EE412 Foundation of Big Data Analytics, Fall 2021
HW2

Name: 함태욱

Student ID: 20180716

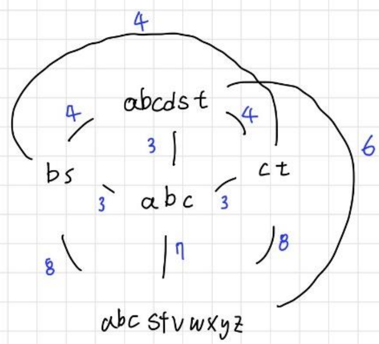
Discussion Group (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

Answer to Problem 1

(a)

Exercise 7.2.6



If we choose minimized the sum of the distances

$\boxed{abc = 3+3+3+7 = 16}$

abcdst = 3+4+4+6 = 17

bs = 3+4+4+8 = 19

ct = 3+4+4+8 = 19

abcstvwxyz = 6+7+8+8 = 29

But, If we choose minimized max distance to other points,

abc = 7

$\boxed{abcdst = 6}$

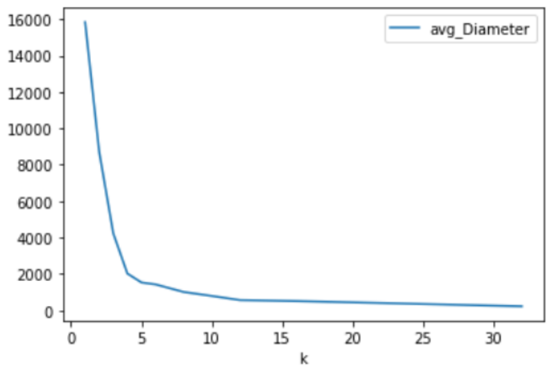
bs = 8

ct = 8

abcstvwxyz = 8.

set { abc, abcdst, bs, ct, abcstvwxyz }.

(b)
(1, 15840.015935162375)
(2, 8712.758000610926)
(3, 4225.773466007261)
(4, 2018.267391109598)
(5, 1526.2137994123182)
(6, 1420.7214130743844)
(8, 1003.3225742993732)
(10, 784.7882449282713)
(12, 557.6125972712002)
(16, 504.6115133174929)
(32, 216.9042145953651)
[Graph]



-The k value with an explanation why it is good for this data.

v = 10 일때, v(=10) ~ 2v(=20)간의 기울기 변화가 적고, v/2 < k < v 를 만족하는 k 값은 6,7,8,9 중의 하나이다. 이때 binary search 를 사용한 결과 나는 k=8 이 ‘best value’ 라고 생각하여 선택하였다.

Answer to Problem 2

(1)

[Code Attachment]

```
import sys
import numpy as np
M1 = np.array([[1,1,1],[1,2,3],[1,3,6]])
x0 = np.array([1,1,1]).T
# (a)
x = x0
prev_x = x
while True:
    x = np.matmul(M1,x)
    x = x/np.linalg.norm(x)
    if np.linalg.norm(x-prev_x)<0.000001:
            break
    prev_x = x
# (b)
eig_vec1 = x
eig_val1=np.matmul(np.matmul(eig_vec1.T,M1),eig_vec1)
# (c)
M2 = M1 - np.dot(eig_val1 ,
np.matmul(eig_vec1.reshape(3,1),eig_vec1.reshape(1,3)))
# (d)
x = eig_vec1
prev_x =x
while True:
    x = np.matmul(M2,x)
    x = x/np.linalg.norm(x)
    if np.linalg.norm(x-prev_x)<0.000001:
            break
    prev_x = x
eig_vec2 = x
eig_val2=np.matmul(np.matmul(eig_vec2.T,M2),eig_vec2)
M3 = M2 - np.dot(eig_val2 ,
np.matmul(eig_vec2.reshape(3,1),eig_vec2.reshape(1,3)))
# (e)
x = eig_vec2
prev_x = x
while True:
    x = np.matmul(M3,x)
    x = x/np.linalg.norm(x)
```

```
    if np.linalg.norm(x-prev_x)<0.000001:
            break
    prev_x = x
eig_vec3 = x
eig_val3=np.matmul(np.matmul(eig_vec3.T,M2),eig_vec3)
```

Eigenvalues = [7.872983346207407, 1.0000000000000713, 0.12701665379258312]
Eigenvectors =
[0.19382269 0.4722473 0.85989258]
[-0.81649653 -0.40824816 0.40824854]
[0.54384382 -0.78122714 0.30646053]

(2)

