Project 4 Causal Inference

Group 7

Daizy Lam, Qiao Li, Chuanchuan Liu, Yingyao Wu, Serena Yuan

Introduction

In this project,our goal is to compare Regression Estimate, Stratification, and Weighted Regression where PS is based on L2 penalized logistic regression.

We implement and evaluate these three causal inference algorithms on treatment datasets (low dimensional/ high dimensional).

ATE estimation is used to evaluate and compare the efficiency of the causal inference algorithms.

Load Data

```
lowDim <- read.csv("../data/lowDim_dataset.csv")
highDim <- read.csv("../data/highDim_dataset.csv")
```

Regression Estimate (with no need of propensity score)

This algorithm builds regression model fit to control groups and treatment groups, and then makes predictions based on the model in order to calculate ATE.

Algorithm

```
Regression_Estimation <- function(df){

  start_time <- Sys.time()

  # regression model for control groups (A=0)
  model0 <- glm(formula=Y~., data=subset(df[which(df$A==0),],select=-c(A)))
  # regression model for treatment groups (A=1)
  model1 <- glm(formula=Y~., data=subset(df[which(df$A==1),],select=-c(A)))

  X = subset(df, select=-c(Y,A)) #input data for prediction

  #prediction using model0
  Y0 <- predict(model0, newdata=X)
  #prediction using model1
  Y1 <- predict(model1, newdata=X)

  # calculate ATE
  ATE <- mean(Y1-Y0)
```

```
  # calculate running time
  end_time <- Sys.time()
  running_time <- end_time - start_time

  return (list(ATE=ATE, running_time=running_time))
}
```

Summarize

- ATE

```
calc_ate_low <- Regression_Estimation(lowDim)$ATE
calc_ate_high <- Regression_Estimation(highDim)$ATE
cat("ATE for Low Dimension Data is", calc_ate_low, "\n")
```

```
## ATE for Low Dimension Data is 2.125138
```

```
cat("ATE for High Dimension Data is", calc_ate_high)
```

```
## ATE for High Dimension Data is -57.42659
```

- Running Time

```
running_time_low <- Regression_Estimation(lowDim)$running_time
running_time_high <- Regression_Estimation(highDim)$running_time
cat("Running Time for Low Dimension Data is", running_time_low, "sec.","\n")
```

```
## Running Time for Low Dimension Data is 0.01399589 sec.
```

```
cat("Running Time for High Dimension Data is", running_time_high, "sec.")
```

```
## Running Time for High Dimension Data is 0.3170061 sec.
```

- Performance

```
true_ate_low = 2.0901
true_ate_high = -54.8558
performance_low <- abs(true_ate_low - calc_ate_low)
performance_high <- abs(true_ate_high - calc_ate_high)
cat("Performance for Low Dimension Data is", performance_low, "\n" )
```

```
## Performance for Low Dimension Data is 0.03503807
```

```
cat("Performance for High Dimension Data is", performance_high)
```

```
## Performance for High Dimension Data is 2.570787
```

- Result

```
RE <- matrix(c(calc_ate_high,calc_ate_low,running_time_high,running_time_low,performance_high,performan
colnames(RE) = c("ATE","Computational Efficiency","Performance")
rownames(RE) = c("High Dimension", "Low Dimension")
RE
```

```
##                       ATE Computational Efficiency Performance
## High Dimension -57.426587               0.31700611  2.57078732
## Low Dimension    2.125138               0.01399589  0.03503807
```

Propensity Score

L2 Penalized Logistic Regression

The logistic regression model represents the class-conditional probabilities through a linear function of the predictors:

$$logit[Pr(T = 1|X)] = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

$$Pr(T = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)}}$$

To avoid overfitting of the logistic regression model, we introduce regularization term to decrease the model variance in the loss function Q.

