Machine Intelligence 2

Exercise 09

K-means Clustering

Group Members:
Xugang Zhou
Fangzhou Yang

Tutor:
Timm Lochmann

June 26, 2013

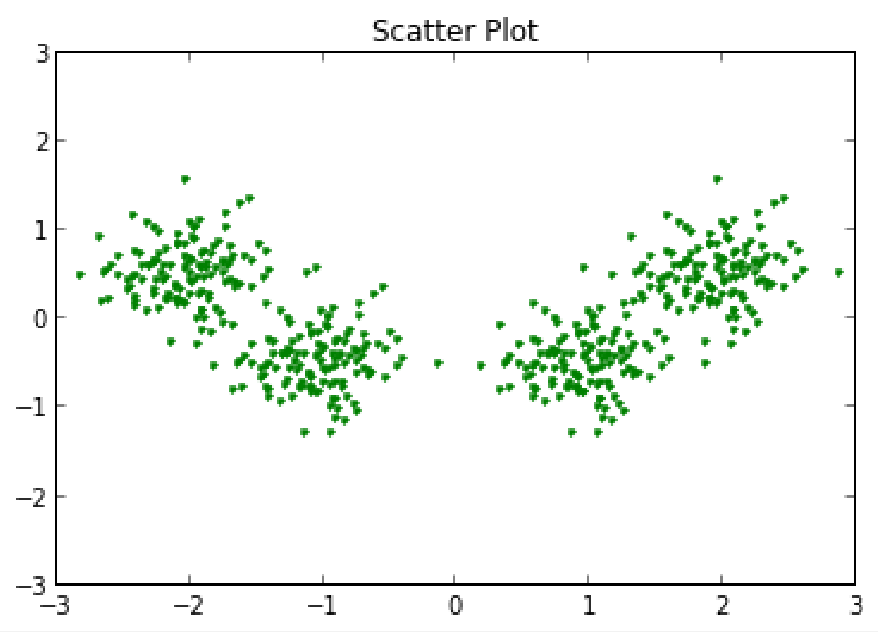
1 9.1 K-Means Clustering

```
#function
from numpy import *
import matplotlib
import matplotlib.pyplot as plt

def plotScatter(X,Y,title,c,ran):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.axis([-ran,ran,-ran,ran])
    ax.plot(X,Y,c+’.’)
    ax.set_title(title)
    plt.show()

def plotCluster(X,Y,title,colors,ran,assign, W):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.axis([-ran,ran,-ran,ran])
    for i in range (len(X)):
        p = assign[i]
        ax.plot(X[i],Y[i],colors[p]+’.’)
    for i in range (len(W)):
        ax.plot(W[i][0],W[i][1],colors[i]+’D’)
        ax.plot(W[i][0],W[i][1],’y’+’H’)
    ax.set_title(title)
    plt.show()

def plotLine(data,title):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    xaxis = [i+1 for i in range(len(data))]
    ax.plot(xaxis,data,’.-’)
    ax.set_title(title)
    plt.show()

def distance(x1,y1,x2,y2):
    return math.sqrt((y2-y1)**2 + (x2-x1)**2)

def assignPoint(W,x,y):
    k = len(W)
    for i in range(k):
        if(i==0):
            min_distance = distance(W[i][0],W[i][1],x,y)
            min_p = i
        else:
            d = distance(W[i][0],W[i][1],x,y)
            if(d < min_distance):
                min_distance = d
                min_p = i
    return min_p
```

```
def drawBoundary(X,Y,title,colors,ran,assign, W, fine):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.axis([-ran,ran,-ran,ran])
    for i in range (len(X)):
        p = assign[i]
        ax.plot(X[i],Y[i],colors[p]+’.’)
    for i in range (len(W)):
        ax.plot(W[i][0],W[i][1],colors[i]+’D’)
        ax.plot(W[i][0],W[i][1],’y’+’H’)
    ax.set_title(title)

    ran = float(ran)
    unit = 2*ran/fine
    print ’unit’,unit
    for i in range(fine):
        x = -ran + i*unit
        for j in range(fine):
            y = -ran + j*unit
            #print x,y
            p0 = assignPoint(W,x,y)
            p1 = assignPoint(W,x+unit,y)
            p2 = assignPoint(W,x-unit,y)
            p3 = assignPoint(W,x,y+unit)
            p4 = assignPoint(W,x,y-unit)
            if(p0==p1 and p1==p2 and p2==p3 and p3==p4):
                continue
            else:
                ax.plot(x,y,’y.’)
    plt.show()
```

```
#read data
data = loadtxt("cluster.dat")
print data.shape

X = data[0,:]
Y = data[1,:]

plotScatter(X,Y,"Scatter Plot", ’g’, 3)
```

```
(2, 500)
```

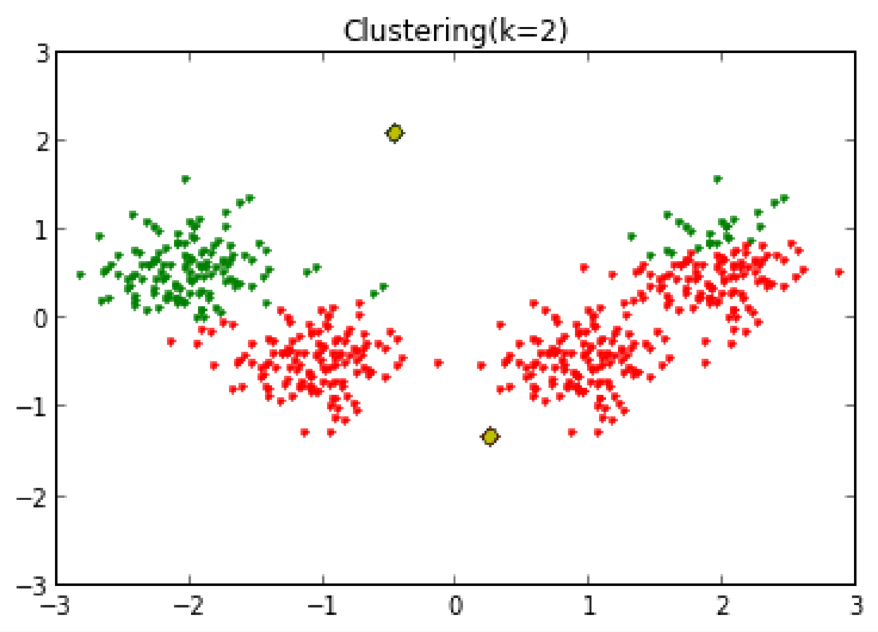


```
def distance(x1,y1,x2,y2):
    return math.sqrt((y2-y1)**2 + (x2-x1)**2)

def K_Means(X,Y,k,t_max,W_init):
    assign = [0 for i in range(len(X))]
    error = [0 for i in range(t_max)]
    W = W_init[:][:]
    for i in range (t_max):
        for n in range (len(X)):
            for p in range(k):
                if(p == 0):
                    min_distance = distance(X[n],Y[n],W[p][0],W[p][1])
                    min_p = p
                else:
                    tmp = distance(X[n],Y[n],W[p][0],W[p][1])
                    if (tmp < min_distance):
                        min_distance = tmp
                        min_p = p
            assign[n] = min_p

        #draw
        print ’iteration ’ + str(i+1) +’ done!’
        colors = [’r’,’g’,’b’,’c’,’m’]
        plotCluster(X,Y,"Clustering(k="+str(k)+")", colors, 3, assign, W)

        #update W
        count = [0 for o in range(k)]
        for p in range(k):
            W[p][0] = 0
            W[p][1] = 0
        for n in range(len(X)):
            tmp = assign[n]
            W[tmp][0] += X[n]
            W[tmp][1] += Y[n]
            count[tmp] += 1
        for p in range(k):
            if(count[p] >0):
                W[p][0] /= count[p]
                W[p][1] /= count[p]

        #claculate the error
        error[i] = 0
        for n in range(len(X)):
            error[i] += distance(X[n],Y[n],W[assign[n]][0],W[assign[n]][1])
        error[i] /= 2*len(X)
    return assign,error,W
```

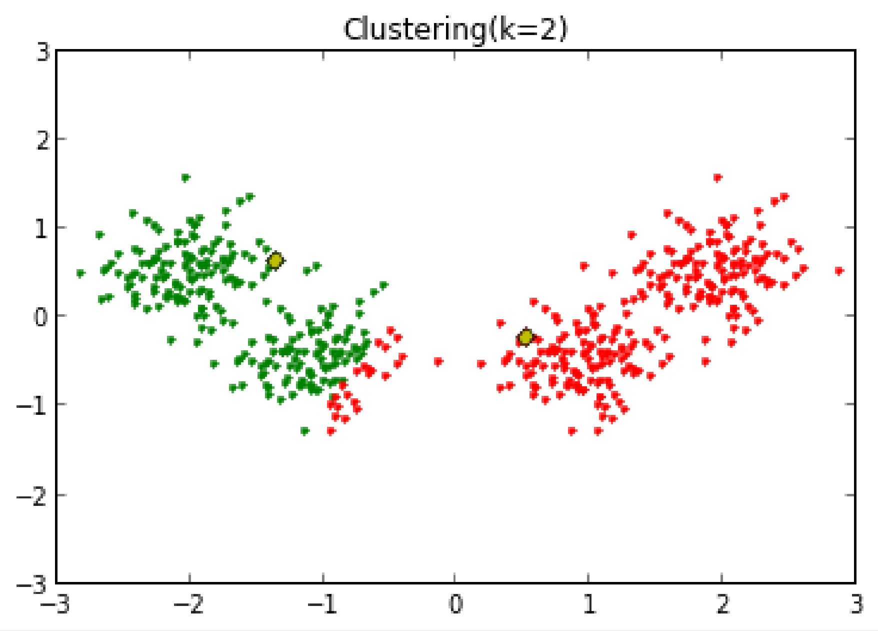
```
#init W
def init_w(k):
    random.seed(100)
    W_init = [[0 for j in range(2)] for i in range(k)]
    for p in range(k):
        W_init[p][0] = random.random() * 6 - 3
        W_init[p][1] = random.random() * 6 -3
    return W_init
```

```
# k=2
t_max = 5

k = 2
W_init = init_w(k)
print matrix(W_init)


assign,error,W = K_Means(X,Y,k,t_max,W_init)
```

