Better Online Learning

Su Chen

September 18, 2016

Part A

```r
Sto_Gradient_backtracking = function(x, y, m, alpha0, rho, c, lambda, iter_max) {
    N = nrow(x)
    P = ncol(x)
    # initialize all the variables
    beta = matrix(0, P, iter_max)
    nll_ini = Neg_ll(rep(0, P), x, y, m)
    nll = rep(nll_ini, iter_max)
    nll_avg = rep(nll_ini, iter_max)
    nll_ex_avg = rep(nll_ini, iter_max)
    iter = 1

    while (iter < iter_max) {
        index = sample(1:N, 1)  #sample 1 data point from the whole data set
        xi = x[index, , drop = F]
        yi = y[index]
        mi = m[index, ]
        g = Gradient_Cal(beta[, iter], xi, yi, mi)
        desc_direction = -g
        # desc_direction = desc_direction / sqrt(sum(desc_direction^2))
        step = Backtracking(f = Neg_ll, df = g, t = beta[, iter], p = desc_direction,
            alpha0, rho, c, xi, yi, mi)
        beta[, iter + 1] = beta[, iter] + step * desc_direction  #update beta
        nll[iter + 1] = Neg_ll(beta[, iter + 1], x, y, m)  #calculate neg log likelihood for whole data
        nll_xi = Neg_ll(beta[, iter + 1], xi, yi, mi) * N  #calculate neg log likelihood for single dat
        nll_avg[iter + 1] = (nll_xi + iter * nll_avg[iter])/(iter + 1)
        nll_ex_avg[iter + 1] = lambda * nll_xi + (1 - lambda) * nll_ex_avg[iter]
        iter = iter + 1  #keep track of iteration
    }
    return(list(iter, beta[, 1:iter], nll[1:iter], nll_avg[1:iter], nll_ex_avg[1:iter]))
}
```

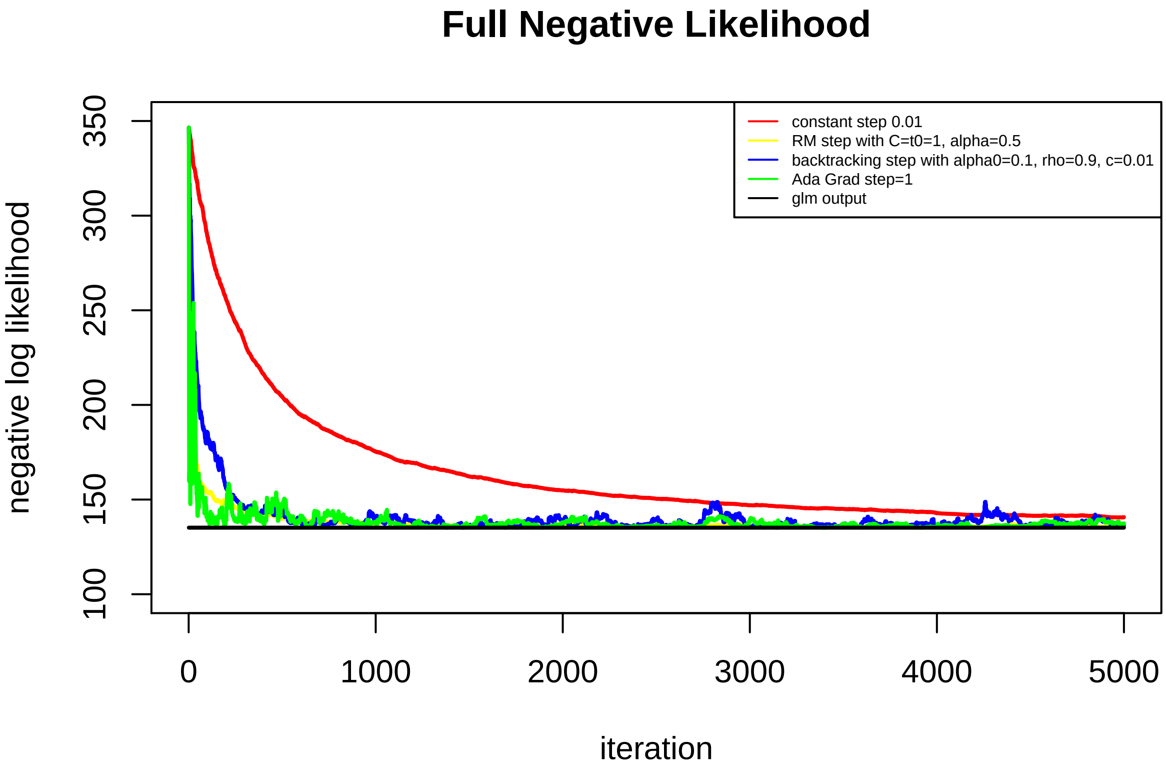
Part B

```r
Ada_Grad = function(x, y, m, step, lambda, iter_max) {
    N = nrow(x)
    P = ncol(x)
    # initialize all the variables
    beta = matrix(0, P, iter_max)
    nll_ini = Neg_ll(rep(0, P), x, y, m)
    nll = rep(nll_ini, iter_max)
    nll_avg = rep(nll_ini, iter_max)
    nll_ex_avg = rep(nll_ini, iter_max)
    iter = 1
    G = rep(0, P)
    while (iter < iter_max) {
        index = sample(1:N, 1)  #sample 1 data point from the whole data set
        xi = x[index, , drop = F]
        yi = y[index]
        mi = m[index, ]
        g = Gradient_Cal(beta[, iter], xi, yi, mi)
        G = G + g^2
        beta[, iter + 1] = beta[, iter] - step * g/(sqrt(G) + 1e-08)  #update beta
        nll[iter + 1] = Neg_ll(beta[, iter + 1], x, y, m)  #calculate neg log likelihood for whole data
        nll_xi = Neg_ll(beta[, iter + 1], xi, yi, mi) * N  #calculate neg log likelihood for single dat
        nll_avg[iter + 1] = (nll_xi + iter * nll_avg[iter])/(iter + 1)
        nll_ex_avg[iter + 1] = lambda * nll_xi + (1 - lambda) * nll_ex_avg[iter]
        iter = iter + 1  #keep track of iteration
    }
    return(list(iter, beta[, 1:iter], nll[1:iter], nll_avg[1:iter], nll_ex_avg[1:iter]))
}
```

