SDS 385: Exercises 2 - Online Learning

September 19, 2016

Professor Scott

Spencer Woody

Problem 1

(A) From the previous excercise, we have the gradient of β

$$\nabla \ell(\beta) = \sum_{i=1}^{N} (m_i w_i - y_i) x_i. \tag{1}$$

We can think of $m_i w_i$ as the fitted value of y_i, or $\hat{y}_i$, when given β, so the gradient becomes

$$\nabla \ell(\beta) = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i \tag{2}$$

$$= \sum_{i=1}^{N} g_i(\beta) \tag{3}$$

$$g_i(\beta) = (\hat{y}_i - y_i) x_i. \tag{4}$$

(B)

$$\mathrm{E}(n g_i(\beta)) = n \mathrm{E}(g_i(\beta)) \tag{5}$$

In this expectation, the only random variable is i because the data X and y and the coefficients β are all fixed. The variable i is a random draw so it follows a discrete uniform distribution such that

$$P(i = j) = \begin{cases} \frac{1}{n} & j \in \{1, 2, \ldots, n\} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Then we can compute the expectation.

$$\mathrm{E}(g_i(\beta)) = \sum_{j=1}^{n} g_j(\beta) P(i = j) \tag{7}$$

$$= \sum_{j=1}^{n} g_j(\beta) \frac{1}{n} \tag{8}$$

$$= \frac{1}{n} \sum_{j=1}^{n} g_j(\beta) \tag{9}$$

$$= \frac{1}{n} \nabla \ell(\beta), \tag{10}$$

$$\Rightarrow \mathrm{E}(n g_i(\beta)) = n \mathrm{E}(g_i(\beta)) = n \frac{1}{n} \nabla \ell(\beta) = \nabla \ell(\beta) \tag{11}$$

(C) In this case, we perform SGD with $100,000$ iterations of samples with replacement and a step size α of 0.0001. Our initial guess for β, $\hat{\beta}_0$ is some distance from the result of β from Newton's method. That distance is 5 plus some noise from the Exp(1) distribution in either the positive or negative direction. Trace plots of all elements of $\hat{\beta}_k$ for each iteration are shown below in Figure 1. The red horizontal lines represent the values of $\hat{\beta}$ obtained from Newton's method.

Also shown is trace plot of the exponential moving average (EMA) over the previous all iterations of the log-likelihood in Figure 2. Note that at each iteration, we are only computing the "partial" likelihood, that is, the likelihood using the randomly chosen data point and the newly computed $\hat{\beta}$ at that iteration. We use the EMA to reduce the level of noise in our log-likelihood plot. The EMA at the kth iteration, S_k, when the ith data point is sampled is calculated recursively by the formula

$S_k = a \times \ell_i(\hat{\beta}_k) + (1-a) \times S_{k-1}$ with $S_1 = \ell_i(\beta_1)$, $a \in (0,1)$. We choose a small value of $a = 0.5/n$ to reduce noise sufficiently and produce an interpretable graph.

Interestingly, our likelihood seems to converge to stationarity fairly quickly, before the $20,000$th iteration. Most elements of $\hat{\beta}$ converge to their "true" value by around the $10,000$th iteration and then bounce around that value for each subsequent iterations. However, not all of the elements of $\hat{beta}$ reach this point of convergence, specifically $\hat{\beta}_2$ and $\hat{\beta}_5$ shown in Figure 1. Despite this, our likelihood has still converged. This suggests that these coefficients have high variance (i.e., are "insignificant" to our model). $100,000$ iterations is significantly more than the number of iterations used in batch gradient descent (fewer than $50,000$) but this is still pretty remarkable considering that we only touch one data point at a time.

Problem 1 continued on next page...

