

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 X s, Y s and then calculated the distance between all the points and that point as the Z value. Here is the code I used :

```
1 import pandas as pd
2 import numpy as np
3 get_ipython().magic(u'matplotlib inline')
4 import matplotlib.pyplot as plt
5 from IPython.display import display, HTML
6 from mpl_toolkits.mplot3d import Axes3D
7 from matplotlib import cm
8
9 data = pd.read_csv('Linear.csv', sep=',')
10 plt.plot(data[data.Class==1].X, data[data.Class==1].Y, 'ro')
11 plt.plot(data[data.Class==0].X, data[data.Class==0].Y, 'bo')
12 (mX, mY) = (np.mean(data.X), np.mean(data.Y))
13 plt.plot(mX, mY, 'go')
14 plt.axis([min(data.Y)-1, max(data.X)+1, min(data.Y)-1, max(data.Y)+1])
15 plt.show()
16
17 # The three levels is :
18 # - X
19 # - Y
20 # - Z : distance from the mean point
21 lst = []
22 for i in range(len(data.X)):
23     lst.append(np.linalg.norm(np.array((data.X[i], data.Y[i])) - np.array((mX, mY))))
24
25 data['Z'] = pd.Series(np.array(lst), index=data.index)
26
27 fig = plt.figure()
28 ax = fig.add_subplot(111, projection='3d')
29 ax.scatter(data[data.Class==1].X, data[data.Class==1].X, data[data.Class==1].Z, c='r', marker='o')
30 ax.scatter(data[data.Class==0].X, data[data.Class==0].X, data[data.Class==0].Z, c='b', marker='o')
31
32 xx, yy = np.meshgrid(range(6), range(6))
33 z1 = np.reshape(np.repeat(1.65, 36), (6, 6))
34
35 ax.plot_surface(xx, yy, z1, color='blue', rstride=3,
36                cstride=3,
37                alpha=0.3, # transparency of the surface
38                cmap=cm.coolwarm)
39
40 ax.set_xlabel('X Label')
41 ax.set_ylabel('Y Label')
42 ax.set_zlabel('Z Label')
43
44 plt.show()
```

The previous code output The following figures :

