Model Selection and Diagnostics

Michael Winton

June 7, 2018

Variable Selection

Start by using subject matter knowledge to reduce number of variables before using algorithms. Reducing set of variables in a model helps with interpretability. Also, if $p > n$, then it's a requirement.

Also, we have the bias-variance tradeoff. Bias comes from inability to guess the correct relationship between variables, as well as errors in the choice of variables in the model. But, estimating parameters adds variability to any predicted values, so addinng more variables makes the variance worse. For prediction, it might be better to use a model with too few variables (to avoid excess variance). For inference, we might prefer to have more variables (so we don't miss fine contributions).

Model Comparison Criteria

Hypothesis tests can be used to compare models *only* when one is nested in the other. Residual deviance is not a good way to compare models, because it always goes down when you add variables to the model.

Measures based on *information criteria* add a penalty to the log likelihood function to penalize each added variable. Different criteria use different k values, but the same k must be chosen in advance and used for all of the models. n is the sample size, r is the number of parameters in the model - including intercepts, regression coefficients, and other model parameters (e.g. variance for linear regression).

$$IC(k) = -2log[L(\hat{\beta}|y_1, ...y_n)] + kr$$

Commonly used information criteria:

$$AIC = IC(2) = -2log[L(\hat{\beta}|y_1, ...y_n)] + 2r$$

$$\text{Corrected } AIC = AIC_c = IC(\frac{2n}{n-r-1}) = AIC + \frac{2r(r+1)}{n-r-1}$$

$$BIC = IC(log(n)) = -2log[L(\hat{\beta}|y_1, ...y_n)] + [log(n)]r$$

The penalty for AIC_c is always greater than AIC, but they become similar at large n in comparion to r. BIC gets more severe for larger samples; it's always larger than AIC for $n \geq 8$. BIC tends to favor the smallest models, and AIC the largest. We look for the model with the smallest IC; those within $IC(k) \pm 2$ are considered similar.

BIC has the property of *consistency*, which means asymptotically it selects the true model. AIC has the property of *efficiency*, which means asymptotically it minimizes the mean squared error of prediction. AIC_c was developed to bring efficiency to smaller samples, although it's empirical (theory isn't proven).

We should prefer AIC_c or BIC. We use `AIC(...)` to calculate these by setting $k = 2$ (default) or $k = log(n)$ in the function call. AIC_c must be calculated manually from AIC result.

All Subsets Regression

Only computationally feasible for up to ~10 explanatory variables. There are deterministic and stochastic (random process) variants that can be used when it's not feasible to try all variables. The "genetic" stochastic algorithm tends to do a good job converging on a good model, but run it a few times to confirm since it's random.

These can be run with `glmulti(...)` (slow and may have java problems, but supports more models), `bestglm(...)`, and `BMA(...)`. See pages 270-271 for details.

Stepwise Selection

Stepwise Forwards & Backwards Selection are greedy algorithms, either adding or eliminating one variable at a time. Use `step(...)` function.

Alternating Stepwise Selection gives variables that have been removed a chance to be re-added. It proceeds by forward selection, but each time a variable is added, a round of backward elimination is carried out so that variables can be removed from the model. This involves manual manipulation of `step(...)`.

IMPORTANT: Stepwise procedures have to be done with information criteria, not hypothesis tests. Also, significant tests for variables in a model selected by an information criterion are not valid tests, so you can't follow this with another method try to reduce number of variables!

LASSO

Variable selection tends to select variables where random chance made them appear more significant vs. less; thus parameter estimates for selected variables are biased (often estimated further away from zero than actual). LASSO attempts to select variables and shrink bias back towards zero.

LASSO estimates parameters $\hat{\beta}_{0,LASSO}, \hat{\beta}_{1,LASSO}, ...$ by maximizing:

$$log[L(\beta_0, \beta_1, ...\beta_p|y_1, ...y_n)] - \lambda \sum_{j=1}^{p} |\beta_j|$$

The hyperparameter λ is usually chosen through cross-validation. Larger λ causes more shrinkage and smaller models. LASSO often results in better estimates than MLEs, but *inference procedures have not been well-developed yet.* Also note that it can perform poorly when there are many variables in the pool, but few are important.