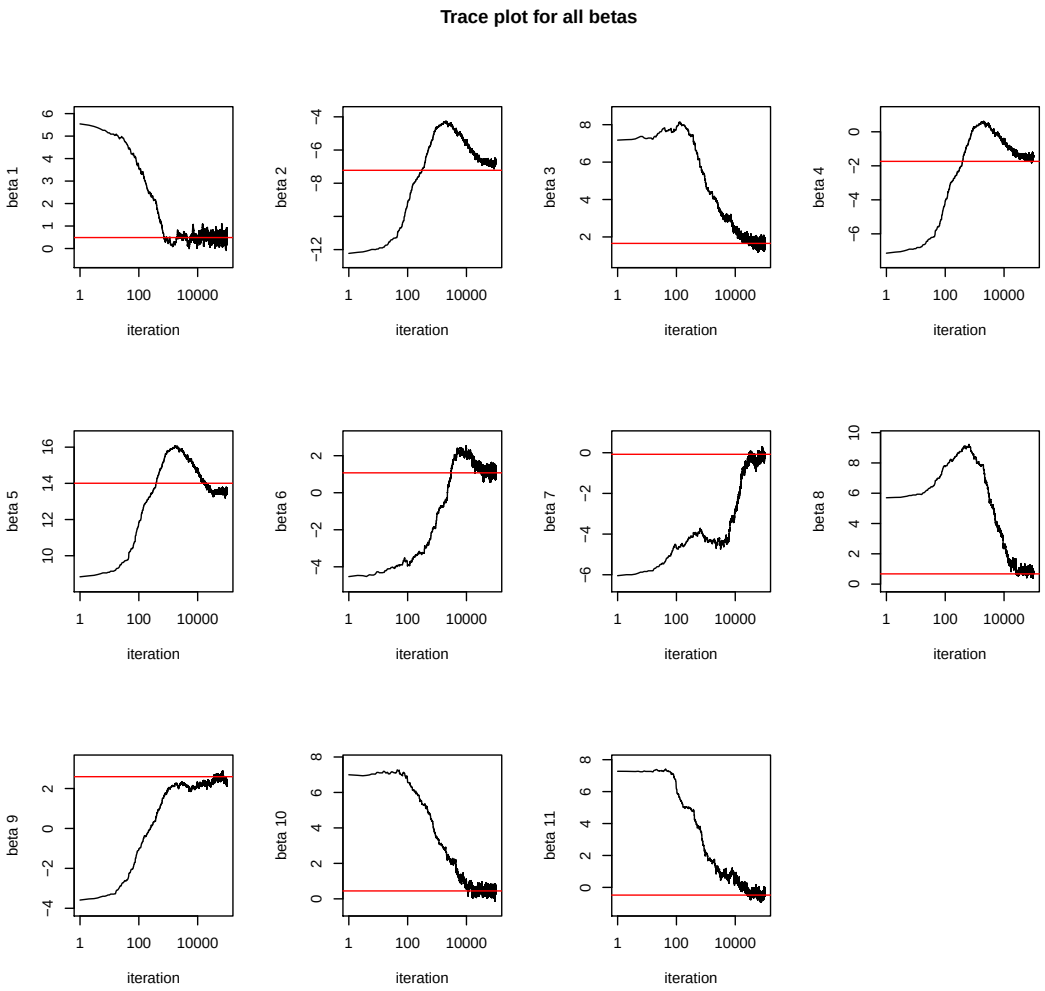

Figure 1: Trace plot of all elements of $\hat{\beta}$ from SGD

$$\hat{\beta}_{t+1} = \hat{\beta}_t - \gamma \nabla \ell_i(\hat{\beta}_t) \tag{12}$$

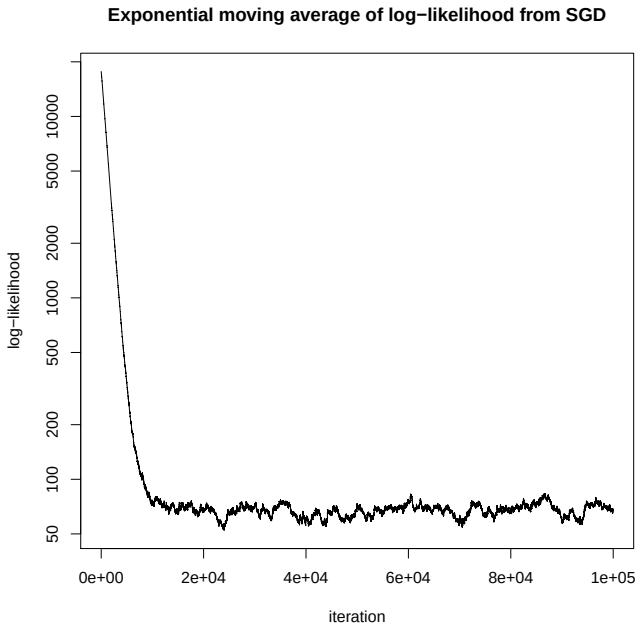

Figure 2: Trace plot of EMA of log-likelihood from SGD

(D) Now we substitute our fixed step size for a step size that decays over time. Here we use the Robbins-Monro rule for decaying step size, that is,

$$\gamma t = C(t + t_0)^{-alpha}. \tag{13}$$

In this case, we choose $C = 0.1$, $\alpha = 0.6$, and $t_0 = 1$. Trace plots for all elements of estimates $\hat{\beta}$ are shown in Figure 3. We have a measured . The rationale for having a decaying step size is that we eventually get closer to the "true" values of $\hat{\beta}$ and then we our step sizes should decay so that we do not deviate too far from the . However, it appears to be the case for some of the $\hat{\beta}$s that our step size decays too rapidly before we reach the true values. For example, $\hat{\beta}_2$ in Figure 3 never gets close to its true value but the step size decays to the point where it can't jump back up. However, for the most part we have a tight distribution around the true values. Figure 4 shows the EMA for the partial log-likelihood.