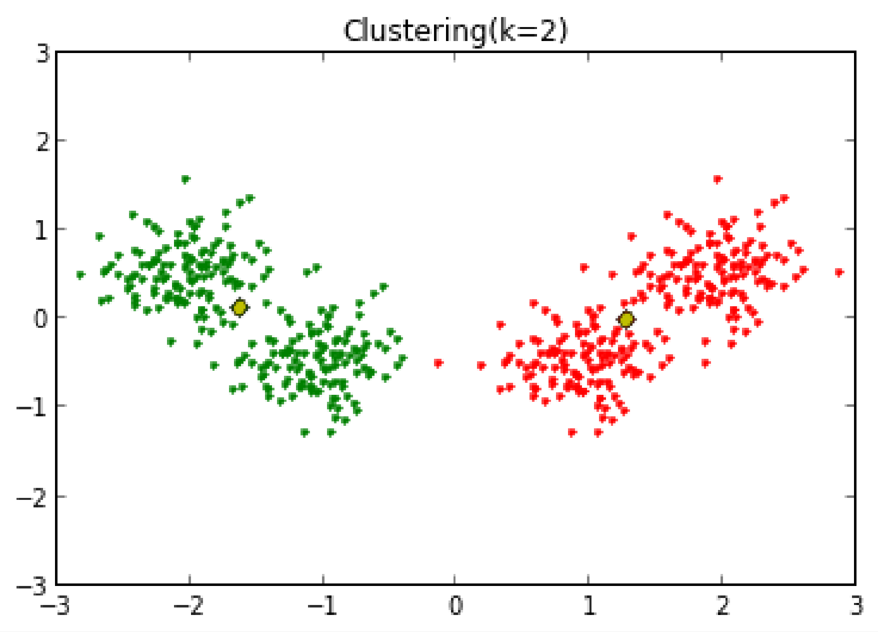
```
[[ 0.26042965 -1.32978369]
 [-0.45289446  2.06865679]]
iteration 1 done!
```



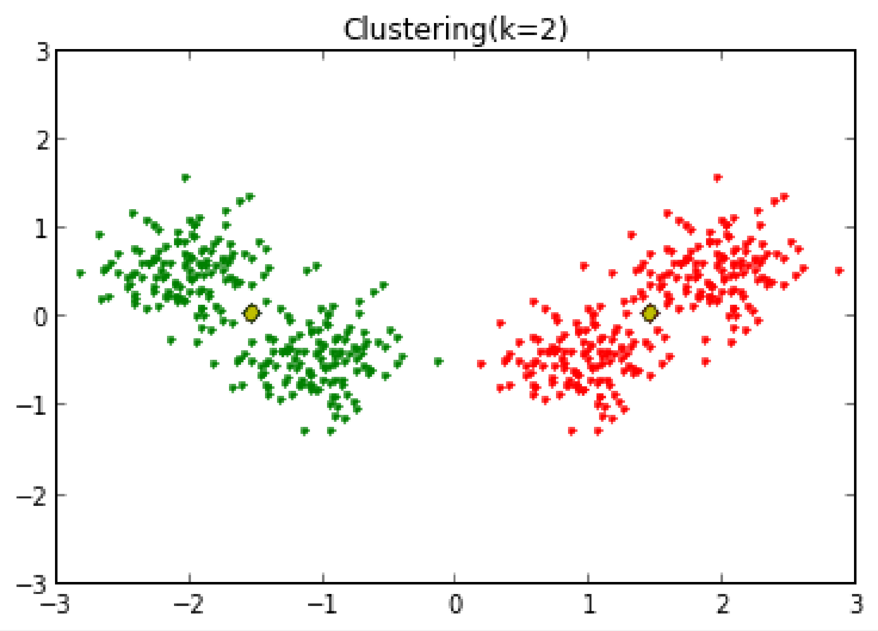
```
iteration 2 done!
```



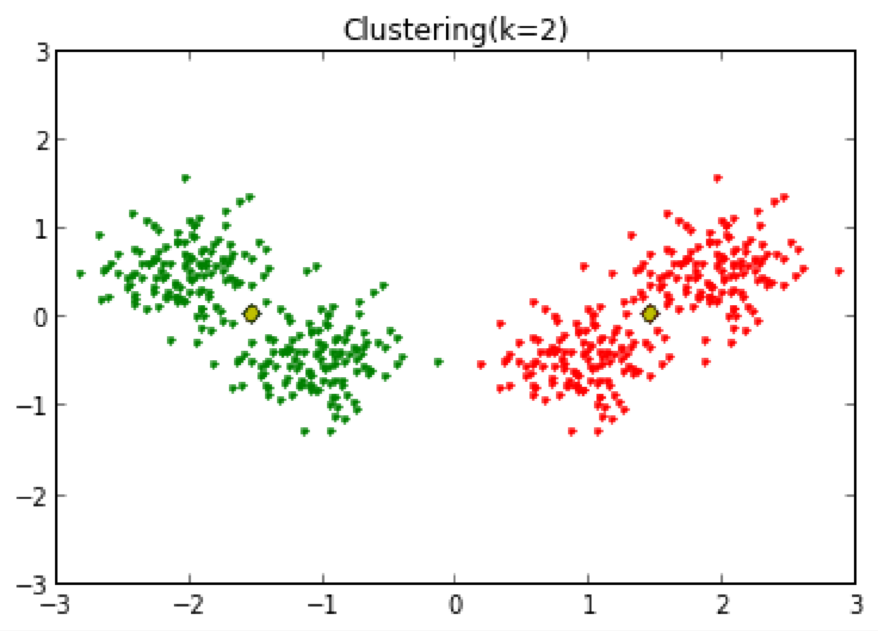
```
iteration 3 done!
```



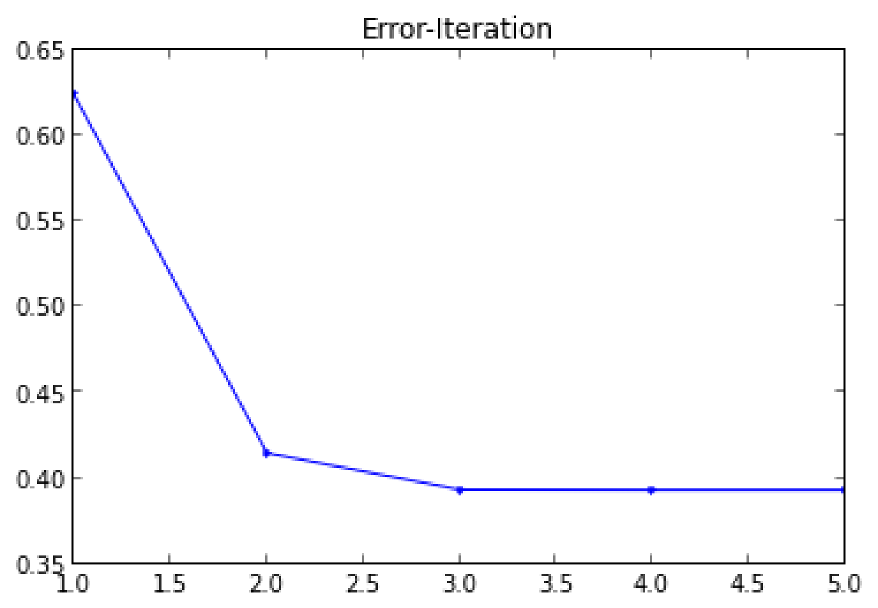
```
iteration 4 done!
```



```
iteration 5 done!
```

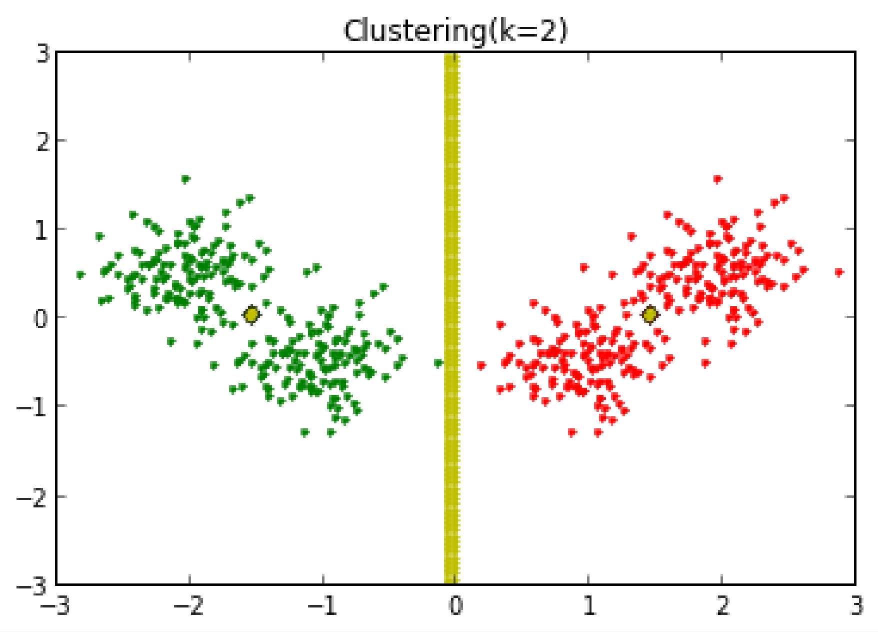


```
plotLine(error,"Error-Iteration")
```



