cs208 HW 3

Anthony Rentsch

4/16/2019

Question 1

(a)

The centralized version of the SQ algorithm simply takes in a conjunction matrix - a matrix with a conjunction of x_j and y in each column - computes an estimate of p_j for $j \in \{1, ..., d\}$ and then adds Laplace noise with scale factor $\frac{d}{n\epsilon}$ to each of the d estimates. For each $\hat{p_j} < t$ for some threshold t, j is returned.

In the localized version, I perform randomized response on each row of the conjunction matrix with the local randomizer defined as follows:

$$Q(x_i) = \begin{cases} x_i & w.p. \frac{e^{\epsilon/d}}{e^{\epsilon/d}+1} \\ 1 - x_i & w.p. \frac{1}{e^{\epsilon/d}+1} \end{cases}$$

From here, we are left with a new conjunction matrix $\hat{x}$ with each row the result of a randomized response process. We can then calculate the estimate of p_j for each column of the new conjunction matrix. However, we will have to correct the output of this, as it will be biased in expectation. I'll consider correcting the numerator of the estimate of p_j and in my implementation I'll divide this by the number rows to get the appropriate estimate of the probability.

$$\begin{aligned} E\left[\hat{p_j}_{numerator}\right] &= E\left[\sum_{i=1}^{n} \hat{x_i}[j]\right] \\ &= \sum_{i=1}^{n} E[\hat{x_i}[j]] \\ &= \sum_{i=1}^{n} \left[x_i \frac{e^\epsilon}{e^\epsilon + 1} + (1 - x_i)\frac{1}{e^\epsilon + 1}\right] \\ &= \sum_{i=1}^{n} \left[x_i \frac{e^\epsilon - 1}{e^\epsilon + 1} + \frac{1}{e^\epsilon + 1}\right] \\ &= nx_i \frac{e^\epsilon - 1}{e^\epsilon + 1} + \frac{n}{e^\epsilon - 1} \end{aligned}$$

Thus, our correction will need a multiplicative and an additive term. The appropriate terms are

$$\begin{aligned} c &= \frac{e^\epsilon + 1}{e^\epsilon - 1} \\ d &= \frac{-n}{e^\epsilon - 1} \\ E[c * \hat{p_j}_{numerator} + d] &= \frac{e^\epsilon + 1}{e^\epsilon - 1}\left[nx_i \frac{e^\epsilon - 1}{e^\epsilon + 1} + \frac{n}{e^\epsilon - 1}\right] - \frac{n}{e^\epsilon - 1} \\ &= nx_i + \frac{n}{e^\epsilon - 1} - \frac{n}{e^\epsilon - 1} \\ &= nx_i \end{aligned}$$

As I did in the centralized version, j is returned if $\hat{p_j} > t$ for some threshold t. For the implmentation of these algorithms, see the Appendix.

(b)

Centralized

$$\begin{aligned}
P[\hat{S} \not\supset S] &= \sum_{j\in S} P[\hat{p_j} > t] \\
&= \sum_{j\in S} P\left[Lap\left(\frac{d}{n\epsilon}\right) > t\right] \\
&= |S|\int_t^{\infty} \frac{e^{-|y|n\epsilon/d} * n\epsilon}{2d} dy \\
&= |S|\frac{n\epsilon}{2d}\frac{-d}{n\epsilon}\left[e^{-yn\epsilon/d}\right]_t^{\infty} \\
&= \frac{|S|}{2}e^{-tn\epsilon/d}
\end{aligned}$$

Now consider that $P[\hat{S} \not\supset S] \leq 0.1$.

$$\begin{aligned}
\frac{|S|}{2}e^{-tn\epsilon/d} &\leq 0.1 \\
-\frac{tn\epsilon}{d} &\leq log\left(\frac{0.2}{|S|}\right) \\
t &\geq -\frac{d\epsilon}{n}log\left(\frac{0.2}{|S|}\right) \\
t &\geq -\frac{d\epsilon}{n}log\left(\frac{0.2}{d}\right) \quad (|S| \leq d)
\end{aligned}$$

