Vignette for Logistic Lasso Regression Model

Zhixing Hong

12/1/2020

Setting up

First we need to load the required functions and packages.

```
library(tidyverse)
library(tidymodels)
source("make_tidy.R")
set.seed(77777)
```

Simulating data

To preform the logistic lasso regression model, we need to stimulate the data for the regression first. We created the dataset as following: `y` is the response variable, and the other variables are predictors. Especialy, `x2` is designed to have no relationship with y.

```
## # A tibble: 6 x 5
##      x1    x2     w y     cat
##   <dbl> <dbl> <dbl> <fct> <chr>
## 1 -3    -2    -3    1     b
## 2 -2.99 -2.00 -3.00 1     c
## 3 -2.99 -1.99 -3.00 1     b
## 4 -2.98 -1.99 -3.00 1     c
## 5 -2.98 -1.98 -2.99 1     c
## 6 -2.97 -1.98 -2.99 1     a
```

We also split the dateset into training and testing datasets, with ratio 3:1. To fitting in the logistic regression model build in parsnip style, the recipe for the data is also created.

Fitting the model

Here we are going to use the training data to train the logistic lasso regression model. We first tried $\lambda = 1$ as the penalty term. The coefficients for the predictors are shown in the below table. In order to check whether we have overfitting problems for the model, we also run the model with the test dataset. The stats from confusion table shows that the model preformed fairly good.

```
lambda = 1
spec <- log_lasso(penalty=lambda) %>% set_engine("fit_logistic_lasso")

fit <- workflow() %>% add_recipe(rec) %>% add_model(spec) %>% fit(train)
fit$fit$fit$fit$bet
```

```
##     intercept            x1            x2             w         cat_b
##  8.549376e-17 -3.503763e+00 -7.557897e-16  2.070255e+00  2.361610e-01
```

```
##         cat_c
##  2.329479e-01
```

```
predict(fit, new_data = test) %>% bind_cols(test %>% select(y)) %>%
  conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 109  16
##          1  12 112
```

Here is another example for using the logistic lasso regression model. In this example, we used $\lambda = 0.005$. Similarly, the coefficients for the predictors and the confusion table for the test dataset are shown as below.

```
lambda = 0.005
spec <- log_lasso(penalty=lambda) %>% set_engine("fit_logistic_lasso")

fit <- workflow() %>% add_recipe(rec) %>% add_model(spec) %>% fit(train)
fit$fit$fit$fit$beta
```

```
##     intercept            x1            x2             w         cat_b
##  1.359729e-16 -3.640659e+00  0.000000e+00  2.163744e+00  2.752211e-01
##         cat_c
##  2.709903e-01
```