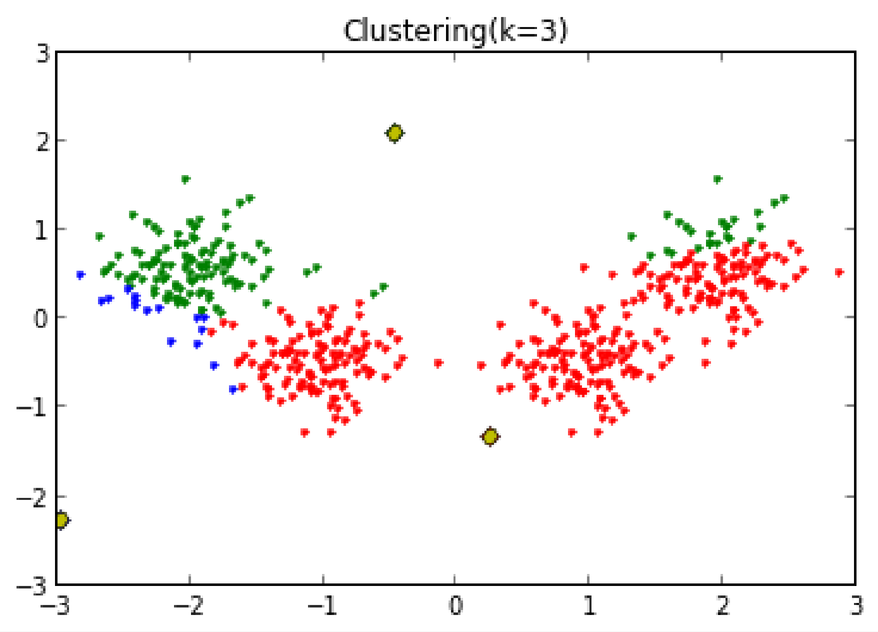
```
colors = ['r','g','b','c','m']
drawBoundary(X,Y,"Clustering(k="+str(k)+")",colors, 3, assign, W, 100)
```

```
unit 0.06
```

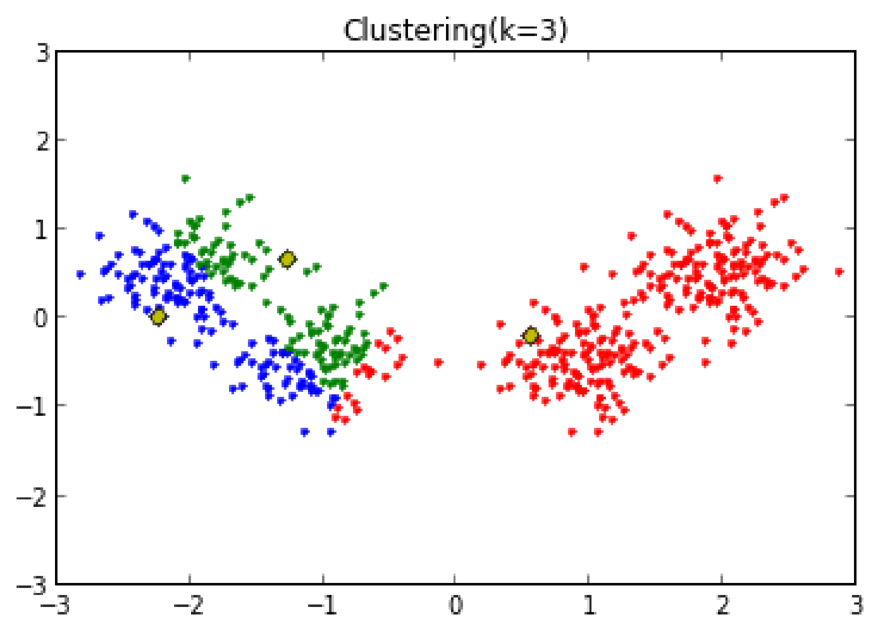


```
# k=3
t_max = 5

k = 3
W_init = init_w(k)
print matrix(W_init)


assign,error,W = K_Means(X,Y,k,t_max,W_init)
```

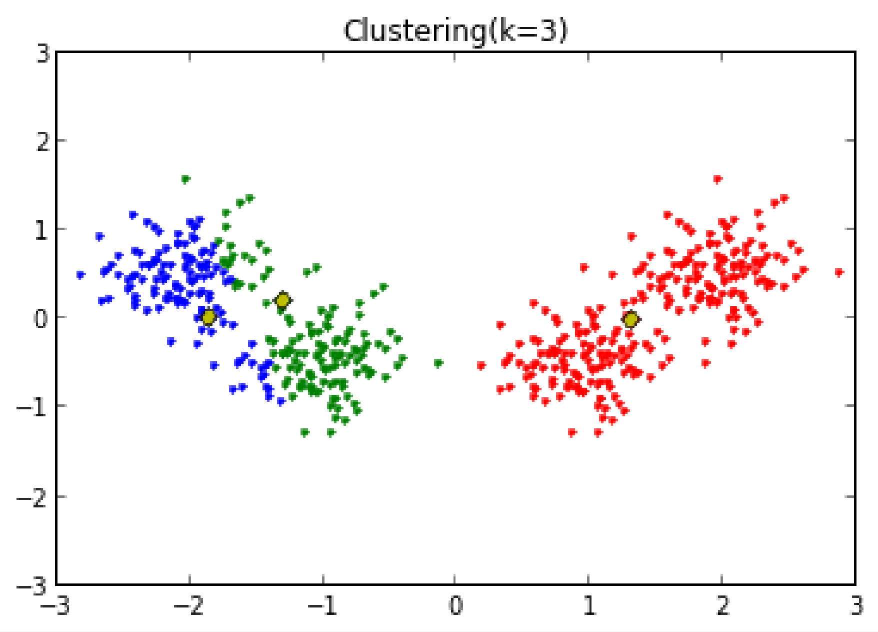
```
[[ 0.26042965 -1.32978369]
 [-0.45289446  2.06865679]
 [-2.97168686 -2.27058528]]
iteration 1 done!
```



