# cs208 HW 4b

*Anthony Rentsch*

*4/30/2019*

## Question 1

*(a)*

(i) Consider $x = \{0, ..., 0\}$ and $x' = \{0, ..., \infty\}$. Then we have that $GS_f = \infty$.

(ii) The same $x$ from above can be used to consider the best case dataset. Even then, since $x \in \mathbb{R}^N$, we have that $minLS_f = \infty$.

(iii) Now that we restrict the domain of $x$, we can consider $x = \{a, ..., a\}$ and $x' = \{a, ..., b\}$. Now we have that $RS_f^{\mathbb{H}} = \frac{b-a}{n}$.

A Lipschitz-extension function is the clipped mean, i.e. $f'(x) = \frac{1}{n}\sum_{i=1}^{n}[x_i]_a^b$. This is a valid Lipschitz-extension because $f$' agrees with $f$ on $\mathbb{H} = [a, b]^n$ and $GS_{f'} = RS_f^{\mathbb{H}}$.

*(b)*

(i) For illustrative purposes, consider $x \in \mathbb{R}^3$ such that $x = \{0, 0, \infty\}$ and $x' = \{0, \infty, \infty\}$. From this we see that $GS_f = \infty$.

(ii) The best case dataset here is one in which all values of $x \in \mathbb{R}^N$ are the same. However, the value of $minLS_f$ based on changes based on the values of $N$. For example, when $N = 1$ we have $x = \{0\}$ and $x' = \{\infty\}$, which results in $minLS_f = \infty$. Similarly, for $N = 2$ we have $x = \{0, 0\}$ and $x' = \{0, \infty\}$, which also leads to $minLS_f = \infty$. When $N > 2$, though we have $x = \{0, 0, 0\}$ and $x' = \{0, 0, \infty\}$, so $minLS_f = 0$. Thus for $x \in \mathbb{R}^N$, we have

$$minLS_f = \begin{cases} \infty & N \in \{1, 2\} \\ 0 & N > 2 \end{cases}$$

(iii) Consider $x = \{a, a, ..., b, ..., b\}$ where there is one more $a$ than $b$. Then, if we were to switch one $a$ to a $b$, the median would switch from $a$ to $b$. Thus, $RS_f^{\mathbb{H}} = b - a$.

*(c)*

(i) Consider $x$ where there are $n$ nodes with no edges between them and then, to create $x'$, we add one node that connects to all the other $n$ nodes. This would imply that $GS_f = n - 0 = n$.

(ii) Here, the best case dataset is one in which all $n$ nodes in $x$ are fully connected. Then, if we add one isolated node to $x$ and call this $x'$, we have that $minLS_f = 1$.

(iii) Consider that $x$ has $n$ isolated nodes and $x'$ where there are $n = 1$ nodes, with the new node having $d$ edges, since $x' \in \mathbb{H}$. Then $f(x) = n$ and $f(x') = n - d$. Thus, $RS_f^{\mathbb{H}} = d$.

