SDS 385 Homework 1

Qiaohui Lin & Yuguang Yue

17th Sep, 2018

1 Question 1 Convex functions

Part(1): Use definiition to prove e^x

$$\begin{aligned} f(tx+(1-t)y) - tf(x) - (1-t)f(y) &= e^{tx+(1-t)y} - te^x - (1-t)e^y \\ &= e^y e^{(x-y)t} - (e^x - e^y)t - e^y \\ &= g(t) \end{aligned}$$

$$g(0) = 0$$

suppose x¡y,

$$\begin{aligned} g'(t) &= (1+(x-y)e^{(x-y)t})e^y - e^x \\ &= (1+(x-y)e^{(x-y)t} + e^{x-y})e^y \\ &\text{since } 0 < t < 1,\ x < y \\ &< [1+(x-y) - e^{x-y}]e^y \\ &\text{since } 1 + x < e^x \\ &< 0 \end{aligned}$$

Thus, $g(t) \leq 0$ on $t \in [0,1]$
e^x is convex.

Part(2): Affine transform

$$g(x) = f(Ax+b)$$

$$\begin{aligned} g(tx+(1-t)y) &= f(A(tx+(1-t)y)+b) \\ &= f(t(Ax+b)+(1-t)(Ax+b)) \\ &\leq tf(Ax+b) + (1-t)f(Ax+b) \\ &= tg(x) + (1-t)g(x) \end{aligned}$$

$g(x)$ is convex.

Part(3): Pointwise maxima

$$\begin{aligned} f_1(tx+(1-t)y) &\leq tf_1(x)+(1-t)f_1(y) \\ f_2(tx+(1-t)y) &\leq tf_2(x)+(1-t)f_2(y) \\ ...f_n(tx+(1-t)y) &\leq tf_n(x)+(1-t)f_1(y) \end{aligned}$$

$$f(x) = max(f_1(x), f_2(x), ..f_n(x))$$

$$\begin{aligned} f(tx+(1-t)y) &= max(f_1(tx+(1-t)y)), f_2(tx+(1-t)y)), ..f_n(tx+(1-t)y))) \\ &\leq maxtf_1(x)+(1-t)f_1(y), tf_2(x)+(1-t)f_2(y), ...tf_n(x)+(1-t)f_n(y) \\ &\leq tmax(f_1(x), f_2(x), ...f_n(x))+(1-t)max(f_1(y), f_2(y), ...f_n(y)) \end{aligned}$$

$= tf(x)+(1-t)f(x)$

f(x) is convex.

Part(d):
log-likelihood:

$$\begin{aligned} L &= (\frac{1}{1+e^{-\theta^T x}})^{\sum y_i}(\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}})^{n-\sum y_i} \\ l &= logL = -nlog(1+e^{-\theta^T x})+(n-\sum y_i)loge^{-\theta^T x} \end{aligned}$$

Show that loglikelihood is concave:

$$\begin{aligned} \frac{dl}{dx_j} &= n\frac{\theta_j e^{-\theta^T x}}{1+e^{-\theta^T x}}+(n-\sum y_i)\frac{-\theta_j e^{-\theta^T x}}{e^{-\theta^T x}} \\ &= n\frac{\theta_j e^{-\theta^T x}}{1+e^{-\theta^T x}}+(n-\sum y_i)(-\theta_j) \\ \frac{d^2 l}{dx_j^2} &= -\frac{n\theta_j^2 e^{-theta^T x}}{(1+e^{-\theta^T x})} < 0 \\ \frac{d^2 l}{dx_i dx_j} &= -\frac{n\theta_j\theta_i e^{-theta^T x}}{(1+e^{-\theta^T x})} < 0 \end{aligned}$$

loglikelihood is concave.

