Super Learning (Q)

Josiah Davis

4/12/2017

Set up working environment

```
library(dplyr)
library(SuperLearner)
rm(list = ls()); gc()
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 418013 22.4     750400 40.1   592000 31.7
## Vcells 617367  4.8    1308461 10.0   807606  6.2
```

```
set.seed(5000)
```

Define convenience functions

```
get_data <- function(dir, file_in){
  # Get the HR data in data-frame format
  #
  # Args:
  #   dir: working directory as character vector
  #   file_in: name of the file as character vector
  #
  # Returns:
  #   dataframe object with no transformations

  df <- read.csv(paste0(dir, file_in), stringsAsFactors = FALSE)
  return(df)
}

bucket_covariate <- function(x, k){
  # Approximate a single quantitative variable based off of the quantiles
  #
  # Args:
  #   x: The quantitative vector to be approximated
  #   k: The number of quantiles to use in the approximation
  #
  # Returns:
  #   Numeric vector with numbers corresponding to quantiles (e.g., 1 = 1st quantile)

  x_split <- cut(x, breaks = quantile(x, probs = seq(0, 1, 1/k)), include.lowest = TRUE)
  if(any(is.na(x_split))) stop('There are NA values in')
  return(as.numeric(factor(x_split, labels = 1:k)))
}

format_data <- function(df){
  # Handle all custodial details relating to the formatting of the data
```

```r
# This incluseds appriximating covariates, and removing strata that
# violate the positivity assumption.
#
# Args:
#   df: observed data as a data-frame
#
# Returns:
#   Dataframe with variables ready for super learning

# Remove people with medium salary level
df <- df %>% filter(salary != 'medium')

# Approximate all five quantitative variables as dichotomous variables
df <- df %>%
  mutate(
    satisfaction_level = bucket_covariate(satisfaction_level, 2)
    , last_evaluation = bucket_covariate(last_evaluation, 2)
    , number_project = bucket_covariate(number_project, 2)
    , average_montly_hours = bucket_covariate(average_montly_hours, 2)
    , time_spend_company = bucket_covariate(time_spend_company, 2)
  )

# Check for violations of the positivity assumption
df <- df %>%
group_by(
  satisfaction_level
  , last_evaluation
  , number_project
  , average_montly_hours
  , time_spend_company
  , Work_accident
  , promotion_last_5years
  , sales
) %>%
mutate(
  both_treatments = (sum(salary == 'high') > 0) & (sum(salary == 'low') > 0)
  , salary = as.numeric(ifelse(salary == 'high', 1, 0))
) %>% ungroup()

# Count up and print number of removed observations
n_obs <- df %>% nrow()
n_dropped_obs <- df %>% filter(!both_treatments) %>% nrow()
cat('Number of positivity violations ', n_dropped_obs
    , ' (', round(100*n_dropped_obs/n_obs, 2), '%)', sep = '')

# Drop observations from sample with positivity assumption violations
df <- df %>% filter(both_treatments)

# Generate dummy variables
mm <- data.frame(model.matrix( ~ sales - 1, data = df))
# Treat technical department as reference level
mm <- mm[,!(colnames(mm) %in% 'salestechnical')]
df <- cbind(df, mm)
```

```
  # Drop columns no longer needed
  df <- df %>% dplyr::select(-both_treatments, -sales)

  # Create attribute to store columns
  attr(df, 'x_cols') <- df %>% dplyr::select(-left) %>% colnames()

  return(df)
}

get_super_learner <- function(df, learning_library){
  # Train SuperLearner and create counterfactual outcomes
  #
  # Args:
  #   df: observed data as a data-frame
  #   learning_library: character vector of libraries used for ensembling
  #
  # Returns:
  #   model object with predictions, and other results (e.g., cvRisk)


  # Set treatment to 0, 1 for generating the counterfactual outcomes
  X_0 <- df[,attr(df, 'x_cols')] %>% mutate(salary = 0)
  X_1 <- df[,attr(df, 'x_cols')] %>% mutate(salary = 1)

  # Run Super Learner
  model <- SuperLearner(Y = df$left
                        , X = df[,attr(df, 'x_cols')]
                        , newX = rbind(X_0, X_1)     # Note: this data is not used for training
                        , SL.library = learning_library
                        , cvControl = list(V = 5)
                        , family = 'binomial'
                        , verbose = FALSE)

  # Show some output
  print(model)
  run_time <- model$times$everything['elapsed'] %>% unname() / 60
  cat('Model training and predicting took', round(run_time, 2), 'minutes')

  return(model)
}

get_counterfact_outcomes <- function(df, model){
  # Retreive Y_a under each potential outcome from the super learner object
  #
  # Args:
  #   df: observed data as a data-frame
  #   model: observed data as a data-frame
  #
  # Returns:
  #   model object with predictions, and other results (e.g., cvRisk)


  #
```

```
  df$Y_0 <- model$SL.predict[1:nrow(df)]
  df$Y_1 <- model$SL.predict[(nrow(df)+1):(2*nrow(df))]

  return(df)
}

save_output <- function(df, model, dir, out_name, save_model = TRUE){
  # Write counterfactual outcomes, model summary results to local disk
  #
  # Args:
  #   df: observed data as a data-frame
  #   model: observed data as a data-frame
  #   dir: working directory as character vector
  #   out_name: name of output file with counterfactual outcomes as character vector
  #   save_model: Whether or not to save the full model as binary value
  #
  # Returns:
  #   Nothing to the R environment

  # 1. Write dataframe containing counterfactual outcomes to csv
  df %>% write.csv(paste0(dir, out_name), row.names = FALSE)

  # 2. Save some modeling results
  model_summary <- list(
    risk = model$cvRisk
    , coefficients = model$coef
    , times = model$times
    )
  saveRDS(model_summary, paste0(dir, 'SL_summary'))

  # 3. Save full (large) super learning object
  if(save_model) saveRDS(model, paste0(dir, 'SL_full.rds'))

}
```