In order to achieve this, we modifying the loss function with a penalty term which effectively shrinks the estimates of the coefficients. The regularization term is the "L2 norm":

$$Q = -\frac{1}{n}\sum[y_i(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p) + log(1 + \exp(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p))] + \lambda \sum \beta_j^2$$

```
propensity_score <- function(data){
  #L2 penalized logistic regression was used to estimate propensity scores
  #Input:
  #data - the dataframe we want to evaluate
  #Return:
  #p_score - propensity score
  #Run time for calculating p_score
  set.seed(0)
  start_time <- Sys.time()
  A<-data$A
  Y<-data$Y

  glm <- cv.glmnet(data.matrix(data[,-c(1,2)]), A, family = "binomial", alpha = 0)

  p_score <- predict(glm$glmnet.fit,
                     s = glm$lambda.min,
                     newx = data.matrix(data[,-c(1,2)]),
                     type = "response")

  end_time <- Sys.time()
  running_time = end_time - start_time
  return(p_score)
}
```

Propensity Scores

```
ps_high = propensity_score(highDim)
ps_low = propensity_score(lowDim)
```

Stratification

Stratification (sometimes referred to as subclassification) is commonly used in observational studies to control for systematic differences between the control and treated groups. This technique consists of grouping subjects into strata determined by observed background characteristics. (D'Agostino, 1998)

We will estimate the Average Treatment Effect (ATE) using stratification based on L2 penalized Logistic Regression propensity scores. The algorithm will be based on the following equation:

$$\hat{\Delta}_S = \sum_{j=1}^{K} \frac{N_j}{N} \{N_{1j}^{-1} \sum_{i=1}^{N} T_i Y_i I(\hat{e}_i \in \hat{Q}_j) - N_{0j}^{-1} \sum_{i=1}^{N} (1 - T_i) Y_i I(\hat{e}_i \in \hat{Q}_j)\}$$

where K is the number of strata, N_j is the number of individuals in stratum j, N_{1j} is the number of "treated" individuals in stratum j and N_{0j} is the number of "controlled" individuals in stratum j. $\hat{Q}_j = (\hat{q}_{j-1}, q_j]$ is the interval from (j-1)th sample quantile to jth sample quantile of the estimated propensity scores. (Lunceford and Davidian, 2004)

Estimate Average Treatment Effect (ATE)

We first determine the number of strata K and create equally spaced intervals starting from 0 to 1. In the example below we are using K = 5, and the resulting intervals would be (0, 0.2, 0.4, 0.6, 0.8, 1).

```
K = 5 # number of strata
q = seq(0, 1, by = 1/K) # sample quantile
```

Then we form K strata according to the sample quantiles of the $\hat{e}_i$ (estimated propensity scores), i = 1, ..., N (sample size/number of observations), where the jth sample quantile $\hat{q}_j$, for j = 1, ..., K, is such that the proportion of $\hat{e}_i \leq \hat{q}_j$ is roughly j/K, $\hat{q}_0 = 0$ and $\hat{q}_K = 1$. (Lunceford and Davidian, 2004)

Within each stratum, we subset the observations for the specific strata, i.e. observations whose propensity scores fall in the interval $(q_{j-1}, q_j]$. We then split the subsetted data set into "treated" and "controlled" groups using the binary treatment indicator 'A'. For each group within the stratum, we calculate the summation over the multiplication of treatment indicator 'A' and outcome 'Y', and divide the sum by the number of individuals for each group.

Then the estimated ATE would be the summation over all K strata of the weighted sum of the difference of "treated" and "controlled" groups as described above.

```
stratification = function(df, ps){
  tm_start = Sys.time()
  # split into K strata
  stratum = rep(NA, length(q))
  for (i in 1:length(q)){
    stratum[i] = quantile(ps, q[i])
  }
  # ATE
  ate_strata = rep(NA, K)
  for (j in 1:K){
    # select observations whose propensity score is within (q_{j-1},q_j]
    curr.obs = df[which(stratum[j] < ps & ps <= stratum[j+1]),]

    # subset/select treatment and control groups
    treatment_strata = curr.obs[curr.obs$A == 1,]
    control_strata = curr.obs[curr.obs$A == 0,]

    # calculate sum within stratum
```

```
    treatment_sum = sum(treatment_strata$A * treatment_strata$Y) / nrow(treatment_strata)
    control_sum = sum((1 - control_strata$A) * control_strata$Y) / nrow(control_strata)
    ate_strata[j] = (nrow(curr.obs)/nrow(df)) * (treatment_sum - control_sum)
  }
  tm_end = Sys.time()
  cat("The estimated ATE is", sum(ate_strata), "\n")
  cat("Run time is", (tm_end - tm_start), "s")
  return(list(sum(ate_strata), (tm_end - tm_start)))
}
```