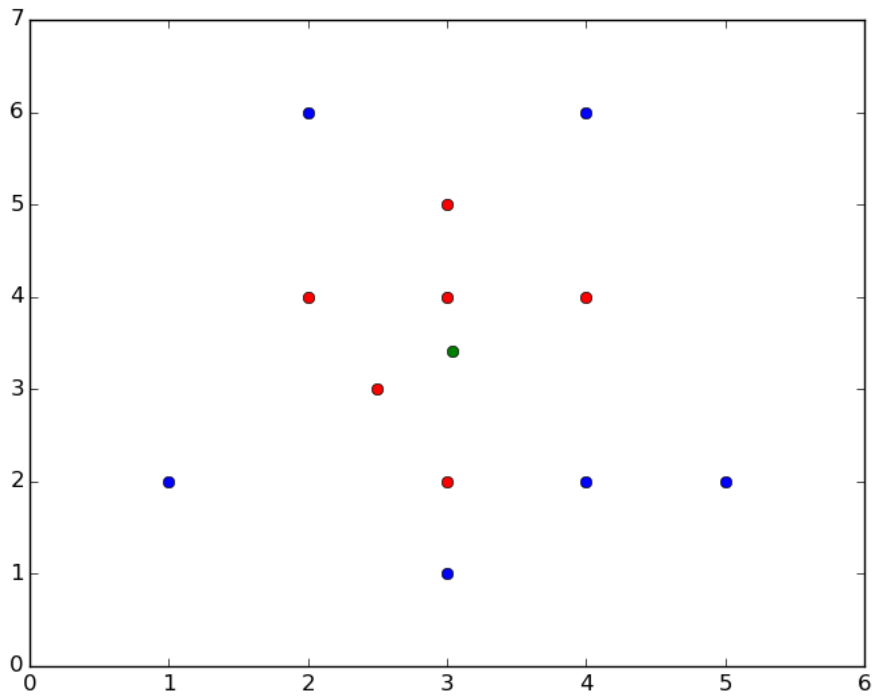


Fig. 1: Show the points in the plane with the mean point

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

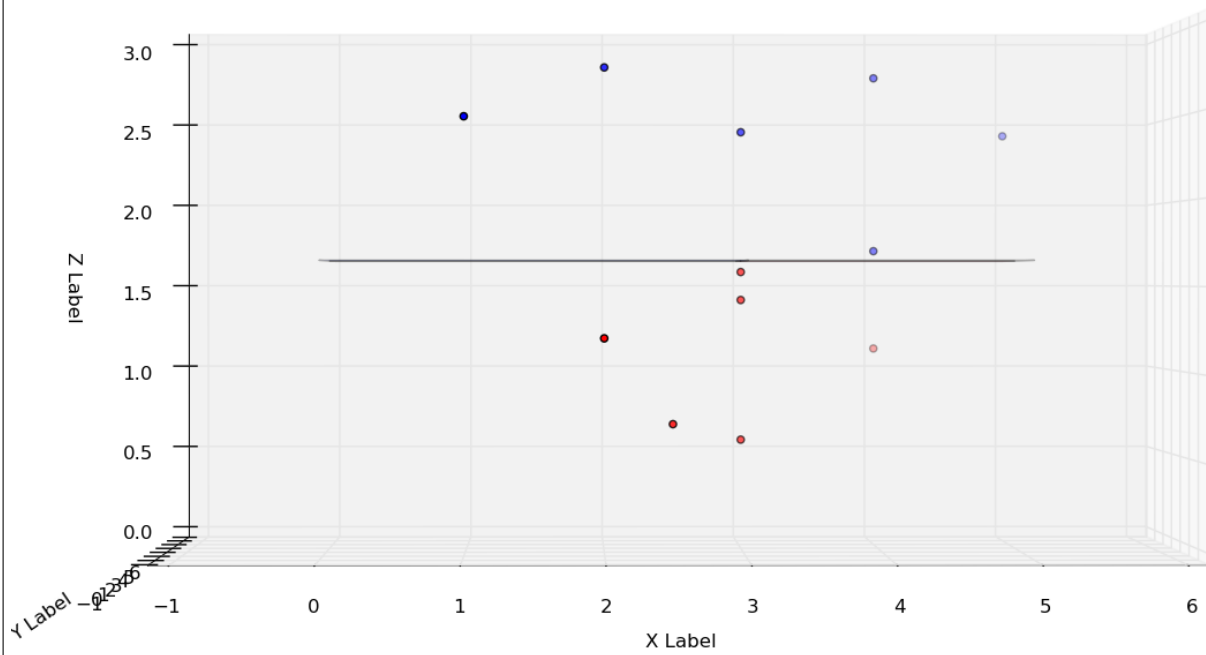


Fig. 2: The hyper plan in 2d prospective

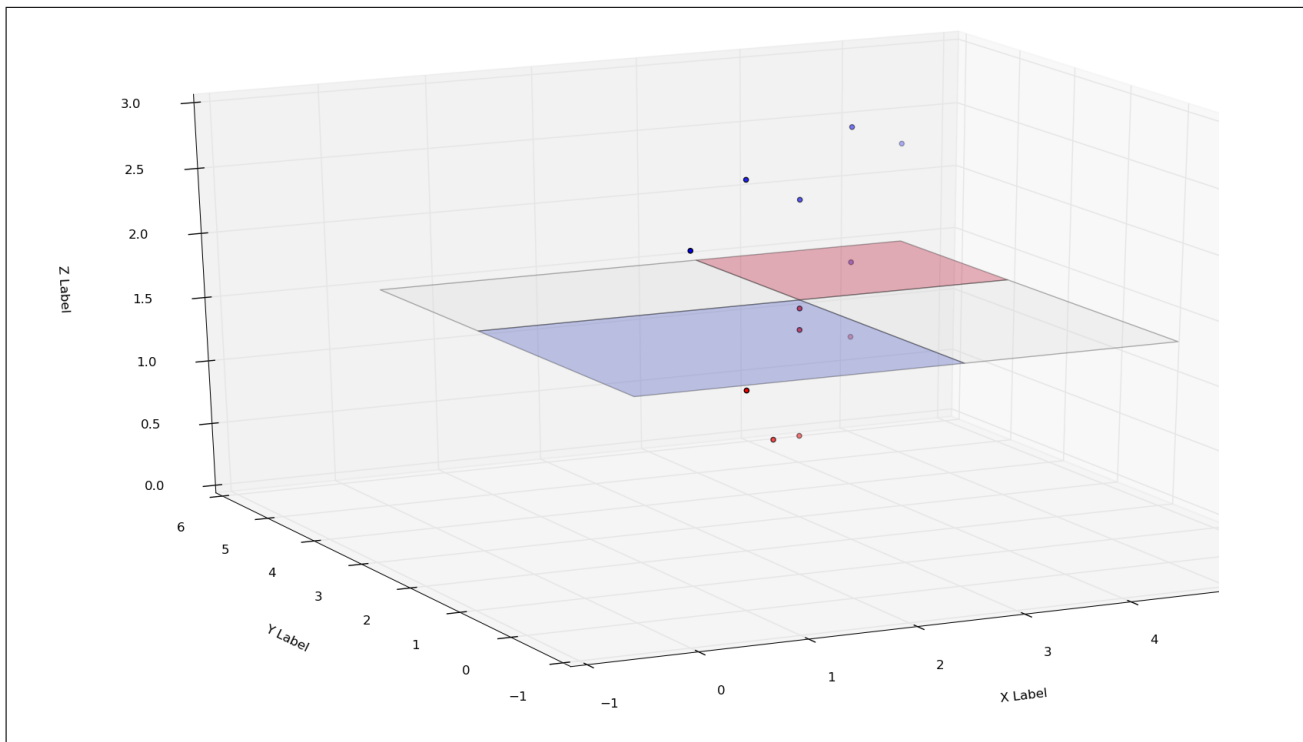


Fig. 3: The hyper plane in 3d prospective

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 :

```
1 import pandas as pd
2 import numpy as np
3 get_ipython().magic(u'matplotlib inline')
4 import matplotlib.pyplot as plt
5 from IPython.display import display, HTML
6 from mpl_toolkits.mplot3d import Axes3D
7 from matplotlib import cm
8 from operator import attrgetter
9
10 points = pd.read_csv('Clustering.csv', sep=',')
11
12 plt.plot(points.X, points.Y, 'ro')
13 plt.axis([min(points.X)-1, max(points.X)+1, min(points.Y)-1, max(points.Y)+1])
14 for i, txt in enumerate(points.index):
15     plt.annotate(txt+1, (points.X[i], points.Y[i]))
16 plt.show()
17
18 class Dist:
19     def __init__(self, p1indx, p2indx, dist):
20         self.p1 = p1indx
21         self.p2 = p2indx
22         self.dist = dist
23
24 distances = {}
25 distancelst = []
26 for i in range(len(points.index)):
27     for j in range(len(points.index)):
28         if i == j:
29             continue
30         if (i, j) not in distances.keys():
31             dist = np.linalg.norm(np.array([points.X[i], points.Y[i]]) - np.array([points.X[j],
32             points.Y[j]]))
33             distances[(i, j)] = dist
34             distances[(j, i)] = dist
35             distancelst.append(Dist(i, j, dist))
36
37 distancelst.sort(key=lambda x: x.dist, reverse=False)
```

