Problem

Consider the following entry set:

0 1
1 1
5 6
4 7
6 7

Requirements:

Classify the above patterns into two distinct classes (you will have to find all the patterns that belongs to class 1 and all the patterns that belongs to class 2) using the dynamic kernels/clusters method

Steps and hints

"In various scientific areas (medicine, biology, archeology, economy, etc) vast sets of objects represented by a finite number of parameters frequently appear ; for the specialist, obtaining the groupings "natural and homogeneous" together with the most representative elements of such a set constitutes an important stage in the understanding of his data. A good approach for the solution of the problem is provided by clustering techniques which consist of finding a partition for a finite set such that each object resembles more to the objects within its group than the objects outside."

E Diday

More information on dynamic kernels/clusters method:
http://www.numdam.org/article/RO_1976__10_2_75_0.pdf

It is a convergent algorithm and it is used in data mining. It clusters the data into M groups (classes) where M is predefined.

Compared with other clustering algorithms, this algorithm requires less machine time and storage. It can be also referred as a generalization of the K-means algorithm.

The K-means algorithm is one of the most widely used clustering algorithm that uses an explicit distance measure to partition the data set into clusters.

Usage and applications

It has been successfully used in various topics, including market segmentation, computer vision, geostatistics, astronomy and agriculture. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

http://dni-institute.in/blogs/k-means-clustering-examples-and-practical-applications/

https://arxiv.org/ftp/arxiv/papers/1002/1002.2425.pdf - prediction of Students' Academic Performance

https://pdfs.semanticscholar.org/2ba1/942a08f3b9da97afa3b55719b5005ae2e5d0.pdf

```
dynamicKernels  (x[n][p], n, p, M, iclass[n])
        *) c[i] = x[i], i = 1,M // c[i] si x[i]  - arrayes with p number of elements
        *) iclass[i] = 0, i=1,n
        repeat
                done = true
                *) g[k][j]=0, j=1,p si k=1,M // initialize center of gravity matrix
                *) miu[k]=0 // counter of number of patterns for class
                for  i = 1, n execute
                {
                        dmin = MAX_VALUE
                        for  k =1, M execute
                                dik= calculateDistance(x[i], c[k])
                                if dik <dmin then
                                        dmin = dik
                                        kmin = k
                        miu[kmin] += 1
                        for  j=1,p execute
                                g[kmin][j] += x[i][j]
                        if iclass[i] != kmin then
                                iclass [i] = kmin
                                done = false
                }
                if  done ==false then
                        for k =1, M execute
                                for  j=1,p execute
```

g[k][j] /= miu[k]

***) c[k]=x[i0], where x[i0] is the clossest pattern to g[k]**

While done== false

end