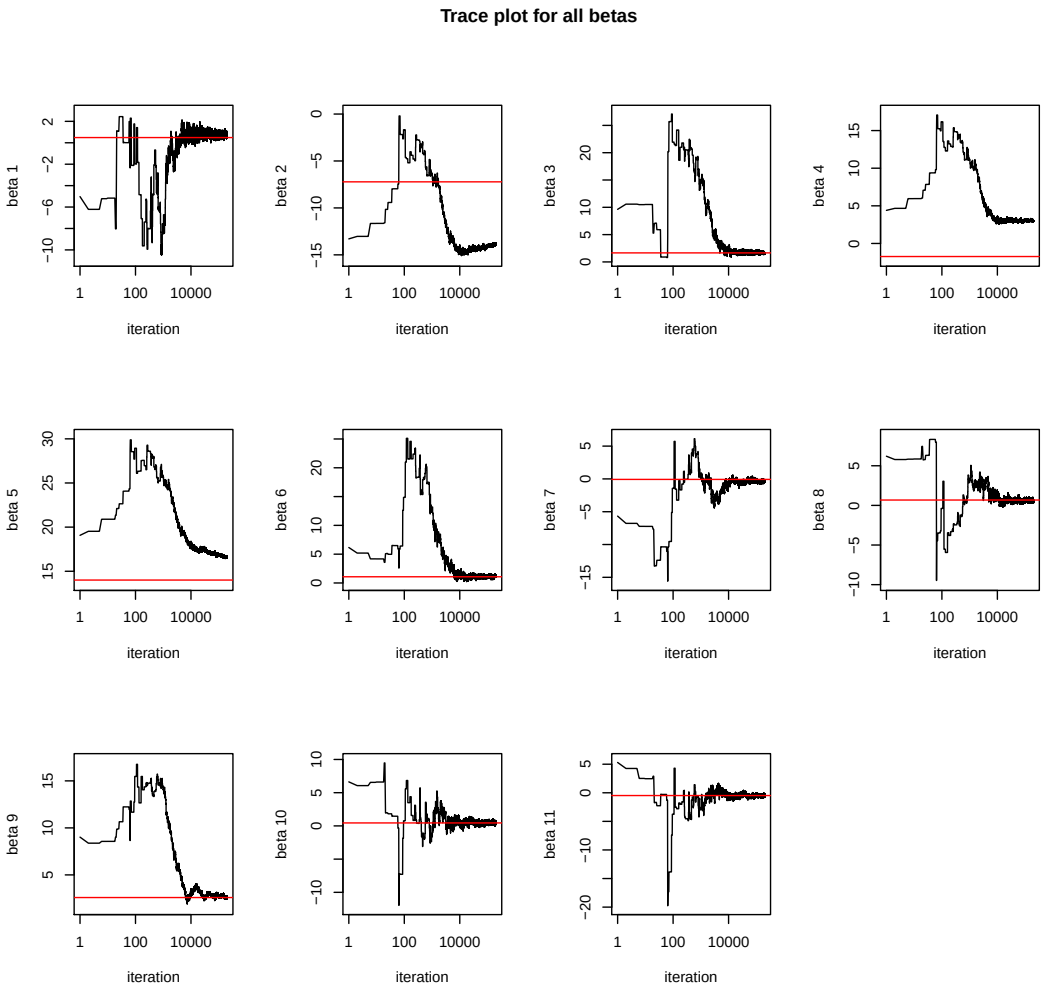

Figure 3: Trace plot of all elements of $\hat{\beta}$ from SGD with decaying step