## Question 2

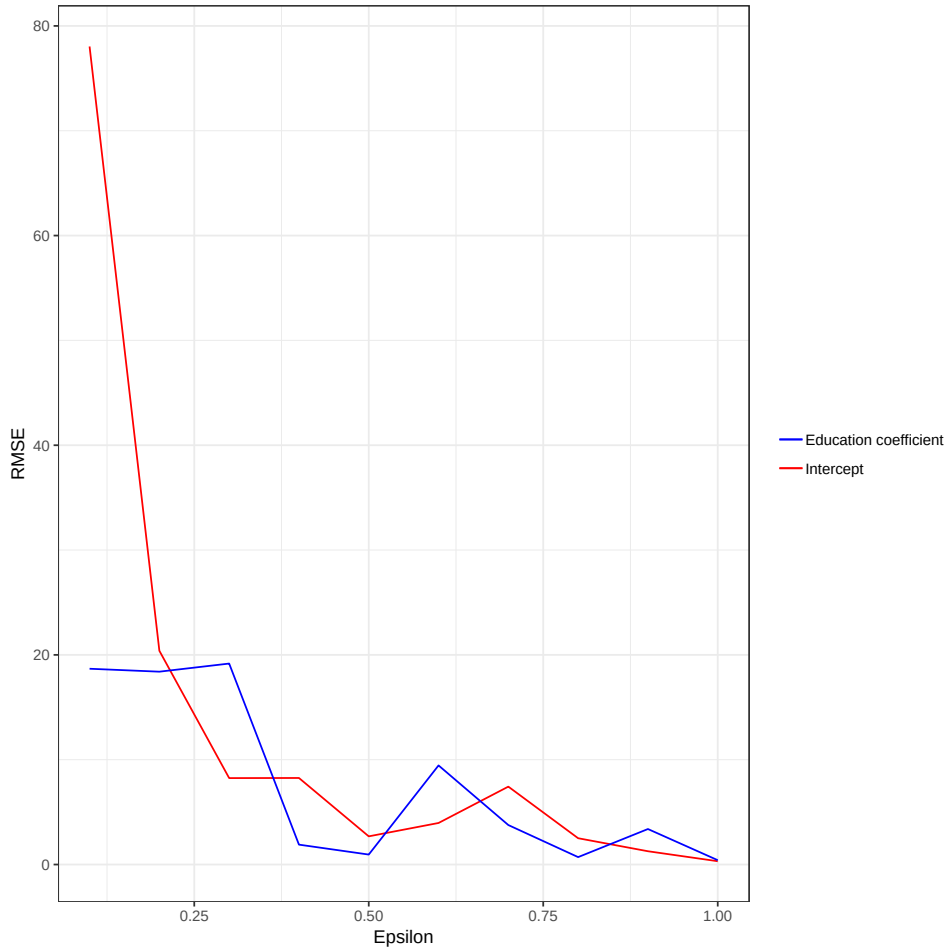
To implement local differential privacy for stochastic gradient descent, I follow these steps:

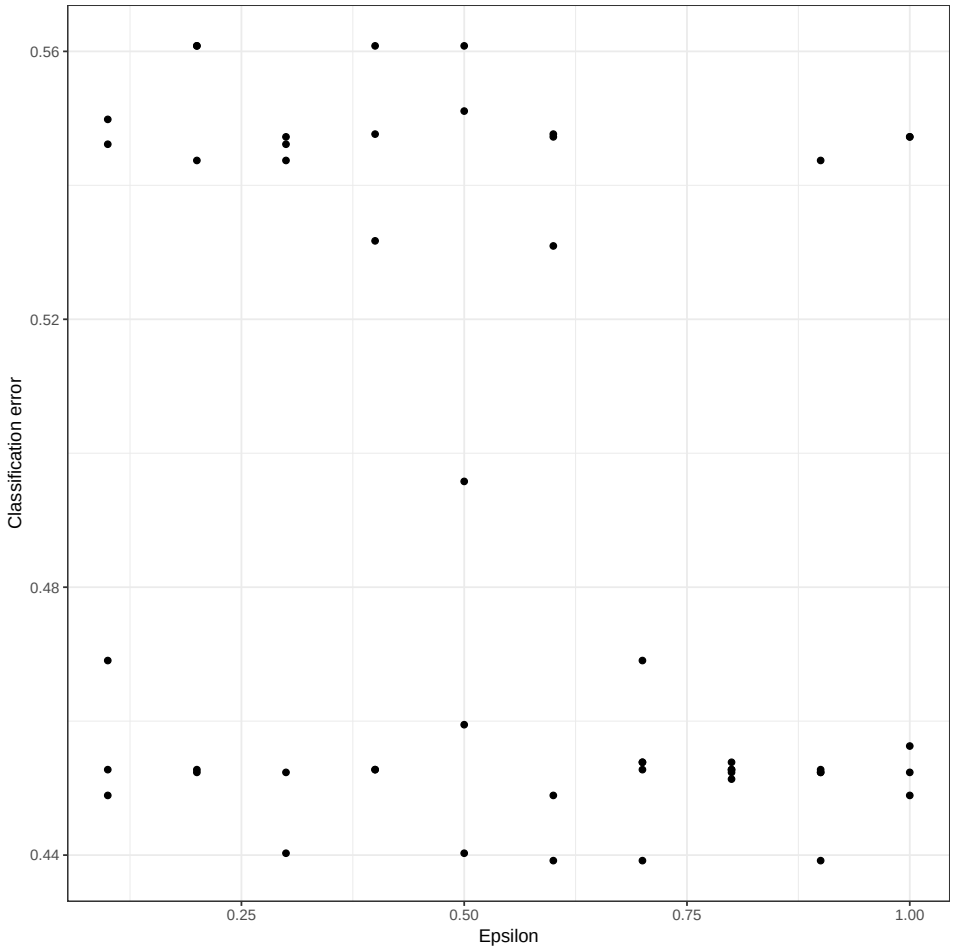
- Split the data up into disjoint batches
- For each batch
  - Compute the gradient for each observation
  - To each observation's gradient, add Gaussian noise with mean 0 and standard deviation