36
37
38

```
for i in distancelst:
    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.

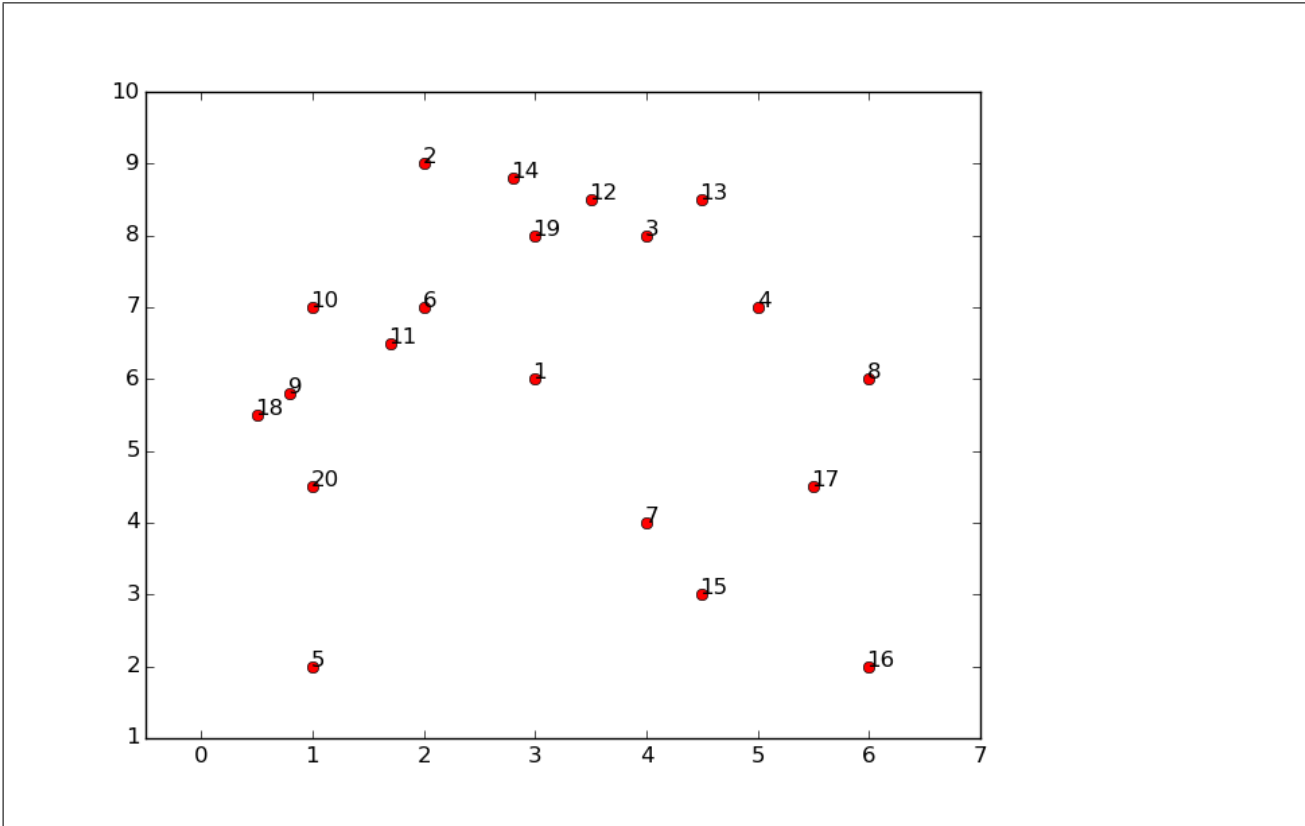


Fig. 4: Points in the plane numbered

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:

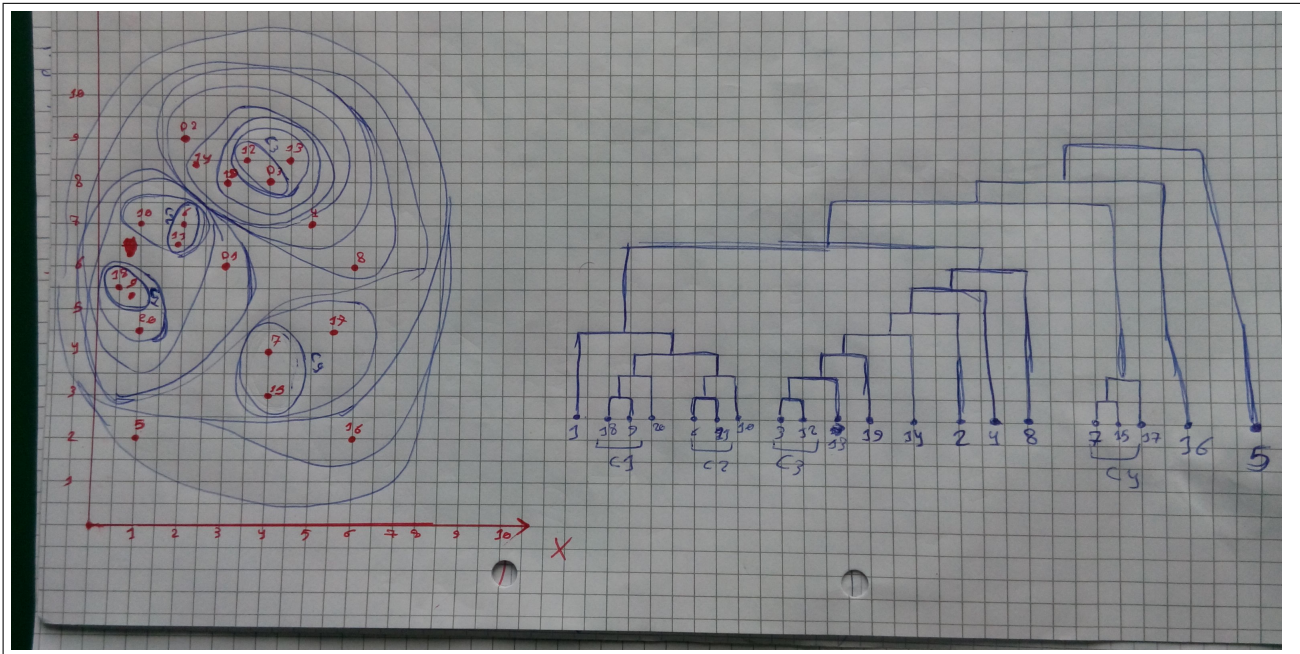


Fig. 5: The clusters build

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.

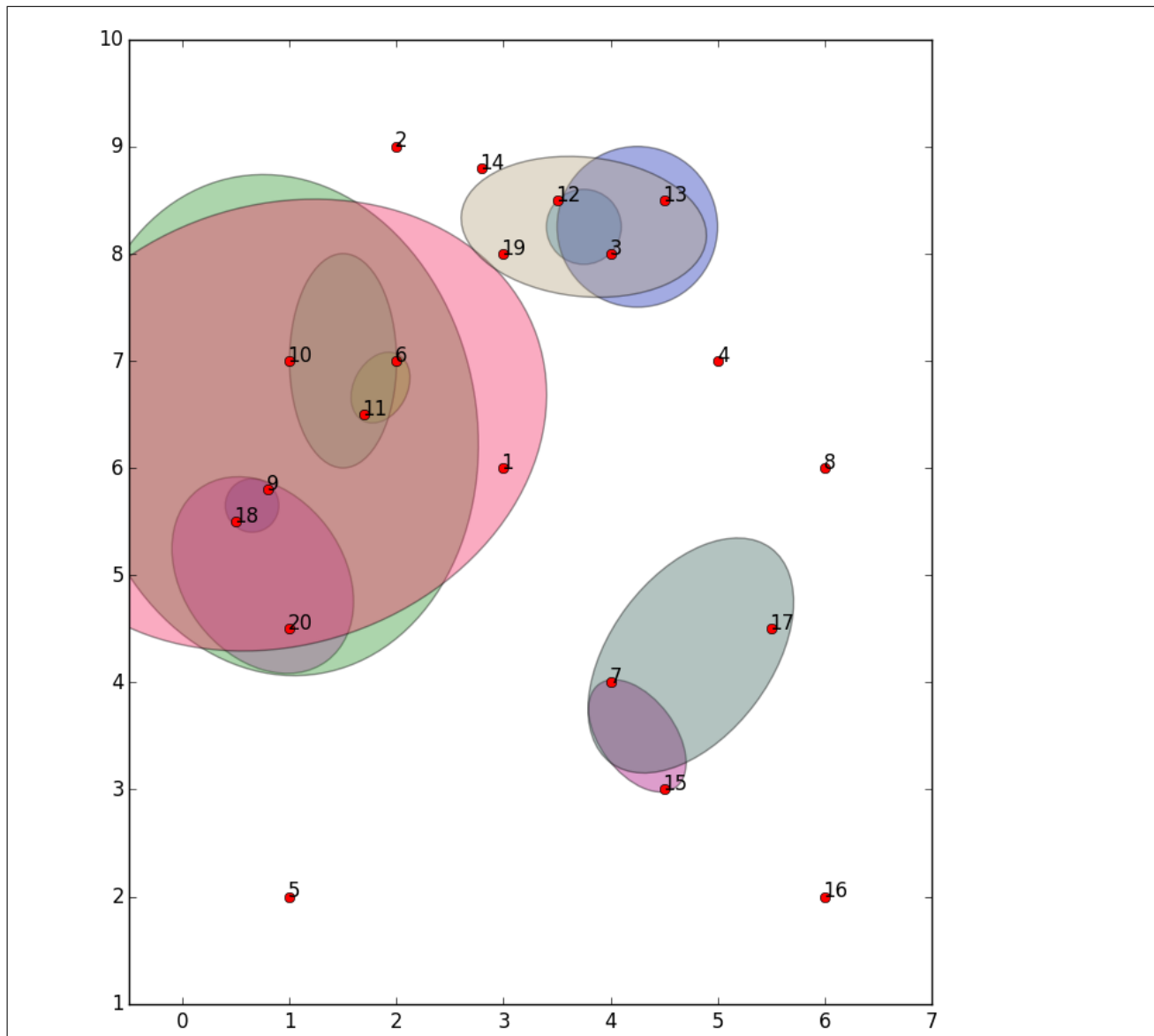


Fig. 6: Show some groups