LASSO parameter estimates can be calculated using `glmnet(...)` and `cv.glmnet(...)`. These do not support categorical explanatory variables, though. For those, use `grplasso(...)`.

```
placekick <- read.csv("placekick.csv")
# get data into separate x, y matrices
yy <- as.matrix(placekick[, 9])
xx <- as.matrix(placekick[, 1:8])

library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
lasso_fit <- glmnet(y = yy, x = xx, family = "binomial")
cols <- c(1, 2, 30, 31, 62, 63)
round(coef(lasso_fit)[, cols], 3)
```

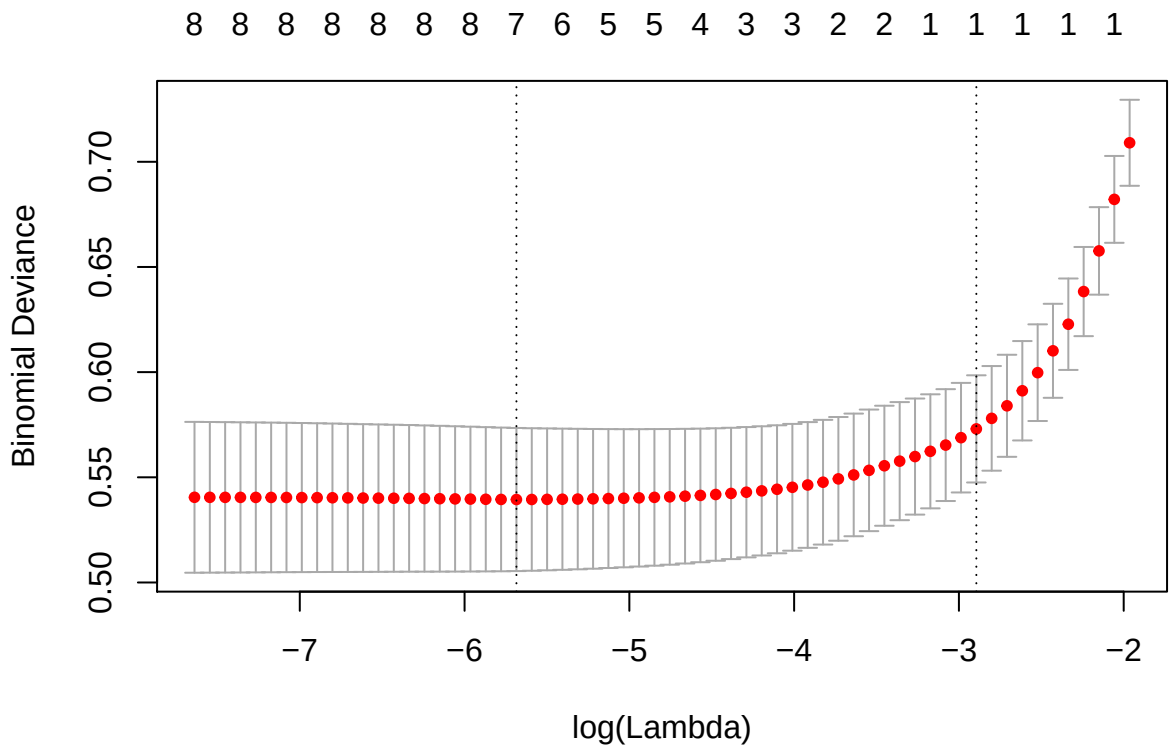
```
## 9 x 6 sparse Matrix of class "dgCMatrix"
##                 s0     s1    s29    s30    s61    s62
## (Intercept) 2.047  2.359  4.671  4.696  4.786  4.785
## week        .      .     -0.002 -0.004 -0.023 -0.024
## distance    .     -0.011 -0.086 -0.086 -0.086 -0.086
## change      .      .     -0.180 -0.195 -0.342 -0.342
## elap30      .      .      .      .      0.004  0.004
## PAT         .      .      0.779  0.812  1.214  1.216
## type        .      .      .      .      0.265  0.269
## field       .      .      .      .     -0.175 -0.178
## wind        .      .     -0.160 -0.189 -0.590 -0.593
```

In this case, LASSO found 63 values of λ that maximize the log likelihood equation. Use cross-validation to find the best one:

```
cv_lasso_fit <- cv.glmnet(y = yy, x = xx, family = "binomial")
coef(cv_lasso_fit)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  4.09748542
## week         .
## distance    -0.06697266
## change       .
## elap30       .
## PAT          .
## type         .
## field        .
## wind         .
```

```
plot(cv_lasso_fit)
```



