Data Mining
Home work 11
ML, Clustering, projects...

Aqeel Labash
Lecturer: Jaak Vilo

21 April 2016

First Question

For this question I simply created a new point depending on the mean value of all Xs, Ys and then calculated the distance between all the points and that point as the Z value. Here is the code I used :

```
import pandas as pd
import numpy as np
get_ipython().magic(u'matplotlib inline')
import matplotlib.pyplot as plt
from IPython.display import display, HTML
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

data =pd.read_csv('Linear.csv',sep=',')
plt.plot(data[data.Class==1].X, data[data.Class==1].Y, 'ro')
plt.plot(data[data.Class==0].X, data[data.Class==0].Y, 'bo')
(mX,mY) = (np.mean(data.X),np.mean(data.Y))
plt.plot(mX,mY,'go')
plt.axis([min(data.Y)-1, max(data.X)+1, min(data.Y)-1, max(data.Y)+1])
plt.show()

# #The three levels is :
# - X
# - Y
# - Z :distance from the mean point
lst =[]
for i in range(len(data.X)):
    lst.append(np.linalg.norm(np.array((data.X[i],data.Y[i]))-np.array((mX,mY))))

data['Z'] = pd.Series(np.array(lst), index=data.index)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data[data.Class==1].X, data[data.Class==1].X, data[data.Class==1].Z, c='r', marker=
    'o')
ax.scatter(data[data.Class==0].X, data[data.Class==0].X, data[data.Class==0].Z, c='b', marker=
    'o')

xx, yy = np.meshgrid(range(6), range(6))
z1 = np.reshape(np.repeat(1.65,36),(6,6))

ax.plot_surface(xx,yy,z1, color='blue',                rstride=3,
                cstride=3,
                alpha=0.3,            # transparency of the surface
                cmap=cm.coolwarm)

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

plt.show()
```

The previous code output The following figures :

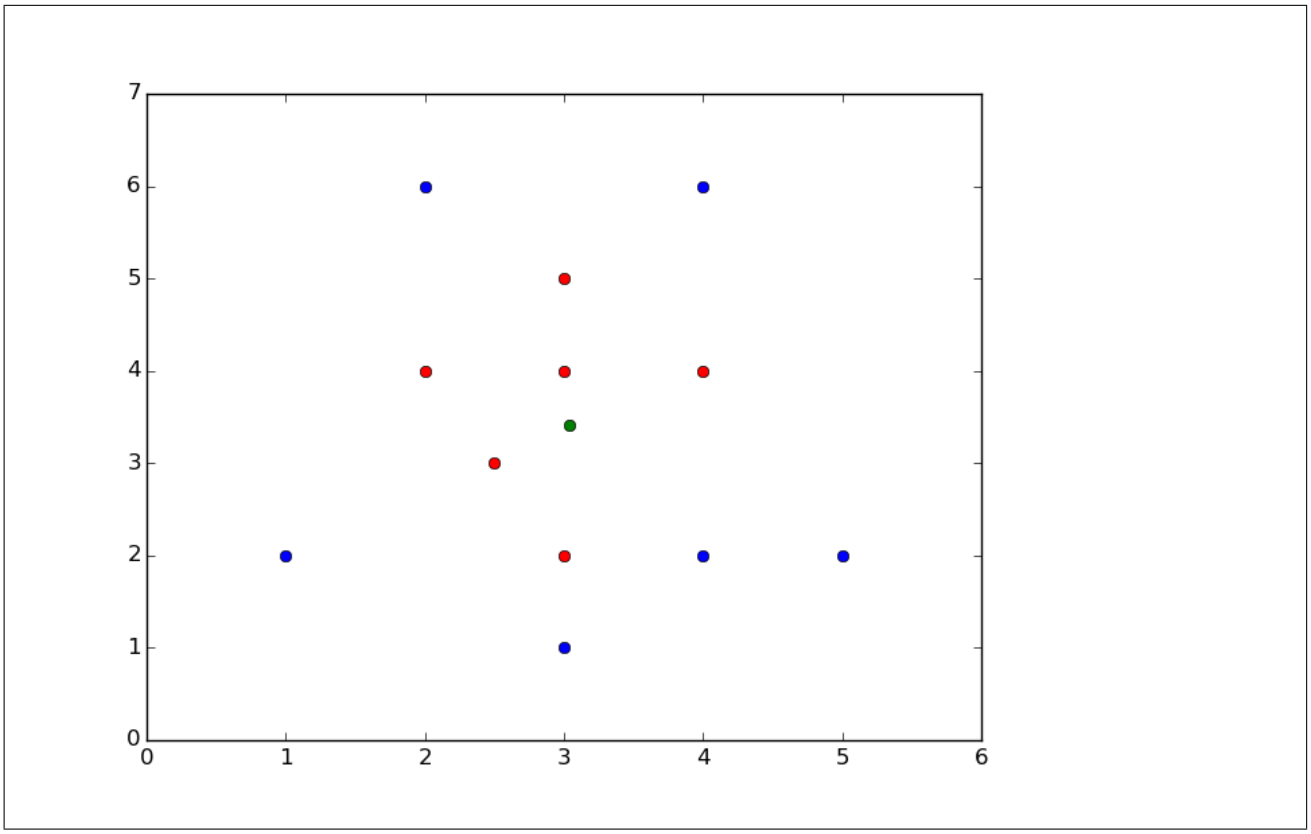

Fig. 1: Show the points in the plane with the mean point

In the previous figure we can see the green point which represent the mean point between all the points.