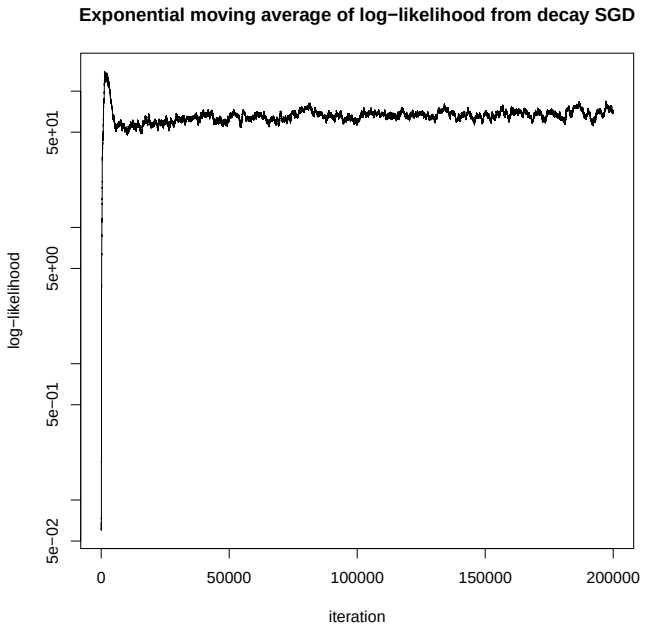

Figure 4: Trace plot of EMA of log-likelihood from SGD with decaying step

(E) Now, we incorporate. Remember that $\hat{\beta}$ eventually reaches a point of converging to the true values and then oscillates between that true value. Then, to get a final estimate, it makes sense to average out the estimates after some burn-in period with the Polyak-Ruppert (PR) averaging formula

$$\bar{\beta}^{(t)} = \frac{1}{t}\sum_{k=0}^{t-1} \beta^{(k)}. \tag{14}$$

Figure 5 shows the result of PR-averaging with burn-in of the first two-thirds of estimates for SGD with a fixed stepsize, and Figure 6 shows a trace plot of the corresponding log-likelihood for each iterated $\bar{\beta}^{(t)}$. We achieve a nice smooth decrease in the log-likelihood, and the averaging scheme gets us closer to the true values for most elements in $\hat{\beta}$, provided that we achieved convergence for those elements. Figure 7 and Figure 8 are the corresponding plots for SGD with decaying step size from the previous section.

R code is shown in the appendix of this report.