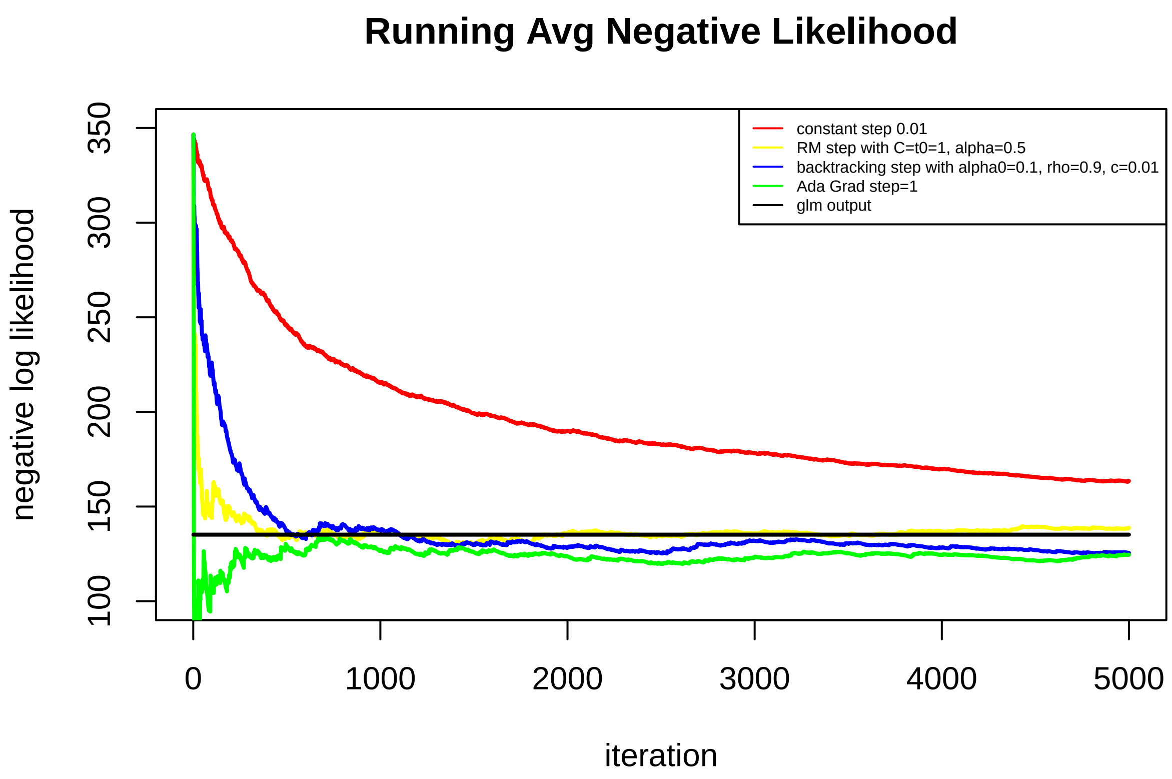
```
N = 500
P = 3
X_sim = as.matrix(cbind(rep(1, N), rnorm(N), rnorm(N)))
beta_sim = c(1, 2, -3)
W_sim = 1/(1 + exp(-X_sim %*% beta_sim))
Y_sim = matrix(rbinom(N, 1, W_sim), N, 1)
M_sim = matrix(1, N, 1)

# run glm and save output for comparision later
glm_sim = glm(Y_sim ~ X_sim - 1, family = "binomial")
result_glm = as.vector(glm_sim$coefficients)
nll_glm = Neg_ll(result_glm, X_sim, Y_sim, M_sim)
```

```
# stochastic gradient decsent with different step size
Iter_Max = 5000

Step_constant = rep(0.01, Iter_Max)

Step_RM = rep(0, Iter_Max)
for (i in 1:Iter_Max) {
    Step_RM[i] = RM_Step_Cal(C = 1, t = i, t0 = 1, alpha = 0.5)
}


result_sgd_constant = Sto_Gradient_Desc(x = X_sim, y = Y_sim, m = M_sim, step = Step_constant,
    lambda = 0.01, iter_max = Iter_Max)

result_sgd_rm = Sto_Gradient_Desc(x = X_sim, y = Y_sim, m = M_sim, step = Step_RM,
    lambda = 0.01, iter_max = Iter_Max)

result_sgd_backtracking = Sto_Gradient_backtracking(x = X_sim, y = Y_sim, m = M_sim,
    alpha0 = 0.1, rho = 0.9, c = 0.01, lambda = 0.01, iter_max = Iter_Max)

result_ada_grad = Ada_Grad(x = X_sim, y = Y_sim, m = M_sim, step = 1, lambda = 0.01,
    iter_max = Iter_Max)
```

