Lasso method for Logistic Regression

Author: Eric Bataller Thomas

I. EXERCISE 1:

Implement LASSO for logistic regression on the spam dataset. Then, compare MLE, LASSO-CV and LASSO-BIC estimates indicating the number of selected variables and 10-fold cross-validated R^2 for each of these 3 methods. Include the R code you used and a discussion on the following question: if you were asked to choose one of these methods to obtain results for your applied research paper, which one would you pick, and why?

The following table contains the results of the comparison after fitting the logistic regression curve by using 3 different methods (namely, MLE, LASSO setting λ via CV and LASSO setting λ via BIC).

Method	Variables selected	Efron's Pseudo R^2	McFadden Pseudo R^2
MLE	58	0.75269	0.62863
LASSO-CV	54	0.73514	0.64176
LASSO-BIC	55	0.75219	0.64327

TABLE I: Comparison of the resulting number of variables, Cross-Validated Efron's Pseudo R^2 and Cross-Validated McFadden Pseudo R^2 obtained by 3 different methods after fitting the Logistic regression curve on the spam data.

As it can be seen in the table, in all 3 cases the number of variables selected with non-zero coefficients almost contains all the variables in the dataset. This is obvious for the MLE method, as we don't penalize more complex models, but is also expected for the other 2 methods given the nature of the dataset. Our data was build by counting the frequency of appearance of several common words seen in Spam emails. Apart from the fact that those words haven't been selected at random and they are most likely somehow related to real spam emails, we also have to consider that spam emails vary a lot on it's content. Is therefore quite difficult for a model that penalizes complexity to reach a tipping point, where the benefits of increasing explanatory power by adding a new parameter, are offset by the increase in model complexity, as there aren't many variables on the dataset whose explanatory power are "obviously" low suggesting that we can therefore get rid of them in prone of reducing complexity.

In order to get a sense of how well our fitted curve is able to predict new data, Cross validated R-squared is a statistic generated in ordinary least squares (OLS) regression that is often used as a goodness-of-fit measure. When analyzing data with a logistic regression, an equivalent statistic to R-squared does not exist. The model estimates from a logistic regression are maximum likelihood estimates arrived at through an iterative process. They are not calculated to minimize variance, so the OLS approach to goodness-of-fit does not apply. However, to evaluate the goodness-of-fit of logistic models, several pseudo R-squareds have been developed. These are "pseudo" R-squareds because they look like R-squared in the sense that they are on a similar scale, ranging from 0 to 1 (though some pseudo R-squareds never achieve 0 or 1) with higher values indicating better model fit, but they cannot be interpreted as one would interpret an OLS R-squared and different pseudo R-squareds can arrive at very different values.
Two of the most famous pseudo R^2 metrics are the Efron's and the McFaddent R^2. The first one is computed as

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{\pi}_i)}{\sum_{i=1}^{N}(y_i - \hat{y}_i)} \tag{1}$$

While the second one is computed as follows:

$$R^2 = 1 - \frac{\hat{L}(M_{full})}{\hat{L}(M_{intercept})} \tag{2}$$

Where the $\hat{L}(M_{intercept})$ in McFadden's formula refers to the log-likelihood of the intercept model, which is the simplest model that one can think in glm where the dependent variable doesn't depend on any predictor from the dataset. In contrast, $\hat{L}(M_{full})$ refers to the log-likelihood of the full model (in this case, the one that makes use of the variables that best explain the data). Efron's pseudo R-squared can be thought as a way to measure the variability that our model can explain by adding the variables. The more variability explained, the better the model. The dependent variables in a logistic regression, however, are not continuous and the predicted value (a probability) are, so you have to bear that in mind.
Meanwhile, McFadden's pseudo R^2 aims to understand the improvement from the null model to the fitted model. The ratio is indicative of the degree to which the model parameters improve upon the prediction of the null model. The ratio of the likelihoods suggests the level of improvement over the intercept model offered by the full model. Thus, a small ratio of log likelihoods indicates that the full model is a far better fit than the intercept model, and our R^2 will be closer to 1.
With all this in mind, if we were asked to choose one of these methods to obtain results for your applied research paper, the decision would depend on the resulting pseudo R^2 and our underlying interest. In this case, LASSO-BIC seems to be the best candidate as it is the one that seems to explain better the true model (according to McFadden), and it also resulted in a high Efron's Pseudo R^2 (just 0.0005 below MLE).