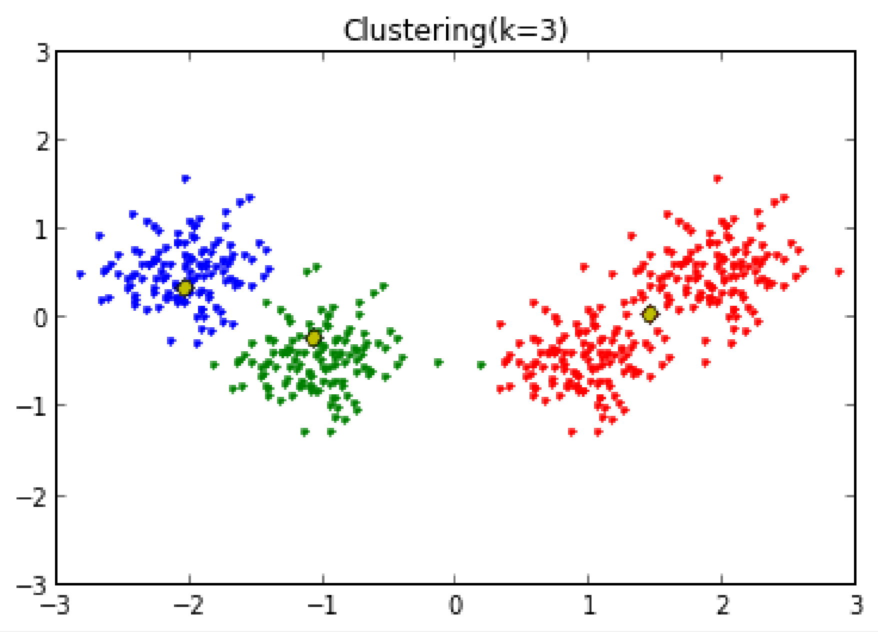
```
iteration 2 done!
```



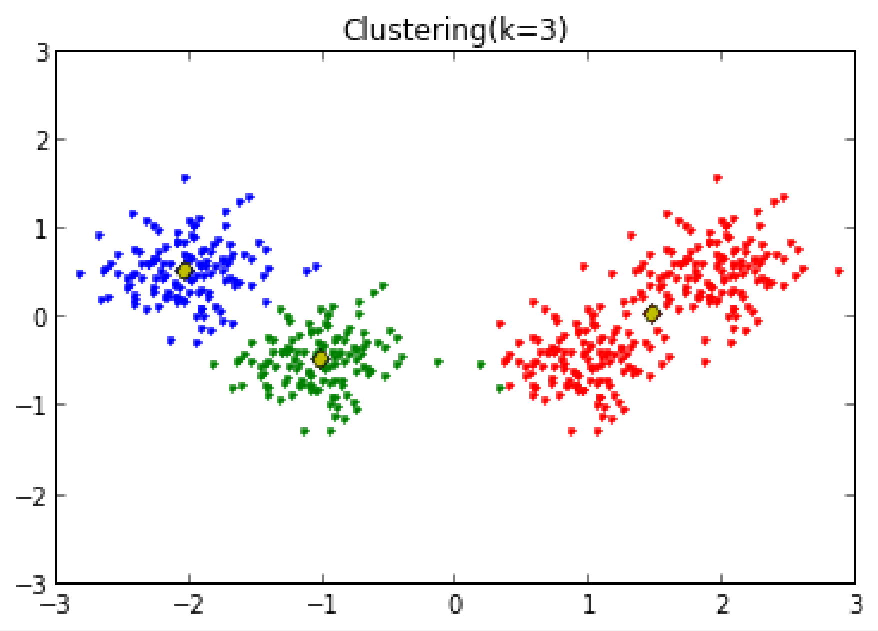
```
iteration 3 done!
```



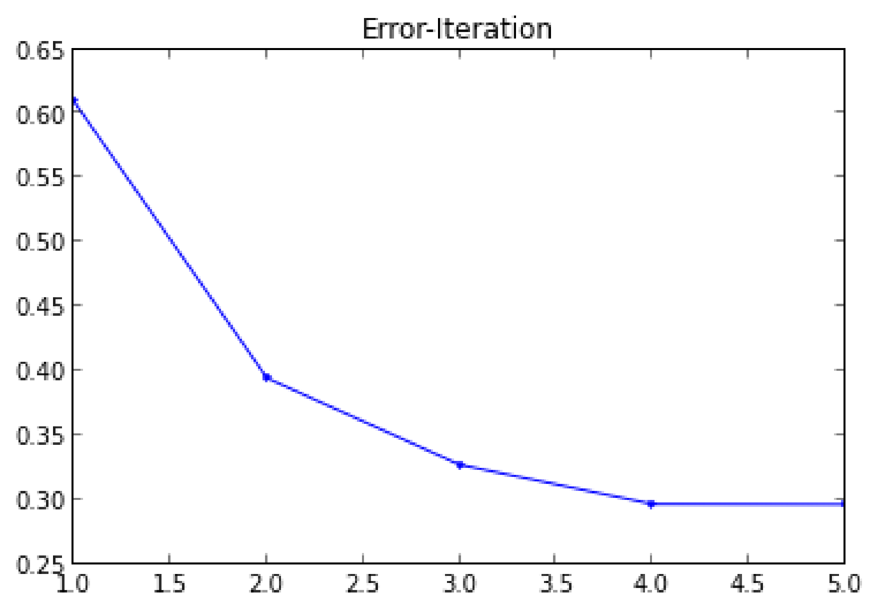
```
iteration 4 done!
```



```
iteration 5 done!
```

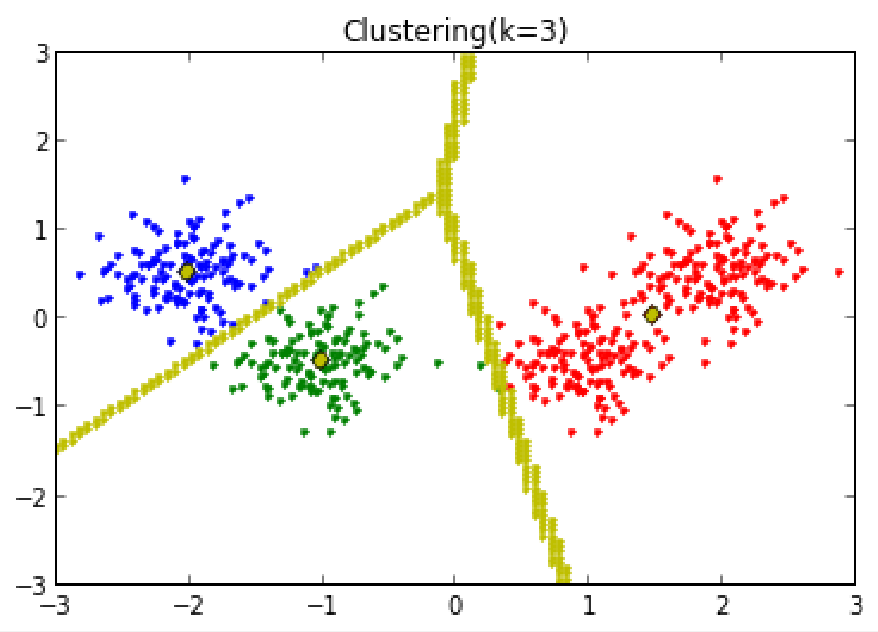


```
plotLine(error,"Error-Iteration")
```



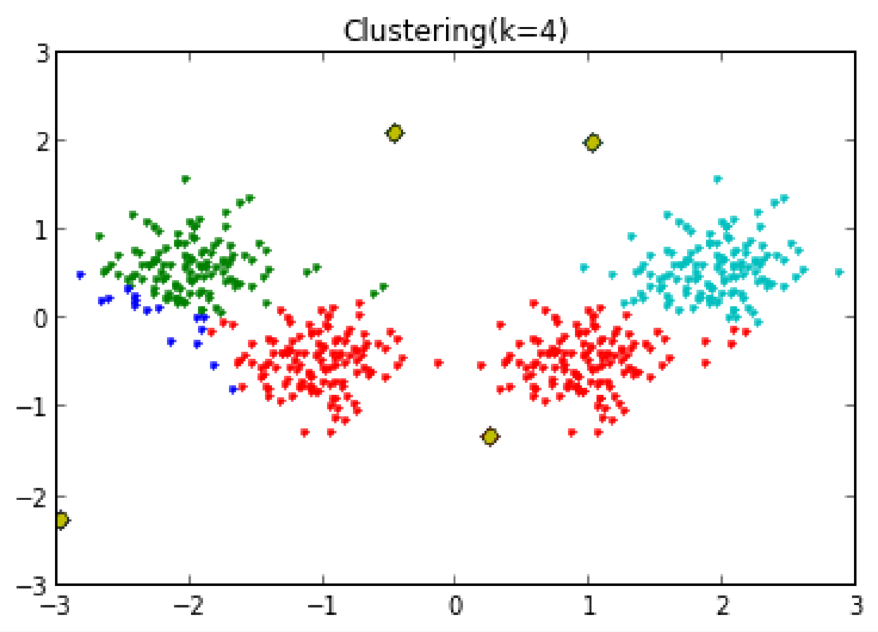
```
colors = ['r','g','b','c','m']
drawBoundary(X,Y,"Clustering(k="+str(k)+")",colors, 3, assign, W, 100)
```

```
unit 0.06
```

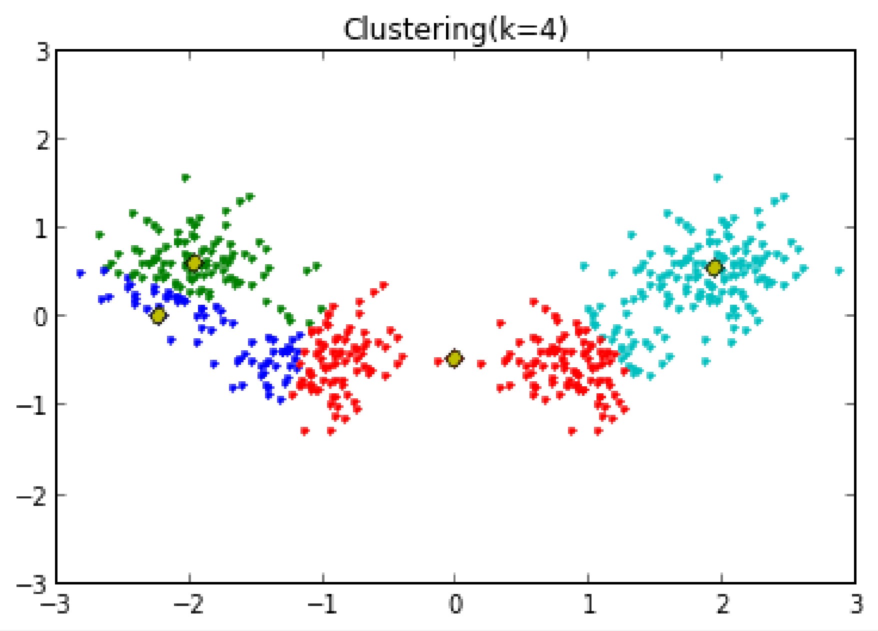


```
# k=4
t_max = 5

k = 4
W_init = init_w(k)
print matrix(W_init)


assign,error,W = K_Means(X,Y,k,t_max,W_init)
```