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:

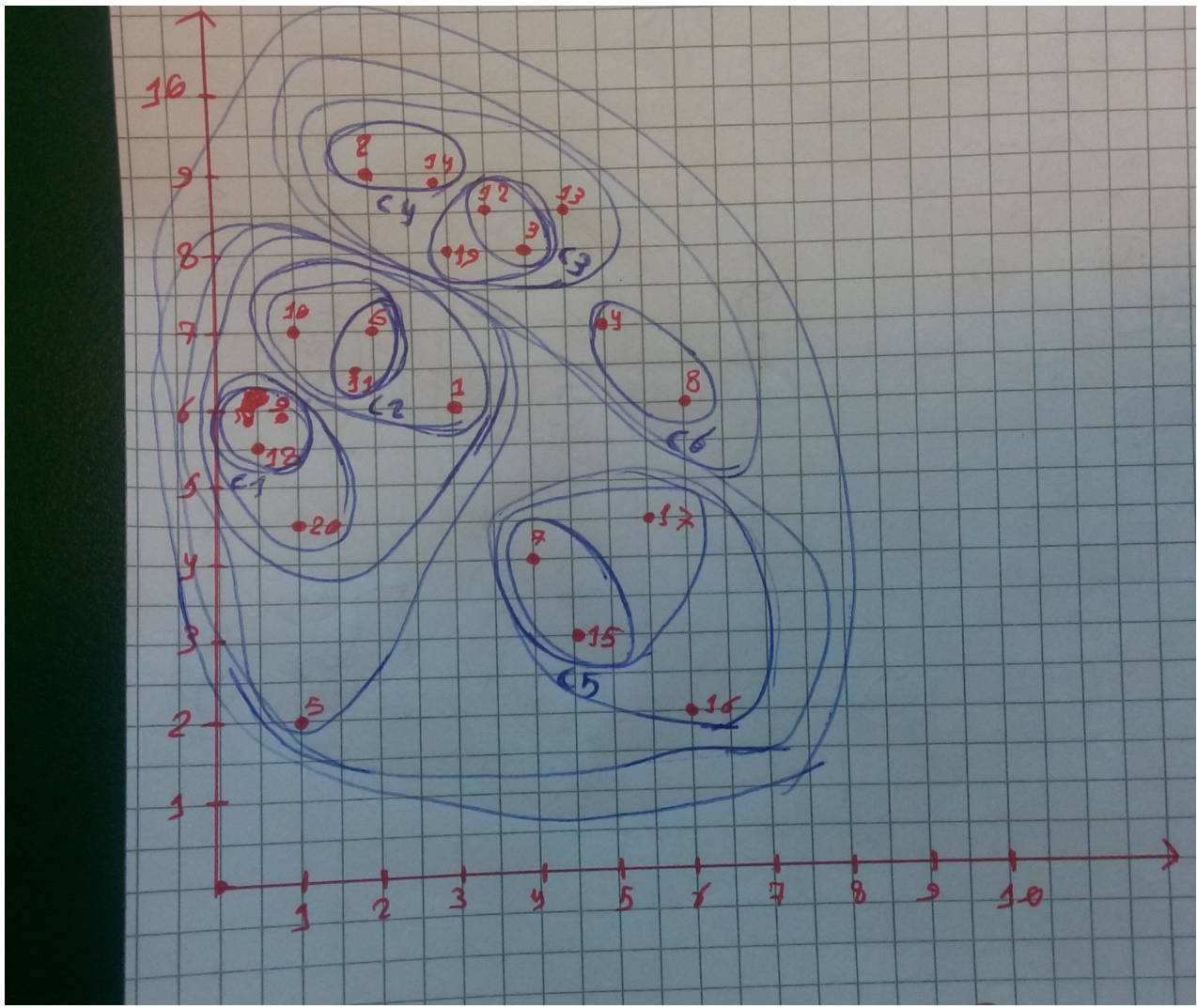


Fig. 7: Joining points map

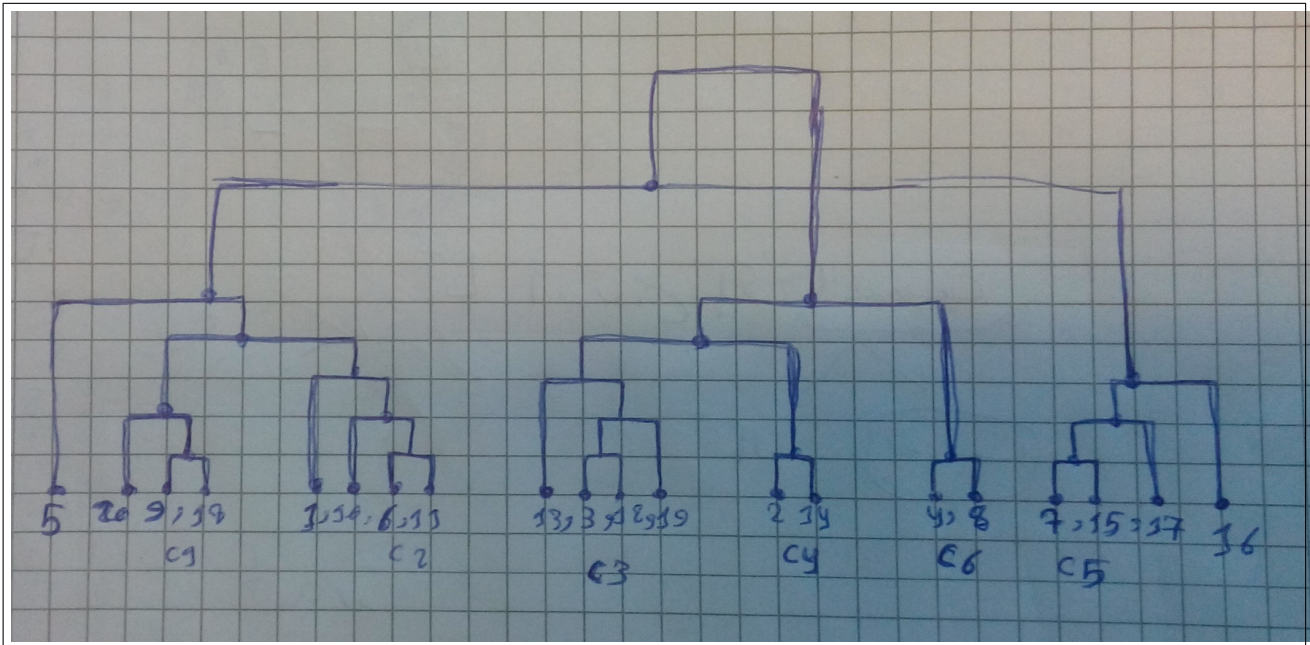


Fig. 8: The tree

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 :

