Recitation 10 imputation, ridge, and lasso

Seamus Wagner

November 1, 2021

Lab overview

I cannot remember how far we got through in the demo. It seemed as though it was a bit rushed, so we will go over some of the code here. The goal here will be to go through line-by-line to make sure you all understand why we are calling each thing and what they are doing. To do this, we will do it outside of a function and then as a function.

```
rm(list=ls())
set.seed(717933232)
n_groups = 10
n_obs = 1000
# reminder to go over auto-organizing code chunks
group <- sample(n_groups, n_obs, replace = TRUE)

# Why are we doing this?
X <- model.matrix(~ factor(group))

n_groups <- ncol(X)

mu_beta = 0
sigma_beta = 1
# Why are we doing this
beta <- rnorm(n_groups, mu_beta, sigma_beta)
beta <- cbind(beta)
beta
```

```
##              beta
##  [1,]  0.19358945
##  [2,]  0.88918069
##  [3,] -0.83732238
##  [4,] -0.34173113
##  [5,] -1.52242792
##  [6,]  0.97699805
##  [7,]  0.62499521
##  [8,]  0.79874898
##  [9,]  0.05844331
## [10,]  0.44635154
```

```
# What is this bit doing?
sigma_upsilon = .1
```

```
data <- data.table(group, upsilon = rnorm(n_obs, 0, sigma_upsilon))
data[, epsilon := X %*% beta + upsilon]
data
```

```
##       group     upsilon    epsilon
##    1:     3 -0.02747576 -0.6712087
##    2:     9 -0.13231055  0.1197222
##    3:     6  0.02424474  1.1948322
##    4:     4  0.02107268 -0.1270690
##    5:     9  0.14511369  0.3971464
##   ---
##  996:     2 -0.15205745  0.9307127
##  997:     2 -0.03198083  1.0507893
##  998:     9 -0.05327459  0.1987582
##  999:     6  0.02273329  1.1933208
## 1000:     7  0.23796271  1.0565474
```

```
mean_group_ate = 0
sd_group_ate = 1
group_ate <- cbind(rnorm(n_groups, mean_group_ate, sd_group_ate))

alpha = .3
data[, `:=`(Y0 = alpha + epsilon,
    Y1 = alpha + X %*% group_ate + epsilon # <- and here
  )]

mu_beta_selection = 0
sigma_beta_selection = 1
# Why are we adding this?
beta_selection <- cbind(rnorm(n_groups, mu_beta_selection, sigma_beta_selection))
beta_selection
```

```
##             [,1]
##  [1,] -1.3879861
##  [2,]  0.8441334
##  [3,]  1.1307211
##  [4,]  0.2069393
##  [5,]  0.1412494
##  [6,] -0.3087460
##  [7,]  0.1937443
##  [8,]  0.5421652
##  [9,] -0.4732922
## [10,] -0.2292644
```

```
# What is prob_D unit of analysis?
data[, prob_D := plogis(X %*% beta_selection)]
data[, D := rbinom(.N, 1, prob_D)]
  data[, Y := D * Y1 + (1 - D) * Y0]
data
```

```
##       group     upsilon    epsilon         Y0          Y1    prob_D D
##    1:     3 -0.02747576 -0.6712087 -0.3712087 -1.99216893 0.4360362 1
##    2:     9 -0.13231055  0.1197222  0.4197222 -0.13006525 0.1345541 0
##    3:     6  0.02424474  1.1948322  1.4948322  3.06338285 0.1548926 0
##    4:     4  0.02107268 -0.1270690  0.1729310  0.21369600 0.2348640 0
##    5:     9  0.14511369  0.3971464  0.6971464  0.14735899 0.1345541 0
##   ---
##  996:     2 -0.15205745  0.9307127  1.2307127  0.31363447 0.3672918 0
##  997:     2 -0.03198083  1.0507893  1.3507893  0.43371109 0.3672918 0
##  998:     9 -0.05327459  0.1987582  0.4987582 -0.05102929 0.1345541 0
##  999:     6  0.02273329  1.1933208  1.4933208  3.06187140 0.1548926 0
## 1000:     7  0.23796271  1.0565474  1.3565474  1.78288499 0.2325012 0
##                 Y
##    1: -1.9921689
##    2:  0.4197222
##    3:  1.4948322
##    4:  0.1729310
##    5:  0.6971464
##   ---
##  996:  1.2307127
##  997:  1.3507893
##  998:  0.4987582
##  999:  1.4933208
## 1000:  1.3565474
```

```
# Why D?
coefficient_estimator <- coef(lm(Y ~ D + factor(group), data))["D"]
coefficient_estimator
```

```
##          D
## -0.3629397
```

```
ate <- data[, mean(Y1 - Y0)]
ate
```

```
## [1] -0.2570647
```

```
coefficient_bias <- abs(coefficient_estimator - ate)
coefficient_bias
```

```
##         D
## 0.1058749
```