II. CODE:

```
#Logistic regression

fit.mle= glm(y ~ x[,-1], family = "binomial")
b.mle= coef(fit.mle)
summary(fit.mle)
cor(fit.mle$fitted.values,y)^2 #In Sample Efron's Pseudo-R^2=0.7726014

cv.mle= kfoldCV.glm(y=y,x=data.frame(x[,-1]),K=10,seed=1, family = "binomial",
predict_type = "response")
r2.mle= cor(cv.mle$pred, y)^2
r2.mle #10-fold cross-validated Efron's Pseudo-R^2 = 0.7526937

logLik_fit = sum(log(cv.mle$pred**(y)*(1-cv.mle$pred)**(1-y))) #Log likelihood of the fitted model
logLik_null = sum(log((sum(y)/length(y))**(y)*(1-sum(y)/length(y))**(1-y))) #LogLik of Null model
# where we don't take into account any covariate --> p = spam/emails (independently of any x)
r2_McFadden.mle = 1-logLik_fit/logLik_null # McFadden Pseudo-r2 = 0.62863

#Lasso
fit.lasso = cv.glmnet(x=x[,-1], y=y, nfolds=10, family = "binomial")
fit.lasso #Optimal lambda = 0.000335 --> Nonzero = 54 --> Index 69 of the lambdas tried
length(fit.lasso$lambda) #91 lambdas tried
fit.lasso$lambda.min #Optimal lambda = 0.000335
b.lasso= as.vector(coef(fit.lasso, s='lambda.min'))
rownames(coef(fit.lasso, s='lambda.min'))
round(b.lasso, 3)

rownames(coef(fit.lasso, s = 'lambda.min'))[coef(fit.lasso, s = 'lambda.min')[,1]!= 0]
# returns names of nonzero coefs
coef(fit.lasso, s = 'lambda.min')[coef(fit.lasso, s = 'lambda.min')[,1]!= 0]
# returns nonzero coefs (Takes also the intercept --> 55)

lasso_cv.mle = kfoldCV.lasso.glm(y=y,x=x[,-1], K=10, seed=1, family ="binomial",
predict_type = "response")
lasso_r2.mle= cor(lasso_cv.mle$pred, y)^2
lasso_r2.mle #10-fold cross-validated Efron's Pseudo-R^2 = 0.7351376

logLik_fit = sum(log(lasso_cv.mle$pred**(y)*(1-lasso_cv.mle$pred)**(1-y)))
#Log likelihood of the fitted model
```

```
logLik_null = sum(log((sum(y)/length(y))**(y)*(1-sum(y)/length(y))**(1-y)))
#LogLik of Null model where we don’t take into account any covariate
#p = spam/emails (independently of any x)
lasso_r2_sMcFadden.mle = 1-logLik_fit/logLik_null # McFadden Pseudo-r2 = 0.6417641

#Lasso Bic
fit.lassobic = lasso.bic.glm(x=x[,-1], y=y, family = "binomial", extended = FALSE)
fit.lassobic #Optimal lambda = 0.000335 --> Nonzero = 55 --> Index 69 of the lambdas tried
length(fit.lassobic$lambda[['lambda']]) #91 lambdas tried
fit.lassobic$lambda.opt #Optimal lambda = 0.0001591292
b.lassobic= as.vector(fit.lassobic[["coef"]])
rownames(coef(fit.lasso, s='lambda.min'))
round(b.lasso, 3)

lassobic_cv.mle = kfoldCV.lasso.glm(y=y,x=x[,-1], K=10, seed=1, family="binomial",
predict_type = "response", criterion="bic")
lassobic_r2.mle= cor(lassobic_cv.mle$pred, y)^2
lassobic_r2.mle #10-fold cross-validated Efron’s Pseudo-R^2 = 0.7521989

logLik_fit = sum(log(lassobic_cv.mle$pred**(y)*(1-lassobic_cv.mle$pred)**(1-y)))
#Log likelihood of the fitted model
logLik_null = sum(log((sum(y)/length(y))**(y)*(1-sum(y)/length(y))**(1-y)))
#LogLik of Null model where we don’t take into account any covariate
lassobic_r2_McFadden.mle = 1-logLik_fit/logLik_null # McFadden Pseudo-r2 = 0.643267
```