Define Functions for Super Learning

```
# Get knn with different sizes
create_SL_knn <- function(k = c(20, 30)) {
  for(mm in seq(length(k))){
    eval(parse(text = paste('SL.knn.', k[mm], '<- function(..., k = ', k[mm],
      ') SL.knn(..., k = k)', sep = '')), envir = .GlobalEnv)
  }
  invisible(TRUE)
}
create_SL_knn(c(10, 15, 20, 25))

# Get Neural networks of different sizes
create_SL_nnet <- function(size = c(2, 3)) {
  for(mm in seq(length(size))){
    eval(parse(text = paste('SL.nnet.', size[mm], '<- function(..., size = ', size[mm],
      ') SL.nnet(..., size = size)', sep = '')), envir = .GlobalEnv)
```

```
  }
  invisible(TRUE)
}
create_SL_nnet(c(2, 3, 4, 5, 6))

# Get both the ridge and lasso regressions
SL.glmnet.0 <- function(..., alpha = 0) SL.glmnet(..., alpha = 0) # Ridge
SL.glmnet.1 <- function(..., alpha = 1) SL.glmnet(..., alpha = 1) # Lasso
```

Define constants

```
# Set constants
SL_LIBRARY <- c(

  # Linear methods
  'SL.glm'
  , 'SL.glmnet.0' # Ridge
  , 'SL.glmnet.1' # Lasso

  # Additive models, Trees and other methods
  , 'SL.gam'
  , 'SL.xgboost'
  , 'SL.randomForest'
  , 'SL.rpartPrune'
  , 'SL.polymars'

  # Neural Network Methods
  , 'SL.nnet.2'
  , 'SL.nnet.3'
  , 'SL.nnet.4'
  , 'SL.nnet.5'
  , 'SL.nnet.6'

  # Prototype Methods
  , 'SL.knn.10'
  , 'SL.knn.15'
  , 'SL.knn.20'
  , 'SL.knn.25'

  # Other
  , 'SL.mean'
  )

DIR <- '/Users/josiahdavis/Documents/Berkeley/PH252D/data/' # <= UPDATE AS NEEDED
FILE_IN_NAME <- 'HR_comma_sep_2.csv'
FILE_OUT_NAME <- 'SL_output_2.csv'
```

Run Everything

```r
d <- get_data(DIR, FILE_IN_NAME)

d <- format_data(d)
```

```
## Number of positivity violations 1209 (14.14%)
```

```r
super_learner <- get_super_learner(d, SL_LIBRARY)
```

```
## Warning: package 'xgboost' was built under R version 3.3.2
```

```
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## warning - model size was reduced
## step half ouch...
## step half ouch...
## warning - model size was reduced
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## warning - model size was reduced
## step half ouch...
## step half ouch...
## step half ouch...
## warning - model size was reduced
## step half ouch...
## step half ouch...
## step half ouch...
## warning - model size was reduced
## step half ouch...
## step half ouch...
## step half ouch...
## warning - model size was reduced
## warning - model size was reduced
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
```

```
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
## step half ouch...
##
## Call:
## SuperLearner(Y = df$left, X = df[, attr(df, "x_cols")], newX = rbind(X_0,
##     X_1), family = "binomial", SL.library = learning_library, verbose = FALSE,
##     cvControl = list(V = 5))
##
##
##                           Risk         Coef
## SL.glm_All          0.15800512 0.0000000000
## SL.glmnet.0_All     0.15798507 0.0000000000
## SL.glmnet.1_All     0.15803079 0.0000000000
## SL.gam_All          0.15800512 0.0000000000
## SL.xgboost_All      0.08125157 0.6417040019
## SL.randomForest_All 0.08973321 0.0690668407
## SL.rpartPrune_All   0.08620364 0.0007638034
## SL.polymars_All     0.08280282 0.2734436428
## SL.nnet.2_All       0.17847655 0.0140479684
## SL.nnet.3_All       0.17390744 0.0000000000
## SL.nnet.4_All       0.19178302 0.0000000000
## SL.nnet.5_All       0.17854615 0.0000000000
## SL.nnet.6_All       0.16597486 0.0009737430
## SL.knn.10_All       0.09029267 0.0000000000
## SL.knn.15_All       0.09536192 0.0000000000
## SL.knn.20_All       0.09708870 0.0000000000
## SL.knn.25_All       0.09858463 0.0000000000
## SL.mean_All         0.20308630 0.0000000000
## Model training and predicting took 4.74 minutes
```

```r
d <- get_counterfact_outcomes(d, super_learner)

save_output(d, super_learner, DIR, FILE_OUT_NAME)
```