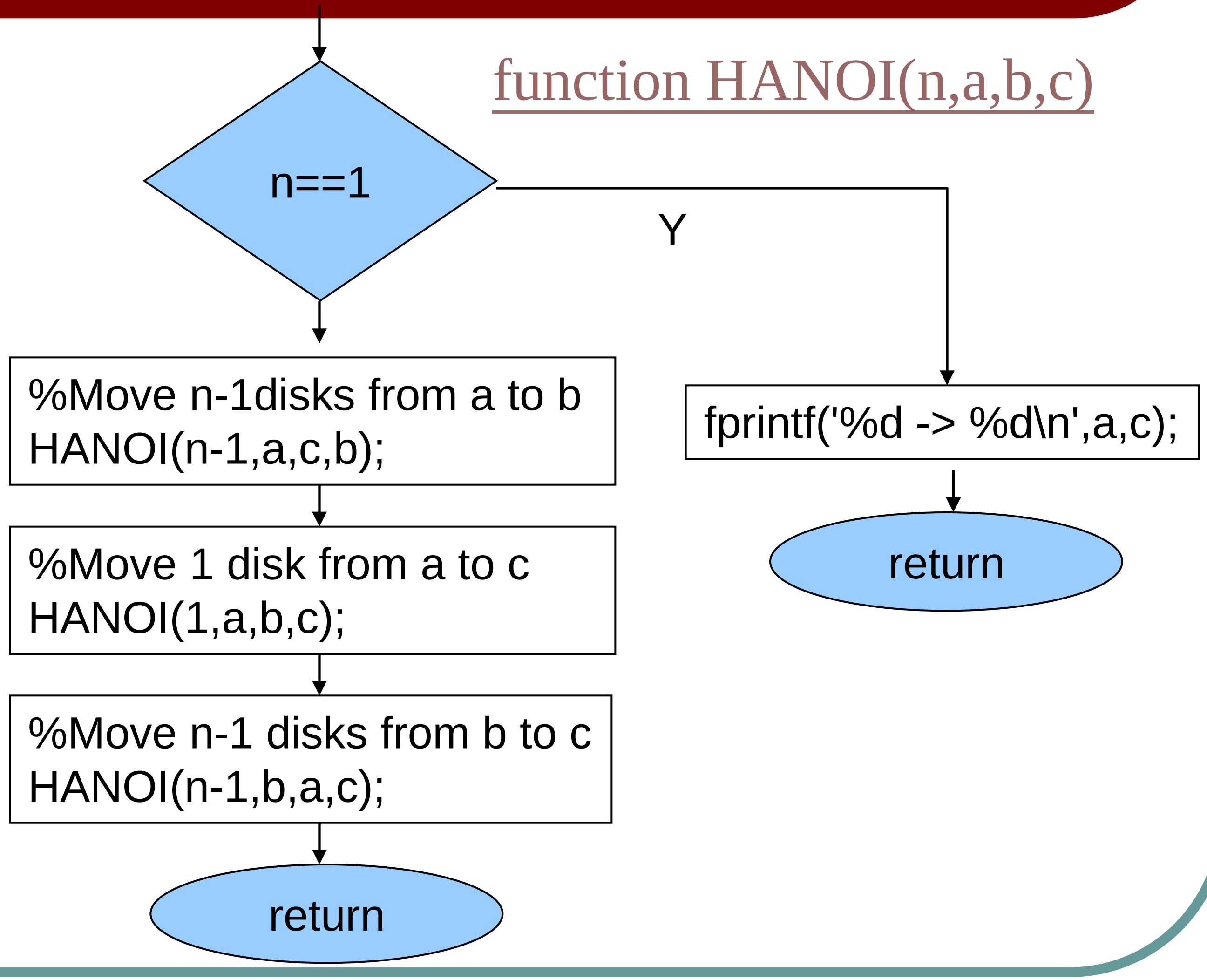
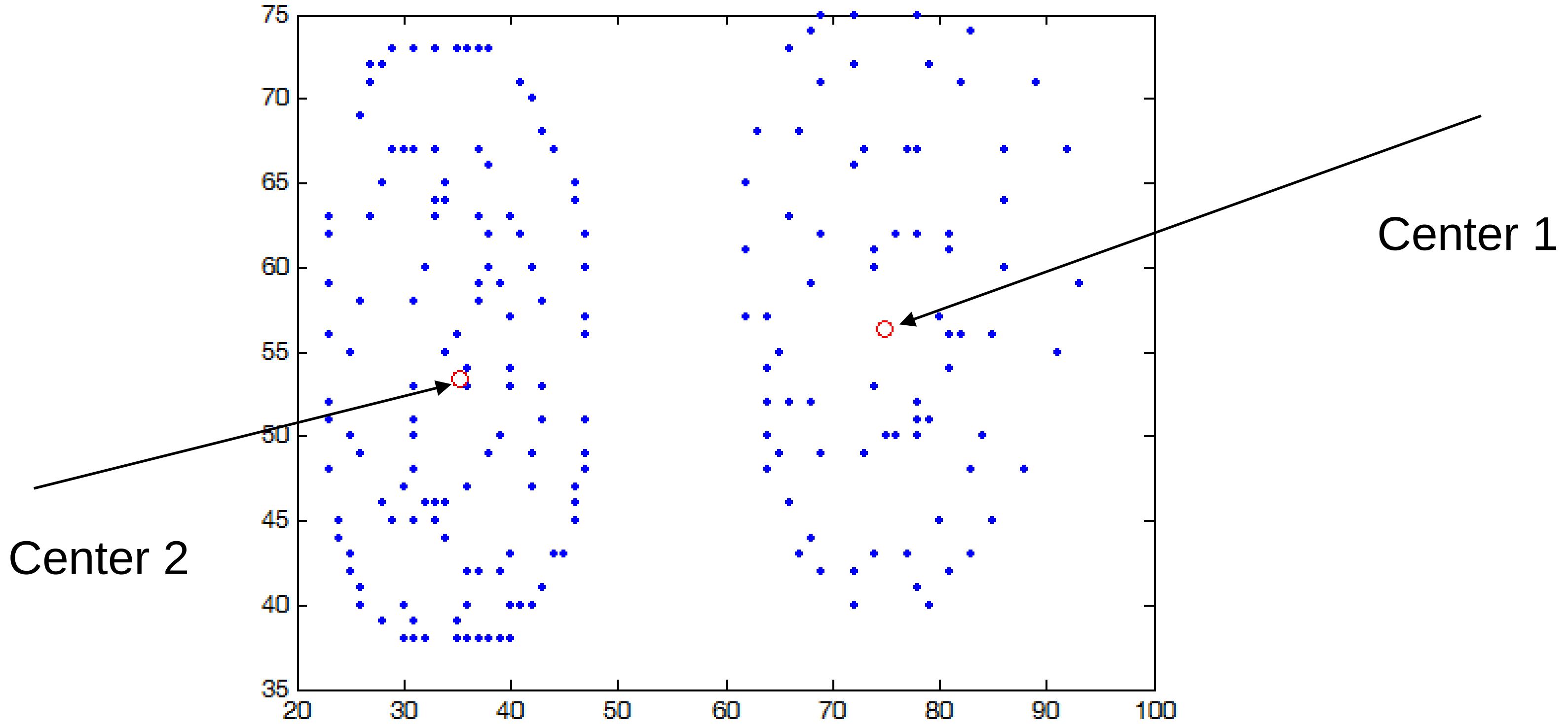
Lecture 8 K-means for clustering