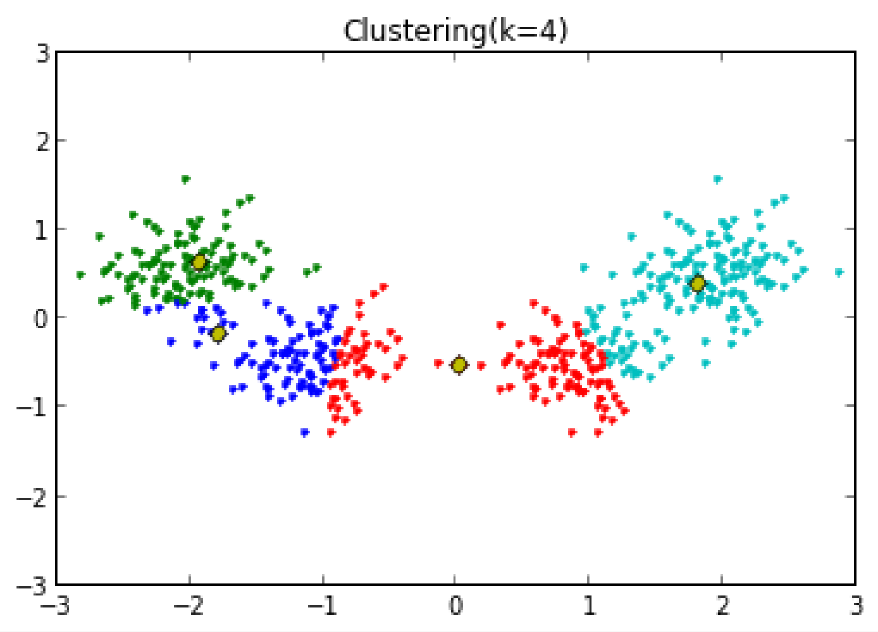
```
[[ 0.26042965 -1.32978369]
 [-0.45289446  2.06865679]
 [-2.97168686 -2.27058528]
 [ 1.02449451  1.95511653]]
iteration 1 done!
```



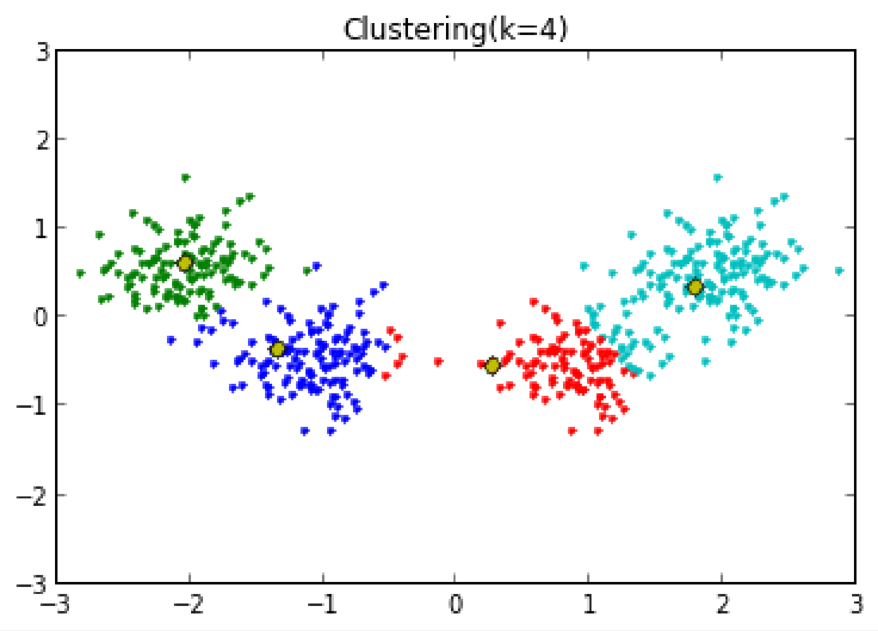
```
iteration 2 done!
```



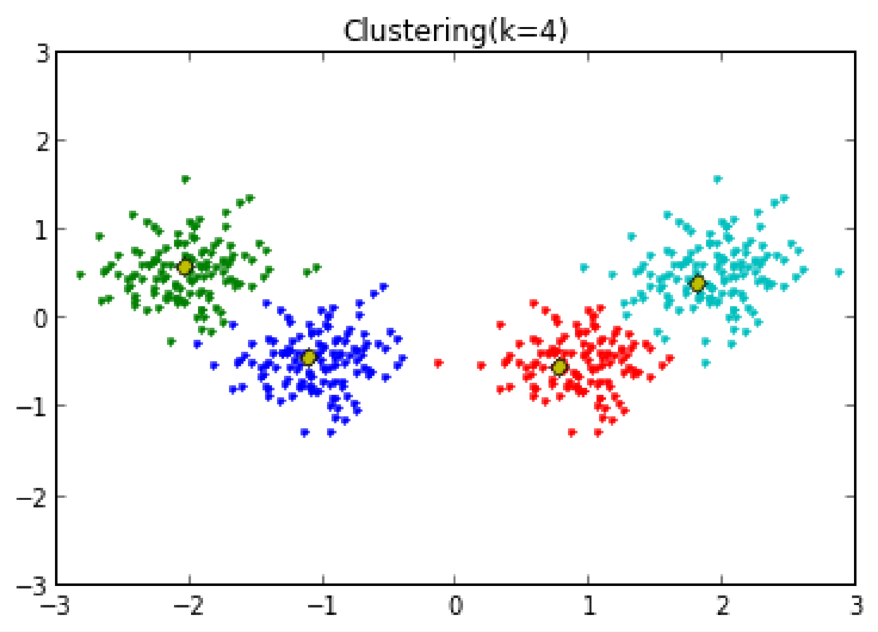
```
iteration 3 done!
```



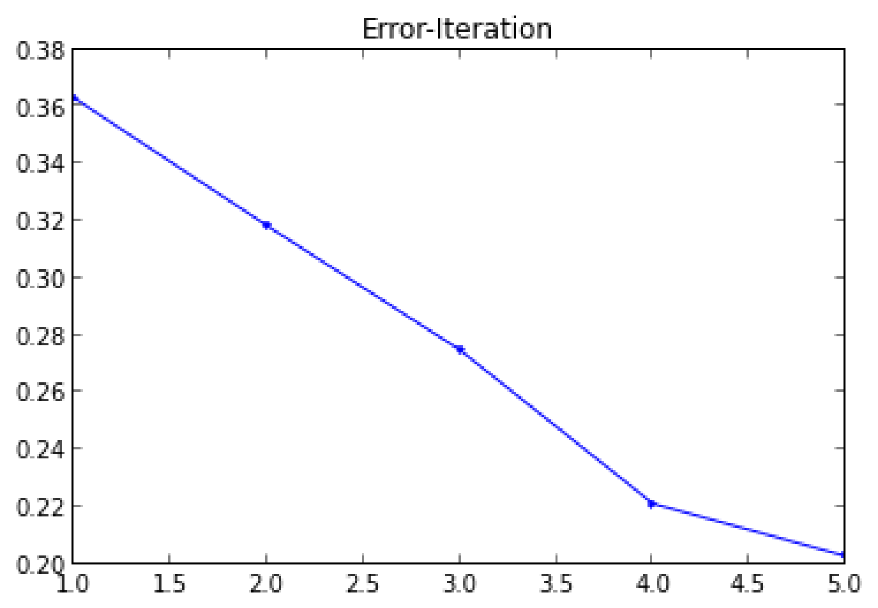
```
iteration 4 done!
```



```
iteration 5 done!
```

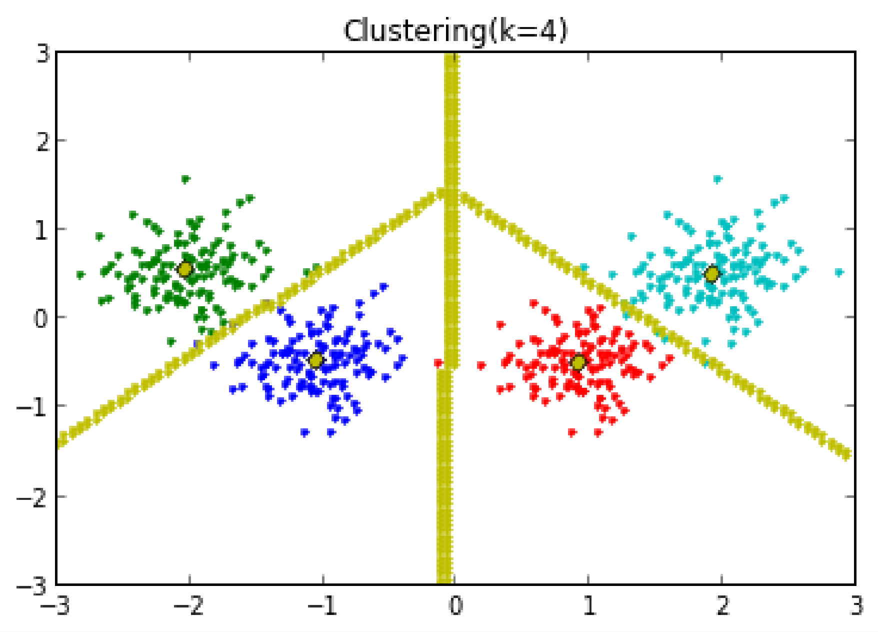


```
plotLine(error,"Error-Iteration")
```