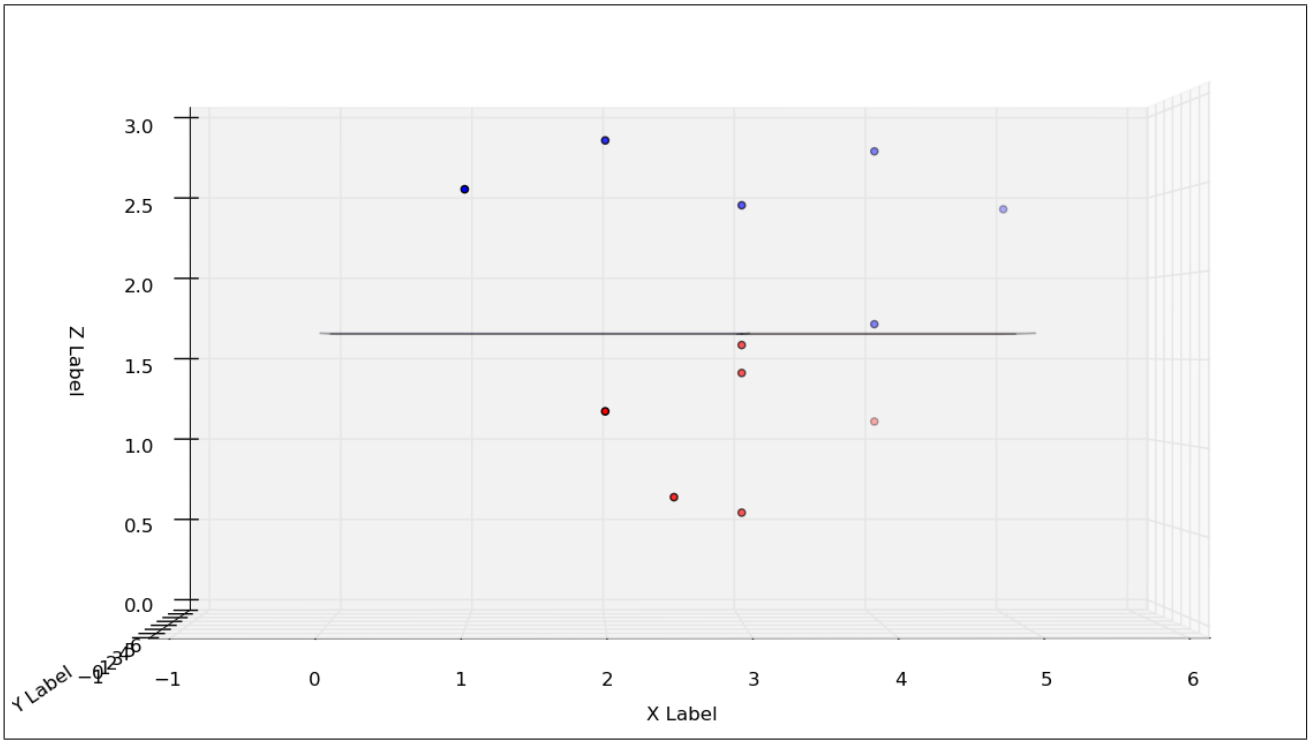

Fig. 2: The hyper plan in 2d prospective

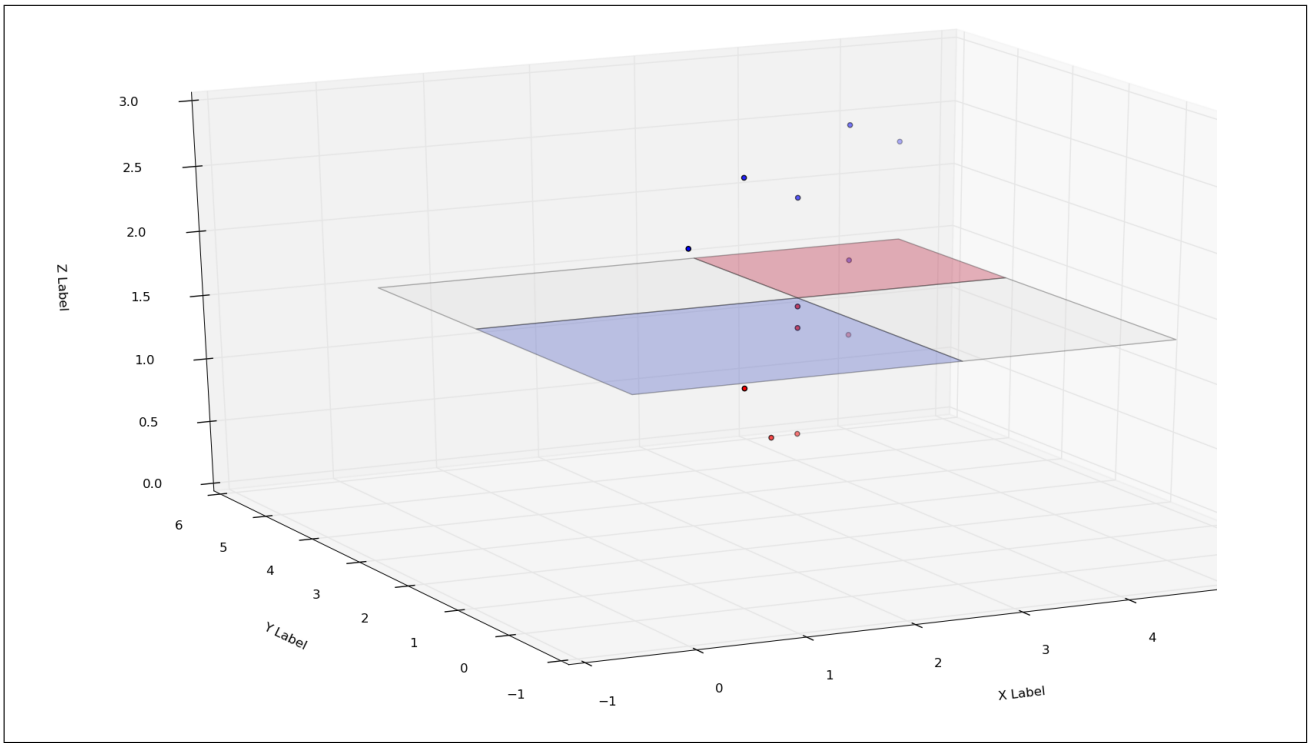

Fig. 3: The hyper plane in 3d prospective

Please notice that we can add angle to the hyper plane which would give a bigger margin even.

Second Question

For this question I used python code to calculate the distances and make things easier (Just to print the distance list and draw a picture of the points). Here is the code :

```
import pandas as pd
import numpy as np
get_ipython().magic(u'matplotlib inline')
import matplotlib.pyplot as plt
from IPython.display import display, HTML
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from operator import attrgetter

points = pd.read_csv('Clustering.csv',sep=',')

plt.plot(points.X, points.Y, 'ro')
plt.axis([min(points.X)-1, max(points.X)+1, min(points.Y)-1, max(points.Y)+1])
for i, txt in enumerate(points.index):
    plt.annotate(txt+1, (points.X[i],points.Y[i]))
plt.show()

class Dist:
    def __init__(self,p1indx,p2indx,dist):
        self.p1 = p1indx
        self.p2 = p2indx
        self.dist=dist
distances={}
distancelst = []
for i in range(len(points.index)):
    for j in range(len(points.index)):
        if i==j:
            continue
        if (i,j) not in distances.keys():
            dist = np.linalg.norm(np.array([points.X[i],points.Y[i]])-np.array([points.X[j],
    points.Y[j]]))
            distances[(i,j)]=dist
            distances[(j,i)]=dist
            distancelst.append(Dist(i,j,dist))

distancelst.sort(key=lambda x: x.dist, reverse=False)
```

```
36
37 for i in distancelst:
38     print i.dist,i.p1+1,i.p2+1
```

In the previous code I just calculate the distance between all the points and order them min to max so I can pick what points to connect first.Here is the outcome from the previous code.