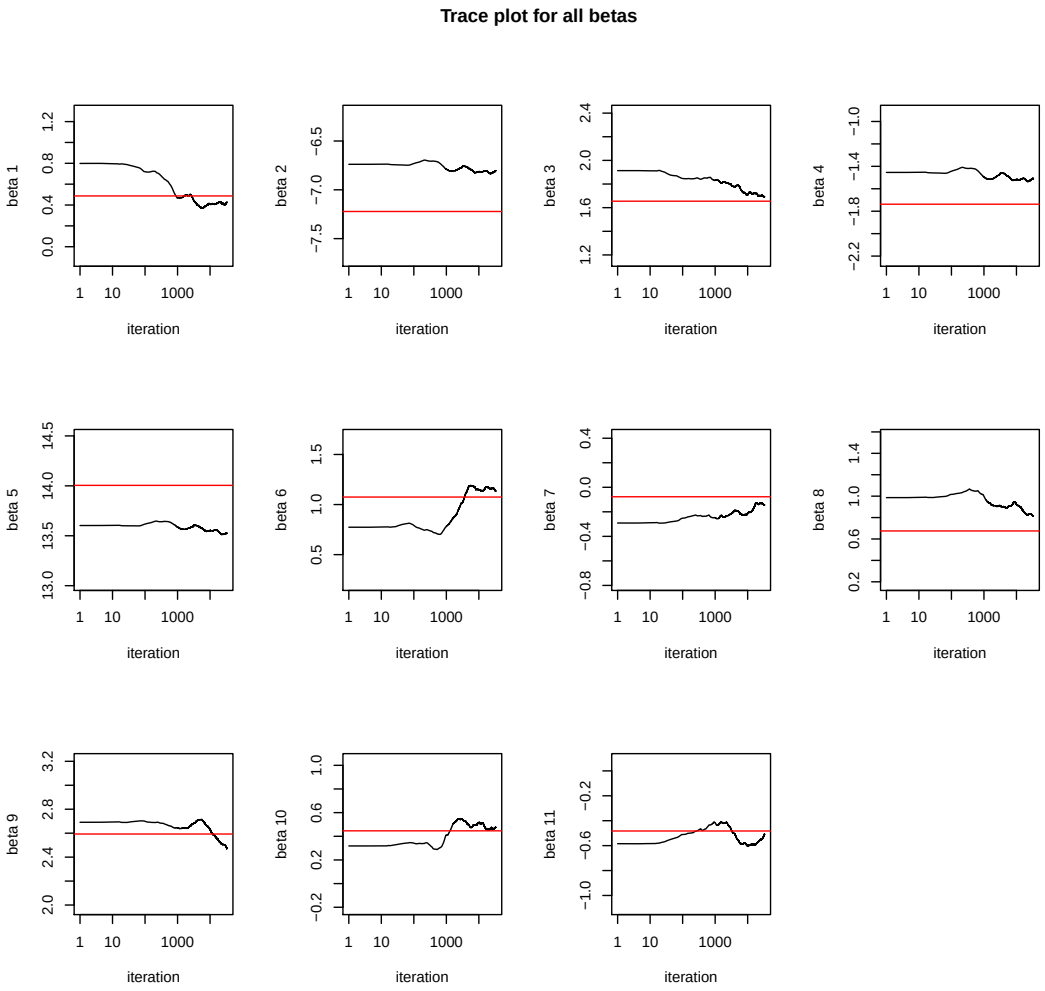

Figure 5: Trace plot of all elements of $\hat{\beta}$ from SGD with PR-averaging

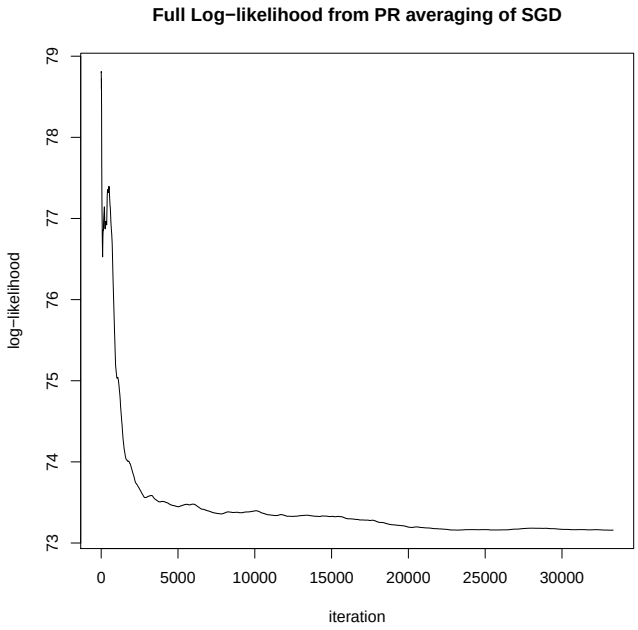

Figure 6: Trace plot of EMA of log-likelihood from SGD with PR-averaging

```
############################################################
######### Created by Spencer Woody on 13 Sep 2016 #########
############################################################

library(TTR)

# Read in data file, scale X (y = 1 represents a malignant tumor)

wdbc <- read.csv("wdbc.csv", header = FALSE)

X <- as.matrix(wdbc[, 3:12])
X <- scale(X)
X <- cbind(rep(1, nrow(X)), X)

y <- wdbc[, 2]
y <- y == "M"
m.i <- 1

# # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # #

# Function for computing w.i (logit transform of Xtbeta)

comp.wi <- function (X, beta) {
    wi <- 1 / (1 + exp(-X %*% beta))
    return(wi)
}
```