The error bars are ± 1 standard error for each value of λ. The numbers on the top of the plot are the numbers of variables in the corresponding model.

```
predict_lasso <- predict(cv_lasso_fit, newx = xx, type = "response")
head(cbind(placekick$distance, round(predict_lasso, 3)))
```

```
##              1
## [1,] 21 0.936
## [2,] 21 0.936
## [3,] 20 0.940
## [4,] 28 0.902
## [5,] 20 0.940
## [6,] 25 0.919
```

Model Evaluation and Diagnostics

When estimating a GLM, we have 3 assumptions to evaluate: 1) Data are generated iid by the model we propose (binomial, Poisson, etc...) 2) The mean of the distribution is linked to the explanatory variables by the link function we chose 3) The link relates to the variables in a linear fashion.

Residuals

Raw residuals ($y_i - \hat{y}_i$) aren't very helpful, as they scale with the mean (ie. not standardized). Instead, we use Pearson residuals, which account for variance in the response variables. Note that for Bernoulli trials, it's important to aggregate them into a binomial format before evaluating residuals.

Pearson residuals:

$$e_i = \frac{y_i - \hat{y}_i}{\sqrt{\hat{var}(Y_i)}} \forall i = 1..M$$

The $\hat{var}(Y_i)$ term is just $\hat{y}_i$ for Poisson, and $n_i\pi_i(1-\pi_i)$ for binomial. The Pearson residuals behave approximately as a sample from a normal distribution, especially for the values are large. However, the denominator overestimates the standard deviation of the numerator. To correct, we use the standardized Pearson residual:

$$e_i = \frac{y_i - \hat{y}_i}{\sqrt{\hat{var}(Y_i - \hat{Y}_i)}} = \frac{y_i - \hat{y}_i}{\sqrt{\hat{var}(Y_i)(1-h_i)}}$$

h_i is the i'th diagonal element of the hat matrix. Standardized Pearson residuals are preferred when they are available.

In R, we can get raw or Pearson residuals from `residuals(...)`. We can get standardized Pearson residuals from `rstandard(...)`.

We can interpret standardizing residuals approximately as observations from a standard normal distribution. This allows us to look at threshholds of ~2 (5% of observations) and ~3 (~no observations) for identifying unusual residuals. However, the normal approximation is poor for binomial models with small n. For example, in the binary cases, every raw residual is either $0 - \hat{\pi}$ or $1 - \hat{\pi}$, with similar problem for Pearson residuals. The normal approximation is also poor for $\hat{\pi}$ near 0 or 1. Similar problems occur for Poisson models with $\hat{y}_i < 0.5$ (because most observations will be 0 or 1).

In cases like that, we would want to calculate the probability of observing such an extreme residual, assuming that $\hat{y}_i$ is correct. In other words, what's the probability of observing a count at least as extreme as the observed y_i.

Diagnostic Plots

1) *Plot standardized residuals vs. each explanatory variable.* This helps confirm *the form* of the explanatory variable is correct. We should see no serious fluctuations of the mean, and approximately equal variance throughout the range of x. Curvature may suggest need for a transformation. Don't overreact to changes in loess curves near extremes.

2) *Plot residuals vs. fitted values.* This is against $\hat{y}$ for Poisson and $\hat{\pi}$ for binomial. This helps identify whether the link function fits well. We should see roughly constant variance and no curvature.

2b) *Alternately, plot residuals vs. linear predictor.* Sometimes patterns are more clear without the link function.

3) *Check for extreme outliers* on any of these plots. Use ± 2 and ± 3 threshholds. This may indicate overdispersion - possibly explanatory variables are missing, or a different distribution is needed to model the data. (See pages 302-306 for more on overdispersion.) Investigate outliers and make decision whether to keep or remove.

Expect extreme values for binomial models at $\hat{\pi}$ near 0 or 1. Calculate probability of seeing them. Expect "bands" of points when we have just a few discrete responses. This could put approximate normality of residuals in doubts, weakening potential for the residuals plots to detect violations of assumptions.

There are not good diagnostic tools for multicategory response models. See pages 301-302.