2 Question 2: Convergence of GD

Part(a): show that $f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$
Proof: first, construct $g(x) = \frac{L}{2}x^Tx - f(x)$, show g(x) is convex.

$$\nabla g(x) = Lx - \nabla f(x)$$

$$\begin{aligned}(\nabla g(x) - \nabla g(y))(x-y) &= (Lx - \nabla f(x) - Ly + \nabla f(y))(x-y) \\ &= L(x-y)^2 - (\nabla f(x) - \nabla f(y))(x-y) \\ &\text{as } ||\nabla f(x) - \nabla f(y)|| \leq L||x-y||_2 \\ &\geq L(x-y)^2 - L||x-y||_2 * (x-y) \\ &\geq 0\end{aligned}$$

Thus, $g(x)$ is convex. For convex function $g(x)$ we have $g(y) \geq g(x) + \nabla g(x)(y-x)$. Side proof:

$$\begin{aligned}g(ty + (1-t)x \leq tg(y) + (1-t)g(x) \\ g(t(y-x)+x) \leq t(g(y) - g(x)) + g(x) \\ \frac{g(t(y-x)+x) - g(x)}{t} \leq g(y) - g(x) \\ t \to 0, \nabla(g(x))(y-x) \leq g(y) - g(x) \\ g(y) \geq g(x) + \nabla g(x)(y-x)\end{aligned}$$

Thus follows,

$$\begin{aligned}g(y) \geq g(x) + \nabla g(x)(y-x) \\ \frac{L}{2}y^Ty - f(y) \geq \frac{L}{2}x^Tx - f(x) + (Lx - \nabla f(x))(y-x) \\ f(y) \leq f(x) + \nabla f(x)(y-x) + \frac{1}{2}Ly^Ty + \frac{1}{2}Lx^Tx - Lxy \\ f(y) \leq f(x) + \nabla f(x)(y-x) + \frac{L}{2}||x-y||_2\end{aligned}$$

Proved.

Part(b): $y' = x - t\nabla f(x)$. Show: $f(y') \leq f(x) - t\|\nabla f(x)\|^2/2$
Proof: $y' = x - t\nabla f(x)$, then from part(a) we have

$$
\begin{aligned}
f(y') &\leq f(x) + \nabla f(x)^T(-t\nabla f(x)) + \frac{L}{2}||t\nabla f(x)||^2 \\
&\leq f(x) - (t - \frac{L}{2}t^2)||\nabla f(x)||^2 \\
&(\text{since } t < \frac{1}{L}, \text{ we have } t - \frac{L}{2}t^2 > \frac{1}{2}t) \\
&\leq f(x) - \frac{t}{2}||\nabla f(x)||^2
\end{aligned}
$$

Proved.

Part(3):show that $f(y') - f(x^*) \leq \frac{1}{2t}(\|x - x^*\|^2 - \|y' - x^*\|^2)$
Proof: we have:

$$
\begin{aligned}
f(y) &\leq f(x) + \nabla f(x^T)(y - x) + \frac{L}{2}||y - x||^2 \\
f(x) &\leq f(x^*) + \nabla f(x^*)(y - x) + \frac{L}{2}||x - x^*||
\end{aligned}
$$

adding up,

$$
\begin{aligned}
f(y') - f(x^*) &\leq \nabla f(x^T)(y - x) + \frac{L}{2}||y' - x||^2 + \frac{L}{2}||x - x^*|| \\
&\leq \nabla f(x^T)(-t\nabla f(x)) + \frac{L}{2}||y' - x||^2 + \frac{L}{2}||x - x^*|| \\
&\leq -t\nabla||f(x)||^2 + \frac{1}{2t}||y' - x||^2 + \frac{L}{2}||x - x^*|| \\
&= -t(\frac{1}{t^2}||y' - x||^2) + \frac{1}{2t}||y' - x||^2 + \frac{L}{2}||x - x^*|| \\
&= -\frac{1}{2t}(||x - x^*||^2 - ||y' - x^*||^2)
\end{aligned}
$$

Proved.