$\sqrt{\left(\frac{C}{\epsilon}\right)^2 * \frac{TB}{n} * ln(1/\delta)}$, where $C$ is the clipping parameter, $T$ is the number of steps, $B$ is the batch size, and $n$ is the total number of observations
  - Take the average of the noisy individual gradients
  - Update parameters by stepping in direction of negative gradient by the learning rate

The code for this question can be found in the appendix. I test out my implementation for different values of $\epsilon$ and plot the RMSE of the estimated values of the coefficients obtained via the non-private and private methods as well as the classification error rate for each of the 5 simulations at each value of $\epsilon$. As the value of $\epsilon$ gets larger, the RMSE between the non-private and private coefficients tends toward 0, which is the expected behavior as larger values of $\epsilon$ tend to lead to less privacy and greater utility.

The classification error rate does not seem to shrink as drastically This is likely due to the fact that the prediction problem is an extremely hard one. Recall that we are trying to predict binary marital status from ordinal level of education. The classification error rate of the non-private logistic regression is relatively high (45%). Thus, the private logistic regression struggles to classify observations well not necessarily because there is too much noise added during optimization but more likely because it is tough to discriminate marital status from education.






## Appendix

My code can also be found in my Github repository.

```
rm(list = ls())
setwd("~/Desktop/Harvard/S19/cs208/cs208_AnthonyRentsch/hw4b/")
require(plyr); require(dplyr); require(ggplot2)
pums <- read.csv("data/MAPUMS5full.csv")
pums <- pums[c("married","educ")]

# Implementation
calcllik<-function(b,data){      # function borrowed from class
  y<-data[,1]
  x<-data[,2]

  pi<- 1/(1+exp(-b[1] - b[2]*x))

  if(pi == 1){ pi = 0.999 }

  llik<-y * log(pi) + (1-y) * log(1-pi)
  return(-llik)
}
```

```
gaussianReleaseNoise <- function(size=1, epsilon, delta, clip, num_steps, batch_size, n){
  scale <- sqrt((clip/epsilon)^2 * (num_steps*batch_size/n) * log(1/delta))
  noise <- rnorm(n=size, mean=0, sd=scale)
  return(noise)
}

clip <- function(x, lower, upper){     # function borrowed from class
  x.clipped <- x
  x.clipped[x.clipped<lower] <- lower
  x.clipped[x.clipped>upper] <- upper
  return(x.clipped)
}

calcgradient <- function(B, C, theta, fun){     # function borrowed from class
  dx <- 0.0001

  out1 <-   eval(fun(b=theta, data=B))
  out2 <- eval(fun(b=theta + c(0,dx), data=B))
  out3 <- eval(fun(b=theta + c(dx,0), data=B))

  Del.1 <- (out3 - out1)/dx
  Del.1 <- clip(Del.1,-C,C)
  mean.Del.1 <- mean(Del.1)


  Del.2 <- (out2 - out1)/dx
  Del.2 <- clip(Del.2,-C,C)
  mean.Del.2 <- mean(Del.2)