High Dimension Data Set

```
strata_high = stratification(highDim, ps_high)
```

```
## The estimated ATE is -57.93487
## Run time is 0.02869701 s
```

Low Dimension Data Set

```
strata_low = stratification(lowDim, ps_low)
```

```
## The estimated ATE is 2.376539
## Run time is 0.005393028 s
```

Performance

- True ATE for High Dimension Data Set: -54.8558
- True ATE for Low Dimension Data Set: 2.0901

```
true_ate_high = -54.8558
true_ate_low = 2.0901
cat("Performance for High Dimension Data is", abs(true_ate_high - strata_high[[1]]), "\n")
```

```
## Performance for High Dimension Data is 3.079069
```

```
cat("Performance for Low Dimension Data is", abs(true_ate_low - strata_low[[1]]))
```

```
## Performance for Low Dimension Data is 0.2864389
```

```
res = matrix(rep(NA,4), ncol = 2)
colnames(res) = c("Performance", "Computational Efficiency")
rownames(res) = c("High Dimension", "Low Dimension")
res[1,1] = abs(true_ate_high - strata_high[[1]])
res[1,2] = abs(true_ate_low - strata_low[[1]])
res[2,1] = strata_high[[2]]
res[2,2] = strata_low[[2]]
res
```

```
##                Performance Computational Efficiency
## High Dimension  3.07906898              0.286438938
## Low Dimension   0.02869701              0.005393028
```

Weighted Regression

Weighted least square estimation of regression function:

$$Y_i = \alpha_0 + \tau \cdot T_i + \alpha_1' \cdot Z_i + \alpha_2'(Z_i - \bar{Z}) \cdot T_i + \epsilon_i$$

where the weights are of inverse propensity weighting.

Z_i are a subset of covariates X_i with sample average $\bar{Z}$, and τ is an estimate for ATE.

We will use the following method to select Z.

Estimate linear regressions $Y_i = \beta_{k0} + \beta_{k1} T_i + \beta_{k2} \cdot X_{ik} + \epsilon_i$

The weights are given by $\hat{\omega}(t,x) = \frac{t}{\hat{e}(x)} + \frac{1-t}{1-\hat{e}(x)}$.

Then estimate K linear regressions of type $Y_i = \beta_{k0} + \beta_{k1} \cdot T_i + \beta_{k2} \cdot X_{ik} + \epsilon_i$ and then calculate the t-statistic for the test of null hypothesis that the slope $\beta_{k2}$is equal to zero. We select the subset where the t-statistic is significant. So select for Z all covariates with a t-statistic which is larger in absolute value than t_{reg}. Then, in the final regression, include all covariates in this subset, and use the weights as the values $\hat{\omega}(t,x)$.

```
#assign variables for data frame
X_low <- data.matrix(lowDim[,-c(1,2)])
A_low <- data.matrix(lowDim[,2])
X_high <- data.matrix(highDim[,-c(1,2)])
A_high <- data.matrix(highDim[,2])
```

```
Weighted_Regression <- function(X,A,df, threshold){
  start_time <- Sys.time()
  # Finding weights
  t<- as.numeric(A)
  wt<- t/(propensity_score(df)) + (1-t)/(1-propensity_score(df))

  # Estimate linear regression
  Y <- df$Y
  model<- lm(Y~., data = df)
  feature_z <- summary(model)$coef[,4][-c(1:2)]<threshold
  Z <- as.data.frame(cbind(A,X[,feature_z]))
  Z<-sapply(Z, as.numeric)

  #Weighted Regression
  weighted_reg <- lm(Y ~ Z, weights = wt)

  #coef of T is an estimate for ATE
  ATE<- coef(weighted_reg)[2]
  end_time <- Sys.time()
  running_time <- end_time - start_time
  cat("The estimated ATE is", ATE, "\n")
  cat("Run time is", running_time , "s \n")
  return (list(ATE=ATE, running_time=running_time))

}
```

