Problem 1

a) $D = \{(x^i, y^i) \mid 1 \le i \le m\}$

Feature mapping function ϕ

$$K(x,z) = \langle \phi(x), \phi(z) \rangle$$

Uni-variance radial basis kernel

$$K(x,z) = \exp\left(\frac{-1}{2}\|x-z\|^2\right)$$

$$= \|\phi(x) - \phi(z)\|^2$$

$$= \langle \phi(x), \phi(x) \rangle + \langle \phi(z), \phi(z) \rangle - 2\langle \phi(x) \cdot \phi(z) \rangle$$

$$= K(x,x) + K(z,z) - 2K(x,z)$$

$$= 1 + 1 \cancel{0} - 2\exp\left(\frac{-1}{2}\|x-z\|^2\right)$$

$$\|\phi(x) - \phi(z)\|^2 < 2 \Rightarrow \sqrt{\|\phi(x) - \phi(z)\|^2} < \sqrt{2}$$

Hence the

the distance between the ~~dist~~ feature mapping of x & z i.e $\phi(x)$ & $\phi(z)$ is at most $\sqrt{2}$

A function is a valid kernel if it is Real-value positive semi definite and is symmetric

Let us define a kernel matrix for a dataset $\{x_i\}_{i=1}^{n} \in \mathbb{R}^n$

gram matrix

$$K = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & & \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{bmatrix}$$

K is a valid kernel if for any set of points $\{x_i\}_{i=1}^{n} \in \mathbb{R}^n$

it gives rise to matrix K that is positive semidefinite

i.e if $\boxed{\forall a \in \mathbb{R}^n \quad , \quad a^T K a \geq 0}$ PSD.

A matrix is PSD if all its eigen values are non-negative

b) $K(x,z) = K_1(x,z) + K_2(x,z)$

K_1 & K_2 are both valid Kernels.

By gram matrix

$$K = \sum_{j=1}^{2} K_j$$

Now for positive semi definite

$\forall a \in \mathbb{R}^n$, $a^T K a$

$$= \sum_{j=1}^{2} a^T K_j a \geq 0$$

Now if K_j is valid

$$\boxed{K(x,z) = K_1(x,z) + K_2(x,z)}$$

Hence it is a valid kernel

c) $K(x,z) = K_1(x,z) - K_2(x,z)$

$\because$ K_1 & K_2 are valid kernels

$$K_1 = \sum_{i=1}^{n} \lambda_i v_i v_i^T \qquad K_2 = \sum_{j=1}^{n} \alpha_j v_j v_j^T$$

$$= \sum_{i=1}^{n} \lambda_i v_i v_i^T - \sum_{j=1}^{n} \alpha_j v_j v_j^T$$

which cannot be broken as

$K(x,z) = \cancel{as}$, $a^T K a$

Hence it is not a valid kernel

d) $K(x,z) = a K_1(x,z) \quad (a>0)$

$K = \alpha K_1 \qquad \alpha > 0$

$\forall a \in \mathbb{R}^n, \quad a^T K a = \alpha a^T K_1 a \geq 0$

due to $\alpha > 0$ & validity of K_1

$\boxed{K = \alpha K_1(x,z) \text{ is a valid kernel}}$

e) $K(x,z) = b K_1(x,z)$

$K = b K_1 \qquad b < 0$

$\forall a \in \mathbb{R}^n, \quad a^T K a = a^T b K_1 a < 0$

Now b is negative & K_1 is ≥ 0. and valid.