Part(d): $f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|^2}{2tk}$

Proof:

$$\begin{aligned}
\frac{||x^{(0)} - x^*||^2}{2tk} &= \frac{1}{2tk}(||x^{k-1} - x^*||^2 - ||x^k - x^*||^2 \\
&\quad + ||x^{k-2} - x^*||^2 - ||x^{k-1} - x^*||^2) \\
&\quad + ... \\
&\quad + ||x^1 - x^*||^2 - ||x^0 - x^*||^2) \\
&\geq \frac{1}{2tk}(2tf(x^k - f(x^*)) \\
&\quad + 2tf(x^{(k-1)} - f(x^*)) \\
&\quad + 2tf(x^{(1)} - f(x^*)) \\
&= \frac{1}{2tk} * 2t(\sum_{i=1}^{k} f(x^i) - kf(x^*)) \\
&= \frac{\sum_{i=1}^{k} f(x^i)}{n} - f(x^*) \\
&\geq f(x^k) - f(x^*)
\end{aligned}$$

3 Implement Lecun2004

3.1 Implement Code of 2 Algorithms, Q3 part a and b

Code for both batch newton and online kalman are attached at the end of the homework(see section 5). See report in section 4 for write-up and discussion.

3.2 Reproduce Fig1

Here we present fig1 with a even axis and with a logarithm 10 axis. (The original LSOL paper uses a log10 axis, so the right -side figure should be taken as our final answer.)

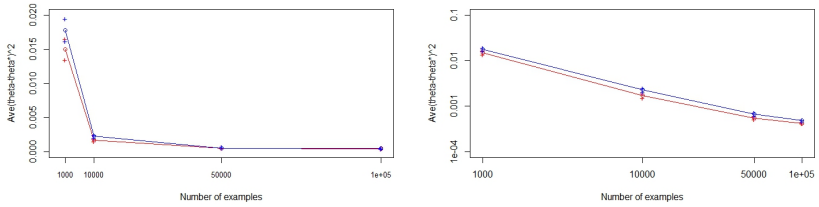

Figure 1: Average$((\theta - \theta_*)^2)$ as a function of number of examples used.Red curve stands for the batch newton, blue cureve for online Kalman.The plus sign is 95% of confidence interval. The left is usual axis and right is log10 axis (which is the same as Fig1 in LSOL.)

3.3 Reproduce Fig2

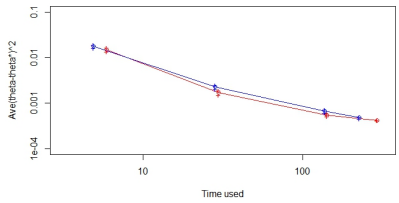

Figure 2: Average$((\theta - \theta_*)^2)$ as a function of time used in seconds on a PC. Red curve stands for the batch newton, blue cureve for online Kalman.The plus sign is 95% of confidence interval. Axis are log10 axis (as Fig2 in LSOL.)

3.4 Reproduce Fig3

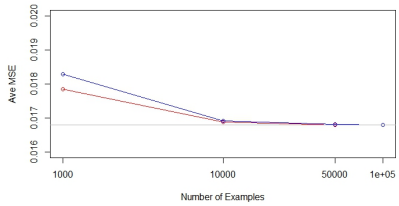

Figure 3: Average(MSE) as a function of number of examples. Red curve stands for the batch newton, blue cureve for online Kalman.The grey line stands for the smallest mse achieved in our experiment. Axis are log10 axis (see Fig3 in LSOL.)

3.5 Reproduce Fig4

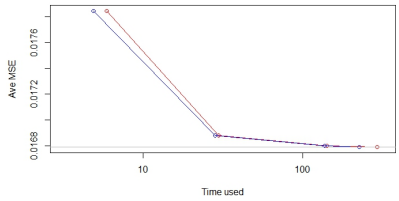

Figure 4: Average(MSE) as a function of time used in seconds on a PC. Red curve stands for the batch newton, blue cureve for online Kalman.The grey line stands for the smallest mse achieved in our experiment. Axis are log10 axis (see Fig4 in LSOL.)

Report on the next page

4 Report of Experiment

We conducted an experiment with the Batch-Newton Algorithm and the Online Kalman algorithm presented in Bottou and LeCun(2004). We compared the two algorithm based on the convergence of parameter θ, test MSE and running time. Online Kalman achieves a better convergence and prediction in a shorter time than Batch Newton.