[Code Attachment]

```
import numpy as np
M = np.array([[1,2,3],[3,4,5],[5,4,3],[0,2,4],[1,3,5]])

row = 5
col = 3
rank = 2
# (a)
MTM = np.matmul(M.T,M)
MMT = np.matmul(M,M.T)

# (b)
val_V,vec_V = np.linalg.eig(MTM)
val_U,vec_U = np.linalg.eig(MMT)

# (c)
sigma = np.array([])

V = np.array([]) #col x rank
V = np.append(V,vec_V.T[0])
V = np.append(V,vec_V.T[1])
V = V.reshape(rank,col).T

U = np.array([]) # row x rank
U = np.append(U,vec_U.T[0])
U = np.append(U,vec_U.T[2])
U = U.reshape(rank,row).T
for val in val_V:
    if val > 0.01:
        sigma = np.append(sigma,val)
sigma = np.sqrt(sigma)
```

```
sigma_diag = np.diag(sigma)
energy = np.linalg.norm(sigma)
two_dim_M = np.matmul(np.matmul(U,sigma_diag),-V.T)

# (d)
index = np.argmin(sigma)
sigma_diag[index]=0
U.T[index]=0
V.T[index]=0
one_dim_M = np.matmul(np.matmul(U,sigma_diag),-V.T)

# (e)
E_retained = sigma[0]**2/(np.sum(sigma**2))
```

(a)

```
MTM =
[[36 37 38]
 [37 49 61]
 [38 61 84]]
MMT =
[[14 26 22 16 22]
 [26 50 46 28 40]
 [22 46 50 20 32]
 [16 28 20 20 26]
 [22 40 32 26 35]]
```

(b)

Eigenpairs of MTM

```
[1.53566996e+02, 1.54330035e+01, 2.99519331e-15]
```

(third eigenvalue is almost zero)

```
[[-0.40928285 -0.81597848  0.40824829]
 [-0.56345932 -0.12588456 -0.81649658]
 [-0.7176358   0.56420935  0.40824829]]
```

Eigenpairs of MMT

```
[ 1.53566996e+02, -2.16919039e-15,  1.54330035e+01, -2.69964138e-15, -1.63115212e-16]
```

(second, fourth, fifth eigenvalue is almost zero)

```
[[ 0.29769568  0.94131607 -0.15906393 -0.57735012 -0.21094872]
 [ 0.57050856 -0.17481584  0.0332003  -0.22666834  0.06716429]
 [ 0.52074297 -0.04034212  0.73585663  0.10591706 -0.13512315]
 [ 0.32257847 -0.18826321 -0.5103921  -0.27280206 -0.68074095]
 [ 0.45898491 -0.21515796 -0.41425998  0.72776982  0.68507159]]
```

(c)
U =
```
[[ 0.29769568 -0.15906393]
 [ 0.57050856  0.0332003 ]
 [ 0.52074297  0.73585663]
 [ 0.32257847 -0.5103921 ]
 [ 0.45898491 -0.41425998]]
```

Sigma =
```
[[12.39221516  0.        ]
 [ 0.          3.92848616]]
```

V.T =
```
[[0.40928285 0.56345932 0.7176358 ]
 [0.81597848 0.12588456  -0.56420935]]
```
(Here, I multiply -1 to V.T from my code output)

M = U*Sigma*(V.T)
```
= [[1.00000000e+00 2.00000000e+00 3.00000000e+00]
 [3.00000000e+00 4.00000000e+00 5.00000000e+00]
 [5.00000000e+00 4.00000000e+00 3.00000000e+00]
 [6.66133815e-16 2.00000000e+00 4.00000000e+00]
 [1.00000000e+00 3.00000000e+00 5.00000000e+00]]
```

I multiply -1 to the original V.T from my code because eigenvector multiplied by a constant is also an eigenvector.
And multiplying -1 to the V makes the right output of original M

(d)
One – dimensional approximation to the matrix M
```
[[1.509889   2.0786628  2.64743661]
 [2.89357443 3.98358126 5.0735881 ]
 [2.64116728 3.63609257 4.63101787]
 [1.63609257 2.25240715 2.86872172]
 [2.32793529 3.20486638 4.08179747]]
```

(e)
0.9086804524257934= about 90.87%

Answer to Problem 3

(a)

9.3.1

	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

$\xrightarrow{\text{boolean}}$

	a	b	c	d	e	f	g	h
A	1	1		1	1		1	1
B		1	1	1	1	1	1	
C	1		1	1		1	1	1

a)

	SIM	Jaccard dist
A, B	$\frac{4}{8}$	$\frac{1}{2}$
B, C	$\frac{4}{8}$	$\frac{1}{2}$
C, A	$\frac{4}{8}$	$\frac{1}{2}$

b)

A : B $\Rightarrow$ $\cos\theta = \frac{4}{\sqrt{6}\sqrt{6}} = \frac{2}{3}$, $\theta = 48.2°$

B : C $\Rightarrow$ $\cos\theta = \frac{4}{\sqrt{6}\sqrt{6}} = \frac{2}{3}$, $\theta = 48.2°$