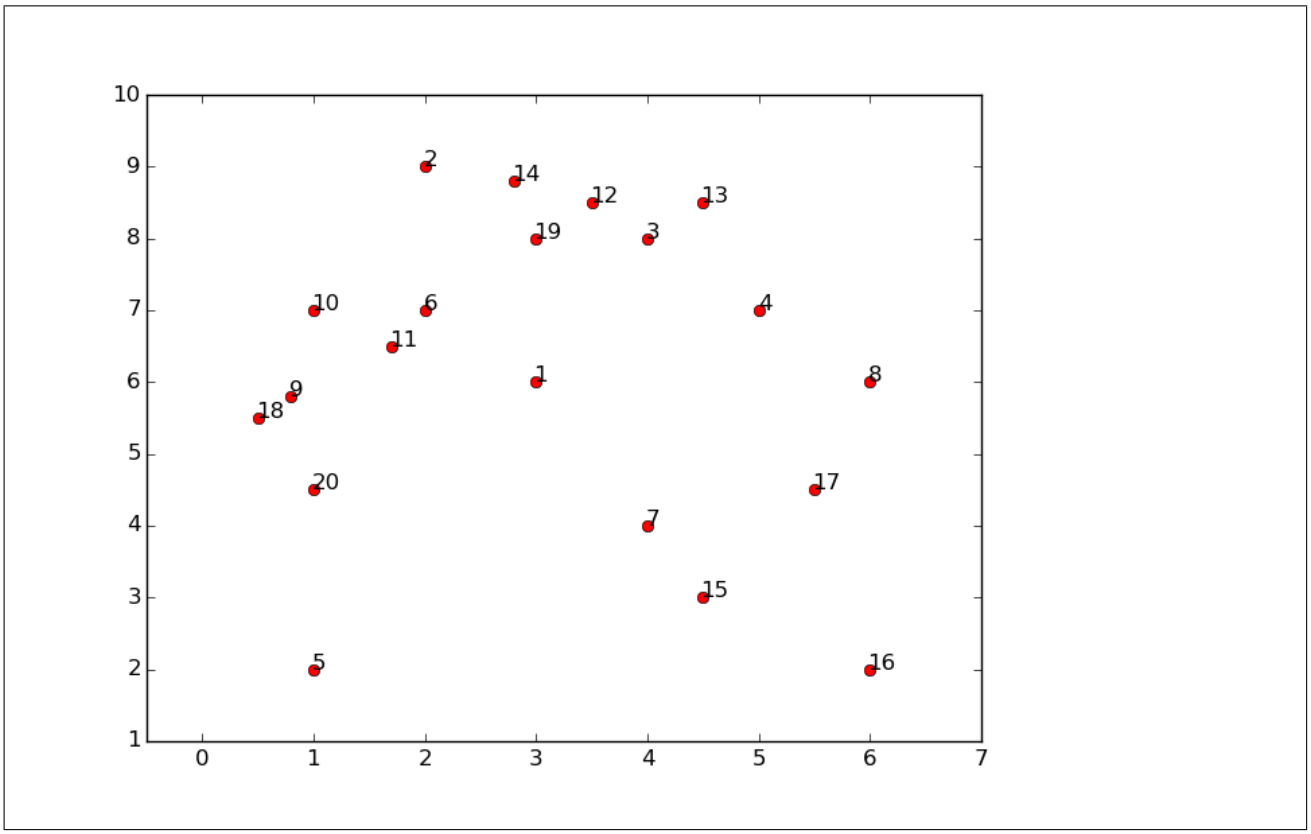

Fig. 4: Points in the plane numbered

The previous code as I said prints the list of ordered points by distance here is the table of it :

Distance	Point 1	Point 2	Notes
0.424264068712	9	18	C1
0.583095189485	6	11	C2
0.707106781187	3	12	C3
0.707106781187	3	13	C3 Extended
0.707106781187	12	19	C3 Extended
0.761577310586	12	14	C3 Extended
0.824621125124	2	14	C3 Extended
0.860232526704	10	11	C2 Extended
1.11803398875	7	15	C4
1.11803398875	18	20	C1 Extended
1.1401754251	9	11	C1&C2 merged→ C1
1.39283882772	1	11	C1 Extended
1.41421356237	3	4	C3 Extended
1.41421356237	4	8	C3 Extended
1.41421356237	6	19	C1&C3 merged→ C1
1.58113883008	7	17	C4 Extended
1.58113883008	8	17	C1&C4 merged→ C1
1.80277563773	15	16	C1 Extended
2.5	5	20	C1 Extended

From the previous table I could see better the shortest distance.And here is what I got on papers:

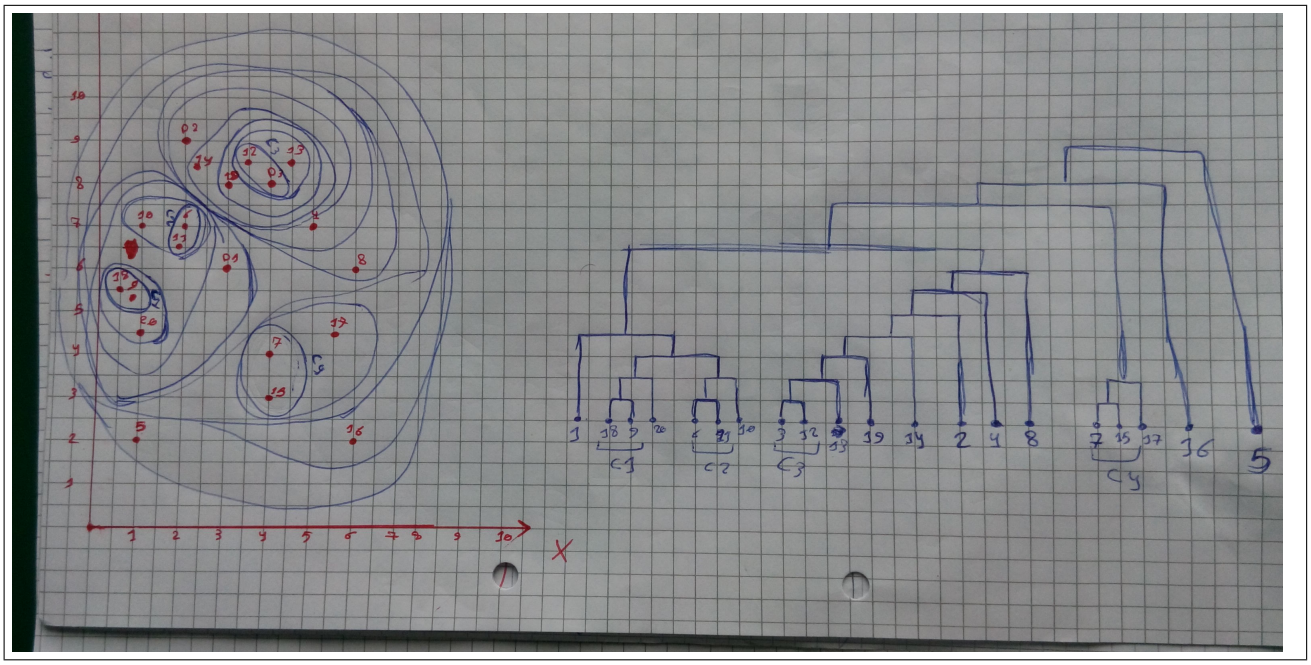

Fig. 5: The clusters build

Please notice that the previous plot may not be accurate due to many points (I might mixed some points) but the tree is correct I believe because I depended on automated calculation for it.I tried to automate the drawing but didn't continue it not to take more time.

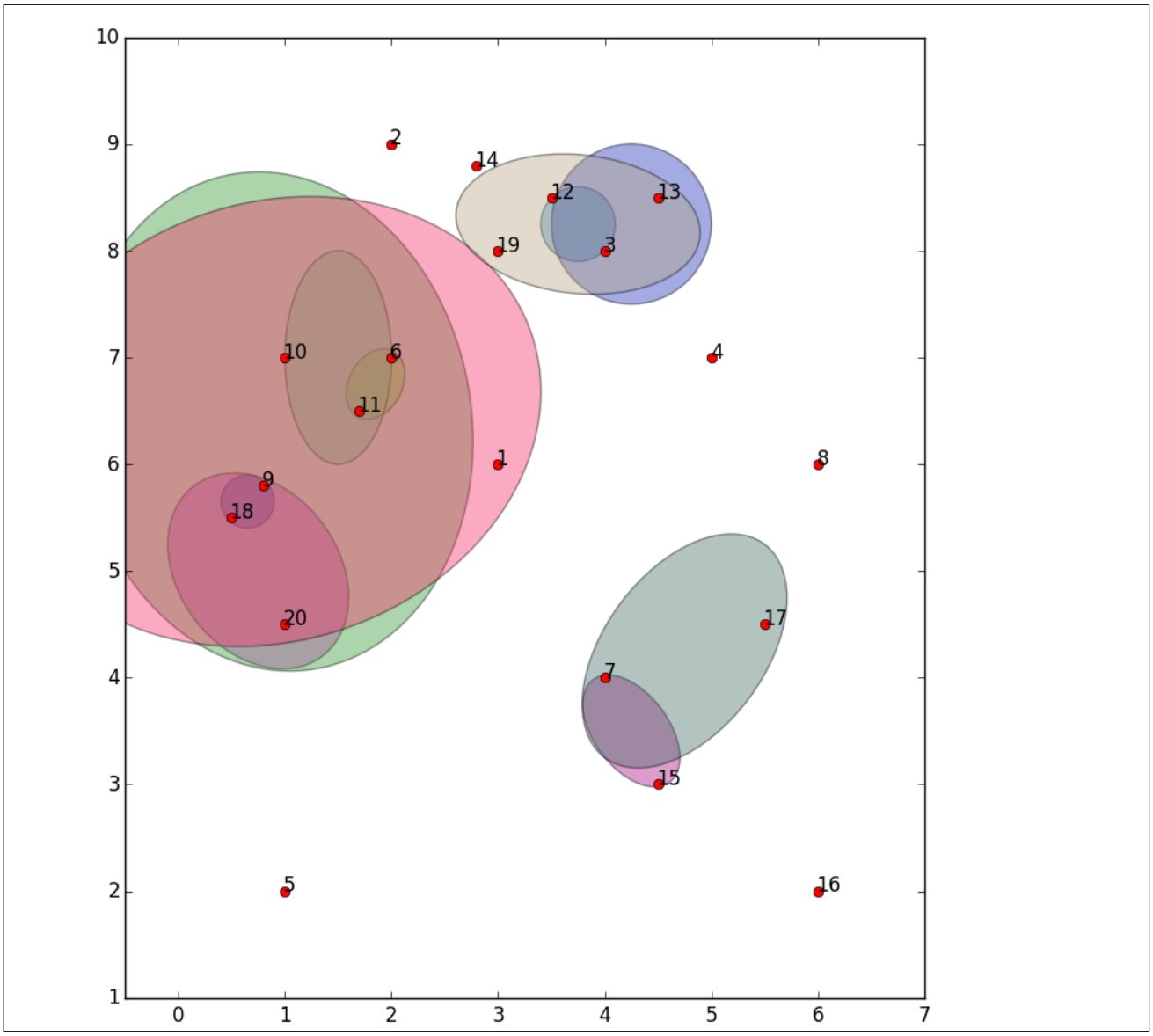

Fig. 6: Show some groups

Now for the **Complete link**, the first stage is the same when we create join the first clusters (at the beginning each point is a cluster).And here is the distance table (New one):

