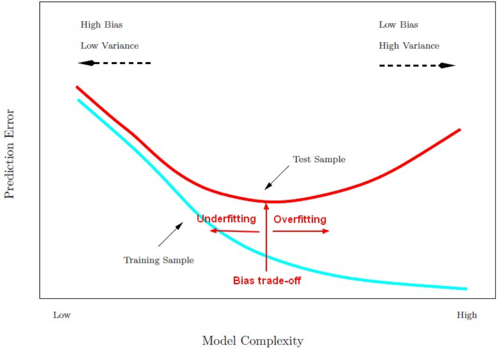
Assessing predictive performance

Sometimes, we are just interested in developing a model with a good predictive performance. To evaluate predictive performance, randomly split your data into

- training data set
- testing data set

Use the training data set to build your model and then use your testing data set to evaluate the fit.

Overfitting



https://gerardnico.com/wiki/data_mining/overfitting

Example splitting in R

```
d <- data.frame(X = X) %>% mutate(y = 10 * X.1 + 10 * X.2 + 10 * X.3 +
                                           X.4 +      X.5 +      X.6 +
                                       .1 * X.7 + .1 * X.8 + .1 * X.9 +
                                     rnorm(n()),
                                   train = rbinom(n(), 1, 0.5))
train <- d %>% filter(train == 1) %>% dplyr::select(-train)
test <- d %>% filter(train == 0) %>% dplyr::select(-train)

m <- step(lm(y ~ ., data = train), k = log(nrow(train)), trace=FALSE)
m


Call:
lm(formula = y ~ X.1 + X.2 + X.3 + X.4 + X.5 + X.6 + X.7 + X.8 +
    X.28 + X.39, data = train)

Coefficients:
(Intercept)          X.1          X.2          X.3          X.4          X.5          X.6          X.7
    0.04179     10.02015      9.99906     10.00197      0.94576      0.92638      0.94104      0.14772       0.1
       X.28         X.39
    0.12241      0.11305


test <- test %>% bind_cols(data.frame(prediction = predict(m, newdata = test)))

# Calculate mean sum of squared errors
with(test, mean((y-prediction)^2))


[1] 1.111346
```

Cross-validation

The model chosen by the test-training split will be sensitive to the test data set chosen. To avoid this sensitivity, we can perform the split several times and average the result.

http://blog.goldenhelix.com/goldenadmin/cross-validation-for-genomic-prediction-in-svs/

Two special cases:

- Leave-one-out cross-validation (LOO-CV)
- k-fold cross validation

Cross-validation in R

```
library(DAAG)
m <- lm(y~., data = d %>% dplyr::select(-train))
cv <- cv.lm(data = d, form.lm = m, m=nrow(d)/5, plotit=FALSE)
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq  F value  Pr(>F)    
X.1        1  94884   94884 9.31e+04 < 2e-16 ***
X.2        1 103067  103067 1.01e+05 < 2e-16 ***
X.3        1  99278   99278 9.74e+04 < 2e-16 ***
X.4        1   1070    1070 1.05e+03 < 2e-16 ***
X.5        1   1011    1011 9.92e+02 < 2e-16 ***
X.6        1    883     883 8.66e+02 < 2e-16 ***
X.7        1     30      30 2.94e+01 7.5e-08 ***
X.8        1     21      21 2.10e+01 5.1e-06 ***
X.9        1      8       8 8.06e+00  0.0046 ** 
X.10       1      1       1 1.02e+00  0.3138    
X.11       1      2       2 1.99e+00  0.1586    
X.12       1      2       2 1.78e+00  0.1831    
X.13       1      1       1 6.60e-01  0.4175    
X.14       1      0       0 5.00e-02  0.8200    
X.15       1      0       0 0.00e+00  0.9592    
X.16       1      0       0 2.00e-02  0.8974    
X.17       1      1       1 7.40e-01  0.3915    
X.18       1      1       1 6.40e-01  0.4225    
X.19       1      0       0 2.00e-01  0.6568    
X.20       1      2       2 2.44e+00  0.1183    
X.21       1      0       0 3.70e-01  0.5408    
X.22       1      0       0 1.80e-01  0.6726    
```