We generate 10^5 data points for both the training set and test set with $X \in R^20$ and $y = \pm 1$. X are 20 dimensional Gaussian with mean $(1, 1, , , , 1)$ or mean $(-1, -1, , , -1)$. We specify the loss function as:

$$L = (1.5y - f(\theta))^2 \tag{1}$$

$$f(x\theta) = 1.71 * tanh(0.66 * x\theta) \tag{2}$$

Batch-Newton algorithm updates θ with:

$$g = \sum_i \frac{\partial L}{\partial \theta}(x_i, y_i, \theta_{k-1}) \quad H = \sum_i (f'(x_i\theta_{k-1}))^2 x_i x_i^T \tag{3}$$

$$\theta_k = \theta_{k-1} - H^{-1}g \tag{4}$$

Online Kalman does a single sequential sweep over all examples:

$$\theta_t = \theta_{t-1} - \frac{1}{\tau}\Phi_t \frac{\partial L}{\partial \therefore}(x_t, y_t, \theta_{t-1}) \tag{5}$$

$$\Phi_t = ((1 - \frac{2}{\tau})\Phi_{t-1}^{-1} + \frac{2}{\tau}(f'(x_t\theta_{t-1}))^2 x_t x_t^T)^{-1} \tag{6}$$

We first compute the optimal θ^* on the test set using batch-newton. We then conduct the experiment of $10^3, 10^4, 5 * 10^4, 10^5$ examples of the training set with both algorithms. We permute the training set 30 times and the compute the average distance of experiment θ to $theta^*$ and the average of MSE. The results are shown in the 4 figures below:

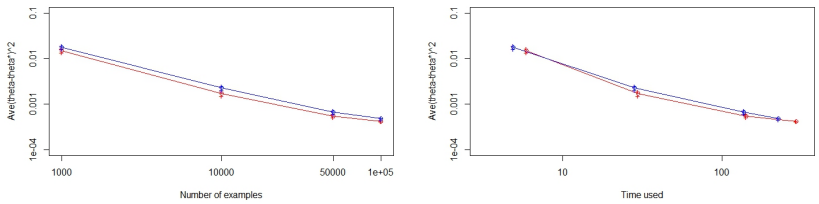

Figure 1 (Left):Average$((\theta - \theta_*)^2)$ as a function of number of examples used. Figure 2 (Right): Average$((\theta - \theta_*)^2)$ as a function of time used in seconds on a PC. Red curve stands for the batch newton, blue cureve for online Kalman.The plus sign is 95% of confidence interval.

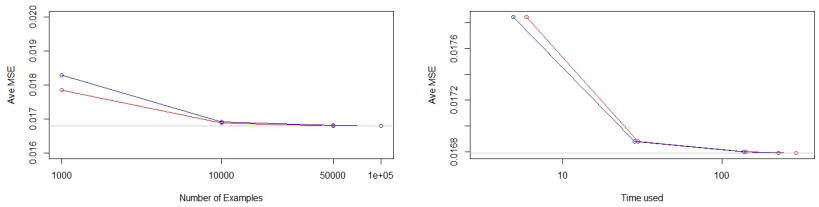

Figure 3 (Left): Average MSE as a function of number of examples used. Figure 4 (Right), Average MSE as a function of time used in seconds on a PC. Red curve stands for the batch newton, blue cureve for online Kalman. The grey line stands for the smallest mse achieved in our experiment.

As shown in Figure 1, both batch-newton and online kalman converges to the ground truth. The distance of the parameter vector to the truth converges to zero as the example sizes grow larger. As Result 6 in LSOL Paper section 3 stated, both sequences have $O(1/t)$ convergence and thus have similar speed and nature of convergence.

However, the online algorithm achieves the same performance using much less time, shown in Figure 2. As the computational cost section in LSOL paper showed, the batch-newton with quadratic convergence takes $NloglogN$ time with N examples, while the online algorithm sweeps sequentially over all N examples and thus takes N times. As presented in Figure 2, the time difference is quite large when the number of examples grows large. This means to achieve the same convergence performance, the batch algorithm has to take $NloglogN - N$ more time than online algorithm. (Each dot of blue line is significantly to the left of the red line.) Eventually, the blue line decreases faster than the red line and would lead to even faster performance if we allow the number of examples to keep blowing.

