6520 Project

Minjia Jia and Joia Zhang

Fall 2023

Simulate data for regression and classification

```
# simulate data: regression
n = 100 # sample size
p = 200 # number of predictors

# beta
k = round(0.05*p, 0) # number of nonzero coefficients
sd_beta = 0.1
nonzero_indexes = sample.int(n=p, size=k)
beta = rep(0, p)
beta[nonzero_indexes] = rnorm(n=k, mean=0, sd=sd_beta)
sum(which(beta !=0) != sort(nonzero_indexes)) # test that we made the right indexes nonzero
```

```
## [1] 0
```

```
beta = as.matrix(beta)

# x
X = matrix(rnorm(n=n*p, mean=10, sd=2), nrow=n)

# epsilon
E = matrix(rnorm(n=n, mean=0, sd=1), nrow=n)

# y

Y = X%*%beta + E

# note that in the online setting, each t^th row of X and Y is for time t

# simulate data: classification
# X, beta same as above
probs = 1/(1+exp(-X%*%beta))
Y = rbinom(n=n, size=1, prob = probs) # Bernoulli
```

OGD

```r
# function for online gradient descent (OGD)
# type: "classification" or "regresssion"
# X: rows are observations, columns are predictors
# Y: response variable
# learning rate (constant)
# beta_0: initialization for the estimate
# N: number of iterations for each data point (each row of X, Y)
my_OGD = function(type, X, Y, lr, beta_0, N) {
  if (type!='classification'&&type!='regression') {
    stop("Argument 'type' must be 'classification' or 'regression'")
  }

  n = nrow(X)
  betahats = array(data=rep(NA, n*p*N), dim = c(n, p, N)) # N arrays that are each nxp arrays
  temp[1, , 1] = beta_0 # initialize
  if (type=='classification') {

  } else {
    # type is regression
    for (t in 1:n) { # for each data point (each row of X, Y)
      for (i in 1:N) { # for each iteration
      d_loss = 2*t(X[t, ]%*%X[t, ]-2*t(X[t, ])%*%Y[t]) # OLS loss
      betahats[t, i+1] = betahats[t, i] - lr*d_loss
      } # end for i
    } # end for t
  } # end else
}
```