STA314 Assignment 2: Using a logistic Lasso model (with coordinate descent) to perform classifications on data by tidymodels workflow

Ruo Ning Qiu

04 December, 2020

```
# Before we begin, let's set the seed to be a student number so it produces the same result
# every time.
set.seed(1004079631)
```

This vignette will demonstrate how we perform classification with on data using **tidymodels** workflow with the helps of 2 functions in functions.R, `fit_logistic_lasso()`, `predict_logistic_lasso()` to set up the logistic Lasso model using coordinate descent on the the penalized iteratively reweighted least squares algorithm with the help of make_tidy.R to set it up using **parsnip**

We need the following library packages to run the functions, make sure you have installed them!

Note: It might be the case that make_tidy.R requires the newest package of `tidymodels`. If there is a error message that says: Error: The values passed to `set_encoding()` had extra arguments: ‘allow_sparse_x’, try to type the following code to install the newest version of `tidymodels` and `tidymodels`: install.packages("tidymodels"), install.packages("tidyverse")

```
# This also checks if the required libraries are installed while loading the functions.
# If not, please follow the return error message and install the required libraries.
source("functions.R")
source("make_tidy.R")
# load the libraries that we need, should be installed at this stage by previous check
library(tidyverse)
library(tidymodels)
```

```
# Before we begin, set the seed to be my student number so it
# produces the same result every time.
set.seed(1004079631)
```

Using a tidymodels workflow, fit a Logistic Regression Lasso with the penalized iteratively reweighted least squares algorithm to this data and by inputting the penalty λ.

Let’s generate some data to fit our new model:

```
# Generate some data (same as week9 tutorial)
n = 1000
dat <- tibble(x = seq(-3,3, length.out = n),
              w = 3*cos(3*seq(-pi,pi, length.out = n)),
              y = rbinom(n,size = 1, prob = 1/(1 + exp(-w+2*x)) )%>% as.numeric %>% factor,
              cat = sample(c("a","b","c"), n, replace = TRUE)
)
split <- initial_split(dat, strata = c("cat"))
train <- training(split)
test <- testing(split)
rec <- recipe(y ~ . , data = train) %>%
  step_dummy(all_nominal(), -y) %>% step_zv(all_outcomes()) %>%
  step_normalize(all_numeric(), -y) %>% # don't normalize y!
  step_intercept() ## This is always last!
```

The lasso linear regression model with penalized penalized iteratively reweighted least squares algorithm is finished setting up by make_tidy.R. We just need to use it (`logistic_lasso()`) with package tidymodels's help.Using a tidymodels workflow, we can fit a Logistic Regression Lasso with the penalized iteratively reweighted least squares algorithm to this data and by inputting the penalty $\lambda = 0.001$

```
lambda = 0.001
spec <- logistic_lasso(penalty = lambda) %>% set_engine("fit_logistic_lasso")
# here fit_logistic_lasso is defined in functions.R, if you are interested in checking it out

# set up the workflow
fit <- workflow() %>% add_recipe(rec) %>% add_model(spec) %>% fit(train)

# check the prediction matrix
predict(fit, new_data = test) %>% bind_cols(test %>% select(y)) %>%
conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 102  16
##          1  13 118
```

Not bad! We can also compare our answer with the `glm` engine:

```
ddat <- rec %>% prep(train) %>% juice
ff <- logistic_reg(penalty = lambda, mixture = 1) %>%
  set_mode("classification") %>%
  set_engine("glm") %>%
  fit(y ~ ., family="binomial", dat = ddat)
compare = ff %>% tidy %>% select(term, estimate) %>%
          mutate(IRLS_estimate = c(fit$fit$fit$fit$beta),
                 err = estimate - IRLS_estimate)
compare
```

```
## # A tibble: 6 x 4
##   term        estimate IRLS_estimate      err
##   <chr>          <dbl>         <dbl>    <dbl>
## 1 (Intercept)  -0.0261      -0.00569 -0.0204
## 2 intercept    NA           -0.0216   NA
## 3 x            -3.27        -3.23    -0.0382
## 4 w             2.16         2.13     0.0297
## 5 cat_b         0.0360       0.0301   0.00585
## 6 cat_c        -0.117       -0.112   -0.00473
```

We can see that the errors are pretty smalls (estimate is what our new model produces). Note that this is because we are penalizing the coefficients with small λ. If λ is big like what assignment had for unit tests, the error will be greater since we are penalizing more!

Using a tidymodels workflow, fit a Logistic Regression Lasso with the penalized iteratively reweighted least squares algorithm to this data and tune the penalty λ.

Let's get the recipe for workflow to prepare the model with tuning penalty this time! Since we do not know the penalty choice for the lasso linear regression model with penalized penalized iteratively reweighted least squares algorithm, let's generate a lambda grid and do cross validation to choose the penalty λ.