Hence

$\boxed{K(x,y) = b K_1(x,y)}$ is not valid

f) $K(x,z) = K_1(x,z)\, K_2(x,z)$

Gram matrix is given by.

$$\boxed{K = K_1 \cdot K_2}$$

K_1 & K_2 positive semi definite
their eigen decomposition

$$K_1 = \sum_{i=1}^{n} \lambda_i u_i u_i^T \qquad K_2 = \sum_{j=1}^{n} \mu_j v_j v_j^T$$

$$\lambda_i \geq 0 \qquad \mu_j \geq 0$$

$$K = \sum_{i=1}^{n} \sum_{j=1}^{n} \underbrace{\lambda_i \mu_j}_{\gamma_k} (u_i u_i^T)(v_j v_j^T)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_k (u_i \cdot v_j)(u_i \cdot v_j)^T$$

$$\boxed{K = \sum_{k=1}^{n^2} \gamma_k w_k w_k^T}$$

with $\gamma_k = \lambda_i \mu_j \geq 0$ $\quad w_k = u \cdot v$

$$\forall a \in \mathbb{R}^n \quad a^T K a = \sum_{k=1}^{n^2} \gamma_k a^T w_k w_k^T a$$

$$K = \sum_{k=1}^{n^2} \gamma_k (w_k^T a)^2$$

Hence valid kernel

g) $K(x,z) = f(x)f(z)$ ($f: \mathbb{R}^n \to \mathbb{R}$, real valued func^n)

$$\phi : \mathcal{X} \to \mathcal{X}^\phi$$

$$x \to \phi(x)$$

A valid kernel is

$$k(x,z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{X}^\phi}$$

for any real valued func^n

$$\boxed{K(x,z) = f(x)f(z)}$$

Hence a valid kernel

h) $K(x,z) = k_3(\phi(x), \phi(z))$ (k_3: another valid kernel over $\mathbb{R}^d \to \mathbb{R}^d$)

Now $\phi(x), \phi(z)$ are transformation to higher dimensional feature space

& k_3 is a valid kernel in that space

Hence $\boxed{K(x,z) = k(\phi(x), \phi(z)) \text{ is a valid kernel}}$

i) $K(x,z) = P(K_1(x,z))$

$p(x)$ a polynomial function with positive coefficients

$P: R \to R$ with (+ve) coefficients.

P is a linear combination of power of kernel K_1 ~~witch~~ which is valid with +ve coefficien

since the power of K_1 of K_1 by itself.

this makes

$K(x,y) = p(K_1(x,z))$ a valid kernel

Q2

April 15, 2020

a) The category of each text article must depend on the meaning of its content. Explain why Naive-Bayes assumption is not too unrealistic for text categorization problem.

Ans. Bag of Words concepts(text categorization problems) treat each word individually and the order in which the words occur does not matter. Using the naive-bayes assumption of conditional independence, a lot of the dependence among features can be explained away by the underlying class.

b)Train the model based on the training data by MLE. For each class k among 4 different classes, you should learn the parameter (j|y)=k, which is the conditional probability p(x(j) |y = k). You should also learn the parameter (y=k), which is the prior probability of each class p(y = k). Report the confusion matrix and training accuracy by predicting the class labels of the training set by your trained Bernoulli Naive-Bayes model.

Ans.

[1]:
```
import os
import re
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
import matplotlib.pylab as plt
import scipy.sparse as sparse
import math
import sklearn.metrics as metrics
```

[2]:
```
os.chdir('C:\\Users\\dhana\\Courses\\MSBA\\Spring_Sem\\IDS575\\Assignment3')
```

[3]:
```
def data(file):
    datafile = open(file, 'r')
    dataLines = datafile.readlines()
    strip = [x.rstrip("\n") for x in dataLines]
    dataTokens = [re.split(":| ",x) for x in strip]
    Y = [int(x[0]) for x in dataTokens]
    Xlist = [np.reshape(np.array(x[1:len(x)],dtype=np.int32),(-1,2)) for x in dataTokens] #word,count array
    Xlist = [np.insert(Xlist[x],0,x,axis=1) for x in range(len(Xlist))]#appending instance number
    Xdata = np.concatenate(tuple(Xlist),axis=0) #concatenating all the instance arrays to one array
```

```
    X = csr_matrix((Xdata[:,2], (Xdata[:,0], Xdata[:,1])))#sparse matrix
    Bernoulli_X = csr_matrix((np.ones((Xdata.shape[0],),dtype=np.int32),␣
↪(Xdata[:,0], Xdata[:,1])))# bernoulli csr_matrix (word in doc implies 1 else␣
↪0)
    return([np.array(Y),X,Bernoulli_X,Xdata])
```