- Cross distances
- Exclusive memberships
- Clustering
  - An iterative approach

Flow chart: move n disks from tower a to c



Two clusters



Cross distances

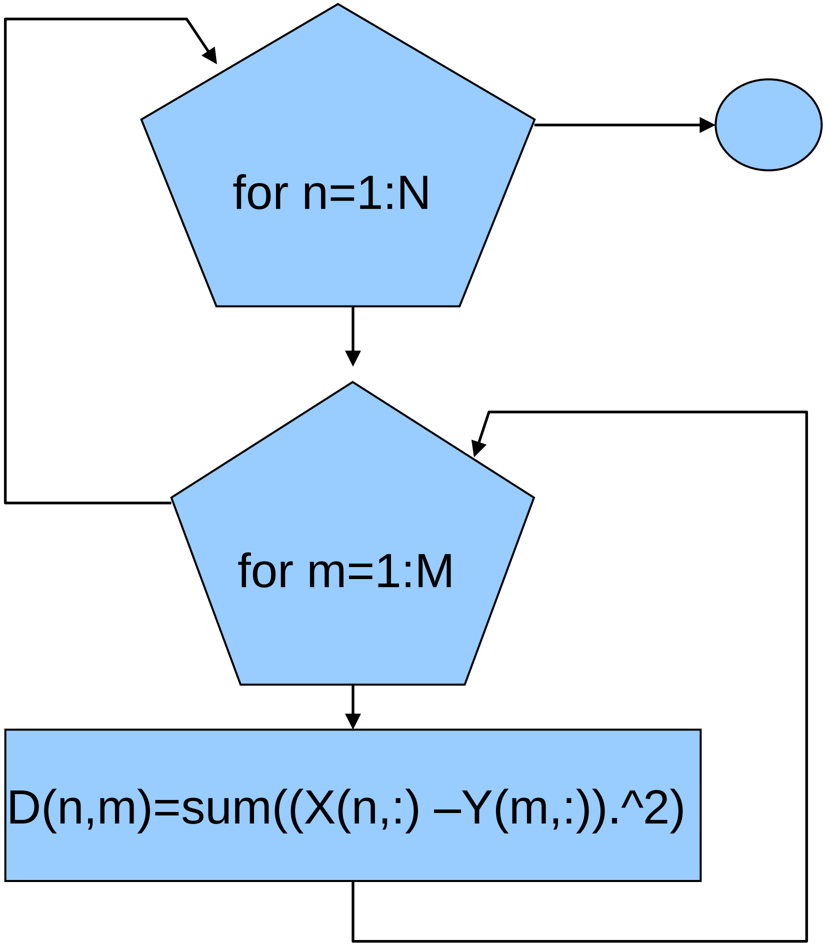
- How to find distances between centers and given points?

Calculation of Cross distances

- Given N points X: Nx2
- M centers Y: Mx2
- D: NxM
- D(i,j) denotes the distance between X(i,:) and Y(j,:)
- Given X and Y, find D

- It needs to calculate cross distances between N points and M centers to determine memberships of N points

Nested loops for cross distances



Matlab codes for nested codes

```
for i=1:N
    for j=1:M
        dd=X(i,:)-Y(j,:);
        D(i,j)=sqrt(sum(dd.^2));
    end
end
```

- Straightforward implementation
- Nested looping
  - A loop within a loop
  - MN calculations of the distance between a point and a center
- Time consuming for large M,N and d