In practice, as we usually care more about the test MSE than the accuracy of parameter vector. Figure 3, as the example number grows, online algorithm can achieve a better prediction eventually, (the minimal mse of all examples are achieved by the blue line at 10^5 examples.) Similar as in Figure 4, online algorithm takes significantly shorter time for each number of example we used and eventually achieves the best prediction performance with a smallest mse over 10^5 examples.

In summary, our experiment shows that though the convergence nature are similar, online Kalman provides a better prediction with a shorter time and the performance difference will be larger when the size of the data set grows.

5 Code Attached

```
set.seed(123)
library(MASS)
# Two potential distribution params
mus <- cbind(rep(-1,20),rep(1,20))
sigs <- seq(1,20,1)
n=10^5

#sample ytrain ytest
ytrain <- sample(c(-1,1), n, replace=T)
ytest <- sample(c(-1,1), n, replace=T)

# Generate Training
train <- matrix(0,nrow=n,ncol=20)
for (i in 1:n){
  train[i,]<- c(mvrnorm(n=1,mu=mus[,(ifelse(ytrain[i]>0,2,1))],Sigma=diag(20)))
  #train[i,]<- c(1,mvrnorm(n=1,mu=mus[,sample(1:2,1)],Sigma=diag(19)))
}
#mvrnorm(n=1,mu=mus[,sample(1:2,1)],Sigma=diag(19)*sample(sigs,1))

#Generate Testing
test <- matrix(0,nrow=n,ncol=20)
for (i in 1:n){
  test[i,]<- c(mvrnorm(n=1,mu=mus[,(ifelse(ytest[i]>0,2,1))],Sigma=diag(20)))
  #test[i,]<- c(1,mvrnorm(n=1,mu=mus[,sample(1:2,1)],Sigma=diag(19)))
}

#permutation of training
ptrain <- matrix(0,n,30)
for (i in 1:30){
  ptrain[,i] <- sample(n,n)
}

#define f
f <- function(x,alpha){
  return(1.71*tanh(0.66*x%*%alpha))
}

fprime <- function(x,alpha){
  return(1.71*0.66*(1-tanh(0.66*x%*%alpha)^2))
}

iter=10^3 #to start with
```

```
#Find ground truth
start <- Sys.time()
thetastar <- matrix(runif(20*iter,min=0,max=1)-1/2,20,(iter+1))
for (i in 1:iter){
  z=2*(1.5*ytest-f(test,thetastar[,i]))*(-fprime(test,thetastar[,i]))
  G=t(z) %*% test
  H=sum((fprime(test,thetastar[,i])^2)*colSums(t(test) * t(test)))
  thetastar[,i+1]=thetastar[,i]-H^(-1)*G
  if (sum((thetastar[,i+1]-thetastar[,i])^2) < (1/n)^2){
    break
  }
  istar <- i
}
end <- Sys.time()
timeused_truth <- end -start
thetatrue <- thetastar[,istar]


#Batch newton
#permutation
thetapres_1000 <- matrix(0,20,30)
theta <- matrix(runif(20*iter,min=0,max=1)-1/2,20,iter)
start  <- Sys.time()
for (p in 1:30){
  trainp1000 <- train[ptrain[,p],][1:1000,]
  ytrainp1000 <- ytrain[ptrain[,p]][1:1000]
  for (i in 1:iter){
    z=2*(1.5*ytrainp1000-f(trainp1000,theta[,i]))*(-fprime(trainp1000,theta[,i]))
    G=t(z) %*% trainp1000
    H=sum((fprime(trainp1000,theta[,i])^2)
      *colSums(t(trainp1000) * t(trainp1000)))
    theta[,i+1]=theta[,i]-H^(-1)*G