[4]:
```
train = data("articles.train")
test = data("articles.test")
```

[5]:
```
Ytrain = train[0]-1
Ytest = test[0]-1
Xtrain = train[1][:,1:]
Xtest = test[1][:,1:]
Berno_Xtrain = train[2][:,1:]
Berno_Xtest = test[2][:,1:]
```

[6]:
```
def count_unique(array):
    unique, count = np.unique(np.asarray(array), return_counts=True)
    return(dict(zip(unique, count)))
```

[7]:
```
def split(cdata):
    cdata1 = cdata[:1000,:]
    cdata2 = cdata[1000:2000,:]
    cdata3 = cdata[2000:3000,:]
    cdata4 = cdata[-1000:,:]
    return([cdata1,cdata2,cdata3,cdata4])
```

[8]:
```
splitM = split(Berno_Xtrain)
frequencyM = split(Xtrain)
```

[9]:
```
def BernoulliNB(csr,X):
    prior=[]
    for i in range(4):
        prior.append(csr[i].sum(axis=0))

    if prior[0].shape[1]<X.shape[1]:
        for i in range(4):
            prior[i]=np.hstack((prior[i],np.zeros((1,X.shape[1]-prior[i].
↪shape[1]))))

    cprob=[]
    for i in range(4):
        cprob.append(X.multiply(np.log((prior[i])/(1000))).tocsr())

    return(np.hstack([cprob[0].sum(axis=1),cprob[1].sum(axis=1),cprob[2].
↪sum(axis=1),cprob[3].sum(axis=1)]))
```

[10]:
```
metrics.confusion_matrix(Ytrain,np.
 ↪argmax(BernoulliNB(splitM,Berno_Xtrain),axis=1))/10
```

```
C:\Users\gitap\.conda\envs\bas575\lib\site-packages\ipykernel_launcher.py:12:
RuntimeWarning: divide by zero encountered in log
  if sys.path[0] == '':
```

[10]:
```
array([[ 99.9,   0. ,   0. ,   0.1],
       [  0.1,  99.2,   0.3,   0.4],
       [  0.2,   0.2,  99.5,   0.1],
       [  0. ,   0. ,   0. , 100. ]])
```

[11]:
```
metrics.confusion_matrix(Ytest,np.
 ↪argmax(BernoulliNB(splitM,Berno_Xtest),axis=1))/6
```

```
C:\Users\gitap\.conda\envs\bas575\lib\site-packages\ipykernel_launcher.py:12:
RuntimeWarning: divide by zero encountered in log
  if sys.path[0] == '':
```

[11]:
```
array([[100.        ,   0.        ,   0.        ,   0.        ],
       [ 94.33333333,   5.33333333,   0.        ,   0.33333333],
       [ 92.        ,   0.        ,   7.83333333,   0.16666667],
       [ 94.5       ,   0.        ,   0.16666667,   5.33333333]])
```

[12]:
```
def MultinomialNB(csr,X):
    prior=[]
    for i in range(4):
        prior.append(csr[i].sum(axis=0))

    if prior[0].shape[1]<X.shape[1]:
        for i in range(4):
            prior[i]=np.hstack((prior[i],np.zeros((1,X.shape[1]-prior[i].
 ↪shape[1]))))

    cprob=[]
    for i in range(4):
        cprob.append(X.multiply(np.log(prior[i]/(prior[i].sum(axis=1)))).
 ↪tocsr())

    return(np.hstack([cprob[0].sum(axis=1),cprob[1].sum(axis=1),cprob[2].
 ↪sum(axis=1),cprob[3].sum(axis=1)]))
```