Distance	Point 1	Point 2	Notes
0.424264068712	9	18	C1
0.583095189485	6	11	C2
0.707106781187	3	12	C3
0.824621125124	2	14	C4
1.0	3	19	C3 Extended
1.0	6	10	C2 Extended
1.11803398875	7	15	C5
1.3152946438	9	20	C1 Extended
1.41421356237	4	8	C6
1.58113883008	13	19	C3 Extended
1.80277563773	15	17	C5 Extended
2.2360679775	1	10	C2 Extended
2.2360679775	2	3	C4&C3 merged →C3
2.69258240357	6	20	C1&C2 merged →C1
2.82842712475	7	16	C5 Extended
5.0	2	8	C3&C6 merged →C3
5.09901951359	5	6	C1 Expanded
7.07106781187	10	16	C1&C5 merged →C1
8.0622577483	2	16	C1&C3 merged →C1

And here is the images:

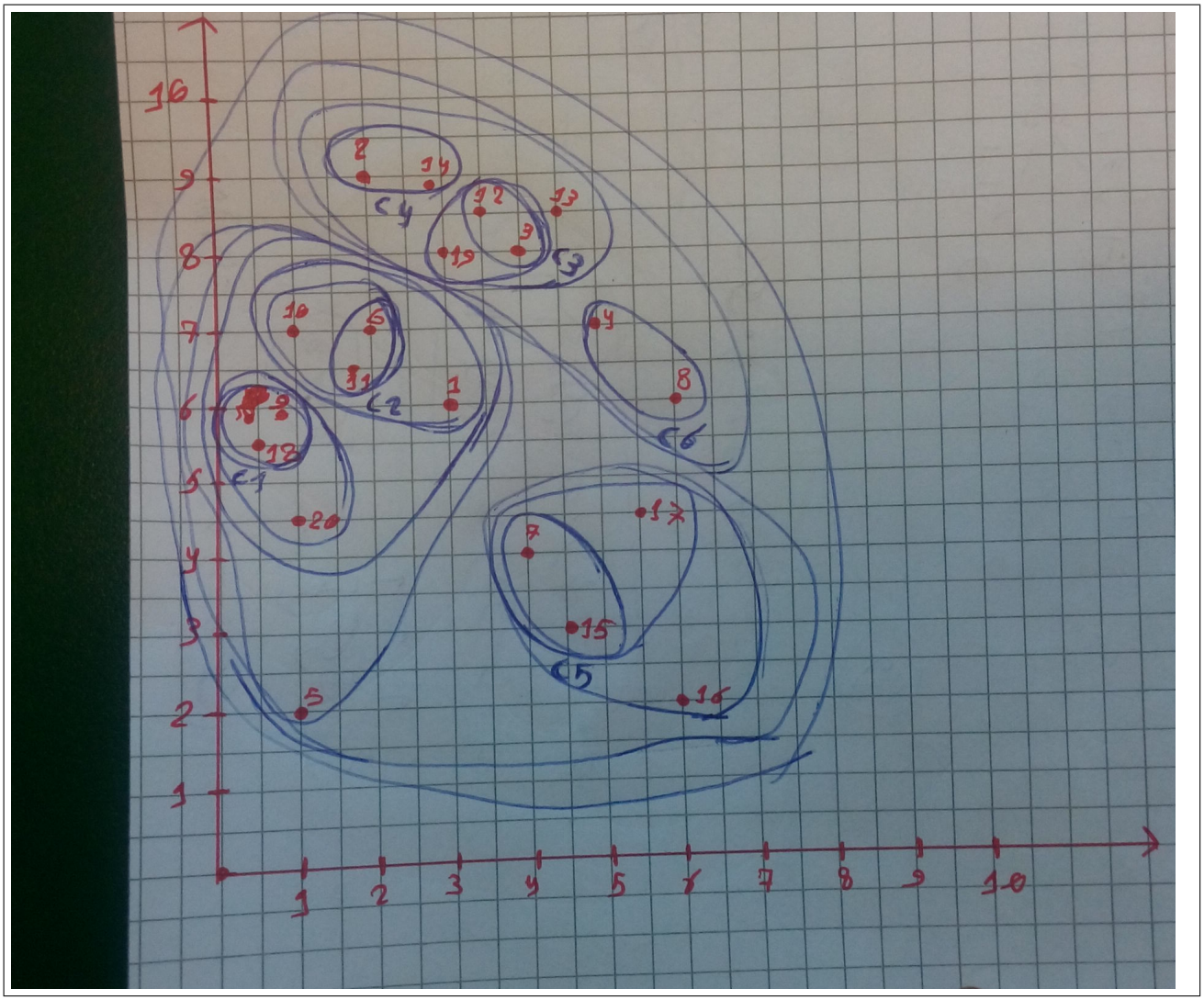

Fig. 7: Joining points map

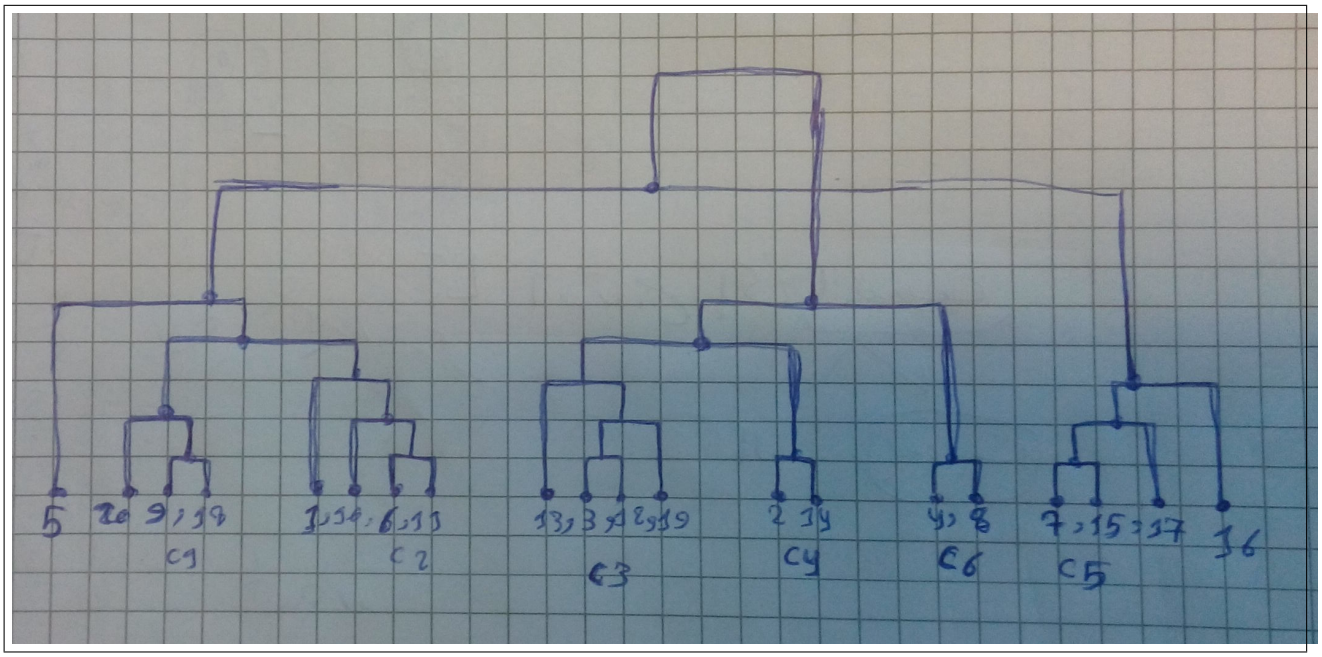

Fig. 8: The tree

Third Question

For this question I wrote a python code,

1. First calculate the distance between the four centers and assign the point to the closest center.
2. Plot the data
3. Calculate the new centers
4. Repeat

Here it is :

```
import pandas as pd
import numpy as np
#%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import display, HTML
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from operator import attrgetter
from matplotlib.patches import Ellipse
from math import atan2,degrees
import numpy.random as rnd

points = pd.read_csv('Clustering.csv',sep=',')

class CenterValue:
    def __init__ (self,p,val,lab):
        self.value=val
        self.point=p
        self.label=lab
    def __str__(self):
        return 'point:{},Value:{},index:{}'.format(self.point,self.value,self.label)

def Distance(p1,p2indx):
    p1 = np.array(p1)
    p2 = np.array((points.X[p2indx],points.Y[p2indx]))
    return np.linalg.norm(p1-p2)

def K_mean(centerpoints):