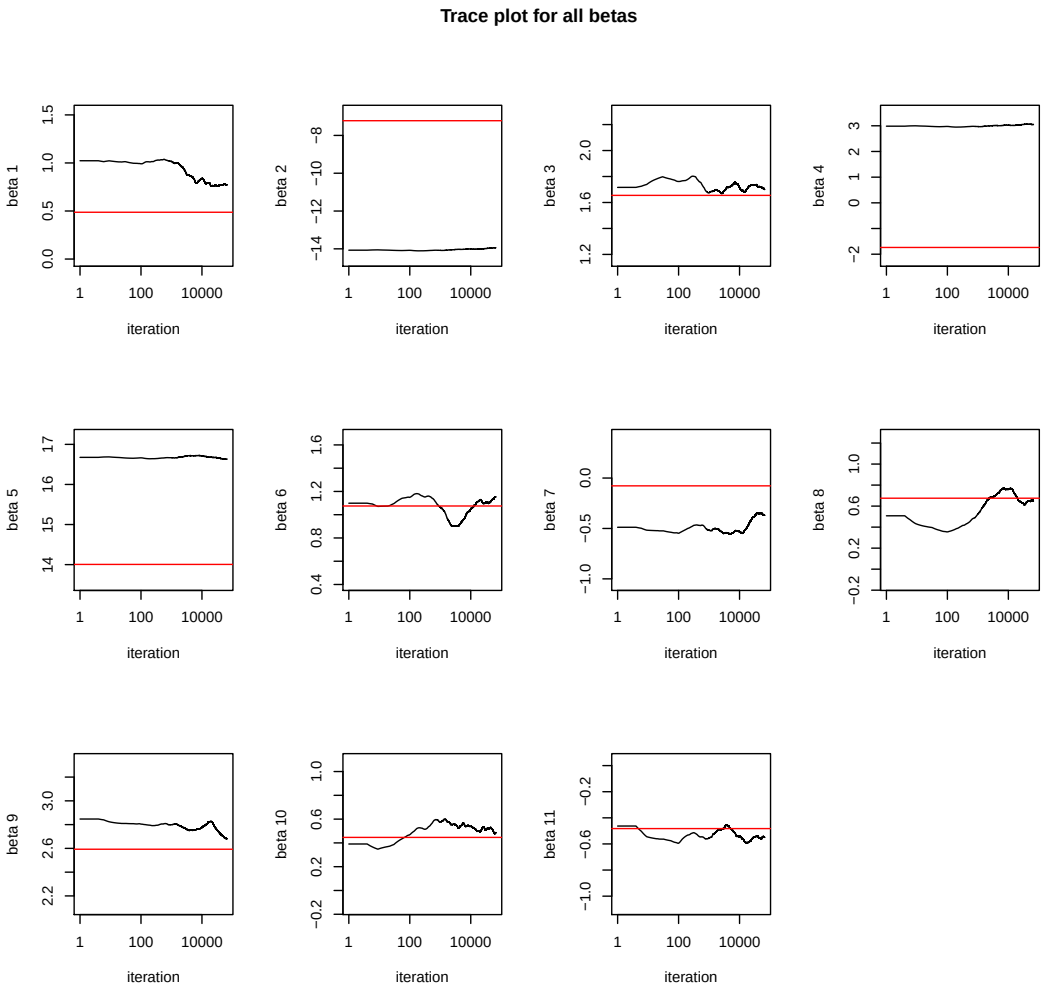

Figure 7: Trace plot of all elements of $\hat{\beta}$ from SGD with decaying steps and PR-averaging