C : A $\Rightarrow$ $\cos\theta = \frac{4}{\sqrt{6}\sqrt{6}} = \frac{2}{3}$, $\theta = 48.2°$

c, d)

	a	b	c	d	e	f	g	h
A	1	1		1			1	
B		1	1	1				
C				1		1	1	1

	SIM	Jaccard dist	cos distance	
A, B	$\frac{2}{5}$	$\frac{3}{5}$	$\frac{2}{\sqrt{4}\sqrt{3}} = 0.577$	$\theta = 54.76°$
B, C	$\frac{1}{6}$	$\frac{5}{6}$	$\frac{1}{\sqrt{3}\sqrt{4}} = 0.289$	$\theta = 73.20°$
C, A	$\frac{2}{6}$	$\frac{2}{3}$	$\frac{2}{\sqrt{4}\sqrt{4}} = 0.5$	$\theta = 60°$

e)

	a	b	c	d	e	f	g	h	Avg.
A	4	5		5	1		3	2	$\frac{10}{3}$
B		3	4	3	1	2	1		$\frac{7}{3}$
C	2		1	3		4	5	3	3

Normalize $\Rightarrow$

	a	b	c	d	e	f	g	h
A	$\frac{2}{3}$	$\frac{5}{3}$		$\frac{5}{3}$	$\frac{-7}{3}$		$\frac{-1}{3}$	$\frac{-4}{3}$
B		$\frac{2}{3}$	$\frac{5}{3}$	$\frac{2}{3}$	$\frac{-4}{3}$	$\frac{-1}{3}$	$\frac{-4}{3}$	
C	-1		-2	0		1	2	0

f)

$$A \sim B: \frac{\frac{10}{9} + \frac{10}{9} + \frac{28}{9} + \frac{4}{9}}{3.6515 \cdot 2.7080} = 0.5843 \quad \theta = 54.25^\circ$$

$$B \sim C \quad \frac{\frac{-10}{3} - \frac{1}{3} - \frac{8}{3}}{2.708 \cdot 3.1622} = -0.7396 \quad \theta = 137.7^\circ$$

$$C \sim A \quad \frac{-\frac{2}{3} - \frac{2}{3}}{3.6515 \cdot 3.1622} = -0.1155 \quad \theta = 96.63^\circ$$

9.3.2

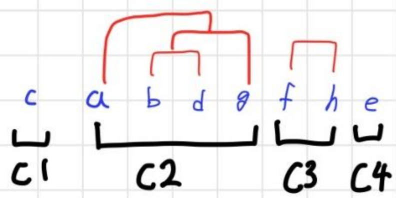
	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

$\Longrightarrow$

	a	b	c	d	e	f	g	h
A	1	1	0	1	0	0	1	0
B	0	1	1	1	0	0	0	0
C	0	0	0	1	0	1	1	1

Jaccard distance = 1- SIM

SIM	a	b	c	d	e	f	g	h
b	$\frac{1}{2}$ ③							
c	0	$\frac{1}{2}$						
d	$\frac{1}{3}$	$\frac{2}{3}$ ②	$\frac{1}{3}$					
e	0	0	0	0				
f	0	0	0	$\frac{1}{3}$	0			
g	$\frac{1}{2}$	$\frac{1}{3}$	0	$\frac{2}{3}$ ②	0	$\frac{1}{2}$		
h	0	0	0	$\frac{1}{3}$	0	1 ⓪	$\frac{1}{2}$	



SIM을 사용한 Jaccard distance를 이용해 clastering을 하면 위와 같다

가장먼저 dist=0 인 f-h가 묶인다. 두번째로 distance = $\frac{1}{3}$인 b-d, d-g를 묶어 {b, d, g}를 생성

{b, d, g} cluster로부터, c 혹은 a 까지의 minimum distance = $\frac{1}{2}$로서 동일하다.

이때 둘중 a를 cluster에 포함시키고 c는 독립된 cluster로 남겨두었다.

	C1	C2	C3	C4
A	0	17/4	2	1
B	4	7/3	2	1
C	1	10/3	3.5	0

A~B: $\dfrac{\frac{119}{12}+4+1}{4.802 \cdot 5.142} = 0.604 \quad \theta = 52.84°$

B~C: $\dfrac{\frac{70}{9}+7+4}{5.142 \cdot 4.936} = 0.7398 \quad \theta = 42.28°$

C~A: $\dfrac{\frac{170}{12}+7}{4.936 \cdot 4.802} = 0.893 \quad \theta = 26.747°$

(b)

175	5.0
261	5.0
440	5.0
480	5.0
527	5.0
5	5.0
318	5.0
364	5.0
785	5.0
1	4.5

(c)

임의의 사용자에 의한 평가되지 않은 영화에 대한 별점 예측을 하기위해서 UV decomposition 을 사용.

Utility Matrix U(n x m) 에 대해 n by 2 matrix U, 2 by m matrix V 를 생성(m = rank 로 설정 시 계산시간이 너무 오래걸려 확인이 힘들었음), 그리고 각 벡터값은 0 부터 1 사이의 임의의 값으로 채웠다. 직접 간단한 K-Fold method 를 이용해 ratings 를 4 개의 set 으로 나누어 진행. RMSE 를 최소로 만들도록 U->V 순으로 최적화 진행, 이후 만들어진 P = UV matrix 로부터 ratings_test.txt 의 user-movie ratings 를 예측.