Vector codes

- How to calculate cross distances without using for-looping or while-looping ?
- Vector codes are loop-free
- Vector codes for cross distances can significantly improve efficiency against nested looping in computation

$$D_{ij} = (\mathbf{x}_i - \mathbf{y}_j)(\mathbf{x}_i^T - \mathbf{y}_j^T)$$

$$= \mathbf{x}_i \mathbf{x}_i^T - 2\mathbf{x}_i \mathbf{y}_j^T + \mathbf{y}_j \mathbf{y}_j^T$$

$$= A_{ij} - 2B_{ij} + C_{ij}$$

D : cross distances between N points and M centers

Matrix D is decomposed to matrices A, B and C

A : elements in a row are identical

B : multiplication of matrix X and transpose of matrix Y

C : elements in a column are identical

Performance comparison

demo_distance2.m

```
for i=1:N
    for j=1:M
        dd=X(i,:)-Y(j,:);
        D(i,j)=sqrt(sum(dd.^2));
    end
end
ss2=cputime;
A=sum(X.^2,2)*ones(1,M);
C=ones(N,1)*sum(Y.^2,2)';
B=X*Y';
DD=sqrt(A-2*B+C);
sum(sum(abs(DD-D)))
```

$$D_{ij} = (\mathbf{x}_i - \mathbf{y}_j)(\mathbf{x}_i^T - \mathbf{y}_j^T)$$

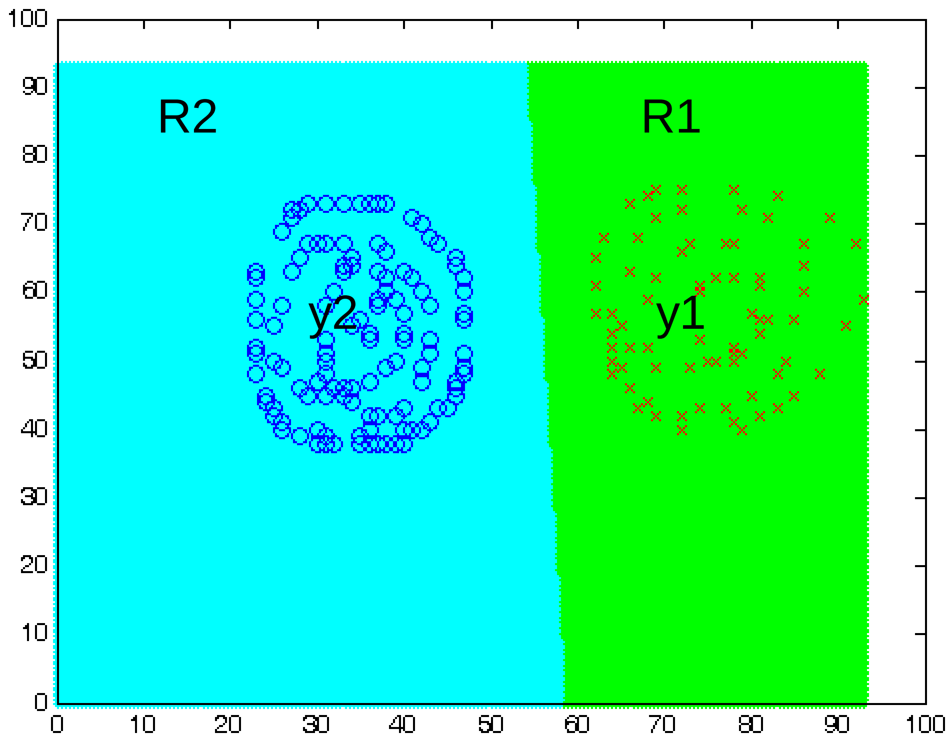
$$= \mathbf{x}_i \mathbf{x}_i^T - 2\mathbf{x}_i \mathbf{y}_j^T + \mathbf{y}_j \mathbf{y}_j^T$$

$$= A_{ij} - 2B_{ij} + C_{ij}$$

```
M=size(Y,1);N=size(X,1);
A=sum(X.^2,2)*ones(1,M);
C=ones(N,1)*sum(Y.^2,2)';
B=X*Y';
D=sqrt(A-2*B+C);
```

Partition to two regions

Each point has its **<u>exclusive membership</u>** to non-overlapping regions partitioned by two centers

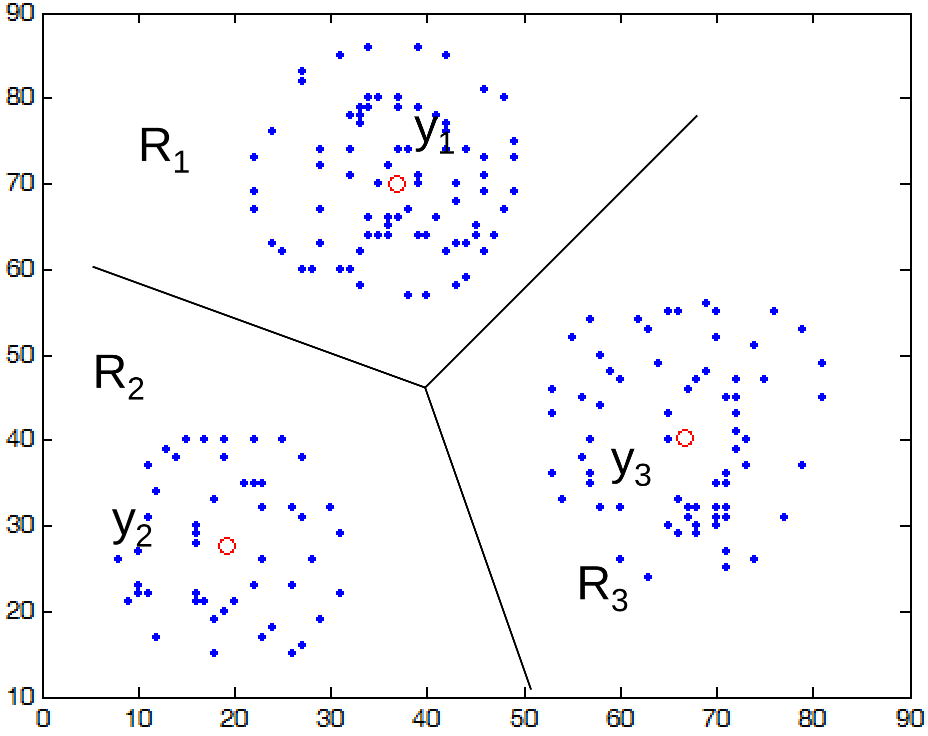


- Step A: calculate cross distances
- Step B: determine memberships
- Step C: determine K means