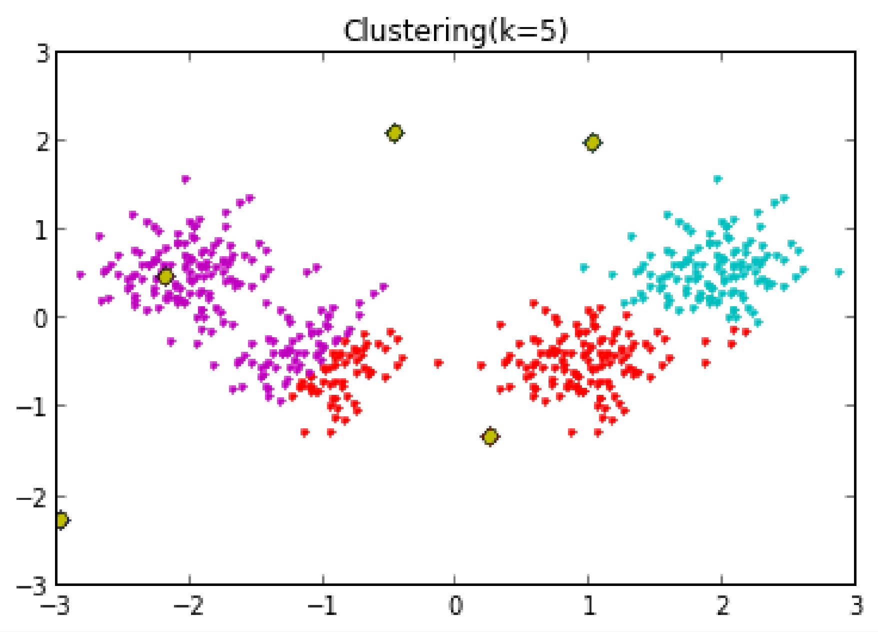
```
colors = ['r','g','b','c','m']
drawBoundary(X,Y,"Clustering(k="+str(k)+")",colors, 3, assign, W, 100)
```

```
unit 0.06
```

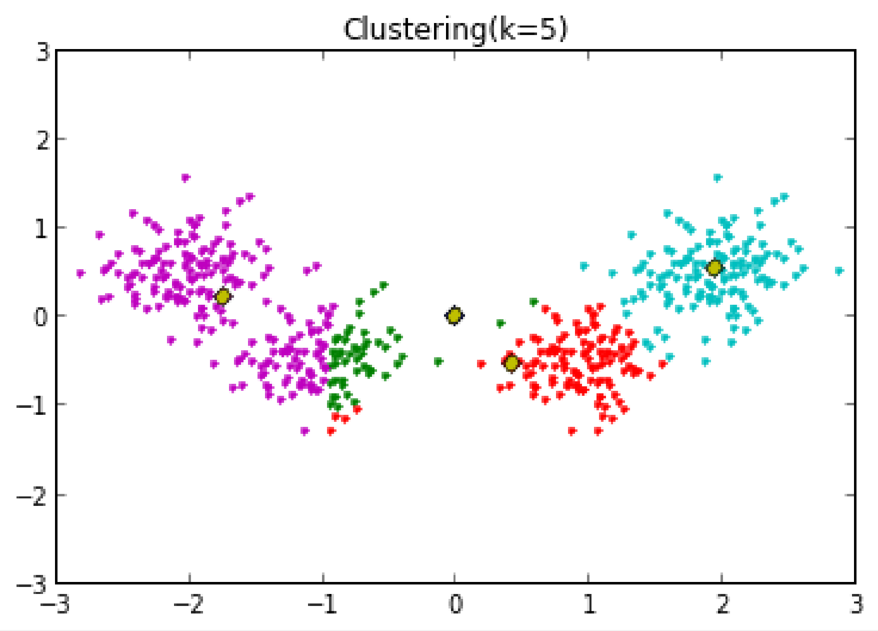


```
# k=5
t_max = 5

k = 5
W_init = init_w(k)
print matrix(W_init)


assign,error,W = K_Means(X,Y,k,t_max,W_init)
```

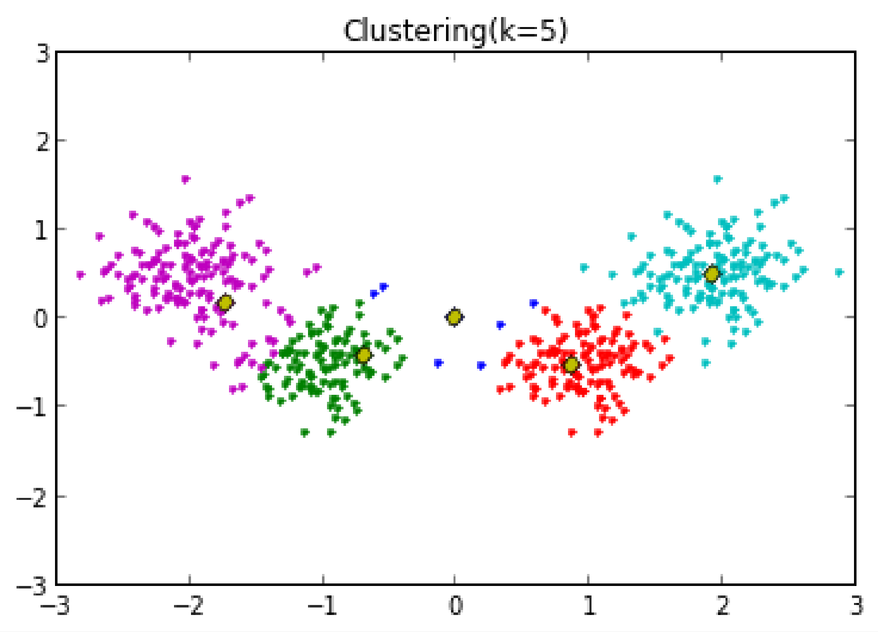
```
[[ 0.26042965 -1.32978369]
 [-0.45289446  2.06865679]
 [-2.97168686 -2.27058528]
 [ 1.02449451  1.95511653]
 [-2.17976046  0.45055998]]
iteration 1 done!
```