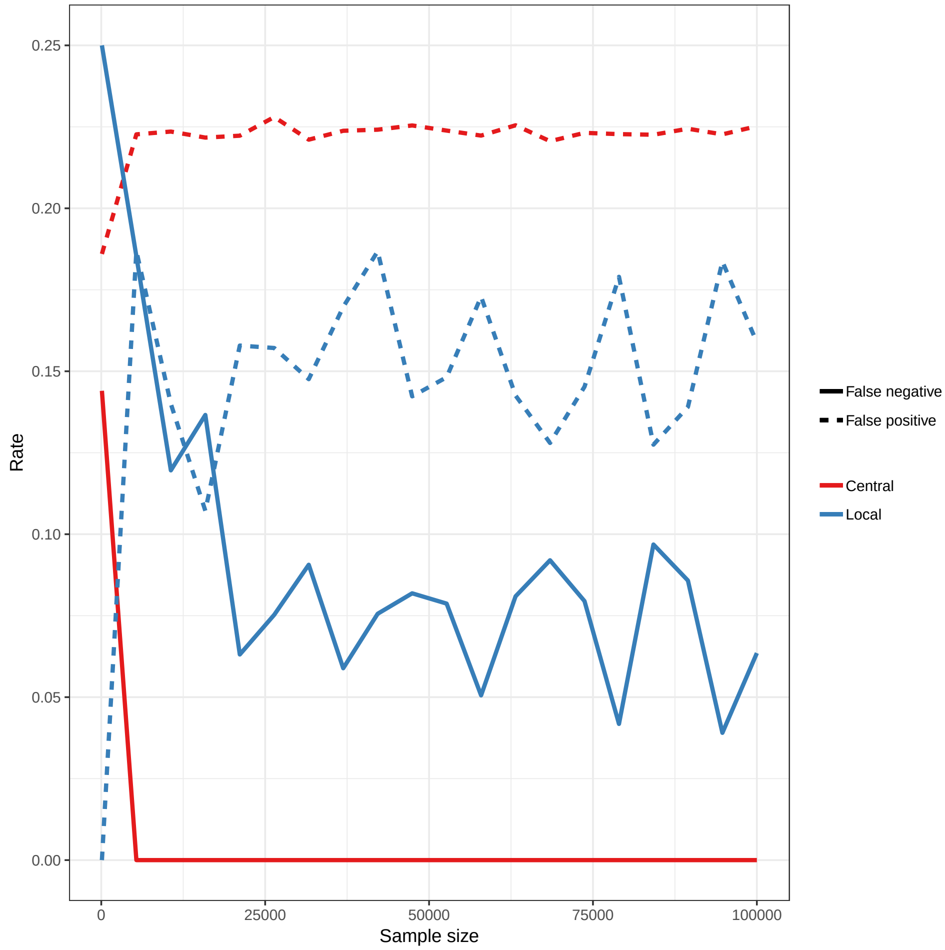
Localized

Intuition is to start from $P[\hat{S} \not\supset S] = \sum_{j\in S} P[\hat{p_j} > t]$ an derive but using randomized response and bias correction term, but presently unsure how to do it.

(c)

My centralized algorithm produces `employed`, `uscitizen`, and `englishability` every time for $t = 0.01$, while the localized model does not produce a consistent set of predictors for the same threshold (some runs it does produce the exact same set as the centralized, while other runs output other predictors or miss one of these three).

I also test my algorithms to predict `targetted` and compute the false positive and false negative rates as a function of the size of the dataset.