```

1 import pandas as pd
2 import numpy as np
3 #matplotlib inline
4 import matplotlib.pyplot as plt
5 from IPython.display import display, HTML
6 from mpl_toolkits.mplot3d import Axes3D
7 from matplotlib import cm
8 from operator import attrgetter
9 from matplotlib.patches import Ellipse
10 from math import atan2, degrees
11 import numpy.random as rnd
12
13 points = pd.read_csv('Clustering.csv', sep=',')
14
15 class CenterValue:
16     def __init__(self, p, val, lab):
17         self.value = val
18         self.point = p
19         self.label = lab
20     def __str__(self):
21         return 'point:{}, Value:{}, index:{}'.format(self.point, self.value, self.label)
22
23 def Distance(p1, p2indx):
24     p1 = np.array(p1)
25     p2 = np.array((points.X[p2indx], points.Y[p2indx]))
26     return np.linalg.norm(p1 - p2)
27
28 def K_mean(centerpoints):
29     centers = {}
30     for i in centerpoints:
31         centers[i] = []
32     for i in range(len(points.index)):
33         values = []

```



```

34     values.append(CenterValue(centerpoints[0],Distance(centerpoints[0],i),i))
35     values.append(CenterValue(centerpoints[1],Distance(centerpoints[1],i),i))
36     values.append(CenterValue(centerpoints[2],Distance(centerpoints[2],i),i))
37     values.append(CenterValue(centerpoints[3],Distance(centerpoints[3],i),i))
38     values.sort(key=lambda x: x.value, reverse=False)
39     centers[values[0].point].append(GetPointsList([i])[0])
40     return centers
41
42 def GetPointsList(Indexs):
43     lst=[]
44     for i in Indexs:
45         lst.append((points.X[i],points.Y[i]))
46     return lst
47
48 def Get_Means(dictionary_of_lists):
49     lst=[]
50     for i in dictionary_of_lists.keys():
51         mX=0
52         mY=0
53         for p in dictionary_of_lists[i]:
54             mX+=p[0]
55             mY+=p[1]
56         lst.append((mX/len(dictionary_of_lists[i]),mY/len(dictionary_of_lists[i])))
57     return lst
58
59 def Plot(k):
60     keys = k.keys()
61     fig = plt.figure(0)
62     ax = fig.add_subplot(111, aspect='equal')
63     ax.plot([t[0] for t in k[keys[0]]], [t[1] for t in k[keys[0]]], 'rs')
64     ax.plot([t[0] for t in k[keys[1]]], [t[1] for t in k[keys[1]]], 'bs')
65     ax.plot([t[0] for t in k[keys[2]]], [t[1] for t in k[keys[2]]], 'gs')
66     ax.plot([t[0] for t in k[keys[3]]], [t[1] for t in k[keys[3]]], 'ms')
67     for k in keys:
68         ax.plot(k[0],k[1], 'k*')
69         ax.annotate('Cntr',(k[0],k[1]))
70     ax.axis([0, 10, 0,10])
71     plt.show()
72
73 #First Iteration
74 k = K_mean(GetPointsList(range(4)))
75 Plot(k)
76
77 #Each Time new Iteration
78 k = Get_Means(k)
79 k = K_mean(k)
80 Plot(k)

```

The previous code output the following :

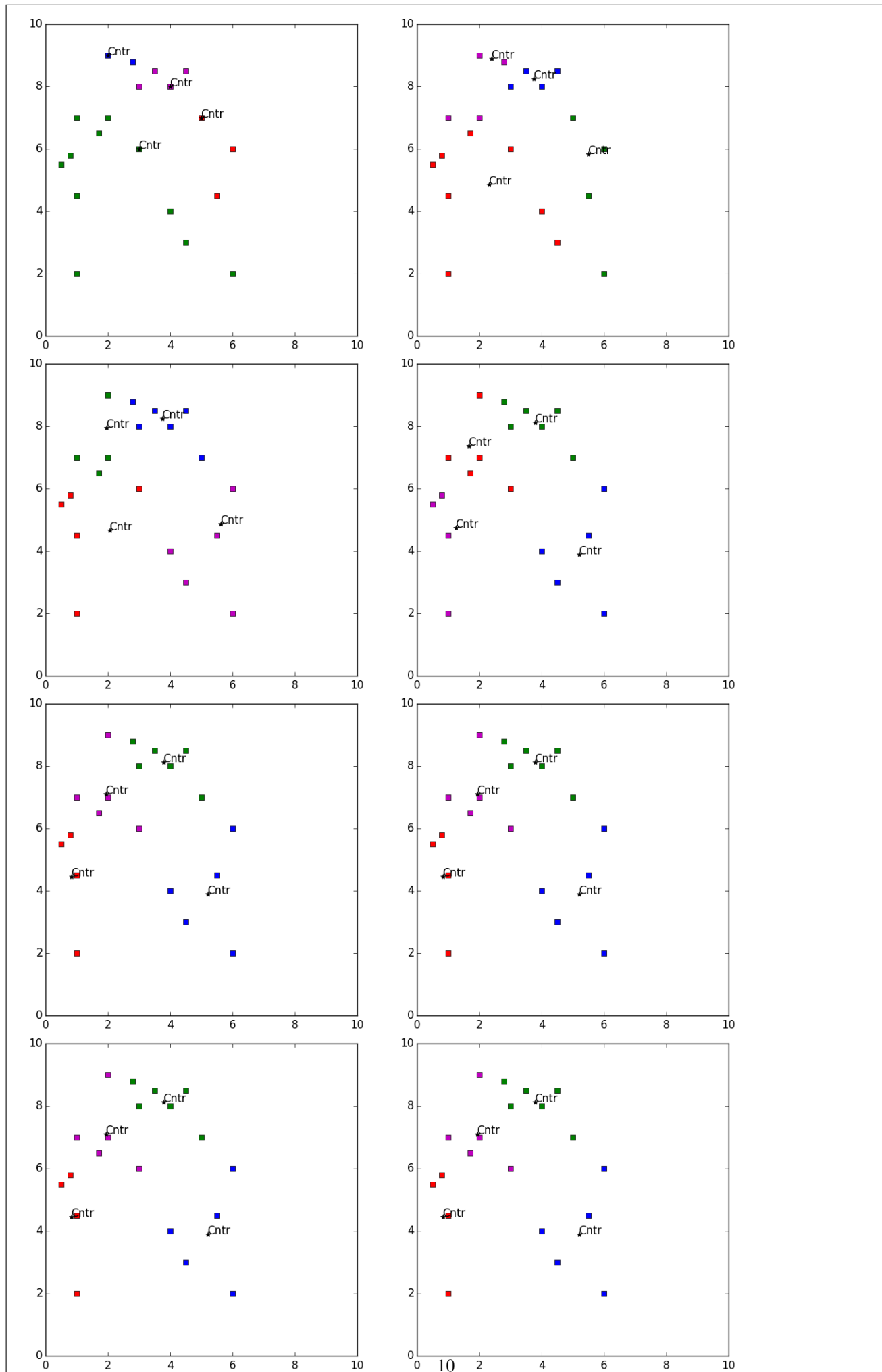


Fig. 9: Shows how centers change counting(1,2),(3,4)

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

To Find something different than the previous information I stemmed the text using the following code :