[13]:
```
def MultinomialNB_laplace(csr,X):
    prior=[]
    for i in range(4):
        prior.append(csr[i].sum(axis=0))
```

```
    if prior[0].shape[1]<X.shape[1]:
        for i in range(4):
            prior[i]=np.hstack((prior[i],np.zeros((1,X.shape[1]-prior[i].
↪shape[1]))))

    cprob=[]
    for i in range(4):
        cprob.append(X.multiply(np.log(prior[i]+1/(prior[i].
↪sum(axis=1)+51949))).tocsr())

    return(np.hstack([cprob[0].sum(axis=1),cprob[1].sum(axis=1),cprob[2].
↪sum(axis=1),cprob[3].sum(axis=1)]))
```

[14]:
```
metrics.confusion_matrix(Ytrain,np.
↪argmax(MultinomialNB(frequencyM,Xtrain),axis=1))/10
```

```
C:\Users\gitap\.conda\envs\bas575\lib\site-packages\ipykernel_launcher.py:12:
RuntimeWarning: divide by zero encountered in log
  if sys.path[0] == '':
```

[14]:
```
array([[99.9,  0.1,  0. ,  0. ],
       [ 0.1, 99.8,  0.1,  0. ],
       [ 0.2,  0.4, 99.4,  0. ],
       [ 0. ,  0.1,  0. , 99.9]])
```

[15]:
```
metrics.confusion_matrix(Ytest,np.
↪argmax(MultinomialNB(frequencyM,Xtest),axis=1))/6
```

```
C:\Users\gitap\.conda\envs\bas575\lib\site-packages\ipykernel_launcher.py:12:
RuntimeWarning: divide by zero encountered in log
  if sys.path[0] == '':
```

[15]:
```
array([[100.        ,   0.        ,   0.        ,   0.        ],
       [ 94.33333333,   5.5       ,   0.16666667,   0.        ],
       [ 92.        ,   0.16666667,   7.83333333,   0.        ],
       [ 94.66666667,   0.        ,   0.16666667,   5.16666667]])
```

c) Learn the model parameters again by performing Laplace smoothing (in Lecture Notes #09a). Report the new confusion matrix and training accuracy when predicting on training data. Report another confusion matrix and test accuracy when predicting on test data. (Note: You cannot report test statistics without Laplace smoothing because there are unseen words in the test data as we experienced at Problem 5 in Homework 2)

Ans.

[16]:
```
def BernoulliNB_laplace(csr,X):
    prior=[]
    for i in range(4):
```

```
        prior.append(csr[i].sum(axis=0))

    if prior[0].shape[1]<X.shape[1]:
        for i in range(4):
            prior[i]=np.hstack((prior[i],np.zeros((1,X.shape[1]-prior[i].
 ↪shape[1]))))

    cprob=[]
    for i in range(4):
        cprob.append(X.multiply(np.log((prior[i]+1)/(1002))).tocsr())

    return(np.hstack([cprob[0].sum(axis=1),cprob[1].sum(axis=1),cprob[2].
 ↪sum(axis=1),cprob[3].sum(axis=1)]))
```

[17]:
```
metrics.confusion_matrix(Ytrain,np.
 ↪argmax(BernoulliNB_laplace(splitM,Berno_Xtrain),axis=1))/10
```

[17]:
```
array([[ 95.1,   0.1,   0. ,   4.8],
       [  0.3,  83.7,   0.1,  15.9],
       [  0.4,   0.1,  94.3,   5.2],
       [  0. ,   0. ,   0. , 100. ]])
```

[18]:
```
metrics.confusion_matrix(Ytest,np.
 ↪argmax(BernoulliNB_laplace(splitM,Berno_Xtest),axis=1))/6
```

[18]:
```
array([[77.33333333,  0.33333333,  0.16666667, 22.16666667],
       [ 0.33333333, 63.        ,  0.16666667, 36.5       ],
       [ 0.33333333,  0.        , 76.66666667, 23.        ],
       [ 0.        ,  0.16666667,  0.5       , 99.33333333]])
```

d) Report part (c) with multinomial Naive-Bayes model. Report correspondingly to part (c).

Ans.