Different Thresholds for Best ATE

```
thresholds <- c(0.1, 0.05, 0.02, 0.01, 0.005)

ate_scores_low <- rep(NA,length(thresholds))
ate_scores_high <- rep(NA,length(thresholds))
```

```
true_ate_high = -54.8558
true_ate_low = 2.0901

count = 1
for (t in thresholds) {
  wreg_low = Weighted_Regression(X_low, A_low, lowDim, t)
  wreg_high = Weighted_Regression(X_high, A_high, highDim, t)
  diff_low = abs(true_ate_low - wreg_low[[1]])
  diff_high = abs(true_ate_high - wreg_high[[1]])
  ate_scores_low[count] = diff_low
  ate_scores_high[count] = diff_high
  count = count + 1
}
```

```
## The estimated ATE is 2.180446
## Run time is 0.540658 s
## The estimated ATE is -58.17327
## Run time is 12.91386 s
## The estimated ATE is 2.180446
## Run time is 0.500773 s
## The estimated ATE is -58.21078
## Run time is 11.96959 s
## The estimated ATE is 2.180446
## Run time is 0.4844079 s
## The estimated ATE is -58.181
## Run time is 12.55749 s
## The estimated ATE is 2.180446
## Run time is 0.5095809 s
## The estimated ATE is -57.85927
## Run time is 12.31895 s
## The estimated ATE is 2.180446
## Run time is 0.4817441 s
## The estimated ATE is -57.8485
## Run time is 12.11534 s
```

Set threshold to the optimal value:

```
opt_low <- which.min(ate_scores_low)
opt_high <- which.min(ate_scores_high)
threshold_low <- thresholds[opt_low]
threshold_high <- thresholds[opt_high]
```

```
Weighted_Regression_low = Weighted_Regression(X_low,A_low,lowDim,threshold_low)
```

```
## The estimated ATE is 2.180446
## Run time is 0.6765339 s
```

```
Weighted_Regression_high = Weighted_Regression(X_high,A_high,highDim, threshold_high)
```

```
## The estimated ATE is -57.8485
## Run time is 12.30315 s
```

```
true_ate_high = -54.8558
true_ate_low = 2.0901
cat("Performance for High Dimension Data is", abs(true_ate_high - Weighted_Regression_high[[1]]), "\n")
```

```
## Performance for High Dimension Data is 2.992701
```

```
cat("Performance for Low Dimension Data is", abs(true_ate_low -Weighted_Regression_low[[1]]))
```

```
## Performance for Low Dimension Data is 0.09034573
```

```
res = matrix(rep(NA,4), ncol = 2)
colnames(res) = c("Performance", "Computational Efficiency")
rownames(res) = c("High Dimension", "Low Dimension")
res[1,1] = abs(true_ate_high - Weighted_Regression_high[[1]])
res[1,2] = abs(true_ate_low - Weighted_Regression_low[[1]])
res[2,1] = Weighted_Regression_high[[2]]
res[2,2] = Weighted_Regression_low[[2]]
res
```

```
##                Performance Computational Efficiency
## High Dimension    2.992701               0.09034573
## Low Dimension    12.303149               0.67653394
```

Model Comparison

Performance: absolute error of the true and estimated ATE

Computational Efficiency: run time in seconds

High Dimension Dataset

```
high = matrix(rep(NA,9), nrow = 3)
rownames(high) = c("Stratification", "Regression Estimate", "Weighted Regression")
colnames(high) = c("ATE", "Performance", "Computational Efficiency")
# stratification
high[1,1] = strata_high[[1]]
high[1,2] = abs(true_ate_high - strata_high[[1]])
high[1,3] = strata_high[[2]]
# regression estimate
high[2,1] = calc_ate_high
high[2,2] = performance_high
high[2,3] = running_time_high
# weighted regression
high[3,1] = Weighted_Regression_high[[1]]
high[3,2] = abs(true_ate_high - Weighted_Regression_high[[1]])
high[3,3] = Weighted_Regression_high[[2]]

high
```

```
##                           ATE Performance Computational Efficiency
## Stratification      -57.93487    3.079069               0.02869701
## Regression Estimate -57.42659    2.570787               0.31700611
## Weighted Regression -57.84850    2.992701              12.30314898
```