Exclusive membership

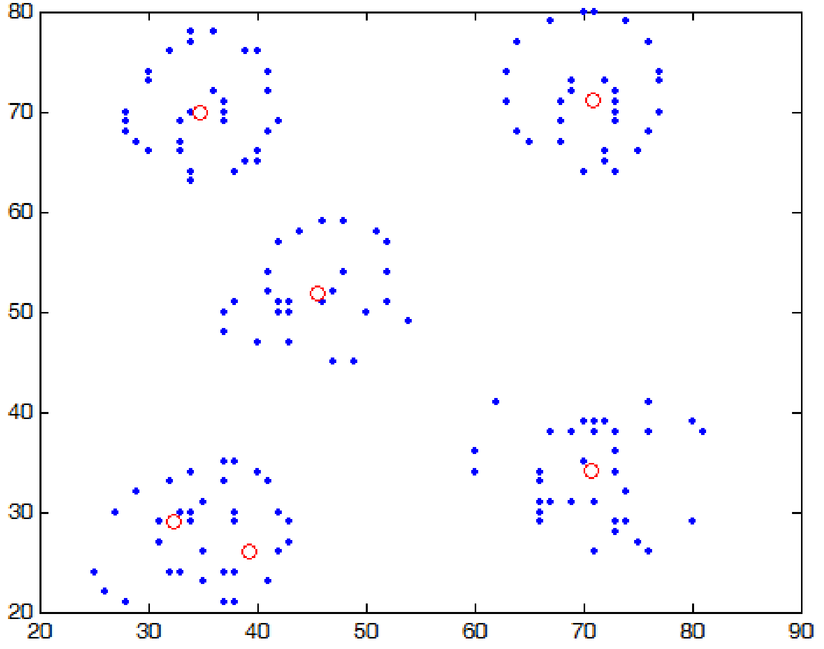
- y_1 and y_2 denote two centers
- R_1 and R_2 denote two regions partitioned by y_1 and y_2
- A point belongs to R_1 if it is closer to y_1
- A point belongs to R_2 if it is closer to y_2

Three clusters



K clusters

- Locating K centers
- Significant geometric features of points in R^d



Exclusive membership

- $y_1 , y_2 ,\ldots, y_K$ denote K distinct centers
- $R_1 , R_2 ,\ldots, R_K$ denote K regions partitioned by $y_1 , y_2 ,\ldots, y_K$
- A point belongs to R_i if it is closest to y_i among K centers

- Let D denote cross distances between N given points and M centers
- Find points nearest to the jth center

Point x(i,:) belongs the jth cluster
if

$$\mathrm{D}(i, j) = \min_{k} D(i, k)$$

Exclusive memberships (step B)

- Given D, exclusive memberships v of N points can be determined by

```
[xx v]=min(D');
```

Updating K-means (step C)

- Determine who belong the jth cluster

```
ind=find(v == j);
```

- Determine their mean

```
Y(j,:) = mean(X(ind,:))
```

Clustering

- Where are K centers ?

K-means

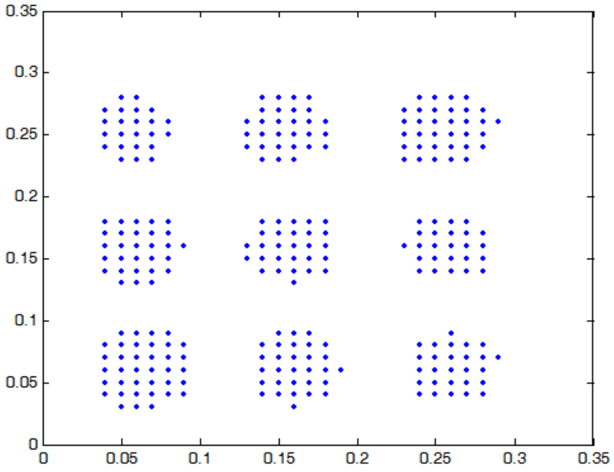
- A popular heuristic approach for clustering analysis
- An iterative approach
  - Step A : cross distance D
  - Step B : exclusive memberships v
  - Step C : updating centers Y

Data and MATLAB codes

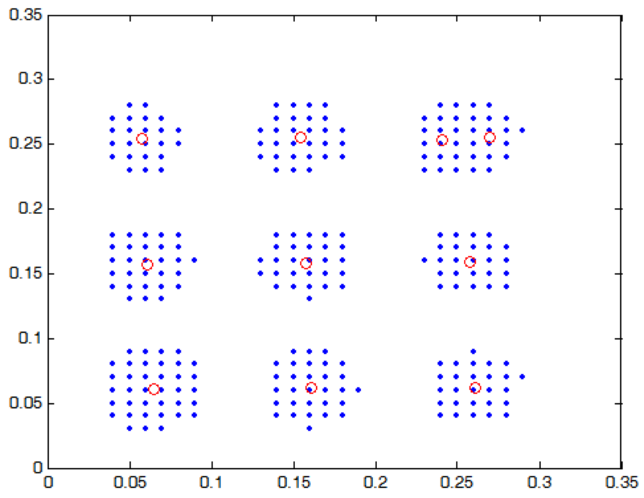
data_9.zip
demo_kmeans.m

```
load data_9.mat
plot(X(:,1),X(:,2),'.');
[cidx, Y] = kmeans(X,10);
hold on;
plot(Y(:,1),Y(:,2),'ro');
```