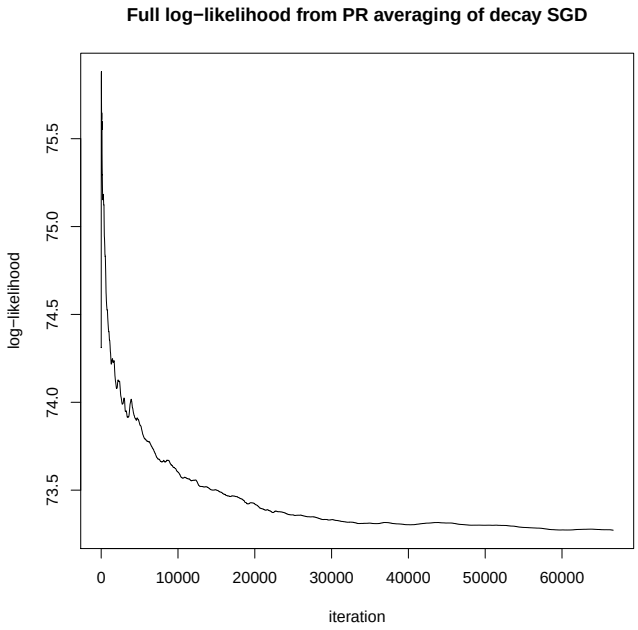

Figure 8: Trace plot of EMA of log-likelihood from SGD with decaying steps and PR-averaging

```
# Function for computing full likelihood

loglik <- function(beta, y, X, m.i) {
    loglik <- apply((m.i - y) * (X %*% beta) + m.i * log(1 + exp(-X %*% beta)), 2, sum)
    return(loglik)
}

# Function for computing gradient of likelihood

grad.loglik <- function(beta, y, X, mi){
  grad <- array(NA, dim = length(beta))
  wi   <- comp.wi(X, beta)
  grad <- apply(X*as.numeric(mi * wi - y), 2, sum)
  return(grad)
}

# # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # #

### NEWTON'S METHOD (results are referred to as beta.N)

beta.N <- as.matrix(rep(0, ncol(X)))

newton.steps <- 10

for (step in 1:newton.steps) {
    Newton.wi <- as.numeric(comp.wi(X, beta.N))
```

```
    Hessian   <- t(X) %*% diag(Newton.wi*(1-Newton.wi)) %*% X
    beta.N    <- beta.N - solve(Hessian, grad.loglik(beta.N, y, X, m.i))
}

# Function for making traceplots of all betas

graph.betatrace <- function(beta, beta.N, graphname) {
    num.vars <- nrow(beta)
    n.graphrows <- floor(sqrt(num.vars))
    n.graphcols <- ceiling(num.vars / n.graphrows)
    pdf(graphname, width = n.graphcols * 2, height = n.graphrows * 2.5)
    par(mfrow = c(n.graphrows, n.graphcols), oma=c(0,0,2,0))
    for (j in 1:num.vars) {
        plot(beta[j, ],
            xlab = "iteration",
            ylab = paste("beta", sprintf("%i", j)),
            type = "l",
            ylim = c(min(beta.N[j], min(beta[j, ])) - 0.5,
            max(beta.N[j], max(beta[j, ])) + 0.5),
            log = "x"
            )
        abline(h = beta.N[j], col = "red")
    }
    title("Trace plot for all betas", outer = TRUE)
    dev.off()
}

# # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # #

# Stochastic gradient

SGD <- function(X, y, m.i, beta.init,
                n.iterSGD, step.size, a, burn.in) {
    #' Perform stochastic gradient descent for a binomial logistic regression
    #'
    #' @param X  N by P matrix of covariate data
    #' @param y  P-vector of responses
    #' @param m.i  n-parameter of binomial (1 for case of binary logistic)
    #' @param beta.init  Initial guess for coefficients (betas)
    #' @param n.iterSGD  Number of iterations to loop through
    #' @param step.size  Constant step size
    #' @param a  Coefficient for weighting decrease for calculating loglik EMA
    #               (larger a discounts older values faster)
    #' @param burn.in  Number of betas to burn before starting PR average
    #
    #' @return A list of four items
    #               1. A matrix, each column is an iteration of computed betas.
    #               2. A vector of EMA of loglikelihood of one data point for
    #                    each iteration
    #               3. A matrix of PR-averaged betas after burn in
    #               4. A vector of the full likelihood for each beta in No. 3
```

```
    #
        # Create zero matrix to store iterative values of beta; initialize beta
        betaSGD <- matrix(rep(NA, ncol(X) * (n.iterSGD+ 1)), nrow = ncol(X))
        betaSGD[, 1] <- beta.init

        EMAloglik <- rep(NA, n.iterSGD)

        PRbeta <- matrix(rep(NA, ncol(X) * (n.iterSGD - burn.in)), nrow = ncol(X))
        EMA.PR <- rep(NA, n.iterSGD - burn.in)
        #
        for (step in 1:n.iterSGD) {
            # Draw random sample of single row of data (with replacement)
            i <- sample(nrow(X), 1)


            # Compute w.i, fitted value of p-parameter of binomial
            w.i <- 1 / (1 + exp(-crossprod(X[i, ], betaSGD[, step])))

            # Compute gradient, the descent direction
            grad <- nrow(X) * (m.i * w.i - y[i]) * X[i, ]

            # Next set of betas
            newbeta <- betaSGD[, step] - step.size * grad
            betaSGD[, step + 1] <- newbeta
            # EMA of loglik
            step.loglik <- loglik(newbeta, y[i], X[i, ], m.i)
            #
            if (step == 1) {
                EMAloglik[step] <- step.loglik
            }
            else {
                EMAloglik[step] <- a * step.loglik + (1 - a) * EMAloglik[step-1]
            }

            if (step > burn.in) {
                j <- step - burn.in
                if (j == 1) {
                    PRbeta[, j] <- newbeta
                }
                else {
                    PRbeta[, j] <- (PRbeta[, j - 1] * (j - 1) + newbeta ) / j
                }
                EMA.PR[j] <- loglik(PRbeta[, j], y, X, m.i)
            }
    }
        mylist <- list(betaSGD, EMAloglik, PRbeta, EMA.PR)
        return(mylist)
}
```

```r
# Create initial values of beta which are some distance away from "true" values
# of beta computed from Newton's method. At least 5 units away plus some
# exponential noise, in either positive or negative direction
beta0 <- beta.N + (-1) ^ rbinom(ncol(X), 1, 0.5) * (5 + rexp(ncol(X), rate = 1))

# Set number of iterations and stepsize
numiter <- 1e5
my.stepsize <- 0.0001
a <- 0.5 / nrow(X)
burn <- floor(numiter / 1.5)

# Perform SGD, graph trace plot of betas
sgd.1 <- SGD(X, y, m.i, beta0, numiter, my.stepsize, a, burn)