[19]:
```
def MultinomailNB_laplace(csr,X):
    prior=[]
    for i in range(4):
        prior.append(csr[i].sum(axis=0))

    if prior[0].shape[1]<X.shape[1]:
        for i in range(4):
            prior[i]=np.hstack((prior[i],np.zeros((1,X.shape[1]-prior[i].
 ↪shape[1]))))

    cprob=[]
    for i in range(4):
        cprob.append(X.multiply(np.log(prior[i]+1/(prior[i].
 ↪sum(axis=1)+51949))).tocsr())
```

```
    return(np.hstack([cprob[0].sum(axis=1),cprob[1].sum(axis=1),cprob[2].
 ↪sum(axis=1),cprob[3].sum(axis=1)]))
```

[20]:
```
metrics.confusion_matrix(Ytrain,np.
 ↪argmax(MultinomialNB_laplace(frequencyM,Xtrain),axis=1))/10
```

[20]:
```
array([[ 99.7,   0. ,   0. ,   0.3],
       [  0.1,  96.7,   0.2,   3. ],
       [  0.3,   0.1,  99.2,   0.4],
       [  0. ,   0. ,   0. , 100. ]])
```

[21]:
```
metrics.confusion_matrix(Ytest,np.
 ↪argmax(MultinomialNB_laplace(frequencyM,Xtest),axis=1))/6
```

[21]:
```
array([[87.66666667,  0.33333333,  1.33333333, 10.66666667],
       [ 0.5       , 82.66666667,  0.        , 16.83333333],
       [ 0.83333333,  0.16666667, 91.        ,  8.        ],
       [ 0.33333333,  0.66666667,  0.33333333, 98.66666667]])
```

e) Compare and contrast the results from part (c) and (d). Justify why one works better than the other in our dataset. Explain, more in general, the weakness of NaıveBayes models by comparing Bernoulli event model and multinomial event model. (Hint: Think about what happen if the same word occurs multiple times in an article)

Ans. The multinomial laplace

Q3. Hidden Markov Model

a. Count the number of parameters to define the initial distribution, the transition distribution, and the emission distribution.

Initial distribution: k-1
Transition distribution: k(k-1)
Emission distribution: k(m-1)

Total: Sum of all three = $k^2 + km - k - 1$

b. Does the number of parameters depend on the number of states? Briefly justify your answer
No it depends on the number of values the states can take.

c. $S_{t+1} \perp S_{t-1} \mid S_t$ (The future is independent of the past, given the present)

S	$\pi(S)$	T_{SA}	T_{SB}	E_{S0}	E_{S1}
A	0.99	0.99	0.01	0.8	0.2
B	0.01	0.01	0.99	0.1	0.9

d) For the sequence $O_1 = 0, O_2 = 1, O_3 = 0$:
Using $\alpha_1(s) = E_{s0_1}\pi(s)$,
$\forall t \geq 2$, $\alpha_t(s) = E_{sO_t} \sum_{s' \in S} T_{s's}\, \alpha_{t-1}(s')$

Time t	1	2	3
Obs O_t	0	1	0
$\alpha_t(A)$	0.792	0.156818	0.124264
$\alpha_t(B)$	0.001	0.008019	0.000950699

$\alpha_1(A) = 0.8 \times 0.99 = 0.792$
$\alpha_1(B) = 0.1 \times 0.01 = 0.001$
$\alpha_2(A) = 0.2(0.792 \times 0.99 + 0.001 \times 0.01)$
$= 0.156818$
$\alpha_2(B) = 0.9(0.792 \times 0.01 + 0.001 \times 0.99)$
$= 0.008019$
$\alpha_3(A) = 0.8(0.156818 \times 0.99 + 0.008019 \times 0.01)$
$= 0.124264$
$\alpha_3(B) = 0.1(0.156818 \times 0.01 + 0.008019 \times 0.99)$
$= 0.000950699$

$p(O_1 = 0, O_2 = 1, O_3 = 0)$
$= \sum_{s \in S} \alpha_T(s)$
$= \alpha_3(A) + \alpha_3(B)$
$= 0.124264 + 0.000950699$
$= 0.125214699$

e) For the same sequence, using the backward algo:
Using $\beta_T(s) = 1$
$\beta_t(s) = \sum_{s' \in S} T_{ss'} E_{s'O_{t+1}} \beta_{t+1}(s')$