Low Dimension Dataset

```
low = matrix(rep(NA,9), nrow = 3)
rownames(low) = c("Stratification", "Regression Estimate", "Weighted Regression")
colnames(low) = c("ATE", "Performance", "Computational Efficiency")
# stratification
low[1,1] = strata_low[[1]]
low[1,2] = abs(true_ate_low - strata_low[[1]])
```

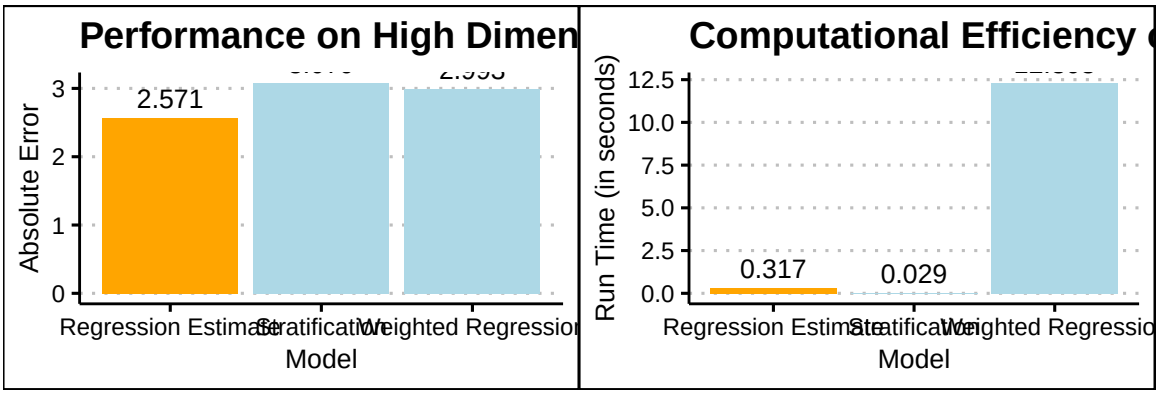
```
low[1,3] = strata_low[[2]]
# regression estimate
low[2,1] = calc_ate_low
low[2,2] = performance_low
low[2,3] = running_time_low
# weighted regression
low[3,1] = Weighted_Regression_low[[1]]
low[3,2] = abs(true_ate_low - Weighted_Regression_low[[1]])
low[3,3] = Weighted_Regression_low[[2]]

low
```

```
##                            ATE Performance Computational Efficiency
## Stratification      2.376539  0.28643894              0.005393028
## Regression Estimate 2.125138  0.03503807              0.013995886
## Weighted Regression 2.180446  0.09034573              0.676533937
```

```
high_perf = data.frame(Model = c("Stratification", "Regression Estimate", "Weighted Regression"),
                       Performance = c(round(abs(true_ate_high - strata_high[[1]]), 3),
                                       round(performance_high, 3),
                                       round(abs(true_ate_high - Weighted_Regression_high[[1]]), 3)))
high_time = data.frame(Model = c("Stratification", "Regression Estimate", "Weighted Regression"),
                       Time = c(as.numeric(round(strata_high[[2]], 3)),
                                round(running_time_high, 3),
                                as.numeric(round(Weighted_Regression_high[[2]], 3))))

hp = ggplot(high_perf, aes(x = Model, y = Performance)) +
  geom_bar(stat = "identity", fill = c("light blue","orange","light blue")) +
  geom_text(aes(label=Performance), vjust = -0.5, size = 3.5) +
  labs(title = "Performance on High Dimension Data", y = "Absolute Error") +
  #ylim(c(0,5)) +
  theme_clean()
ht = ggplot(high_time, aes(x = Model, y = Time)) +
  geom_bar(stat = "identity", fill = c("light blue","orange","light blue")) +
  geom_text(aes(label=Time), vjust = -0.5, size = 3.5) +
  labs(title = "Computational Efficiency on High Dimension Data", y = "Run Time (in seconds)") +
  #ylim(c(0,0.2)) +
  theme_clean()
grid.arrange(hp, ht, ncol=2)
```

