simulation_scratchwork

Thu Pham

2022-11-04

```
y1_child_obs_raw <- read.dta13("y1o_c_long.dta")
y1_teacher_obs_raw <- read.dta13("y1o_t_long.dta")
```

```
## Warning in read.dta13("y1o_t_long.dta"):
##    Factor codes of type double or float detected in variables
##
##    o_t_verbal, o_t_verbal_l, o_t_verbal_t,
##    o_t_whom, o_t_listen_child, o_t_talk_child,
##    o_t_sched, o_t_task, o_t_focus, o_t_tone,
##    o_t_attention, o_t_es
##
##    No labels have been assigned.
##    Set option 'nonint.factors = TRUE' to assign labels anyway.
```

```
y1_coverpage_obs <- read.dta13("y1o_coverpage.dta")
```

```
length(unique(unique(y1_child_obs_raw[, "provid"])))
```

```
## [1] 451
```

```
length(unique(unique(y1_child_obs_raw[, "classid"])))
```

```
## [1] 672
```

```
# looks like there are some providers with more than one class
# students nested in classes, classes nested in schools, schools nested in zip codes
```

```
# helper functions


# need to find a minimum lambda value that we can start from, since too small of a lambda value returns
min_lambda <- function(train_data, formula, rand=list(j=~1)) {
  failed <- TRUE
  lambda_min <- 0.01
  repeat {

  failed <- tryCatch(
      {
        glmmLasso(fix = formula, rnd = rand, data = train_data,
            lambda = lambda_min)
      },
      error = function(e){
          TRUE
      }
   )
```

```
    if (is.logical(failed)) {
      lambda_min <- lambda_min + 0.01
    }

    else {
      break
    }
  }
  return(lambda_min)
}
```

```
# cross validation helper function for the glmmLasso
# need to eventually do this with k-folds, or get as close to the
# cross validation that is happening in the regular lasso
cv_glmmLasso <- function(train_data, formula, criterion, rand=list(j=~1),
                         lambda_step=50) {

  BIC <- rep(NA, lambda_step)
  AIC <- rep(NA, lambda_step)
  pred_error <- rep(NA, lambda_step)
  coefficients <- NULL
  min_lambda <- min_lambda(train_data, formula)

  lambdas <- seq(from=min_lambda, to=100,
                 by=(100 - min_lambda) / lambda_step)

  for (j in 1:lambda_step) {
    # print(paste("Iteration", j))
    mixed_lasso <- glmmLasso(fix = formula, rnd = rand, data = train_data,
              lambda = lambdas[j])
    BIC[j] <- mixed_lasso$bic
    AIC[j] <- mixed_lasso$aic
    coefficients <- cbind(coefficients, mixed_lasso$coefficients)

    y_hat <- predict(mixed_lasso, train_data)
    pred_error[j] <- sum((y_hat - train_data$Y)^2)
  }

  if (criterion == "AIC") {
    return(lambdas[which.min(AIC)])
  }

  if (criterion == "BIC") {
    return(lambdas[which.min(BIC)])
  }

  return(lambdas[which.min(pred_error)])

}
```

```
# Generate
# m -- number of covariates, j -- number of clusters, sigma_r -- variance of
# different intercepts
generate_data <- function(m, j, sigma_r, n=100) {
```

```
  # j -- number of clusters
  colnames_df <- c(paste("X", 1:m, sep = ""), "random_beta")
  # covariance matrix
  # Sigma <- diag(j)
  # Sigma[outer(1:j, 1:j, function(x, y) x != y)] <- rho
  # generate j random betas (random slopes)
  random_beta <- rnorm(n=j, mean=0, sigma_r)
  df <- matrix(nrow=n)
  # generate X_j
  for (i in 1:m) {
    df <- cbind(df, rnorm(n))
  }

  # first column is NA for some reason, definitely need to clean this up later.
  df <- df[, -1]
  data <- as.data.frame(df)
  # make this more efficient later
  data <- cbind(data, rep(random_beta, each=n/j))
  colnames(data) <- colnames_df
  # error term
  epsilon <- rnorm(n)
  data$Y <- rowSums(data[, 1:(m + 1)]) + epsilon
  data$j <- as.factor(rep(1:j, each=n/j))
  return(data)
}
test <- generate_data(5, 2, 0.4, n=4)
```

```
# Analyze
analyze <- function(data, m) {
  # need to do cross validation for LASSO
  train_df <- data[1:(0.80 * nrow(data)), ]
  var.names <- paste("X", 1:m, sep = "")
  com.names <- lapply(seq_along(var.names),
                      function(i) combn(var.names, i, FUN = paste, collapse = " + "))
  covariates <- com.names[[m]]
  X_fixed <- model.matrix(as.formula(paste("Y ~", covariates)),
                          data = train_df)
  X_fixed <- X_fixed[, -1]
  Y <- train_df$Y

  # call cv.glmnet()
  model_lasso_fixed <- cv.glmnet(x = X_fixed, y = Y, alpha = 1)
  cc <- coef(model_lasso_fixed, s = model_lasso_fixed$lambda.min)
  #placeholder lambda value
  coefficients_fixed <- coef(model_lasso_fixed)


  lambda_min <- cv_glmmLasso(train_data = train_df,
                             formula = as.formula(paste("Y ~", covariates)),
                             criterion = "pred_error")
  model_lasso_mixed <- glmmLasso(fix = as.formula(paste("Y ~", covariates)),
                                                    rnd=list(j=~1),
                                                   data = train_df,
                                                    lambda=lambda_min)
```

```
  coefficients_mixed <- model_lasso_mixed$coefficients
  coefficients <- cbind(as.matrix(coefficients_fixed), as.matrix(coefficients_mixed))
  colnames(coefficients) <- c("fixed", "mixed")
  coefs_df <- as.data.frame(coefficients[2:nrow(coefficients), ])
  return(coefs_df)
}
result <- analyze(test, 5)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
# Repeat
one_run <- function() {
  dat <- generate_data(params)
  analyze(dat)
  return(0)
}
# results <- rerun(R, one_run())
# results <- bind_rows(results)

# Summarize
assess_performance <- function(results) {
  # stuff
  return(0)
}
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.