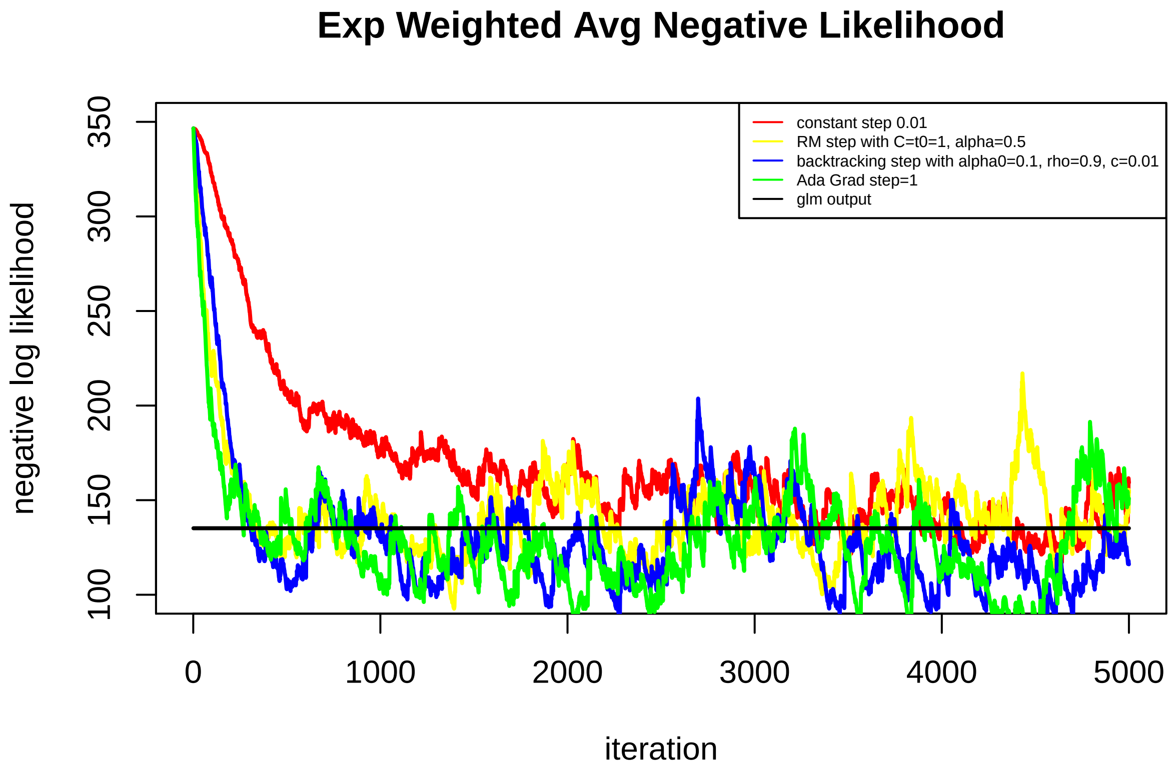
```
### plot likelihood to compare convergence for different step size

plot_x = 1:Iter_Max
plot(x = plot_x, y = result_sgd_constant[[3]], type = "l", xlab = "iteration",
    ylab = "negative log likelihood", ylim = c(100, 350), col = "red", lwd = 2,
    main = "Full Negative Likelihood")
lines(x = plot_x, y = result_sgd_rm[[3]], col = "yellow", lwd = 2)
lines(x = plot_x, y = result_sgd_backtracking[[3]], col = "blue", lwd = 2)
lines(x = plot_x, y = result_ada_grad[[3]], col = "green", lwd = 2)
lines(x = c(1, Iter_Max), y = rep(nll_glm, 2), col = "black", lwd = 2)
legend("topright", cex = 0.5, c("constant step 0.01", "RM step with C=t0=1, alpha=0.5",
    "backtracking step with alpha0=0.1, rho=0.9, c=0.01", "Ada Grad step=1",
    "glm output"), col = c("red", "yellow", "blue", "green", "black"), lty = c(1,
    1, 1, 1, 1))
```



```
plot(x = plot_x, y = result_sgd_constant[[4]], type = "l", xlab = "iteration",
    ylab = "negative log likelihood", ylim = c(100, 350), col = "red", lwd = 2,
    main = "Running Avg Negative Likelihood")
lines(x = plot_x, y = result_sgd_rm[[4]], col = "yellow", lwd = 2)
lines(x = plot_x, y = result_sgd_backtracking[[4]], col = "blue", lwd = 2)
lines(x = plot_x, y = result_ada_grad[[4]], col = "green", lwd = 2)
lines(x = c(1, Iter_Max), y = rep(nll_glm, 2), col = "black", lwd = 2)
legend("topright", cex = 0.5, c("constant step 0.01", "RM step with C=t0=1, alpha=0.5",
    "backtracking step with alpha0=0.1, rho=0.9, c=0.01", "Ada Grad step=1",
    "glm output"), col = c("red", "yellow", "blue", "green", "black"), lty = c(1,
    1, 1, 1, 1))
```



```
plot(x = plot_x, y = result_sgd_constant[[5]], type = "l", xlab = "iteration",
    ylab = "negative log likelihood", ylim = c(100, 350), col = "red", lwd = 2,
    main = "Exp Weighted Avg Negative Likelihood")
lines(x = plot_x, y = result_sgd_rm[[5]], col = "yellow", lwd = 2)
lines(x = plot_x, y = result_sgd_backtracking[[5]], col = "blue", lwd = 2)
lines(x = plot_x, y = result_ada_grad[[5]], col = "green", lwd = 2)
lines(x = c(1, Iter_Max), y = rep(nll_glm, 2), col = "black", lwd = 2)
legend("topright", cex = 0.5, c("constant step 0.01", "RM step with C=t0=1, alpha=0.5",
    "backtracking step with alpha0=0.1, rho=0.9, c=0.01", "Ada Grad step=1",
    "glm output"), col = c("red", "yellow", "blue", "green", "black"), lty = c(1,
    1, 1, 1, 1))
```