beta.SGD  <- sgd.1[[1]]
EMAloglik <- sgd.1[[2]]
PR.beta   <- sgd.1[[3]]
PR.loglik <- sgd.1[[4]]

graph.betatrace(beta.SGD, beta.N, "SGDbetatrace.pdf")

pdf("SGDloglik.pdf")
plot(nrow(X)*EMAloglik,
    log = "y",
    main = "Exponential moving average of log-likelihood from SGD",
    ylab = "log-likelihood",
    xlab = "iteration", type = "l")
dev.off()

graph.betatrace(PR.beta, beta.N, "SGDbetatracePR.pdf")

pdf("SGDloglikPR.pdf")
plot(PR.loglik,
    main = "Full Log-likelihood from PR averaging of SGD",
    ylab = "log-likelihood",
    xlab = "iteration", type = "l")
dev.off()

print(PR.beta[, ncol(PR.beta)])

# > print(PR.beta[, ncol(PR.beta)])
#  [1]  0.4253684 -6.8026923  1.6917163 -1.5052068 13.5295289  1.1387835
#  [7] -0.1433198  0.8158190  2.4698268  0.4751768 -0.5073092

# # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # #

# Decaying steps


SGD.decay <- function(X, y, m.i, beta.init,
```

```
                n.iterSGD, C, t.0, alpha,
                a, burn.in) {
    #' Perform stochastic gradient descent for a binomial logistic regression
    #'
    #' @param X  N by P matrix of covariate data
    #' @param y  P-vector of responses
    #' @param m.i  n-parameter of binomial (1 for case of binary logistic)
    #' @param beta.init  Initial guess for coefficients (betas)
    #' @param n.iterSGD  Number of iterations to loop through
    #' @param C  Constant C in Robbins-Monro rule
    #' @param t.0  Constant t.0 in Robbins-Monro rule
    #' @param alpha Constant alpha in Robbins-Monro rule
    #' @param a  Coefficient for weighting decrease for calculating loglik EMA
    #               (larger a discounts older values faster)
    #' @param burn.in  Number of betas to burn before starting PR average
    #
    #' @return A list of four items
    #               1. A matrix, each column is an iteration of computed betas.
    #               2. A vector of EMA of loglikelihood of one data point for
    #                    each iteration
    #               3. A matrix of PR-averaged betas after burn in
    #               4. A vector of the full likelihood for each beta in No. 3
    #
        # Create zero matrix to store iterative values of beta; initialize beta
        betaSGD <- matrix(rep(NA, ncol(X) * (n.iterSGD+ 1)), nrow = ncol(X))
        betaSGD[, 1] <- beta.init

        EMAloglik <- rep(NA, n.iterSGD)

        PRbeta <- matrix(rep(NA, ncol(X) * (n.iterSGD - burn.in)), nrow = ncol(X))
        EMA.PR <- rep(NA, n.iterSGD - burn.in)
        #
        for (step in 1:n.iterSGD) {
            # Draw random sample of single row of data (with replacement)
            i <- sample(nrow(X), 1)


            # Compute w.i, fitted value of p-parameter of binomial
            w.i <- 1 / (1 + exp(-crossprod(X[i, ], betaSGD[, step])))

            # Compute gradient, the descent direction
            grad <- nrow(X) * (m.i * w.i - y[i]) * X[i, ]

            # Calculate decaying step size
            step.decay <- C * (step + t.0) ^ -alpha

            # Next set of betas
            newbeta <- betaSGD[, step] - step.decay * grad
            betaSGD[, step + 1] <- newbeta
            # EMA of loglik
            step.loglik <- loglik(newbeta, y[i], X[i, ], m.i)
            #
            if (step == 1) {
```

```
                EMAloglik[step] <- step.loglik
            }
            else {
                EMAloglik[step] <- a * step.loglik + (1 - a) * EMAloglik[step-1]
            }

            if (step > burn.in) {
                j <- step - burn.in
                if (j == 1) {
                    PRbeta[, j] <- newbeta
                }
                else {
                    PRbeta[, j] <- (PRbeta[, j - 1] * (j - 1) + newbeta ) / j
                }
                EMA.PR[j] <- loglik(PRbeta[, j], y, X, m.i)
            }
    }
        mylist <- list(betaSGD, EMAloglik, PRbeta, EMA.PR)
        return(mylist)
}

# SGD.decay <- function(X, y, m.i, beta.init,
#               n.iterSGD, C, t.0, alpha,
#               a, burn.in)

beta1 <- beta.N + (-1) ^ rbinom(ncol(X), 1, 0.5) * (5 + rexp(ncol(X), rate = 1))

# Define number of iterations & parameters of Robbins-Monro


numiter2 <- 2e5
a2 <- 0.5 / nrow(X)
burn2 <- floor(numiter2 / 1.5)

C     <-  0.1
t.0   <-  1
alpha <-  0.6

decay.1 <- SGD.decay(X, y, m.i, beta1, numiter2, C, t.0, alpha, a2, burn2)



dbeta.SGD  <- decay.1[[1]]
dEMAloglik <- decay.1[[2]]
dPR.beta   <- decay.1[[3]]
dPR.loglik <- decay.1[[4]]

graph.betatrace(dbeta.SGD, beta.N, "dSGDbetatrace.pdf")

pdf("dSGDloglik.pdf")
plot(nrow(X)*dEMAloglik[10:length(dEMAloglik)],
    log = "y",
    main = "Exponential moving average of log-likelihood from decay SGD",
```

```
    ylab = "log-likelihood",
    xlab = "iteration", type = "l")
dev.off()

graph.betatrace(dPR.beta, beta.N, "dSGDbetatracePR.pdf")

pdf("dSGDloglikPR.pdf")
plot(dPR.loglik,
    main = "Full log-likelihood from PR averaging of decay SGD",
    ylab = "log-likelihood",
    xlab = "iteration", type = "l")
dev.off()



# Running average of beta with decaying steps

# END
```