Data Clustering



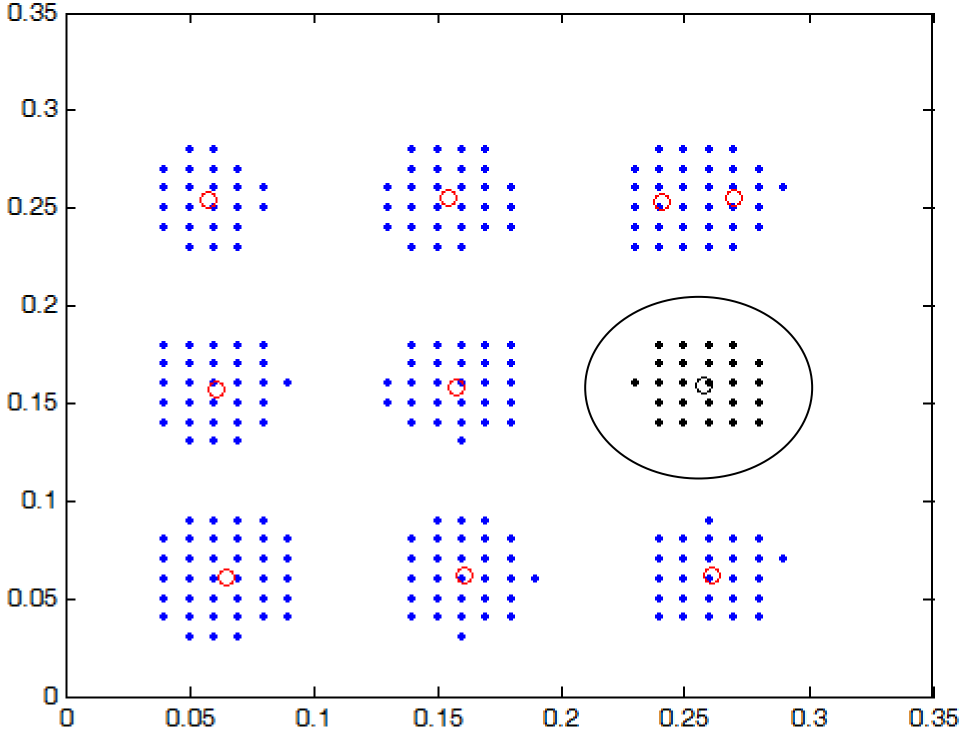
[cidx, Y] = kmeans(X,10);



- Partition given data to K clusters
- The K-means algorithm aims to find means (centers) of K clusters