Time t	1	2	3
Obs O_t	0	1	0
$\beta_t(A)$	0.157977	0.793	1
$\beta_t(B)$	0.096923	0.107	1

$\beta_3(A) = 1$
$\beta_3(B) = 1$
$\beta_2(A) = 0.99 \times 0.8 \times 1 + 0.01 \times 0.1 \times 1$
$= 0.793$
$\beta_2(B) = 0.01 \times 0.08 \times 1 + 0.99 \times 0.01 \times 1$
$= 0.107$
$\beta_1(A) = 0.99 \times 0.02 \times 0.793 + 0.01 \times 0.9 \times 0.107$
$= 0.157977$
$\beta_1(B) = 0.01 \times 0.2 \times 0.793 + 0.99 \times 0.9 \times 0.107$
$= 0.096923$

$p(O_1 = 0, O_2 = 1, O_3 = 0)$
$= 0.792 \times 0.157977 + 0.001 \times 0.096923$
$= 0.125214707$

f) Yes, the two results from the forward & backward algorithms agree.
To find the most likely sequence of values of these states, we compare $\alpha_t(A) \times \beta_t(A)$ with $\alpha_t(B) \times \beta_t(B)$. If $\alpha_t(A) \cdot \beta_t(A)$ is greater, most likely state at time t will be A & vice versa.
The comparisons are:

$\alpha_1(A)\beta_1(A) = 0.792 \times 0.159971 = 0.1257178$

$\alpha_1(B)\beta_1(B) = 0.001 \times 0.096923 = 0.000096923$

For $t=1$, state is A

$\alpha_2(A)\beta_2(A) = 0.156818 \times 0.793 = 0.1243567$

$\alpha_2(B)\beta_2(B) = 0.008019 \times 0.107 = 0.000858033$

For $t=2$, state is A

$\alpha_3(A)\beta_3(A) = 0.124264 \times 1 = 0.124264$

$\alpha_3(B)\beta_3(B) = 0.000950699 \times 1 = 0.000950699$

For $t=3$, state is A

Therefore the most likely sequence is A, A, A.

g) The Viterbi sequence is computed as follows:

$V_1(A) = 0.99 \times 0.8 = 0.792$

$V_1(B) = 0.01 \times 0.1 = 0.001$

For $t=1$, state is A

$V_2(A) = 0.792 \times 0.99 \times 0.2 = 0.156816$

$V_2(B) = 0.792 \times 0.01 \times 0.9 = 0.007128$

For $t=2$, state is A

$V_3(A) = 0.156816 \times 0.99 \times 0.8 = 0.1241963$

$V_3(B) = 0.156816 \times 0.01 \times 0.1 = 0.000156816$

For $t=3$, state is A

Therefore, the most likely sequence using Viterbi Algorithm is A, A, A

h) To find the most likely sequence of values for each state separately, we would just have to compare products of $\pi(S) \cdot E_{S o_t}$.

For sequence 0,1,0:

$t=1$, $p(A)$: $0.8 \times 0.99 = 0.792 \rightarrow A$
$p(B)$: $0.1 \times 0.01 = 0.001$

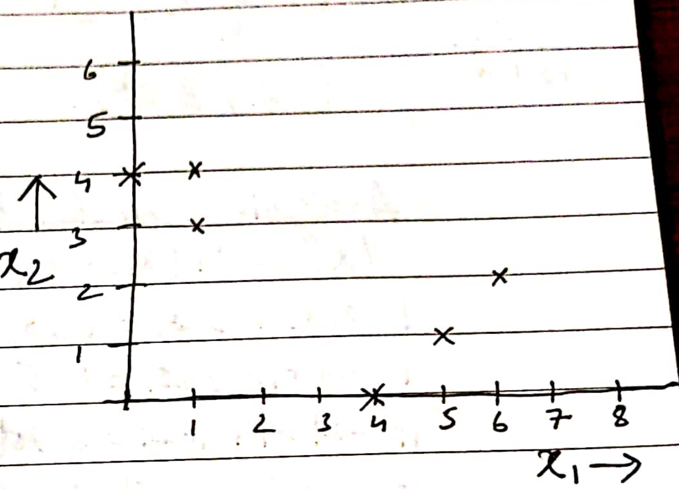
$t=2$, $p(A) = 0.2 \times 0.99 = 0.198 \rightarrow A$
$p(B)$: $0.9 \times 0.01 = 0.009$

