**Exercise 09: Matrix Factorization**

**A sparse matrix factorization:**

In the case of a single factor, $K = 1$, so that $\hat{X} = duv^T$. This gives us a rank-1 approximation to the original matrix. The Witten et. al. paper proposes to estimate $u$ and $v$ by solving the following optimization problem:

$$\underset{u\in\mathbb{R}^N, v\in\mathbb{R}^P}{minimize}\ \|X - duv^T\|_F^2$$
$$\text{subject to } \|u\|_2^2 = 1\ ,\ \|v\|_2^2 = 1,\ \|u\|_1 \le \lambda_u\ ,\ \|v\|_1 \le \lambda_v$$

This is equivalent to:

$$\underset{u\in\mathbb{R}^N, v\in\mathbb{R}^P}{maximize}\ u^T x v$$
$$\text{subject to } \|u\|_2^2 = 1\ ,\ \|v\|_2^2 = 1,\ \|u\|_1 \le \lambda_u\ ,\ \|v\|_1 \le \lambda_v$$

According the the paper, The following iterative algorithm is used to optimize the criterion for the rank-1 PMD:

Step 1. Initialize $v$ to have $L2$-norm 1.

Step 2. Iterate until convergence:

(a) $u = argmax_u u^T X v$ subject to $\|u\|_1 \le \lambda_u$ and $\|u\|_2^2 \le 1$.

The solution $u = \frac{S(Xv,\Delta_u)}{\|S(Xv,\Delta_u)\|_2}$, with $\Delta_u = 0$ if this results in $|u| \le \lambda_u$; otherwise, $\Delta_u$ is chosen s.t. $|u| = \lambda_u$. Here $S$ denotes the soft thresholding operator.

$S(Xv, \Delta_u) = sgn(Xv)(|Xv| - \Delta_u)_+$.

(b) $v = argmax_v u^T X v$ subject to $\|v\|_1 \le \lambda_v$ and $\|v\|_2^2 \le 1$.

The solution $v = \frac{S(X^T u,\Delta_v)}{\|S(X^T u,\Delta_v)\|_2}$, with $\Delta_v = 0$ if this results in $|v| \le \lambda_v$; otherwise, $\Delta_v$ is chosen s.t. $|v| = \lambda_v$. Here $S$ denotes the soft thresholding operator.

$S(X^T u, \Delta_v) = sgn(X^T u)(|X^T u| - \Delta_u)_+$.

Step 3. $d = u^T X v$.

Computation of K factors of PMD:

Step 1. Let $X^1 = X$

Step 2. For $k = 1, 2, \cdots, K$:

(a) Find $u_k$, $v_k$, and $d_k$ by appling the single-factor PMD algorithm to data $X^k$.

(b) $X^{k+1} = X^k - d_k u_k v_k^T$.

**Application to Marketing Data**

I did a square root transformation since the data points are all non-negative integer and look like Poisson Distribution.

Apply the rank-1 PMD. I did some result comparison by changing the constraints $\lambda_u$ and $\lambda_v$, from loose to tight.

From left to right, $\lambda_u = \lambda_v = 5$, $\lambda_u = \lambda_v = 3$, $\lambda_u = \lambda_v = 1$

| | | | | | |
|---|---|---|---|---|---|
| chatter | 0.336809 | health_nutrition | 0.69416 | chatter | 1 |
| health_nutrition | 0.323689 | personal_fitness | 0.395777 | current_events | 0 |
| cooking | 0.312577 | chatter | 0.372328 | travel | 0 |
| politics | 0.306646 | cooking | 0.228312 | photo_sharing | 0 |
| photo_sharing | 0.294122 | photo_sharing | 0.205035 | uncategorized | 0 |
| travel | 0.259083 | outdoors | 0.192348 | tv_film | 0 |
| personal_fitness | 0.212151 | food | 0.178349 | sports_fandom | 0 |
| news | 0.189889 | shopping | 0.127408 | politics | 0 |
| shopping | 0.186372 | politics | 0.117236 | food | 0 |
| food | 0.185805 | current_events | 0.080506 | family | 0 |
| fashion | 0.175683 | sports_fandom | 0.077613 | home_and_garden | 0 |
| religion | 0.169285 | news | 0.073833 | music | 0 |
| beauty | 0.153233 | tv_film | 0.07207 | news | 0 |
| sports_fandom | 0.149917 | religion | 0.055781 | online_gaming | 0 |
| parenting | 0.145122 | parenting | 0.036925 | shopping | 0 |
| current_events | 0.141461 | school | 0.031302 | health_nutrition | 0 |
| computers | 0.134411 | eco | 0.0268 | college_uni | 0 |
| school | 0.129667 | travel | 0.020008 | sports_playing | 0 |
| family | 0.120555 | college_uni | 0.014211 | cooking | 0 |
| dating | 0.120462 | uncategorized | 0 | eco | 0 |
| outdoors | 0.111803 | family | 0 | computers | 0 |
| college_uni | 0.111343 | home_and_garden | 0 | business | 0 |
| automotive | 0.092853 | music | 0 | outdoors | 0 |
| uncategorized | 0.09254 | online_gaming | 0 | crafts | 0 |
| tv_film | 0.09005 | sports_playing | 0 | automotive | 0 |
| music | 0.078662 | computers | 0 | art | 0 |
| sports_playing | 0.071176 | business | 0 | religion | 0 |
| art | 0.057508 | crafts | 0 | beauty | 0 |
| eco | 0.057156 | automotive | 0 | parenting | 0 |
| business | 0.047803 | art | 0 | dating | 0 |
| small_business | 0.046031 | beauty | 0 | school | 0 |
| crafts | 0.039098 | dating | 0 | personal_fitness | 0 |
| home_and_garden | 0.033075 | fashion | 0 | fashion | 0 |
| online_gaming | 0.023963 | small_business | 0 | small_business | 0 |
| spam | 0 | spam | 0 | spam | 0 |
| adult | 0 | adult | 0 | adult | 0 |