Memberships

Black points belong to the cluster centered at black circle

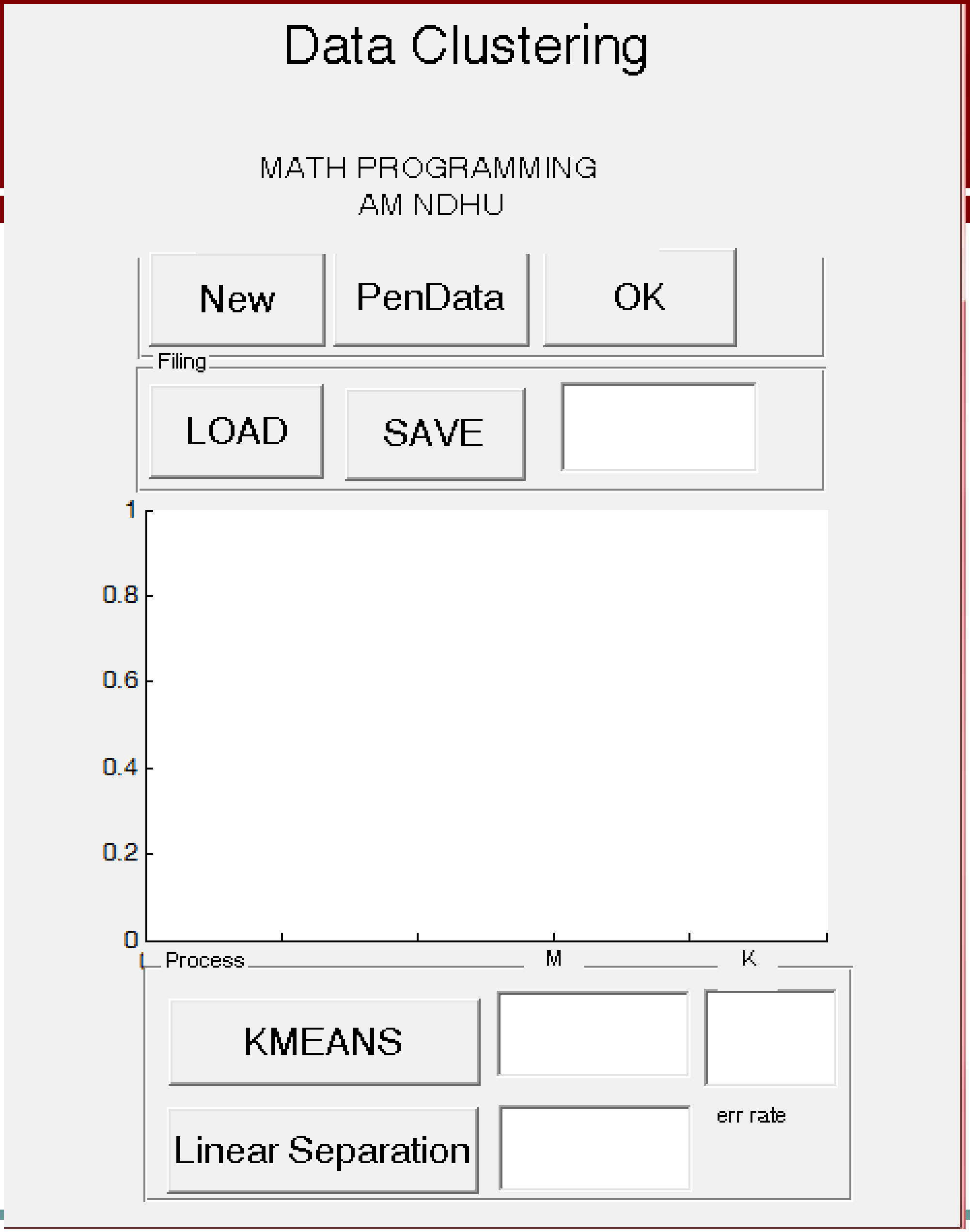


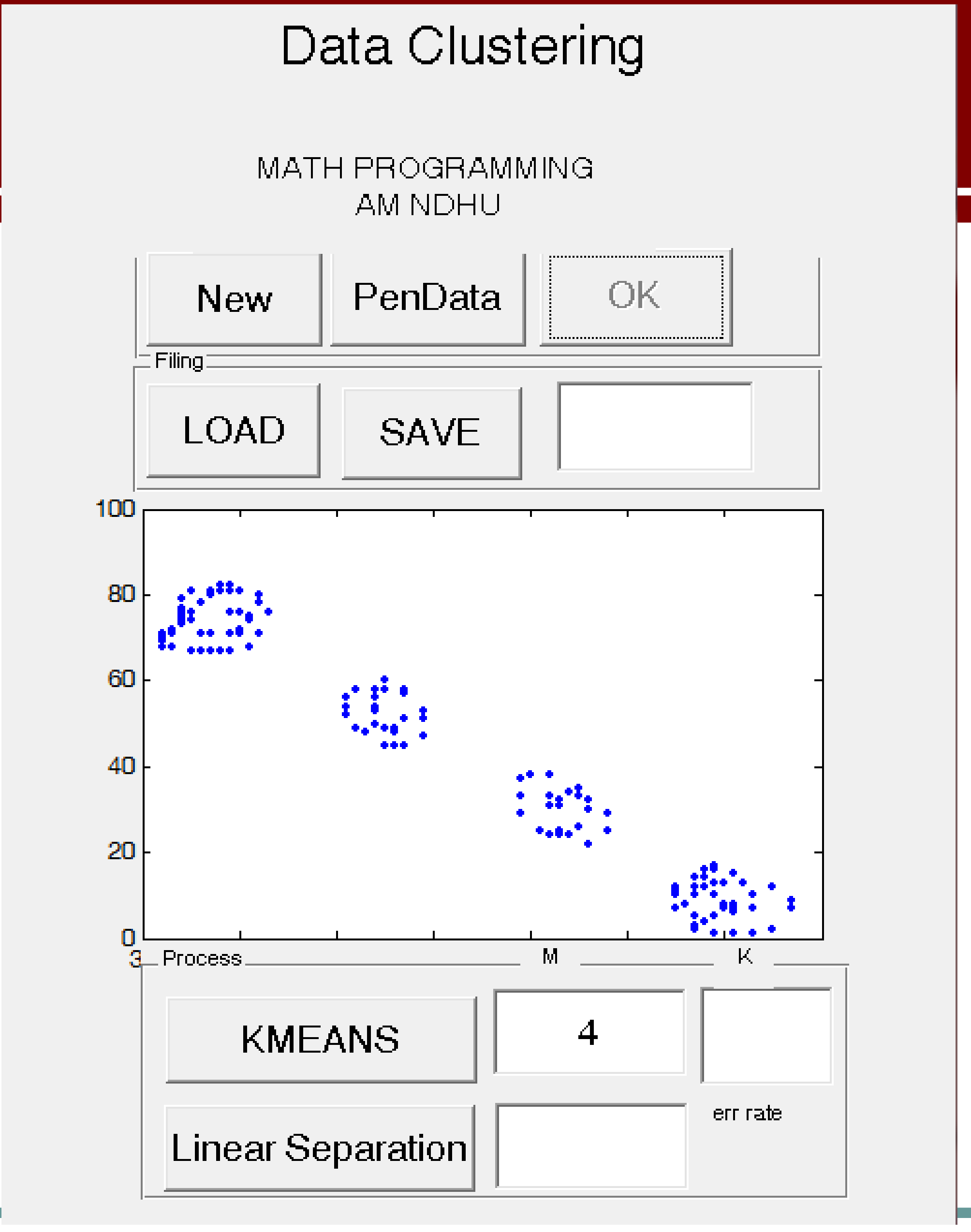
A math tool for data clustering

ClusteringTest.rar

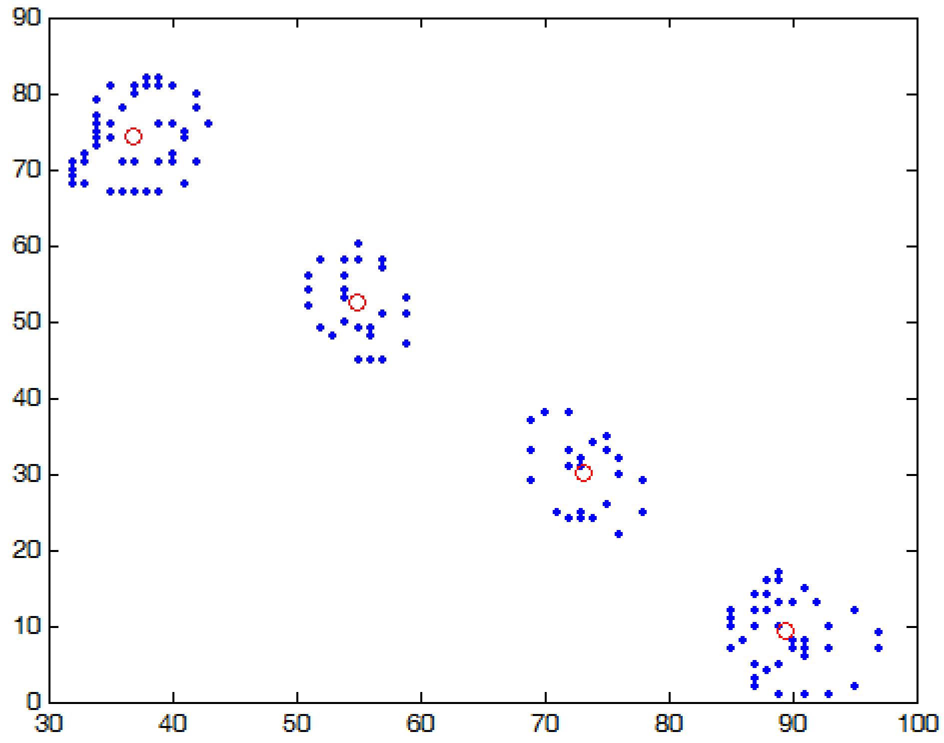
Steps for data clustering:

New
PenData
Enter M
KMEANS





Data Clustering
MATH PROGRAMMING
AM NDHU
New
PenData
OK
Filing
LOAD
SAVE
100
80
60
40
20
0
3
Process
M
K
KMEANS
4
Linear Separation
err rate

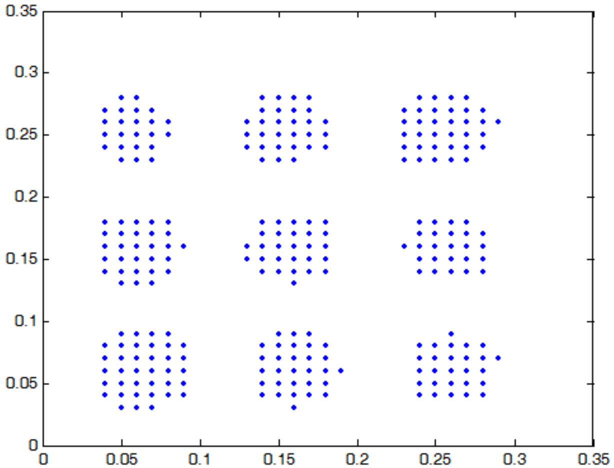

90
80
70
60
50
40
30
20
10
0
30
40
50
60
70
80
90
100

Main steps of K-means clustering

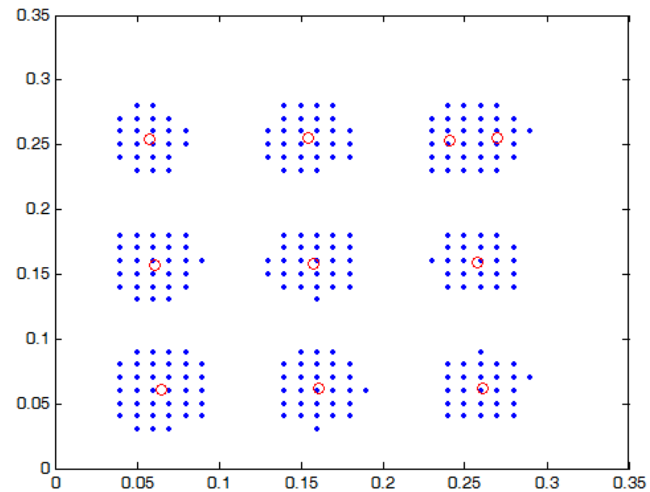
- Iterative execution of steps A, B and C until K means converge
  - A : cross distances D
  - B : exclusive memberships v
  - C : update K means Y

Step A: Calculation of Cross distances

- X: Nx2
- Y: Mx2
- D: NxM
- D(i,j) denotes the distance between X(i,:) and Y(j,:)
- Given X and Y, find D




```
[cidx, Y] = kmeans(X,10);
```

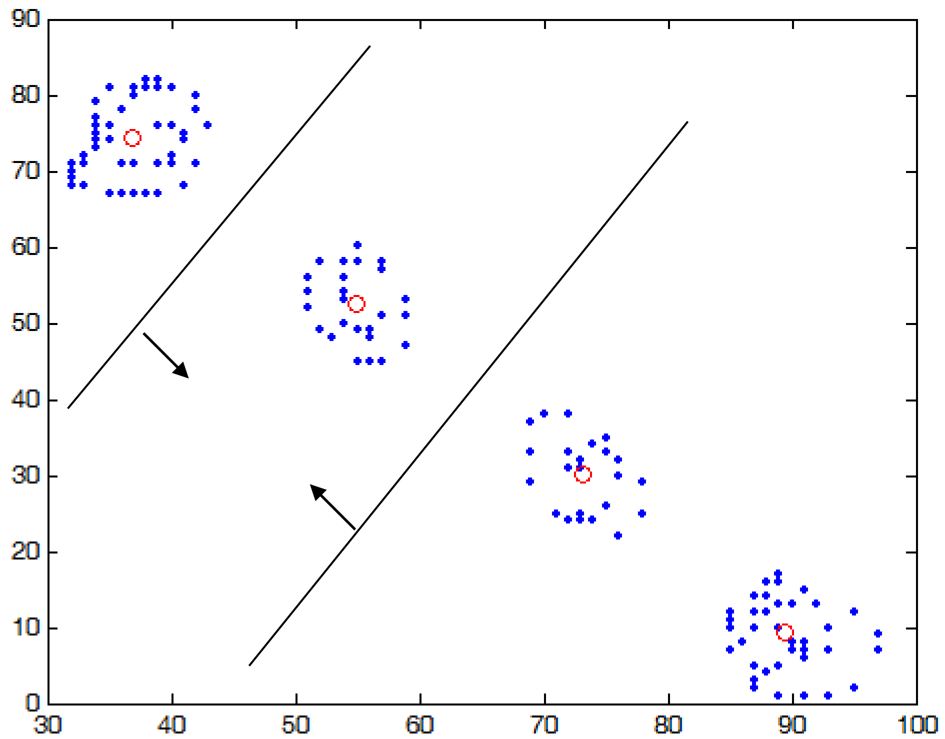



```
M=size(Y,1);N=size(X,1);
A=sum(X.^2,2)*ones(1,M);
C=ones(N,1)*sum(Y.^2,2)';
B=X*Y';
D=sqrt(A-2*B+C);
```

