Super Learning

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 418023 22.4     750400 40.1   592000 31.7
## Vcells 617668  4.8    1308461 10.0   807903  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)))
}

pre_process_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(
    number_project = bucket_covariate(number_project, 2)
    , average_montly_hours = bucket_covariate(average_montly_hours, 5)
    , time_spend_company = bucket_covariate(time_spend_company, 2)
  )

# Check for violations of the positivity assumption
df <- df %>%
group_by(
  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)
```

```r
  # Create attribute to store columns
  attr(df, 'x_cols') <- df %>%
    dplyr::select(-left, -satisfaction_level, -last_evaluation) %>%
    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 <-  df[,attr(df, 'x_cols')]
  X_0 <- X %>% mutate(salary = 0)
  X_1 <- X %>% mutate(salary = 1)

  # Run Super Learner
  model <- SuperLearner(Y = df$left
                        , X = df[,attr(df, 'x_cols')]
                        , newX = rbind(X, 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: super learnerer predicting the probability of outcome (i.e., leaving company)
  #
  # Returns:
  #   data-frame with predictions, and other results (e.g., cvRisk)


  df$Q_bar_AW <- model$SL.predict[1:nrow(df)]
  df$Q_bar_0W <- model$SL.predict[(nrow(df)+1):(2*nrow(df))]
  df$Q_bar_1W <- model$SL.predict[(2*nrow(df)+1):(3*nrow(df))]

  return(df)
}

get_treatment_mechanism <- function(df, model){
  # Retreive g_aw the treatment probability given covariates
  #
  # Args:
  #   df: observed with covariates data as a data-frame
  #   model: super learner predicting the probabability of treatment (i.e., high salary)
  #
  # Returns:
  #   data-frame with treatment mechanisms

  g_hat1_W <- model$SL.predict
  g_hat0_W <- 1 - g_hat1_W
  g_hat_AW <- rep(NA, nrow(obs_data))
  g_hat_AW[obs_data$salary==1] <- g_hat1_W[obs_data$salary==1]
  g_hat_AW[obs_data$salary==0] <- g_hat0_W[obs_data$salary==0]
  df$g_hat_AW <- g_hat_AW
  return(df)
}

save_output <- function(df, q_model, g_model, dir, save_model = TRUE){
  # Write counterfactual outcomes, model summary results to local disk
  #
  # Args:
  #   df: observed data as a data-frame
  #   q_model: super learning object for estimating the conditional probability
  #   g_model: super learning object for estimating the treatment mechanism
  #   dir: working directory as character vector
  #   save_model: whether or not to save the full model
  #
  # Returns:
  #   Nothing to the R environment

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

  # 2. Save modeling results for conditional probability
  cbind(Risk = q_model$cvRisk, Coef = q_model$coef) %>%
    write.csv(paste0(DIR, 'sl_summary_q.csv'), row.names = TRUE)

  # 3. Save modeling results for treatment mechanism
  cbind(Risk = g_model$cvRisk, Coef = g_model$coef) %>%
    write.csv(paste0(DIR, 'sl_summary_g.csv'), row.names = TRUE)

  # 4. Save the model objects themselves
  if(save_model){
      saveRDS(q_model, paste0(dir, 'SL_full_q.rds'))
      saveRDS(g_model, paste0(dir, 'SL_full_g.rds'))
  }
}
```

Define Functions for Super Learning

```r
# 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

```r
# 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'
```

Run Everything

```
# Read data into memory
obs_data <- get_data(DIR, FILE_IN_NAME)

# Preprocess data including bucketing
obs_data <- pre_process_data(obs_data)
```

```
## Number of positivity violations 694 (8.11%)
```

```
# Estimate Probability of Leaving $Q_0(A,W)$
q_hat_sl <- get_super_learner(obs_data, SL_LIBRARY)
```

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

```
## 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...
## warning - model size was reduced
## step half ouch...
## warning - model size was reduced
## warning - model size was reduced
## step half ouch...
## step half ouch...
## warning - model size was reduced
## warning - model size was reduced
## warning - model size was reduced
## 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...
## warning - model size was reduced
## warning - model size was reduced
## 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
##
## Call:
## SuperLearner(Y = df$left, X = df[, attr(df, "x_cols")], newX = rbind(X,
##     X_0, X_1), family = "binomial", SL.library = learning_library, verbose = FALSE,
##     cvControl = list(V = 5))
##
##
##                          Risk       Coef
## SL.glm_All          0.1697886 0.00000000
## SL.glmnet.0_All     0.1696695 0.00000000
## SL.glmnet.1_All     0.1695108 0.00000000
## SL.gam_All          0.1539264 0.00000000
## SL.xgboost_All      0.1091937 0.45210166
## SL.randomForest_All 0.1236473 0.00000000
## SL.rpartPrune_All   0.1145225 0.02960361
## SL.polymars_All     0.1097682 0.35135024
## SL.nnet.2_All       0.2445457 0.03773175
## SL.nnet.3_All       0.1585016 0.01050713
## SL.nnet.4_All       0.1482916 0.00000000
## SL.nnet.5_All       0.2080735 0.00000000
## SL.nnet.6_All       0.1462710 0.01121231
## SL.knn.10_All       0.1123217 0.08974739
## SL.knn.15_All       0.1143094 0.01774591
## SL.knn.20_All       0.1166216 0.00000000
## SL.knn.25_All       0.1181384 0.00000000
## SL.mean_All         0.1977303 0.00000000
## Model training and predicting took 4.26 minutes
```

```
# Calculate counterfactual outcomes
results <- get_counterfact_outcomes(obs_data, q_hat_sl)

# Estimate propensity of treatment g_0(A|W)
g_hat_sl<- SuperLearner(Y=obs_data$salary
                        , X=obs_data[,attr(obs_data, 'x_cols')] %>% select(-salary)
                        , SL.library=SL_LIBRARY, family="binomial")
```

```
## warning - model size was reduced
```

```
results <- get_treatment_mechanism(results, g_hat_sl)
```

```r
# Save results
save_output(results, q_hat_sl, g_hat_sl, DIR)
```