    centers={}
    for i in centerpoints:
        centers[i]=[]
    for i in range(len(points.index)):
        values=[]
```

```
        values.append(CenterValue(centerpoints[0],Distance(centerpoints[0],i),i))
        values.append(CenterValue(centerpoints[1],Distance(centerpoints[1],i),i))
        values.append(CenterValue(centerpoints[2],Distance(centerpoints[2],i),i))
        values.append(CenterValue(centerpoints[3],Distance(centerpoints[3],i),i))
        values.sort(key=lambda x: x.value, reverse=False)
        centers[values[0].point].append(GetPointsList([i])[0])
    return centers

def GetPointsList(Indexs):
    lst=[]
    for i in Indexs:
        lst.append((points.X[i],points.Y[i]))
    return lst

def Get_Means(dictionary_of_lists):
    lst=[]
    for i in dictionary_of_lists.keys():
        mX=0
        mY=0
        for p in dictionary_of_lists[i]:
            mX+=p[0]
            mY+=p[1]
        lst.append((mX/len(dictionary_of_lists[i]),mY/len(dictionary_of_lists[i])))
    return lst

def Plot(k):
    keys = k.keys()
    fig = plt.figure(0)
    ax = fig.add_subplot(111, aspect='equal')
    ax.plot([t[0] for t in k[keys[0]]], [t[1] for t in k[keys[0]]], 'rs')
    ax.plot([t[0] for t in k[keys[1]]], [t[1] for t in k[keys[1]]], 'bs')
    ax.plot([t[0] for t in k[keys[2]]], [t[1] for t in k[keys[2]]], 'gs')
    ax.plot([t[0] for t in k[keys[3]]], [t[1] for t in k[keys[3]]], 'ms')
    for k in keys:
        ax.plot(k[0],k[1],'k*')
        ax.annotate('Cntr',(k[0],k[1]))
    ax.axis([0, 10, 0,10])
    plt.show()

#First Iteration
k = K_mean(GetPointsList(range(4)))
Plot(k)

#Each Time new Iteration
k = Get_Means(k)
k = K_mean(k)
Plot(k)
```

