generate_replicationR.R

moffer

2022-08-30

```
library(reticulate)

use_python('/usr/bin/python')

gammahat <- read.csv('../experiments/results/gammahat.csv', header = FALSE)
muhat <- read.csv('../experiments/results/muhat.csv', header = FALSE)
ys <- read.csv('../experiments/results/ys.csv', header = FALSE)
ws <- read.csv('../experiments/results/ws.csv', header = FALSE)
yobs <- read.csv('../experiments/results/yobs.csv', header = FALSE)
xs <- read.csv('../experiments/results/xs.csv', header = FALSE)
muxs <- read.csv('../experiments/results/muxs.csv', header = FALSE)

T <- dim(gammahat)[1]
K <- dim(gammahat)[2]

# probabilities
np <- import("numpy")
# 3 dimensions: time, contexts, treatment arms
probs_array <- np$load("../experiments/results/probs.npy")

mask <- matrix(1, nrow = T, ncol = T)

for(i in 2:T){
  mask[i, i:T] <- 0
}

# Create point-wise optimal policy matrix (not learned from data)
policy <- matrix(0, nrow = T, ncol = K)
policy[cbind(1:T,apply(muxs, 1, which.max))] <- 1


# Reciprocal of interior of (10) in paper
all_condVars <- sapply(1:T, function(x) rowSums(sweep(1/probs_array[,x,], MARGIN = 2, policy[x,]^2, `*`)
all_condVars_inverse <- matrix(0, ncol = T, nrow = T)

all_condVars_inverse[all_condVars > 1e-6] <- 1 / all_condVars[all_condVars > 1e-6]

expected_condVars <- rowSums(all_condVars * mask)/rowSums(mask)

expected_condVars_inverse <- expected_condVars
expected_condVars_inverse[] <- 0
```

```
expected_condVars_inverse[expected_condVars > 1e-6] <- 1 / expected_condVars[expected_condVars > 1e-6]



estimate <- function(w, gammahat, policy, policy_value){
  # Return bias and variance of policy evaluation via non-contextual weighting.
  #
  # INPUT
  #     - w: non-contextual weights of shape [T]
  #     - gammahat: AIPW score of shape [T, K]
  #     - policy: policy matrix pi(X_t, w), shape [T, K]
  #     - policy_value: ground truth policy value
  #
  # OUTPUT
  #     - np.array([bias, var])
  estimate <- aw_estimate(gammahat, policy, w)
  var <- aw_var(gammahat, estimate, policy, w)
  bias <- estimate - policy_value

  return(c(`estimate` = estimate, `bias` = bias, `var` = var))
}

aw_estimate <- function(scores, policy, evalwts=NULL){
  # Estimate policy value via non-contextual adaptive weighting.
  #
  # INPUT
  #     - scores: AIPW score, shape [T, K]
  #     - policy: policy matrix pi(X_t, w), shape [T, K]
  #     - evalwts: non-contextual adaptive weights h_t, shape [T]
  # OUTPUT
  #     - estimated policy value.
  if(is.null(evalwts)){
    evalwts <- matrix(1, nrow = nrow(scores))
  }

  return(sum(evalwts*rowSums(scores * policy))/sum(evalwts))
}

aw_var <- function(scores, estimate, policy, evalwts=NULL){
  # Variance of policy value estimator via non-contextual adaptive weighting.
  #
  # INPUT
  #     - scores: AIPW score, shape [T, K]
  #     - estimate: policy value estimate
  #     - policy: policy matrix pi(X_t, w), shape [T, K]
  #     - evalwts: non-contextual adaptive weights h_t, shape [T]
  # OUTPUT
  #     - variance of policy value estimate
  #
  # var =  sum[t=0 to T] h[t]^2 * (sum[w] scores[t, w] * policy[t, w] - estimate)^2
  #       ---------------------------------------------------------------------
  #                        (sum[t=0 to T] h[t])^2
```

```r
  if(is.null(evalwts)){
    evalwts <- matrix(1, nrow = nrow(scores))
  }

  return(sum((rowSums(scores * policy)-estimate)^2*evalwts^2)/sum(evalwts)^2)
}

calculate_continuous_X_statistics <- function(h, gammahat, policy, policy_value){
  # Return bias and variance of policy evaluation via contextual weighting.
  #
  # INPUT
  # - h: adaptive weights h_t(X_s) of size (T, T)
  # - gammahat: AIPW score of shape [T, K]
  # - policy: policy matrix pi(X_t, w), shape [T, K]
  # - policy_value: ground truth policy value
  #
  # OUTPUT:
  #   - np.array([bias, var])
  T <- dim(h)[1]
  Z <- colSums(h)  # size (T) \sum_{s=1}^T h_s(X_t)
  gamma_policy <- rowSums(gammahat * policy)
  ht_Xt_Z <- h[cbind(1:T, 1:T)]
  ht_Xt_Z[Z > 1e-6] <- ht_Xt_Z[Z > 1e-6] / Z[Z > 1e-6]  # size (T), h_t(X_t) / Z(X_t)
  B <- ht_Xt_Z * gamma_policy
  h_Z <- h
  h_Z[, Z > 1e-6] <- sweep(h_Z[, Z > 1e-6], 2, Z[Z > 1e-6], `/`)

  estimate <- sum(B)
  var <- sum((B - colSums(h_Z*B))^2)
  bias <- estimate - policy_value

  return(c(`estimate` = estimate, `bias`= bias, `var` = var))
}

policy_value <- 0.5

analysis_output <- list(
  uniform=estimate(1:T, gammahat, policy, policy_value),
  propscore_expected=estimate(expected_condVars_inverse, gammahat, policy, policy_value),
  propscore_X=calculate_continuous_X_statistics(
    all_condVars_inverse, gammahat, policy, policy_value),
  lvdl_expected=estimate(sqrt(expected_condVars_inverse),
                         gammahat, policy, policy_value),
  lvdl_X=calculate_continuous_X_statistics(sqrt(
    all_condVars_inverse), gammahat, policy, policy_value)
)


analysis_output
```

```
## $uniform
##     estimate         bias          var
##  0.464665459 -0.035334541  0.003244216
##
```

```
## $propscore_expected
##     estimate         bias          var
##  0.401707544 -0.098292456  0.003178457
##
## $propscore_X
##     estimate         bias          var
## 0.5147349721 0.0147349721 0.0003302522
##
## $lvdl_expected
##     estimate         bias          var
##  0.445044719 -0.054955281  0.003338309
##
## $lvdl_X
##     estimate         bias          var
## 0.5011886185 0.0011886185 0.0004938294
```