$t=3$, $p(A) = 0.8 \times 0.99 = 0.792 \rightarrow A$
$p(B) = 0.1 \times 0.01 = 0.001$

The resulting sequence A, A, A is equivalent to what we observed in part (f) & (g).
This is because $\pi(B)$ is really low in this case.
However this does not hold in general. In any case where $\pi(B)$ is moderately higher, the sequence would possibly change.

Problem 4: K-Means Clustering

i	$x_1^{(i)}$	$x_2^{(i)}$	k	k_1
1	1	4	1	1
2	1	3	2	1
3	0	4	1	1
4	5	1	1	2
5	6	2	1	1
6	4	0	2	2



Calculating Centroid for Clusters

Centroid for C_1 $(k=1)$

$$\left(\frac{1+0+5+6}{4}, \frac{4+4+1+2}{4}\right)$$

$$\left(\frac{12}{4}, \frac{11}{4}\right)$$

$$(3, 2.75)$$

Centroid for C_2 $(k=2)$

$$\left(\frac{1+4}{2}, \frac{3+0}{2}\right)$$

$$(2.5, 1.5)$$

Calculating distance using Manhattan Method

$\lvert x_1-3\rvert + \lvert x_2-2.75\rvert$	dist	$\lvert x_1-2.5\rvert + \lvert x_2-1.5\rvert$
$\lvert 1-3\rvert + \lvert 4-2.75\rvert = 2+1.25 = 3.25$	$<$	$\lvert 1-2.5\rvert + \lvert 4-1.5\rvert = 4$
$\lvert 1-3\rvert + \lvert 3-2.75\rvert = 2+0.25 = 2.25$	$<$	$\lvert 1-2.5\rvert + \lvert 3-1.5\rvert = 3$
$\lvert 0-3\rvert + \lvert 4-2.75\rvert = 3+1.25 = 4.25$	$<$	$\lvert 0-2.5\rvert + \lvert 4-1.5\rvert = 5$
$\lvert 5-3\rvert + \lvert 1-2.75\rvert = 3.75$	$>$	$\lvert 5-2.5\rvert + \lvert 1-1.5\rvert = 3$
$\lvert 6-3\rvert + \lvert 2-2.75\rvert = 3.75$	$<$	$\lvert 6-2.5\rvert + \lvert 2-1.5\rvert = 4$
$\lvert 4-3\rvert + \lvert 0-2.75\rvert = 3.75$	$>$	$\lvert 4-2.5\rvert + \lvert 0-1.5\rvert = 3$

Assigning datapoints, based on the distance, to the cluster. (k_1)

Calculating new centroids based on new clusters

Centroid for Cluster 1 (updated) → $\left(\frac{1+1+0+6}{4}, \frac{4+3+4+2}{4}\right)$

$(2, 3.25)$.