    if (sum((theta[,i+1]-theta[,i])^2) < (0.01/n)^2){
      break
    }
    thetapres_1000[,p] <- theta[,i+1]
    i1000<- i
  }
  thetapres_1000[,p] <- theta[,i+1]
}
end <- Sys.time()
timeused_1000 <- end-start


thetapres_10000 <- matrix(0,20,30)
theta <- matrix(runif(20*iter,min=0,max=1)-1/2,20,iter)
```

```
start  <- Sys.time()
for (p in 1:30){
  trainp10000 <- train[ptrain[,p],][1:10000,]
  ytrainp10000 <- ytrain[ptrain[,p]][1:10000]
  for (i in 1:iter){
    z=2*(1.5*ytrainp10000-f(trainp10000,theta[,i]))
      *(-fprime(trainp10000,theta[,i]))
    G=t(z) %*% trainp10000
    H=sum((fprime(trainp10000,theta[,i])^2)
      *colSums(t(trainp10000) * t(trainp10000)))
    theta[,i+1]=theta[,i]-H^(-1)*G
    if (sum((theta[,i+1]-theta[,i])^2) < (0.01/n)^2){
      break
    }
    thetapres_10000[,p] <- theta[,i+1]
    i10000<- i
  }
  thetapres_10000[,p] <- theta[,i+1]
}
end <- Sys.time()
timeused_10000 <- end-start

thetapres_50000 <- matrix(0,20,30)
theta <- matrix(runif(20*iter,min=0,max=1)-1/2,20,iter)
start  <- Sys.time()
for (p in 1:30){
  trainp50000 <- train[ptrain[,p],][1:50000,]
  ytrainp50000 <- ytrain[ptrain[,p]][1:50000]
  for (i in 1:iter){
    z=2*(1.5*ytrainp50000-f(trainp50000,theta[,i]))
    *(-fprime(trainp50000,theta[,i]))
    G=t(z) %*% trainp50000
    H=sum((fprime(trainp50000,theta[,i])^2)
       *colSums(t(trainp50000) * t(trainp50000)))
    theta[,i+1]=theta[,i]-H^(-1)*G
    if (sum((theta[,i+1]-theta[,i])^2) < (0.01/n)^2){
      break
    }
    thetapres_50000[,p] <- theta[,i+1]
    i50000<- i
  }
  thetapres_50000[,p] <- theta[,i+1]
}
end <- Sys.time()
timeused_50000 <- end-start
```

```
thetapres_100000 <- matrix(0,20,30)
theta <- matrix(runif(20*iter,min=0,max=1)-1/2,20,iter)
start  <- Sys.time()
for (p in 1:30){
  trainp <- train[ptrain[,p],]
  ytrainp <- ytrain[ptrain[,p]]
  for (i in 1:iter){
    z=2*(1.5*ytrainp-f(trainp,theta[,i]))*(-fprime(trainp,theta[,i]))
    G=t(z) %*% trainp
    H=sum((fprime(trainp,theta[,i])^2)
        *colSums(t(trainp) * t(trainp)))
    theta[,i+1]=theta[,i]-H^(-1)*G
    if (sum((theta[,i+1]-theta[,i])^2) < (0.01/n)^2){
      break
    }
    thetapres_100000[,p] <- theta[,i+1]
    i100000<- i
  }
  thetapres_100000[,p] <- theta[,i+1]
}
end <- Sys.time()
timeused_100000 <- end-start

#Online kalman
thetakpres_1000 <- matrix(0,20,30)
thetak <- matrix(runif(20*(n+1),min=-0.5,max=0.5),20,(n+1))
phi <- diag(20)

start <- Sys.time()
for (p in 1:30){
  trainp <- train[ptrain[,p],]
  ytrainp <- ytrain[ptrain[,p]]
  for (t in 1:10^3){
    tao=max(20,t-40)
    xt=trainp[t,]
    yt=ytrainp[t]
    Gt=2*(1.5*yt-1.71*tanh(0.66*xt %*% thetak[,t]))
     *(-1.71)*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66*xt