Part C

```
library(Rcpp)
library(RcppEigen)
library(Matrix)
sourceCpp(file = "C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment4/Ada_Grad.cpp")
sourceCpp(file = "C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment4/sgdlogit.cpp")

x = readRDS("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment4/url_X.rds")
y = readRDS("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment4/url_y.rds")
x = Matrix(x, sparse = TRUE)
x = t(x)
beta0 = rep(0, nrow(x))


time = proc.time()
result_JS = sparsesgd_logit(x, y, M, eta = 0.01, npass = 1, beta0, lambda = 0.1,
    discount = 0.01)
time_JS = proc.time() - time
plot(result_JS$nll_tracker, type = "l", log = "xy", xlab = "iteration", ylab = "Exp Weighted Avg negati
    main = "Stochastic Gradient Decsent JS code")


time = proc.time()
result_SC = Ada_Grad_sparse(x, y, beta0, step = 1e-08, npass = 1, alpha = 0.01,
    lambda = 0.1)
time_SC = proc.time() - time
plot(result_SC$nll_ex_avg, type = "l", log = "xy", xlab = "iteration", ylab = "Exp Weighted Avg negative
    main = "Stochastic Gradient Decsent SC code")

# compare results of James Scott code with mine code
Output = matrix(0, 6, 2)
dimnames(Output) = list(c("run time", "intercept", "min beta", "median beta",
    "mean beta", "max beta"), c("JS code", "SC code"))
Output[, 1] = c(time_JS[3], result_JS$alpha, min(result_JS$beta), median(result_JS$beta),
    mean(result_JS$beta), max(result_JS$beta))
Output[, 2] = c(time_SC[3], result_SC$intercept, min(result_SC$beta), median(result_SC$beta),
    mean(result_SC$beta), max(result_SC$beta))
Output
```