Centroid for Cluster 2 (updated) $\left(\frac{5+4}{2}, \frac{1+0}{2}\right)$

$(4.5, 0.5)$

Assigning ~~clusters~~ datapts to clusters based on new centroids

$\lvert x_1-2\rvert + \lvert x_2-3.25\rvert$		$\lvert x_1-4.5\rvert + \lvert x_2-0.5\rvert$	updated clusters
$\lvert 1-2\rvert+\lvert 4-3.25\rvert = 1.75$	$<$	$\lvert 1-4.5\rvert+\lvert 4-0.5\rvert = 3.5$	1
$\lvert 1-2\rvert+\lvert 3-3.25\rvert = 1.25$	$<$	$\lvert 1-4.5\rvert+\lvert 3-0.5\rvert = 2.5$	1
$\lvert 0-2\rvert+\lvert 4-3.25\rvert = 2.75$	$<$	$\lvert 0-4.5\rvert+\lvert 4-0.5\rvert = 3.5$	1
$\lvert 5-2\rvert+\lvert 1-3.25\rvert = 5.25$	$>$	$\lvert 5-4.5\rvert+\lvert 1-0.5\rvert = 0.5$	2
$\lvert 6-2\rvert+\lvert 2-3.25\rvert = 5.25$	$>$	$\lvert 6-4.5\rvert+\lvert 2-0.5\rvert = 1.5$	2
$\lvert 4-2\rvert+\lvert 0-3.25\rvert = 5.25$	$>$	$\lvert 4-4.5\rvert+\lvert 0-0.5\rvert = 1$	2

Centroid for C_1

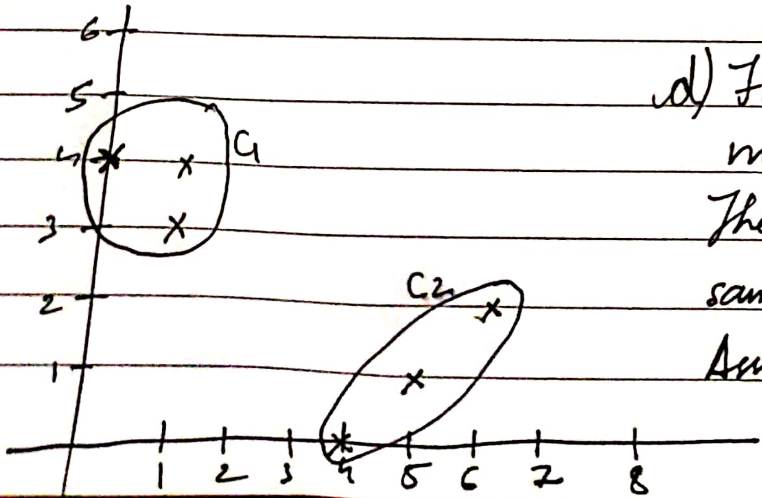
$C_1 = \left(\frac{1+1+0}{3}, \frac{4+3+4}{3}\right) = \left(\frac{2}{3}, \frac{11}{3}\right) = (0.66, 3.66)$

$C_2 = \left(\frac{5+6+4}{3}, \frac{1+2+0}{3}\right)$

$C_2 = (5, 1)$.

$\lvert x_1-0.66\rvert + \lvert x_2-3.66\rvert$		$\lvert x_1-5\rvert + \lvert x_2-1\rvert$	updated cluster (k_3)
$\lvert 1-0.66\rvert+\lvert 4-3.66\rvert = 0.66$	$<$	$\lvert 1-5\rvert+\lvert 4-1\rvert = 7$	1
$\lvert 1-0.66\rvert+\lvert 3-3.66\rvert = 0.99.$	$<$	$\lvert 1-5\rvert+\lvert 3-1\rvert = 6$	1
$\lvert 0-0.66\rvert+\lvert 4-3.66\rvert = 0.99$	$<$	$\lvert 0-5\rvert+\lvert 4-1\rvert = 8$	1
$\lvert 5-0.66\rvert+\lvert 1-3.66\rvert = 6.99$	$>$	$\lvert 5-5\rvert+\lvert 1-1\rvert = 0$	2
$\lvert 6-0.66\rvert+\lvert 2-3.66\rvert = 6.99$	$>$	$\lvert 6-5\rvert+\lvert 2-1\rvert = 2$	2
$\lvert 4-0.66\rvert+\lvert 0-3.66\rvert = 6.99$	$>$	$\lvert 4-5\rvert+\lvert 0-1\rvert = 2$	2

As the centroids & the cluster labels stop changing, ~~update~~ updated clusters are:- (k_3)



d) Final clustering does not match with initial clustering. The ~~also~~ clustering result always same regardless of initial cluster Assignment as shown in the R-Code.