  return(c(mean.Del.1,mean.Del.2))
}

calc_class_error <- function(x, y, theta1, theta2){
  y_preds_raw <- 1 / (1 + exp(-1*(theta1 + theta2*x)))
  y_preds <- round(y_preds_raw)
  error <- 1 - mean(y_preds==y)
  return(error)
}

rmse <- function(y_pred, y_true){
  return(sqrt(mean((y_pred-y_true)^2)))
}

# Run
# non-private model
true.out <- glm(married ~ educ, family="binomial", data=pums)

# private model
N <- nrow(pums)
L <- round(sqrt(nrow(pums)))
steps <- L
new.inds <- sample(1:nrow(pums))
shuffled.pums <- pums[new.inds,]
```

```
C <- 10
nu <- c(0.05, 0.01)

epsilons <- seq(from=0.1, to=1, by=0.1)
num_sims <- 5

res_rmse <- matrix(NA, nrow=length(epsilons), ncol=3)
row_ind_rmse <- 1
res_error <- matrix(NA, nrow=length(epsilons)*num_sims, ncol=3)
row_ind_error <- 1

for(epsilon in epsilons){
  print(paste0("Epsilon: ", epsilon))

  thetas <- matrix(NA, nrow=num_sims, ncol=2)
  for(sim in 1:num_sims){
    print(paste0("Simulation: ", sim))

    theta <- c(0,0)

    for(i in 1:steps){
      startB <- ((i-1)*L+1)
      if(i<L){
        stopB <- i*L
      }else{
        stopB <- nrow(shuffled.pums)
      }

      index <- sample(1:nrow(shuffled.pums), L)
      B <- shuffled.pums[startB:stopB, ]
      grads <- c(0.0, 0.0)
      for(j in 1:nrow(B)){
        grads <- grads + calcgradient(B[j,], C, theta, fun=calcllik) + gaussianReleaseNoise(size=2, eps
      }
      theta <- theta - (grads/nrow(B) * nu)
    }

    thetas[sim, 1] <- theta[1]
    thetas[sim, 2] <- theta[2]

    res_error[row_ind_error, 1] <- calc_class_error(pums$educ, pums$married, theta[1], theta[2])
    res_error[row_ind_error, 2] <- sim
    res_error[row_ind_error, 3] <- epsilon
    row_ind_error <- row_ind_error + 1
  }
  res_rmse[row_ind_rmse, 1] <-  rmse(thetas[1], true.out$coef[1])
  res_rmse[row_ind_rmse, 2] <-  rmse(thetas[2], true.out$coef[2])
  res_rmse[row_ind_rmse, 3] <- epsilon
  row_ind_rmse <- row_ind_rmse + 1
}

res_rmse_df <- as.data.frame(res_rmse)
names(res_rmse_df) <- c("rmse_theta1", "rmse_theta2", "epsilon")
res_error_df <- as.data.frame(res_error)
```

```
names(res_error_df) <- c("class_error", "simulation", "epsilon")

q2_plot_rmse <- ggplot(data=res_rmse_df, aes(x=epsilon)) +
  geom_line(aes(y=rmse_theta1, colour="Intercept")) +
  geom_line(aes(y=rmse_theta2, colour="Education coefficient")) +
  labs(x="Epsilon", y="RMSE") +
  scale_color_manual(values=c("blue","red")) +
  theme_bw() +
  theme(legend.title=element_blank())
pdf("plots/q2_plot_rmse.pdf", width=8, height=8)
q2_plot_rmse
dev.off()

q2_plot_error <- ggplot(data=res_error_df, aes(x=epsilon, y=class_error)) +
  geom_point() +
  labs(x="Epsilon", y="Classification error") +
  theme_bw() +
  theme(legend.title=element_blank())
pdf("plots/q2_plot_error.pdf", width=8, height=8)
q2_plot_error
dev.off()
```