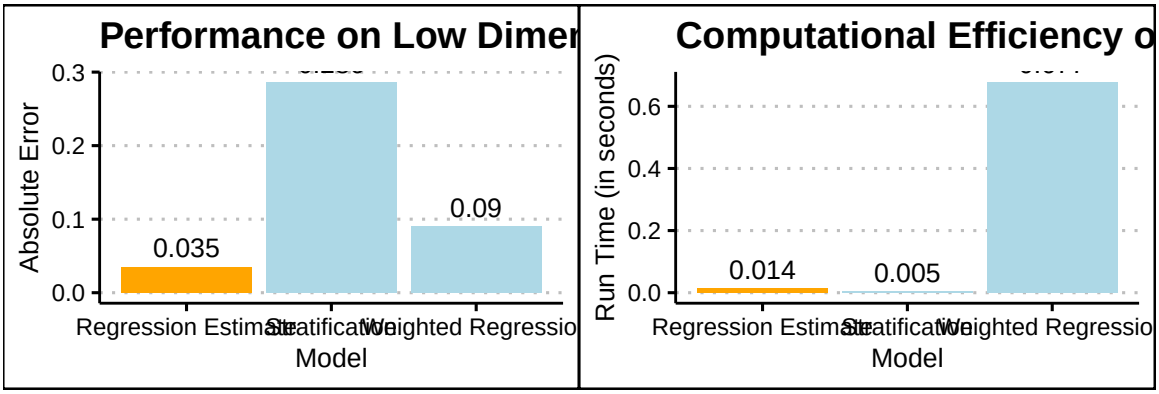



```
# par(mfrow = c(1,2))
# barplot(high_perf$Performance,
```

```
#       main = "Performance on High Dimension Data",
#       names.arg = high_perf$Model,
#       ylab = "Absolute Error")
# barplot(high_time$Time,
#       main = "Computational Efficiency on High Dimension Data",
#       names.arg = high_perf$Model,
#       ylab = "Run Time (in seconds)")
```

```
low_perf = data.frame(Model = c("Stratification", "Regression Estimate", "Weighted Regression"),
                      Performance = c(round(abs(true_ate_low - strata_low[[1]]), 3),
                                      round(performance_low, 3),
                                      round(abs(true_ate_low - Weighted_Regression_low[[1]]), 3)))
low_time = data.frame(Model = c("Stratification", "Regression Estimate", "Weighted Regression"),
                      Time = c(as.numeric(round(strata_low[[2]], 3)),
                               round(running_time_low, 3),
                               as.numeric(round(Weighted_Regression_low[[2]], 3))))

lp = ggplot(low_perf, aes(x = Model, y = Performance)) +
  geom_bar(stat = "identity", fill = c("light blue","orange","light blue")) +
  geom_text(aes(label=Performance), vjust = -0.5, size = 3.5) +
  labs(title = "Performance on Low Dimension Data", y = "Absolute Error") +
  #ylim(c(0,1)) +
  theme_clean()
lt = ggplot(low_time, aes(x = Model, y = Time)) +
  geom_bar(stat = "identity", fill = c("light blue","orange","light blue")) +
  geom_text(aes(label=Time), vjust = -0.5, size = 3.5) +
  labs(title = "Computational Efficiency on Low Dimension Data", y = "Run Time (in seconds)") +
  #ylim(c(0,0.2)) +
  theme_clean()
grid.arrange(lp, lt, ncol=2)
```



References

D'Agostino, R. B. (1998). Propensity score methods for bias reduction in the comparison of a treatment to a non-randomized control group. Statistics in Medicine, 17(19), 2265-2281. doi:10.1002/(sici)1097-0258(19981015)17:193.0.co;2-b

Lunceford, J. K.; Davidian, M. (2004). Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. Statistics in Medicine, 23(19), 2937-2960. doi:10.1002/sim.1903