DMM Assignment 2 Source code

##1. Computing the Loss Function
##1.b

```
## Compute the loss of factorization LR with L2 regularization (parameter lambda)

L<-L0.r10
R<-R0.r10
lambda<-2



loss <- function(L, R, lambda) {
loss1<-0;
loss2<-0;
i<-1;
while(i<(nrow(summary(D))+1)){
loss1<-loss1+ (ds[i]-L[is[i],]%*%R[,js[i]])^2;
i<-i+1;
}
loss2<-(lambda/2)*(norm(L,'F')^2+norm(R,'F')^2)
loss<-loss1+loss2;
return(loss);
}

## verify loss of starting point with lambda=2
loss(L0.r10, R0.r10, 2)
```

##2. Computing the Local Gradients

##2.c

```
## Compute gradient w.r.t. l_i and r_j of the local loss of the p-th revelead
## entry for factorization LR with L2 regularization (parameter lambda). Returns
## a list containing vector elements Li and Rj.

dlossp <- function(L, R, lambda, p) {
   i <- is[p]
   j <- js[p]
   d <- ds[p]
   ## create two length-r vectors dLi and dRj
dLi<-rep(0,r);
dRj<-rep(0,r)
pl<-1;
k<-1;
while(k<(r+1)){
item1<-(-2)* (d-L[i,]%*%R[,j]);
dLi[pl]<-item1* R[k,j]+lambda* L[i,k]/ Nis[i];
dRj[pl]<-item1* L[i,k]+lambda* R[k,j]/ Njs[j] ;
k<-k+1;
pl<-pl+1;
}
   ## * dLi[k] contains the gradient of the local loss with respect to l_ik
   ## * dRi[k] contains the gradient of the local loss with respect to r_kj


   ## return the result as a list
   ## (elements can be accessed with x$Li or x$Ri, where x is the returned list)

list(Li=dLi, Rj=dRj)
}
x<- dlossp(L0.r10, R0.r10, 2, 10)
x$Li
x$Ri


##verify the result
## test local gradient computation for 10th entry
dlossp(L0.r10, R0.r10, 2, 10)
```

##3. Implementing Gradient Descent

```
## a gradient descent epoch
gdepoch <- function(L, R, lambda, eps) {
   ## create gradient matrices
   dL <- matrix(0, m, r)
   dR <- matrix(0, r, n)

   ## fill the gradient matrices using repeated calls to my dlossp function
p<-1;
while(p<(nrow(summary(D))+1)){
x<-dlossp(L, R, lambda,p);
dL[is[p],]<- dL[is[p],]+x$Li;
dR[,js[p]]<- dR[,js[p]]+x$Rj;
p<-p+1;
}
   ## perform a gradient step on L and R with step size eps by using the gradient matrices

L<-L-eps*dL
R<-R-eps*dR

   ## return result
   list(L=L, R=R)
}
## test the gd epoch and print 5th row of L; should give
gdepoch(L0.r10, R0.r10, 2, 0.01)$L[5,]

## Visualize the result after 5, 15, and 50 epochs and discuss.

## Run for 5 epochs
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 2, epochs=5, eps=0.01)
## show result
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

## Run for 15 epochs (continue with the previous result for 10 more epoch)
result.gd.r10.l2 <- runner(gdepoch, result.gd.r10.l2$L, result.gd.r10.l2$R, 2, epochs=10, eps=0.01)
## show result
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

## Run for 50 epochs (continue with the previous result for 35 more epoch)
result.gd.r10.l2 <- runner(gdepoch, result.gd.r10.l2$L, result.gd.r10.l2$R, 2, epochs=35, eps=0.01)
## show result
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)
```

##4. Implementing Stochastic Gradient Descent

```
## Run a stochastic gradient descent epoch. Uses same paramters and return values as gdepoch.
sgdepoch <- function(L, R, lambda, eps) {
   for (p in sample(length(is))) {
      i <- is[p]
      j <- js[p]

      ## perform a stochastic gradient descent step on revealed entry p with step size eps
      ## Use dlossp to compute the local gradient of entry (i,j), then update
      ## L[i,] and R[,j] using step size eps. Without scale up this gradient
x<-dlossp(L, R, lambda,p);
L[i,]<- L[i,]-eps*x$Li;
R[,j]<- R[,j]-eps*x$Rj;
   }
   ## return result
   list(L=L, R=R)
}

## test the sgd epoch and print 5th row of L
set.seed(2, kind="Mersenne-Twister")
sgdepoch(L0.r10, R0.r10, 2, 0.01)$L[5,]

## example run (this takes a while)
result.sgd.r10.l2 <- runner(sgdepoch, L0.r10, R0.r10, 2)

## show result
showgray(result.sgd.r10.l2$L %*% result.sgd.r10.l2$R)
```

##5. Impact of Parameter Choices

```
##case where (r, λ)=(1,0)
set.seed(1, kind="Mersenne-Twister")
r <- 1
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 0,epochs=50)
 tr1<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(1,2)
set.seed(1, kind="Mersenne-Twister")
r <- 1
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 2,epochs=50)
 tr2<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(1,20)
set.seed(1, kind="Mersenne-Twister")
r <- 1
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 20,epochs=50)
 tr3<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(10,0)
set.seed(1, kind="Mersenne-Twister")
r <- 10
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 0,epochs=50)
 tr4<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(10,2)
set.seed(1, kind="Mersenne-Twister")
r <- 10
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 2,epochs=50)
 tr5<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(10,20)
set.seed(1, kind="Mersenne-Twister")
r <- 10
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 20,epochs=50)
 tr6<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)
```

```
##case where (r, λ)=(20,0)
set.seed(1, kind="Mersenne-Twister")
r <- 20
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 0,epochs=50)
 tr7<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(20,2)
set.seed(1, kind="Mersenne-Twister")
r <- 20
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 2,epochs=50)
 tr8<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)

##case where (r, λ)=(20,20)
set.seed(1, kind="Mersenne-Twister")
r <- 20
L0.r10 <- matrix(rnorm(m*r), m, r)/sqrt(r)
R0.r10 <- matrix(rnorm(r*n), r, n)/sqrt(r)
result.gd.r10.l2 <- runner(gdepoch, L0.r10, R0.r10, 20,epochs=50)
 tr9<-result.gd.r10.l2
showgray(result.gd.r10.l2$L %*% result.gd.r10.l2$R)
```