Appendix

Question 1

```
test_data <- read.csv("hw4testdata.csv")
pums <- read.csv("CAPUMS5full.csv")

# a
## general ##
get_conjunction_matrix <- function(x, y, negative_val=0, positive_val=1){
  n <- nrow(x)
  d <- ncol(x)
  conjunction <- matrix(NA, nrow=n, ncol=d)
  for(j in 1:d) {
    conjunction[,j] <- as.integer(x[,j]==negative_val & y==positive_val)
  }
  return(conjunction)
}

calc_pj <- function(conjunction, epsilon, type) {

  if(type=="c"){
```

```
    # centralized - add Laplace noise
    scale <- 1/(epsilon*length(conjunction))
    pj <- mean(conjunction) + rlap(mu=0, b=scale, size=1)
  }
  else if(type=="l"){
    # localized - bias correction for RR
    pj <- correction(release=conjunction, epsilon=epsilon)
  }
  else{
    print("Type not supported.")
  }
  return(pj)
}

## centralized ##
sgn <- function(x) {     # function borrowed from class
  return(ifelse(x < 0, -1, 1))
}

rlap = function(mu=0, b=1, size=1) {     # function borrowed from class
  p <- runif(size) - 0.5
  draws <- mu - b * sgn(p) * log(1 - 2 * abs(p))
  return(draws)
}

SQcentralized <- function(conjunction, t, epsilon=1) {
  d <- ncol(conjunction)
  attributes <- c()
  for(j in 1:d) {
    cur_pj <- calc_pj(conjunction=conjunction[,j], epsilon=epsilon/d, type="c")
    if(cur_pj < t) {
      attributes <- c(attributes, j)
    }
  }
  return(attributes)
}

## localized ##
localRelease <- function(x, values=c(0,1), epsilon){     # function borrowed from class
  draw <- runif(n=1, min=0, max=1)
  cutoff <- 1/(1+exp(epsilon))
  if(draw < cutoff){
    return_val <- values[!values %in% x]
  }
  else{
    return_val <- x
  }
  return(return_val)
}

correction <- function(release, epsilon){     # function updated from class
  n <- length(release)
  mulitiplicative <- (exp(epsilon) + 1)/(exp(epsilon) - 1)
  additive <- -n/(exp(epsilon) - 1)
```

```
  expectation <- (sum(release)*mulitiplicative + additive)/n
  return(expectation)
}

SQlocalized <- function(conjunction, t, epsilon=1, negative_val=0, positive_val=1) {
  n <- nrow(conjunction)
  d <- ncol(conjunction)
  new_conjunction <- matrix(NA, nrow=n, ncol=d)
  attributes <- c()

  # RR
  for(j in 1:d){
    for(i in 1:n){
      new_conjunction[i,j] <- localRelease(conjunction[i,j], values=c(0,1), epsilon=epsilon/d)
    }
  }

  # pj calculation (bias correction happens inside here)
  for(j in 1:d) {
    cur_pj <- calc_pj(conjunction=new_conjunction[,j], epsilon=epsilon/d, type="l")
    if(cur_pj < t) {
      attributes <- c(attributes, j)
    }
  }
  return(attributes)
}

## test ##
test_conj_mat <- get_conjunction_matrix(x=test_data[,1:10], y=test_data[,11])
SQcentralized(test_conj_mat, t=0.01, epsilon=1)
SQlocalized(conjunction=test_conj_mat, t=0.01, epsilon=1)


# c
pums_x <- pums[,c("sex","married","black","asian","collegedegree","employed","militaryservice",
                  "uscitizen","disability","englishability")]
pums_y <- pums$targetted
pums_conj_mat <- get_conjunction_matrix(pums_x, pums_y)

names(pums[SQcentralized(conjunction=pums_conj_mat, t=0.01, epsilon=1)])
names(pums[SQlocalized(conjunction=pums_conj_mat, t=0.01, epsilon=1)])

## look at FPR and FNR ##
bootstrap <- function(x, n){     # function updated from class
  index <- sample(x=1:nrow(x), size=n, replace=TRUE)
  return(x[index,])
}

return_rates <- function(predictors, y_true){
  d <- ncol(predictors)
  if(d > 1){
    y_pred <- ifelse(rowSums(predictors) == d, 1, 0)
  }
  else{
```

```
    y_pred <- ifelse(predictors == d, 1, 0)
  }
  fpr <- mean(y_pred==1 & y_true==0)
  fnr <- mean(y_pred==0 & y_true==1)
  return(list(fpr=fpr, fnr=fnr))
}

ns <- seq(from=100, to=10000, length.out=10)
num_boots <- 10
predictors <- c("sex","married","black","asian","collegedegree","employed","militaryservice",
                "uscitizen","disability","englishability")
thresh <- 0.1
eps <- 1

rates <- matrix(NA, nrow=length(ns), ncol=5)
i <- 1
for(n in ns){
  central_fprs <- c()
  central_fnrs <- c()
  local_fprs <- c()
  local_fnrs <- c()

  new_data <- bootstrap(x=pums, n=n)
  new_conj_mat <- get_conjunction_matrix(new_data[,predictors], new_data$targetted)

  for(boot in 1:num_boots){

    central_res <- names(pums[SQcentralized(conjunction=new_conj_mat, t=thresh, epsilon=eps)])
    local_res <- names(pums[SQlocalized(conjunction=new_conj_mat, t=thresh, epsilon=eps)])

    central_predictor_mat <- as.matrix(new_data[,central_res])
    local_predictor_mat <- as.matrix(new_data[,local_res])
    central_rates <- return_rates(predictors=central_predictor_mat, y_true=new_data$targetted)
    local_rates <- return_rates(predictors=local_predictor_mat, y_true=new_data$targetted)

    central_fprs <- c(central_fprs, central_rates$fpr)
    central_fnrs <- c(central_fnrs, central_rates$fnr)
    local_fprs <- c(local_fprs, local_rates$fpr)
    local_fnrs <- c(local_fnrs, local_rates$fnr)
  }

  rates[i, 1] <- n
  rates[i, 2] <- mean(central_fprs)
  rates[i, 3] <- mean(central_fnrs)
  rates[i, 4] <- mean(local_fprs)
  rates[i, 5] <- mean(local_fnrs)
  i = i + 1
}
rates_df <- as.data.frame(rates)
names(rates_df) <- c("n", "central_fpr", "central_fnr", "local_fpr", "local_fnr")

q1_plot <- ggplot(rates_df) +
  geom_line(aes(x=n, y=central_fpr, colour="Central", lty="False positive"), size=1.2) +
  geom_line(aes(x=n, y=central_fnr, colour="Central", lty="False negative"), size=1.2) +
```

```r
  geom_line(aes(x=n, y=local_fpr, colour="Local", lty="False positive"), size=1.2) +
  geom_line(aes(x=n, y=local_fnr, colour="Local", lty="False negative"), size=1.2) +
  scale_colour_brewer(palette="Set1") +
  theme_bw() +
  labs(x="Sample size", y="Rate") +
  theme(legend.title=element_blank())
pdf("plots/q1_plot.pdf", width=8, height=8)
q1_plot
dev.off()
```