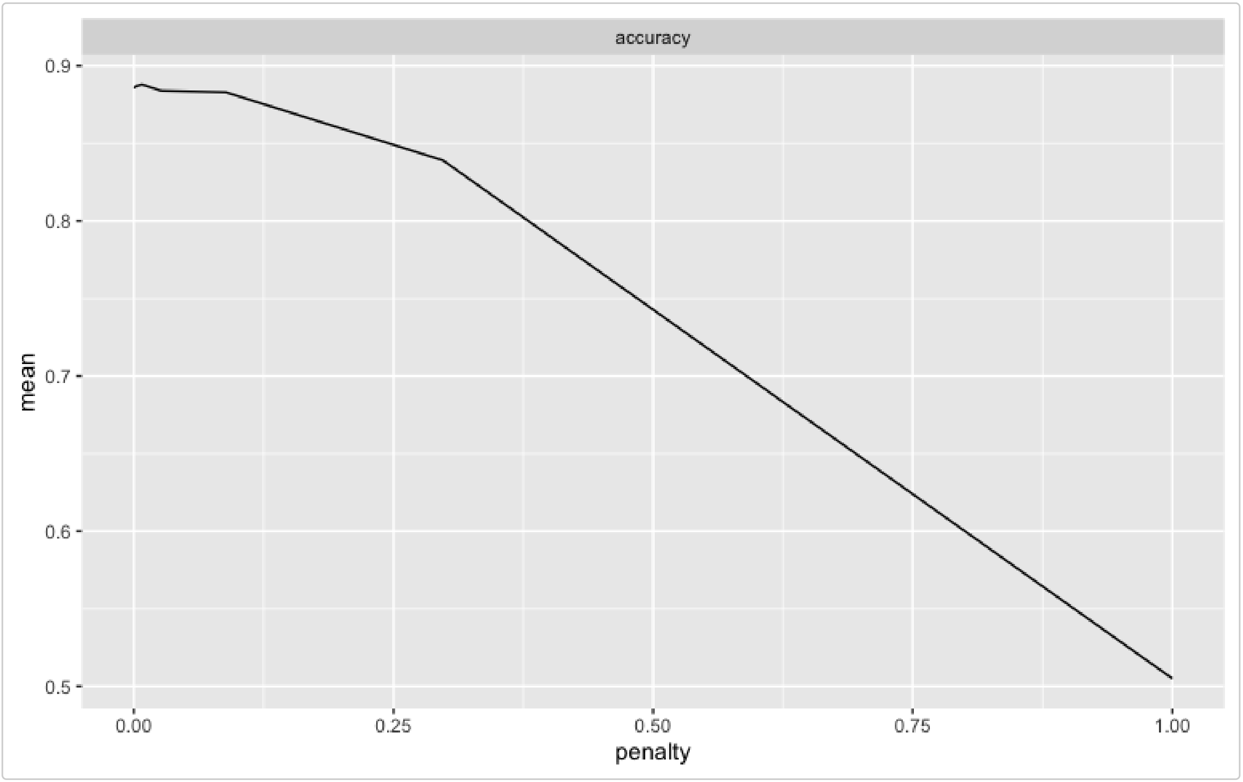
```
lasso <- logistic_lasso(penalty = tune()) %>% set_engine("fit_logistic_lasso")
# use workflow
wf_lasso <- workflow() %>% add_recipe(rec)

# we want to pick a proper penalty, first generate the lambda_grid, candidates for penalty
lambda_grid <- grid_regular(penalty(), levels = 20)
```

```
# generate folds for cross validation, data is split into <v> groups of roughly equal sizes
folds <- vfold_cv(dat, v = 10, strata = y)

# tune our model to find a good value of lambda
wf_lasso <- wf_lasso %>% add_model(lasso)
fit_tune <- tune_grid(wf_lasso, resamples = folds, grid = lambda_grid,
                       metrics = metric_set(accuracy))

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



Let’s select by picking the model that a good accuracy value as the metric for the penalty (using select_best() function) to finalize the model, and check the result. THe accuracy decreases and penalty (lamada) increases, which makes sense as we are penalizing more.

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

# finalize the model (used function in make_tidy.R)
wf_final <- wf_lasso %>% finalize_workflow(penalty_final)
wf_final
```

```
## ══ Workflow ════════════════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: logistic_lasso()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────
## 4 Recipe Steps
##
## • step_dummy()
## • step_zv()
## • step_normalize()
## • step_intercept()
##
## ── Model ───────────────────────────────────────────────────────────────────────
## Model Specification (classification)
##
## Main Arguments:
##   penalty = 0.00784759970351461
```

```
##
## Computational engine: fit_logistic_lasso
```

Penalty is chossen; now it is time to check the confusion matrix!

```
final_fit <- wf_final %>% fit(train)
# check the prediction matrix
predict(final_fit, new_data = test) %>% bind_cols(test %>% select(y)) %>%
conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 102  16
##          1  13 118
```

Let's compare with `glm` engine again with the new finalized penalty for λ.

```
ddat <- rec %>% prep(train) %>% juice
ff <- logistic_reg(penalty = penalty_final$penalty, mixture = 1) %>%
  set_mode("classification") %>%
  set_engine("glm") %>%
  fit(y ~ ., family="binomial", dat = ddat)
compare = ff %>% tidy %>% select(term, estimate) %>%
          mutate(IRLS_estimate = c(final_fit$fit$fit$fit$beta),
                 err = estimate - IRLS_estimate)
compare
```

```
## # A tibble: 6 x 4
##   term        estimate IRLS_estimate      err
##   <chr>          <dbl>         <dbl>    <dbl>
## 1 (Intercept)  -0.0261       -0.0341  0.00807
## 2 intercept    NA             0      NA
## 3 x            -3.27         -3.00   -0.271
## 4 w             2.16          1.95    0.211
## 5 cat_b         0.0360        0       0.0360
## 6 cat_c        -0.117        -0.0794 -0.0375
```

Since the tuned penalty ≈ 0.0078476 which is greater than the previous penalty $= 0.001$, as mentioned earlier, the error would be greater since we are penalizing more.

This is the end of the tutorial for how to using a logistic Lasso model (with coordinate descent) to perform classifications on data by tidymodels workflow. Thanks for bearing with it and have a good winter break!