Now that we did the basics line by line, let’s run it as a function

```
rm(list=ls())
set.seed(717933232)
make_regression_data <- function(n_obs, n_groups,
  mean_group_ate, sd_group_ate,
  alpha = 3, mu_beta = 0, sigma_beta = 1, sigma_upsilon = .1,
  mu_beta_selection = 0, sigma_beta_selection = 1) #This is more inputs than usual, this is where defin
{
  group <- sample(n_groups, n_obs, replace = TRUE)
```

```r
  X <- model.matrix(~ factor(group))
  n_groups <- ncol(X)
  beta <- rnorm(n_groups, mu_beta, sigma_beta)
  beta <- cbind(beta)
  data <- data.table(group, upsilon = rnorm(n_obs, 0, sigma_upsilon))
  data[, epsilon := X %*% beta + upsilon]
  group_ate <- cbind(rnorm(n_groups, mean_group_ate, sd_group_ate))
  data[, `:=`(
    Y0 = alpha + epsilon,
    Y1 = alpha + X %*% group_ate + epsilon
  )]
  beta_selection <- cbind(rnorm(n_groups, mu_beta_selection, sigma_beta_selection))
  data[, prob_D := plogis(X %*% beta_selection)]
  data[, D := rbinom(.N, 1, prob_D)]
  data[, Y := D * Y1 + (1 - D) * Y0]
  return(data)
}

set.seed(717933232)
data <- make_regression_data(
  n_obs = 1000,
  n_groups = 10,
  mean_group_ate = 0,
  sd_group_ate = 1)
coefficient_estimator <- coef(lm(Y ~ D + factor(group), data))["D"]
ate <- data[, mean(Y1 - Y0)]
coefficient_bias <- abs(coefficient_estimator - ate)
coefficient_bias
```

```
##         D
## 0.1058749
```

```r
# so the regression coefficient is biased in these conditions, and we added bias.. so this is good to c
```

Now we will move to the imputation estimator. Does anyone remember what it actually is?

- Imputation estimator: we interact treatment with all X's, then generate individual $\hat{Y}0(x)$ and $\hat{Y}1(x)$
- The we average across all and subtract $\hat{Y}1(x)$ by $\hat{Y}0(x)$

```r
# the imputation estimator ----
# so far we've used regression to adjust.
# but we could use regression to estimate E[Y_1 | D] and E[Y_0 | D] directly,
# and then subtract and average.
model <- lm(Y ~ D * factor(group), data)
data_D1 <- copy(data)
data_D1[, D := 1]
data_D0 <- copy(data)
data_D0[, D := 0]
yhat_D1 <- predict(model, newdata = data_D1)
yhat_D0 <- predict(model, newdata = data_D0)
# What have we done so far, and why?
imputation_estimator <- mean(yhat_D1 - yhat_D0)
```

```r
imputation_bias <- abs(imputation_estimator - ate)
imputation_bias
```

```
## [1] 0.004735689
```

```r
# William mentions this is misleading, why?
```

Penalized techniques

Why would we want to penalize? What options are available to us so far?

Ridge Overview

Ridge regression squares the coefficients and adds the resulting sum as a penalty, scaled by a λ. This λ is chosen by cross-validation. High values mean we weighting the betas to almost nothing (high punishment for inclusion). Low values return the basic linear model as there is almost no penalty. Variables are typically standardized when they go into a ridge regression model, glmnet does this for you for the Y variables when you call gaussian in thr family = portion. glmnet has a stadardize = T/F call within the function, default is true. This is for all X variables. A reminder for standardization (z-transformation here) is the you subtract the mean and divide by the standard deviation for each value.

Bias and variance: bias increases as λ increases. Variance decreases as λ increases.

LASSO regression

Useful for if we want to include (not) a variable. Think of uses you may want here. Essentially it removes variables that are not good for prediction, think about this carefully for causality. This is a useful tool for prediction, but you can lose some of the theoretical reasons for what is included in the model quickly, as we saw when it removes the treatment itself. These methods require thinking, which some people do not consider when using them.

Elastic nets

As with most things, some method that combines them may be most useful. So rather than limiting to include everything but weight it or to remove everything that doesn't meet a cutoff, we could drop really bad variables and weight others according to their contribution and outlierness(?).

How to pick the alpha here, cross-validate (same intuition as bootstrapping as the solution to everything for this). The reason we do this is because the data we input into the model may not be super ideal, so giving it the most options to learn is best.

From here, we will turn to the code in the lab, since William ended right after the formula pasting and basic LASSO/ridge. Excluding the rest of lab, up through part 10 has been covered.