The previous code output the following :

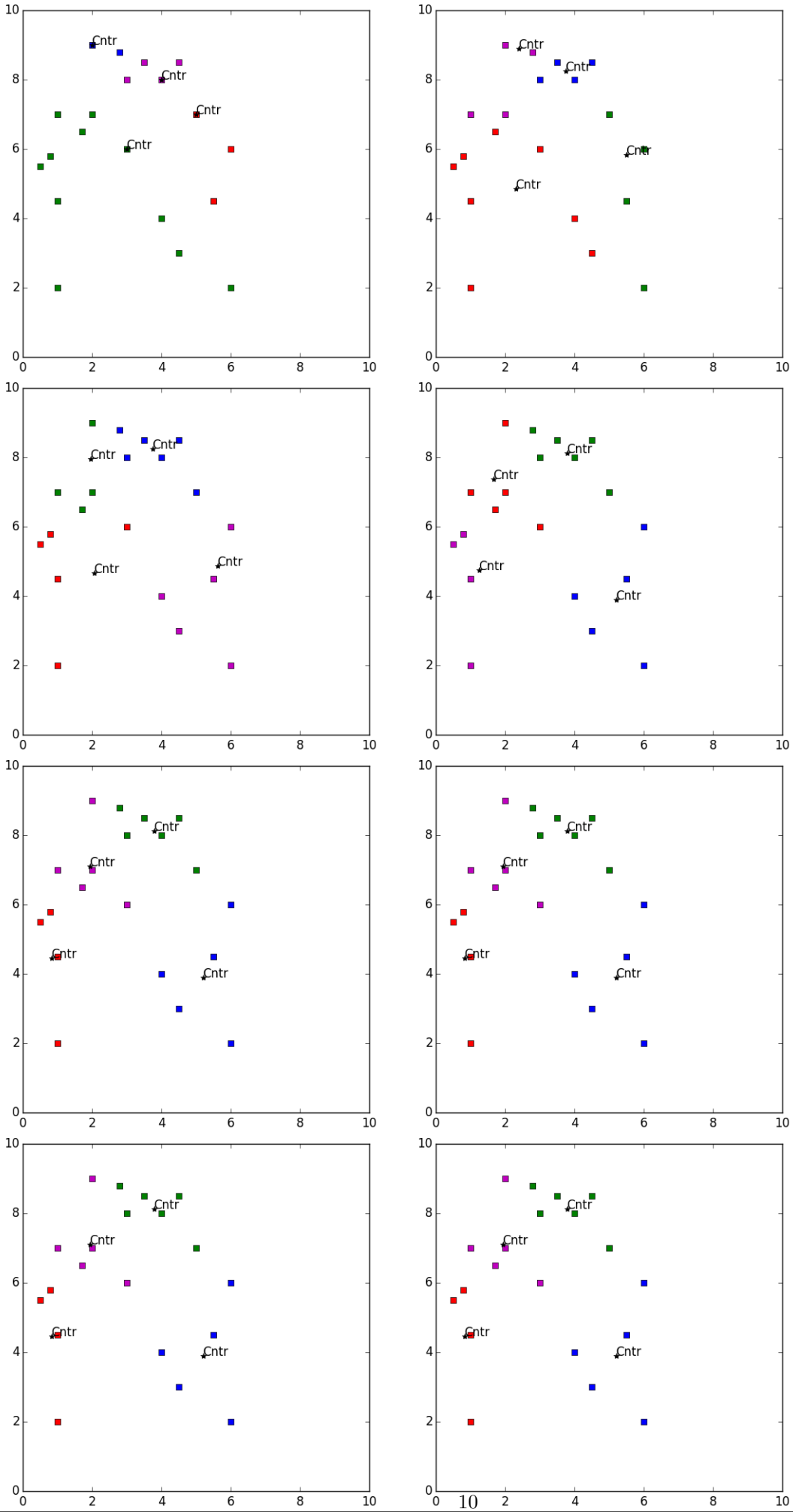

Fig. 9: Shows how centers change counting(1,2),(3,4)

In the previous figures we can see that after the 5th iteration the centers got stable

Fourth Question

The Slide already added.

Fifth Question

Analysis
Note: All code,pythhon ,tex,pdf,etc... files exist in github

E.O.F