    gt2=(1.71*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66)^2
    denom <- 1/(((1-1/2/tao)/(2/tao*gt2))+t(xt)%*%phi%*%xt)[1][1]
    #turn 1*1 matrix to scalar
    phi <- 1/(1-2/tao)*(phi-(denom * (phi %*% xt) %*% t(phi %*% xt)))
    thetak[,t+1]=thetak[,t]-1/tao*phi %*% Gt
  }
  thetakpres_1000[,p] <- thetak[,t+1]
}
```

```
end <- Sys.time()
timeused_k1000 <- end-start


thetakpres_10000 <- matrix(0,20,30)
thetak <- matrix(runif(20*(n+1),min=-0.5,max=0.5),20,(n+1))
phi <- diag(20)
start <- Sys.time()
for (p in 1:30){
  trainp <- train[ptrain[,p],]
  ytrainp <- ytrain[ptrain[,p]]
  for (t in 1:10^4){
    tao=max(20,t-40)
    xt=trainp[t,]
    yt=ytrainp[t]
    Gt=2*(1.5*yt-1.71*tanh(0.66*xt %*% thetak[,t]))
    *(-1.71)*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66*xt
    gt2=(1.71*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66)^2
    denom <- 1/(((1-1/2/tao)/(2/tao*gt2))+t(xt)%*%phi%*%xt)[1][1]
    #turn 1*1 matrix to scalar
    phi <- 1/(1-2/tao)*(phi-(denom * (phi %*% xt) %*% t(phi %*% xt)))
    thetak[,t+1]=thetak[,t]-1/tao*phi %*% Gt
  }
  thetakpres_10000[,p] <- thetak[,t+1]
}
end <- Sys.time()
timeused_k10000 <- end-start

thetakpres_50000 <- matrix(0,20,30)
thetak <- matrix(runif(20*(n+1),min=-0.5,max=0.5),20,(n+1))
phi <- diag(20)
start <- Sys.time()
for (p in 1:30){
  trainp <- train[ptrain[,p],]
  ytrainp <- ytrain[ptrain[,p]]
  for (t in 1:50000){
    tao=max(20,t-40)
    xt=trainp[t,]
    yt=ytrainp[t]
    Gt=2*(1.5*yt-1.71*tanh(0.66*xt %*% thetak[,t]))
    *(-1.71)*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66*xt
    gt2=(1.71*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66)^2
    denom <- 1/(((1-1/2/tao)/(2/tao*gt2))+t(xt)%*%phi%*%xt)[1][1]
    #turn 1*1 matrix to scalar
    phi <- 1/(1-2/tao)*(phi-(denom * (phi %*% xt) %*% t(phi %*% xt)))
    thetak[,t+1]=thetak[,t]-1/tao*phi %*% Gt
```

```
  }
  thetakpres_50000[,p] <- thetak[,t+1]
}
end <- Sys.time()
timeused_k50000 <- end-start

thetakpres_100000 <- matrix(0,20,30)
thetak <- matrix(runif(20*(n+1),min=-0.5,max=0.5),20,(n+1))
phi <- diag(20)
start <- Sys.time()
for (p in 1:30){
  trainp <- train[ptrain[,p],]
  ytrainp <- ytrain[ptrain[,p]]
  for (t in 1:10^5){
    tao=max(20,t-40)
    xt=trainp[t,]
    yt=ytrainp[t]
    Gt=2*(1.5*yt-1.71*tanh(0.66*xt %*% thetak[,t]))
    *(-1.71)*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66*xt
    gt2=(1.71*(1-tanh(0.66*xt %*% thetak[,t])^2)*0.66)^2
    denom <- 1/(((1-1/2/tao)/(2/tao*gt2))+t(xt)%*%phi%*%xt)[1][1]
    #turn 1*1 matrix to scalar
    phi <- 1/(1-2/tao)*(phi-(denom * (phi %*% xt) %*% t(phi %*% xt)))
    thetak[,t+1]=thetak[,t]-1/tao*phi %*% Gt
  }
  thetakpres_100000[,p] <- thetak[,t+1]
}
end <- Sys.time()
timeused_k100000 <- end-start
```