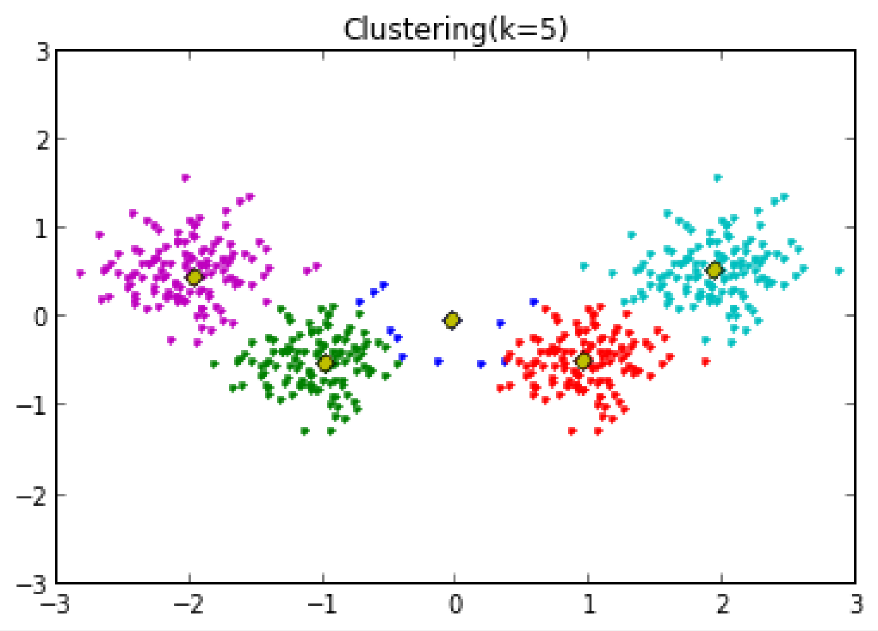
```
iteration 2 done!
```



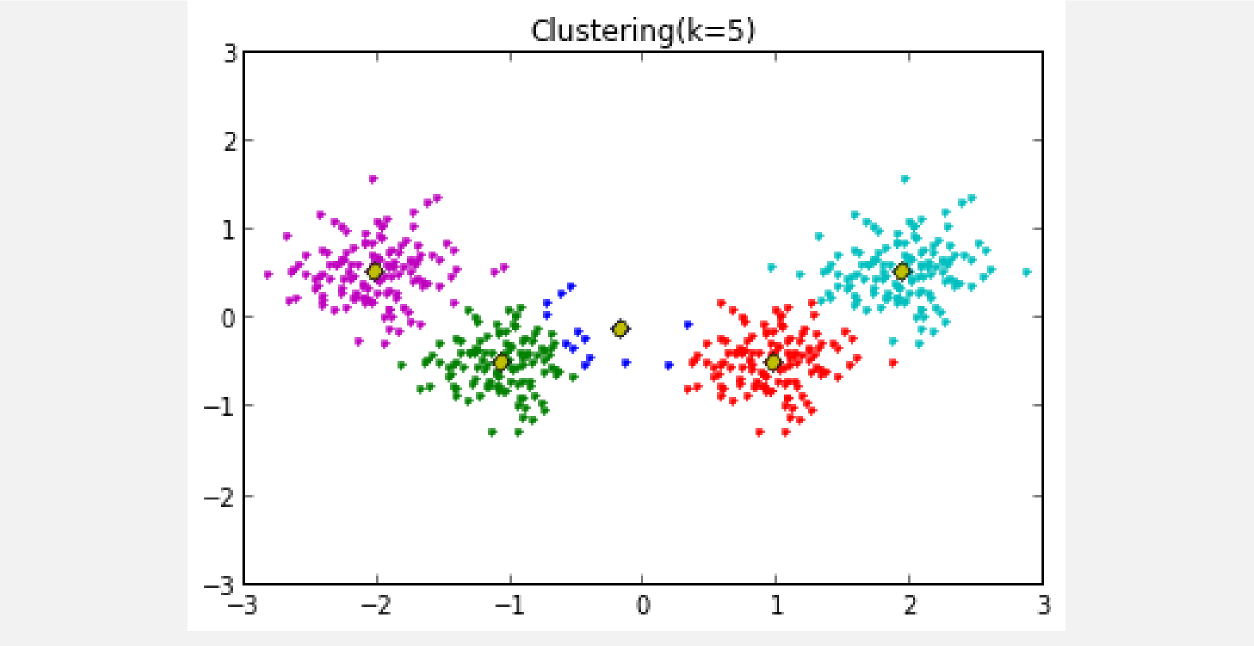
```
iteration 3 done!
```



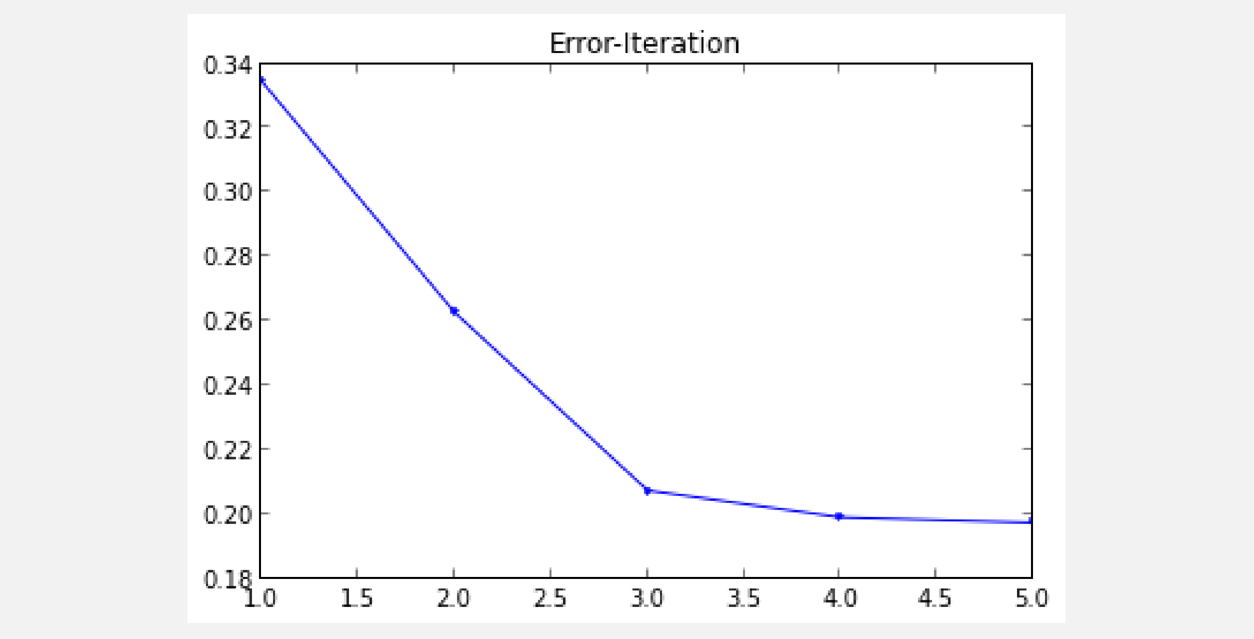
```
iteration 4 done!
```



```
iteration 5 done!
```

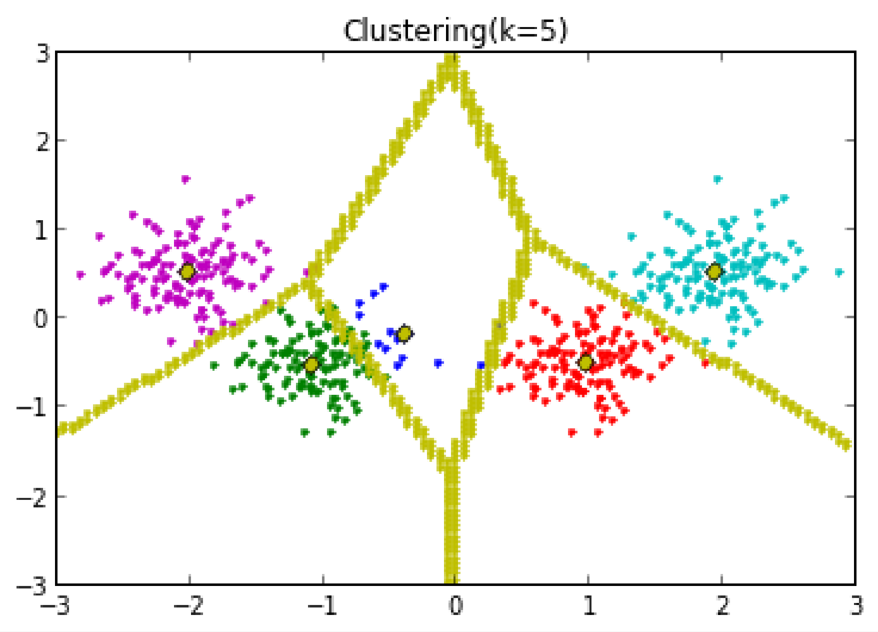


```
plotLine(error,"Error-Iteration")
```



```
colors = ['r','g','b','c','m']
drawBoundary(X,Y,"Clustering(k="+str(k)+")",colors, 3, assign, W, 100)
```

```
unit 0.06
```



2 9.2 Online K-Means Clustering

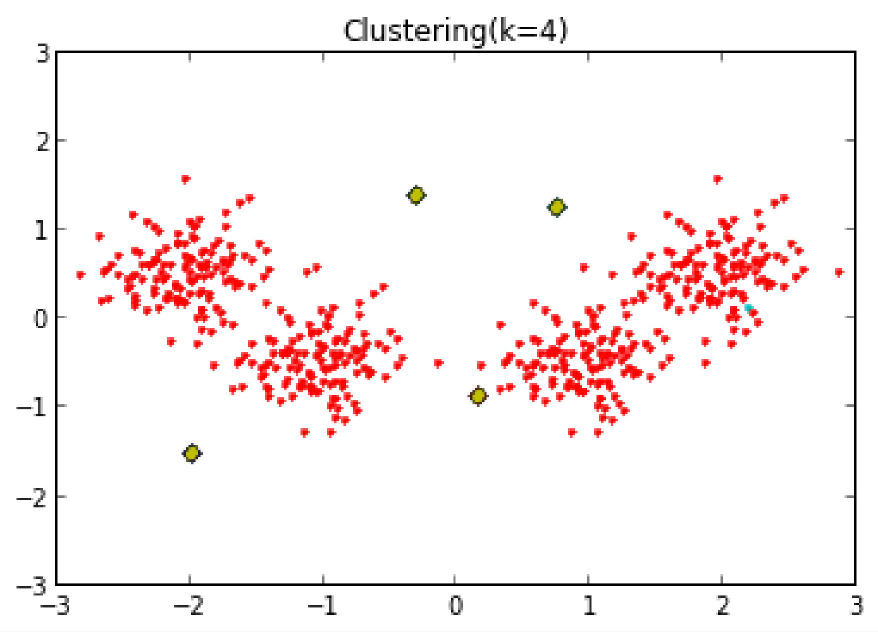
```
#intiliazation
#init W
def init_w_online(k):
    random.seed(100)
    W_init = [[0 for j in range(2)] for i in range(k)]
    for p in range(k):
        W_init[p][0] = random.random() * 4 - 2
        W_init[p][1] = random.random() * 4 -2
    return W_init

t_max = 2* len(X)
k = 4
W_init = init_w_online(k)
print ’W_init’
print matrix(W_init)
eta0 = 0.05
eta = eta0
print ’eta_init:’ , eta0
tao = 0.99
print ’tao_init:’, tao
```

```
W_init
[[ 0.17361977 -0.88652246]
 [-0.30192964  1.37910453]
 [-1.98112458 -1.51372352]
 [ 0.68299634  1.30341102]]
eta_init: 0.05
tao_init: 0.99
```