```
1
2 # coding: utf-8
3
4 # In [8]:
5
6 import time
7 start_time = time.time()
8
9 import numpy as np
10 import pandas as pd
11 from nltk.stem.porter import PorterStemmer
12 stemmer = PorterStemmer()
13 #from nltk.stem.snowball import SnowballStemmer #0.003 improvement but takes twice as long as
14     PorterStemmer
15 import re
16 import random
17 random.seed(1301)
18
19 # In [2]:
20
21 df_train = pd.read_csv('Data Warehouse/train.csv', encoding="ISO-8859-1")
22 df_test = pd.read_csv('Data Warehouse/test.csv', encoding="ISO-8859-1")
23 df_pro_desc = pd.read_csv('Data Warehouse/product_descriptions.csv')
24 df_attr = pd.read_csv('Data Warehouse/attributes.csv', encoding="ISO-8859-1")
25 df_brand = df_attr[df_attr.name == "MFG Brand Name"][["product_uid", "value"]].rename(columns
26     ={"value": "brand"})
27 num_train = df_train.shape[0]
28 df_all = pd.concat((df_train, df_test), axis=0, ignore_index=True)
29 df_all = pd.merge(df_all, df_pro_desc, how='left', on='product_uid')
30 df_all = pd.merge(df_all, df_brand, how='left', on='product_uid')
31
32 # In [9]:
33
34 def stem_sentence(sentence):
35     #if type(sentence)!=types.StringType:
36     #    return sentence
37     #print 'here'
38     try:
39         newsen = [stemmer.stem_word(x) for x in sentence.split()]
40         newsen = filter(None, newsen)
41         return ' '.join(newsen)
42     except:
43         return sentence
44
45 # In [6]:
46
47 stem_sentence('lights lights')
48
49
50 # In [6]:
51
52 df_all['search_term'] = df_all['search_term'].map(lambda x:stem_sentence(x))
53 df_all['product_title'] = df_all['product_title'].map(lambda x:stem_sentence(x))
54 df_all['product-description'] = df_all['product-description'].map(lambda x:stem_sentence(x))
55
56
57 # In [24]:
58
59 words={}
```

```

61 def AddToWords(sentence):
62     sentence = sentence.split()
63     for i in sentence:
64         words[i]=True
65
66
67 # In [12]:
68
69 df_all['search_term'].map(lambda x: AddToWords(x))
70 df_all['product_title'].map(lambda x: AddToWords(x))
71 df_all['product_description'].map(lambda x: AddToWords(x))
72
73
74 # In [14]:
75
76 len(words.keys())
77
78
79 # In [3]:
80
81 df_all['search_term']
82
83
84 # In [10]:
85
86 df_all.to_csv('df_all.csv', sep=',', encoding='utf-8')
87
88
89 # In [10]:
90
91 df_train = pd.read_csv('Data Warehouse/train.csv', encoding="ISO-8859-1")
92 df_test = pd.read_csv('Data Warehouse/test.csv', encoding="ISO-8859-1")
93 df_pro_desc = pd.read_csv('Data Warehouse/product_descriptions.csv')
94 df_attr = pd.read_csv('Data Warehouse/attributes.csv', encoding="ISO-8859-1")
95
96 print 'stemming training'
97 df_train['search_term'] = map(lambda x: stem_sentence(x), df_train['search_term'])
98 df_train['product_title'] = map(lambda x: stem_sentence(x), df_train['product_title'])
99 print 'stemming testing'
100 df_test['search_term'] = map(lambda x: stem_sentence(x), df_test['search_term'])
101 df_test['product_title'] = map(lambda x: stem_sentence(x), df_test['product_title'])
102 print 'stemming description go and sleep'
103 df_pro_desc['product_description'] = map(lambda x: stem_sentence(x), df_pro_desc['product_description'])
104 df_attr['name'] = map(lambda x: stem_sentence(x), df_attr['name'])
105 df_attr['value'] = map(lambda x: stem_sentence(x), df_attr['value'])
106
107 df_train.to_csv('df_train_stemmed.csv', sep=',', encoding='utf-8')
108 df_test.to_csv('df_test_stemmed.csv', sep=',', encoding='utf-8')
109 df_pro_desc.to_csv('df_pro_desc_stemmed.csv', sep=',', encoding='utf-8')
110 df_attr.to_csv('df_attr_stemmed.csv', sep=',', encoding='utf-8')

```

After that I used the kaggle code to generate the same statistics after stem the words and the output is Here.

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

E.O.F