Regularized Logistic Regression

Anson Lai

30/11/2020

```
library(tidymodels)
library(tidyverse)
source("functions.R")
source("make_tidy.R")
```

Introduction

In this Vignette I will be fitting a logistic lasso model using the iterative re-weighted least squares algorithm along with coordinate descent that I have defined in the other two R script files in files/resources panel. I will need to tune the model and choose the correct regularization or penalty. But first I need to load some data to work with.

```
## Rows: 751
## Columns: 5
## $ x     <dbl> -1.703256, -1.699828, -1.696400, -1.692972, -1.689544, -1.686...
## $ w     <dbl> -1.405183, -1.404931, -1.404176, -1.402919, -1.401160, -1.398...
## $ y     <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1...
## $ cat_b <dbl> -0.7038149, 1.4189361, -0.7038149, 1.4189361, -0.7038149, -0....
## $ cat_c <dbl> -0.7229313, -0.7229313, 1.3814153, -0.7229313, 1.3814153, -0....
```

This data has binary response variable Y which is a bunch of 0 and 1's and 4 predictors, x, w, cat_b, cat_c. I have normalized the covariates since it is important for lasso regularization and left the response alone since it is binary so we don't want to normalize y. The recipe is also saved to use in the next step to fit our model.

Fitting logistic lasso with Lambda = 0.3

I have created training data and testing data so now I will fit the training data using the fitting algorithm I have defined. The algorithm basically uses coordinate descent to minimize the objective function which is a quadratic with the L1 penalty term. After each iteration, the weights and working response are updated so we can run coordinate descent again and again until we converge. I will now fit the logistic lasso model using a penalty of 0.3 to illustrate a point.

```
##          x          w      cat_b      cat_c
## -0.2040814  0.0000000  0.0000000  0.0000000
```

```
##    Intercept
## -0.008085722
```

The above output is one logistic lasso model where the coefficients are calculated by the algorithm but how do we know that 0.3 is the correct regularization parameter? As we can see some coefficients have been dropped to 0 and this is expected since the lasso method will bring more coefficients to 0 when we increase the penalty. The higher the penalty, the smaller the algorithm needs to make beta because it needs to make up for the high penalty. We can easily have made the penalty 0.9 and all the coefficients will go to 0 but this doesn't mean the covariates are useless, we just essentially told the model that the covariates are not that important by setting a high parameter so obviously it will go to 0.

So we need a penalty value that will minimize the cross validation prediction error rate but at the same time the penalty should simplify our model the best it can. So we need to the best of two worlds, simple and good accuracy. Some variables in this data may not actually be that important in classifying. They might just be there as extra noise where the other variables are very significant in classifying. The lasso regularization will take care of this by dropping coefficients to 0. This is particular useful in large data sets since most likely, only a few variables will matter.