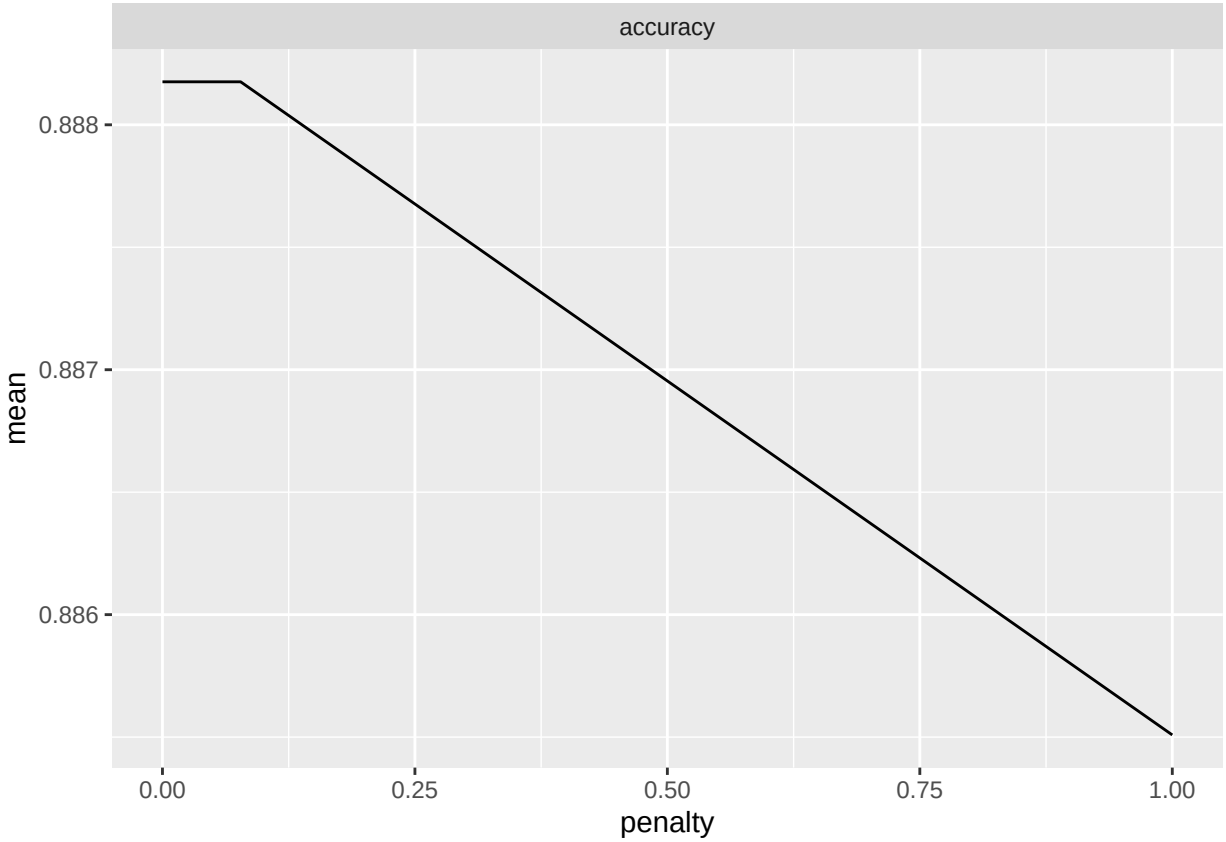
```
predict(fit, new_data = test) %>% bind_cols(test %>% select(y)) %>%
  conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 109  16
##          1  12 112
```

Tune the model

So here we are trying to tune the model. From the graph, the accuracy of prediction is decreasing as penalty term getting larger. To make the model predict more precise, we find the $\lambda = 10^{-10}$ could give the most accurate result.

```
grid <- grid_regular(penalty(), levels = 10)

spec_tune <- log_lasso(penalty = tune()) %>% set_engine("fit_logistic_lasso")

wf <- workflow() %>% add_recipe(rec) %>% add_model(spec_tune)

folds <- vfold_cv(train)

fit_tune <- wf %>%tune_grid(resamples = folds, grid = grid, metrics = metric_set(accuracy))

fit_tune %>% collect_metrics() %>% ggplot(aes(penalty, mean)) + geom_line() + facet_wrap(~.metric)
```




```
penalty_final <- fit_tune %>% select_best(metric = "accuracy")
```

Compare the result with logistic_reg()

By comparing the result from two model, we noticed that there are no different prediction of the data. Therefore, `log_lasso` regression model is safe to use with a specified penalty term.

```
lambda = penalty_final$penalty

spec <- log_lasso(penalty=lambda) %>% set_engine("fit_logistic_lasso")

fit <- workflow() %>% add_recipe(rec) %>% add_model(spec) %>% fit(train)

ddat<- rec %>% prep(train) %>% juice

ff = logistic_reg(penalty = lambda, mixture = 1) %>% # mixture = 1 meaning no L2 penalty
  set_mode("classification") %>%
  set_engine("glm") %>%
  fit(y ~ -1+., family = "binomial", data = ddat)

ff %>% tidy %>% select(term, estimate) %>%
  mutate(log_lasso_estimate = fit$fit$fit$fit$beta, err = estimate - log_lasso_estimate)
```

```
## # A tibble: 6 x 4
##   term      estimate log_lasso_estimate        err
##   <chr>        <dbl>              <dbl>      <dbl>
## 1 intercept    0.178           0.        0.178
```

```
## 2 x1          -3.64           -3.64e+ 0 -0.000514
## 3 x2          NA              -1.66e-15 NA
## 4 w            2.16            2.16e+ 0  0.000297
## 5 cat_b        0.275           2.75e- 1  0.0000439
## 6 cat_c        0.271           2.71e- 1  0.0000374
```

```
test_dat <- rec %>% prep(train) %>% bake(test)
glm_pred <- (predict(ff, test_dat, type = "prob") > 0.5) %>% as.numeric
preds <- predict(fit, new_data=test)
any(preds$.pred_class != preds$glm_pred)
```

```
## [1] FALSE
```