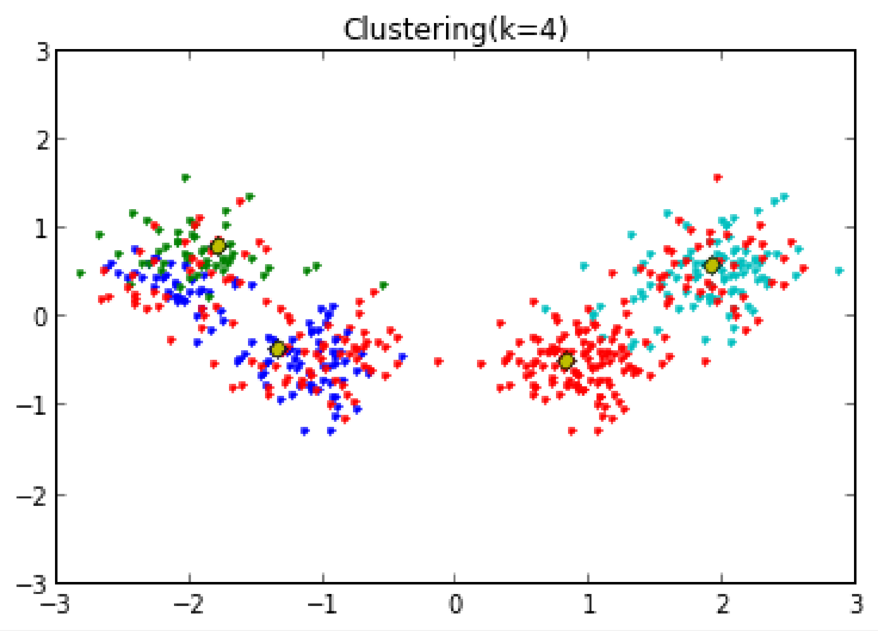
```
#online K_Means
def K_Means_online(X,Y,k,t_max,W_init):
    assign = [0 for i in range(len(X))]
    W = W_init[:][:]
    delta_w = [0,0]
    error = [0 for i in range(t_max)]
    for t in range(t_max):
        n = t%len(X)
        if(t<t_max/4):
            eta = eta0
        else:
            eta = tao * eta
        p = assignPoint(W,X[n],Y[n])
        assign[n] = p
        delta_w[0] = eta*(X[n] - W[p][0])
        delta_w[1] = eta*(Y[n] - W[p][1])
        W[p][0] += delta_w[0]
        W[p][1] += delta_w[1]

        #plot
        #print ’iteration ’ + str(t+1) +’ done!’
        if(t==0 or t==int(float(t_max)/3) or t==int(float(t_max)/3*2) or t==t_max-1):
        #if(:1)
            colors = [’r’,’g’,’b’,’c’,’m’]
            print ’iteration ’ + str(t+1) +’ done!’
            plotCluster(X,Y,"Clustering(k="+str(k)+")", colors, 3, assign, W)

        #claculate the error
        error[t]=0
        for j in range(len(X)):
            error[t] += distance(X[j],Y[j],W[assign[j]][0],W[assign[j]][1])
        error[t] /= 2*len(X)

    return assign,W,error
```

```
assign,W,error = K_Means_online(X,Y,k,t_max,W_init)
```

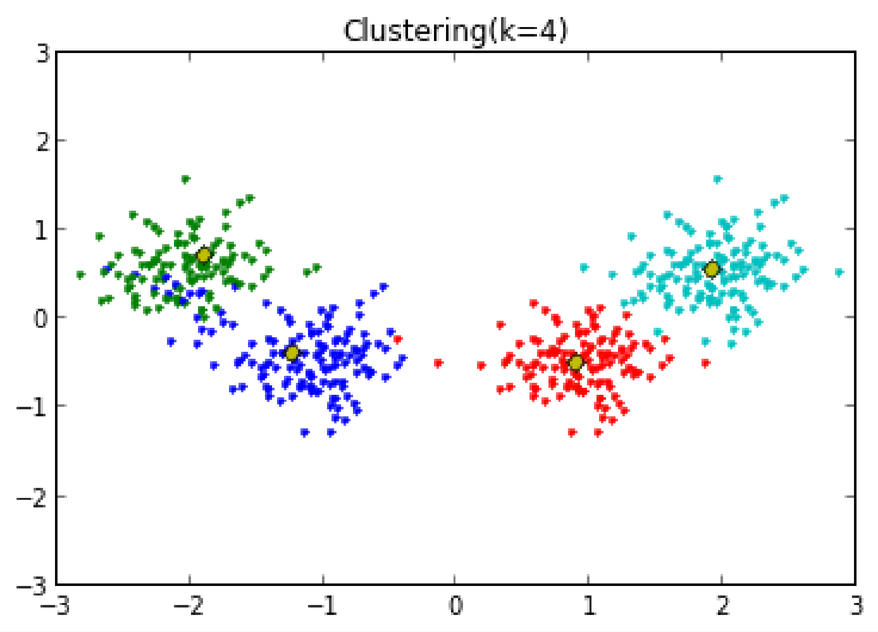
```
iteration 1 done!
```



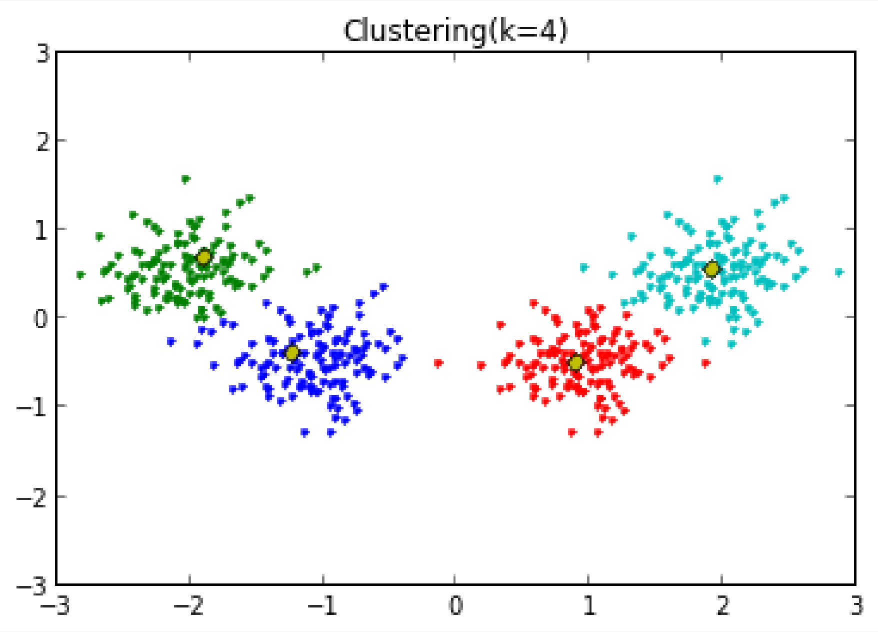
```
iteration 334 done!
```



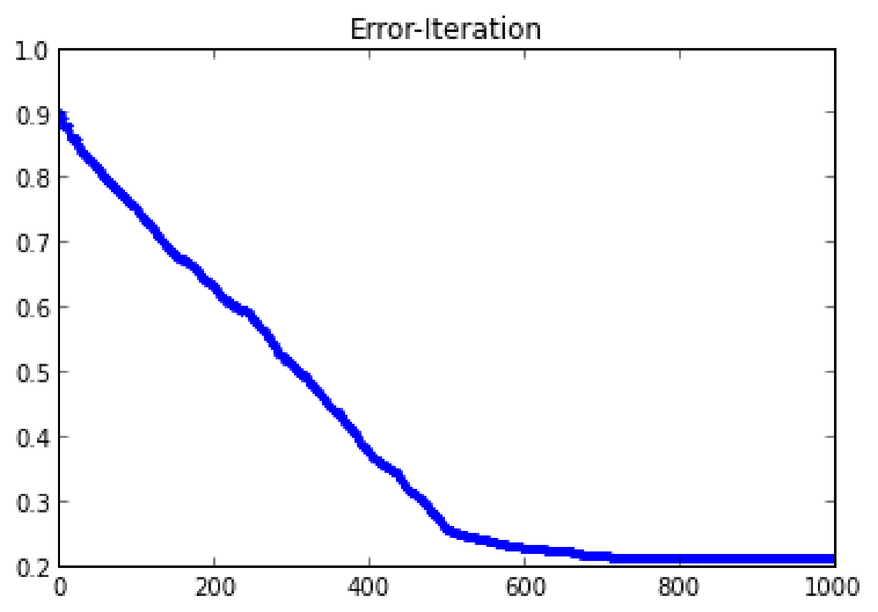
```
iteration 667 done!
```



```
iteration 1000 done!
```

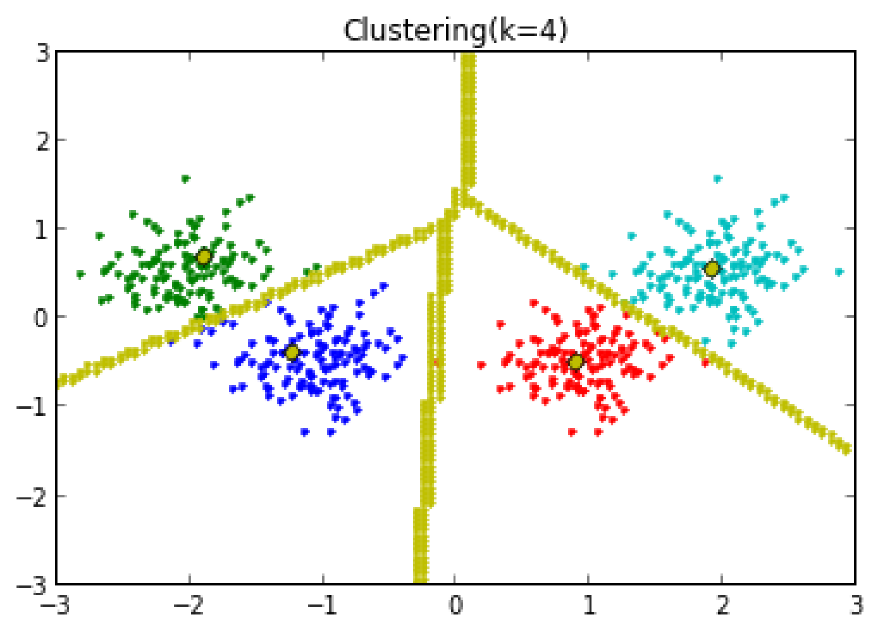


```
plotLine(error,"Error-Iteration")
```



```
colors = [’r’,’g’,’b’,’c’,’m’]
drawBoundary(X,Y,"Clustering(k="+str(k)+")",colors, 3, assign, W, 100)
```

```
unit 0.06
```


Clustering(k=4)