As $\lambda_u$ and $\lambda_v$ get tighter, less factors are selected.

Code for rank-1 & rank-k PMD:

```
#### Generate a simple test version X ####
library(MASS)
N=50
p=5 # Number of indept. variables, i.e. the length of vector beta.
mu.test=rep(1,p)
sigma.test=diag(0.5, p)
x.test=mvrnorm(N, mu=mu.test, Sigma=sigma.test) # Indept. R.V.

#### Functions ####
prox=function(ut, gamma=1, lambda){

prox=sign(ut) * pmax( ( abs(ut) -  gamma* lambda ), 0)

return(prox)
}

Delta= function( a, c  ){
Delta.lower = 0
Delta.upper = max(abs(a))

if(norm(a, type = "2")==0 || sum(abs(a/sqrt(sum(a^2)))) <= c) {
return(0)
}

i=1
while(i<200){
su = prox(a, gamma=1, (Delta.lower + Delta.upper)/2)
if(sum(abs(su/sqrt(sum(su^2)))) < c){
Delta.upper = (Delta.lower + Delta.upper)/2
} else {
```

```
Delta.lower = (Delta.lower + Delta.upper)/2
}
if((Delta.upper - Delta.lower) < 1e-6) {
return((Delta.lower + Delta.upper)/2)
}
i=i+1
}
warning("Didn't quite converge")
Delta = (Delta.lower + Delta.upper)/2

return( Delta )
}

PMD.1=function( X, c.u, c.v, maxiter ){

# Initialize v to have L2 norm=1 #
v = rep(1, p)
v = v / norm(v, type = "2")
u = rep(0, p)

v.t=v
t=1
while( t < maxiter ){
Delta.u.t=Delta( X %*% v.t, c.u )
u.t.prox = prox( ut= X %*% v.t, gamma=1, lambda= Delta.u.t )
u.t = u.t.prox / norm(u.t.prox, type = "2" )
u= cbind(u, u.t)

Delta.v.t=Delta( t(X) %*% u.t, c.v )
v.t.prox = prox( ut= t(X) %*% u.t, gamma=1, lambda= Delta.v.t )
v.t = v.t.prox / norm(v.t.prox, type = "2" )
v= cbind(v, v.t)
```

```
t=t+1
}

u.final= u[,t]
v.final= v[,t]
d= t(u.final) %*% X %*% v.final

return( list( iter=t, u=u, v=v, u.final=u.final, v.final=v.final ,d=d ) )
}

PMD.k = function(X, k, c.u, c.v, maxiter, tol ){



if(k == 1){
PMD1=PMD.1( X, c.u, c.v, maxiter, tol )
return( list( X.final=X, u.PDMk=PMD1$u.final, v.PDMk=PMD1$v.final, d.PDMk=PMD1$d ) )

}

u.PDMk=rep(0, N)
v.PDMk = rep(0,p)
d.PDMk = 0
X1 = X
for(i in 1:k){
r1 = PMD.1(X1, c.u, c.v, maxiter, tol)
d.PDMk = c(d.PDMk, r1$d)
X1 = X1 - as.numeric(r1$d) * r1$u.final %o% r1$v.final
u.PDMk = cbind(u.PDMk, r1$u.final)
v.PDMk = cbind(v.PDMk, r1$v.final)
}
```

```
return( list( X.final=X1, u.PDMk=u.PDMk, v.PDMk=v.PDMk , d.PDMk=d.PDMk ) )
}

fit.PDMk=PMD.k( X=x.test, k=2, c.u=c.u, c.v=c.v, maxiter=maxiter, tol=tol )

maxiter=50
tol=1e-4
c.u=1
c.v=1

fit.PDMk=PMD.k( X=x.test, k=2, c.u=c.u, c.v=c.v, maxiter=maxiter, tol=tol )
```