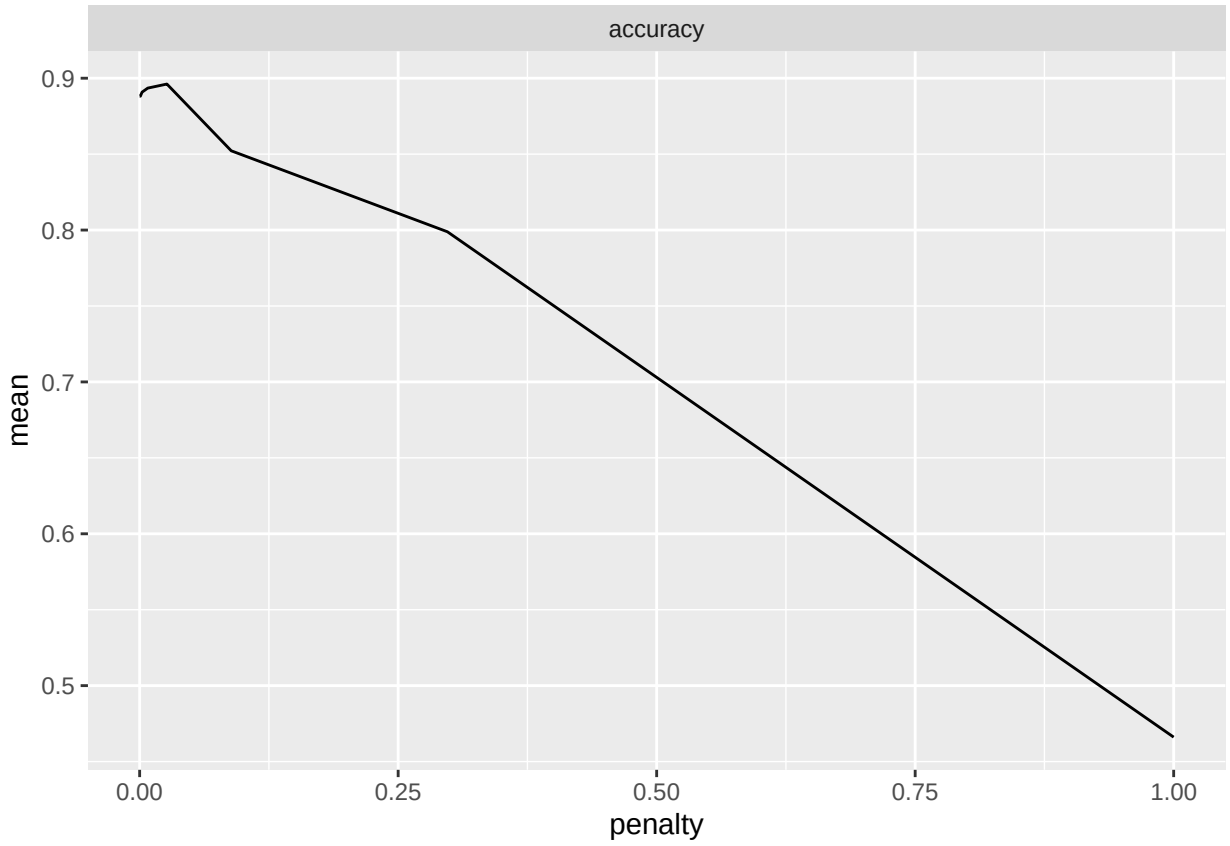
So we need to choose the most optimal penalty by tuning our model so we can do at least as good a job fitting the data as a more complicated model. We will have around the same accuracy as a more complicated model and at the same time be simpler. This is important because this will reduce our chance of over fitting the training data.

```
##            Truth
## Prediction   0   1
##          0 104  25
##          1  22  98
```

This is the confusion matrix for the model we have just fitted with lambda = 0.3. As you can see, although the coefficients drop to 0 and the model is simple, the accuracy is not that good. We have failed to predict around 50 points.

Finally, lets actually tune our model to find the best of two worlds.

Tuning for Lambda



To tune lambda, I have used accuracy as the metric which is basically the proportion predicted correctly. This is a good metric in this case since our data is pretty balanced. As it can be seen by the confusion matrix that the numbers are around the same. By the graph above, the optimal lambda that will have great accuracy seems to be around 0.02 and if we keep increasing lambda, our accuracy will greatly decrease linearly.

Our data set is very small, so the optimal lambda being small is normal since if its bigger all the coefficients will be 0. This process basically works for larger data sets as well and it will be better but for convenience and to just illustrate a point, this data set will do.

Now I have finalized the workflow and chose the best lambda.

Lets fit it on the training data and see if get an improvement from the first result we got when we only tried lambda = 0.3.

Final Model

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_lasso()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
```

```
## * step_dummy()
## * step_zv()
## * step_normalize()
##
## -- Model ---------------------------------------------------------------------
## $intercept
##   Intercept
## -0.04221762
##
## $beta
##         x         w     cat_b     cat_c
## -2.305868  1.165577  0.000000  0.000000
##
## $lambda
## [1] 0.02636651
##
## $fct_levels
## [1] "0" "1"
```

```
##           Truth
## Prediction   0   1
##          0 114  19
##          1  12 104
```

Our final model will be two predictors, x and w. X has coefficient -2.3 and W has coefficient 1.16 as seen above. The other two predictors have gone to 0 which is good because we want a simple model.

The important thing is, our accuracy has now improved. We only have around 30 predictions that are wrong when we have around 50 before. Comparing the confusion matrix now to before (lambda = 0.3), this is an improvement.

If we compare this to tutorial 9, this is a much simpler model but at the same time have around the same accuracy. The model in tutorial 9 has coefficients of cat_b and cat_c not 0. We have those coefficients 0 but also same prediction accuracy. This may be evidence that the cat_b and cat_c predictors are not that useful in predicting the general case.

Conclusion

The logistic lasso is a good classification technique for large data sets. The data in this vignette was small with only 4 predictors and we see good results. If it was for larger data sets, we would expect many coefficients to drop to 0 and also have a model with good accuracy. This will prevent over fitting the training data and help us make good prediction out of sample.