Cross Distances

Membership

- X(:,i) belongs a cluster



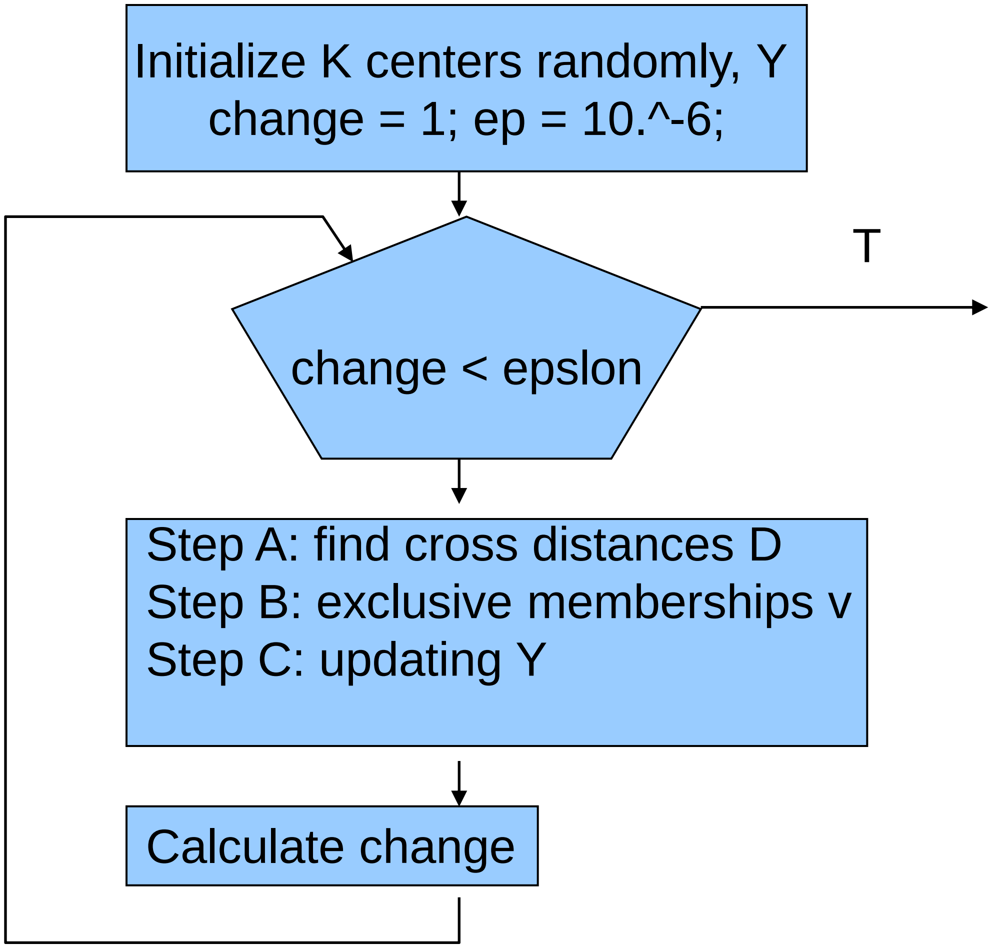
Exclusive memberships (step B)

- Given D, exclusive memberships v of N points can be determined by

```
[xx v]=min(D');
```

Updating K-means (step C)

```
ind=find(v == j);
Y(j,:) = mean(X(ind,:))
```


Initialize K centers randomly, Y
change = 1; ep = 10.^-6;
change < epslon
T
Step A: find cross distances D
Step B: exclusive memberships v
Step C: updating Y
Calculate change

Initialization

- Calculate the mean of N points
- mean_x=mean(X)
- Y=rand(M,2)*0.1-0.05+mean_x

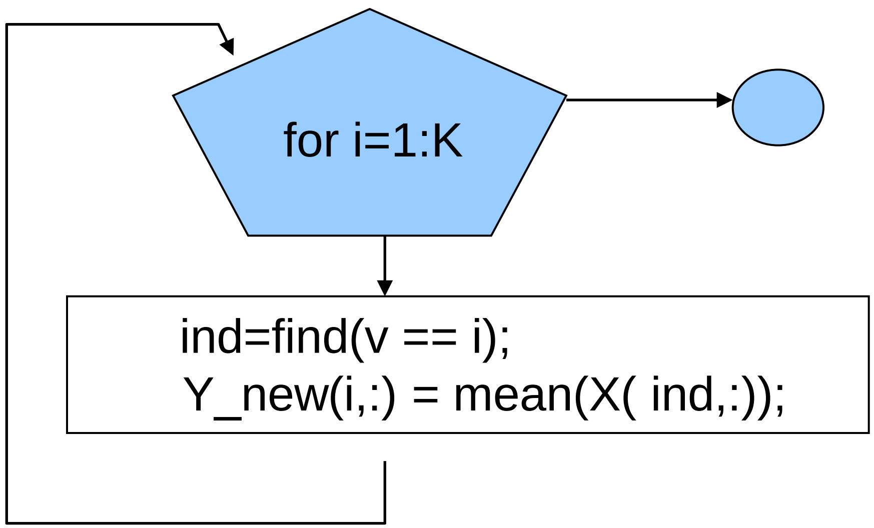
Step A: Cross distances

D = cross_distances(X,Y)

Step B: Exclusive memberships v

- [xx v]=min(D');

Step C: Update centers



Partition N points to K clusters

```
D = cross_distances(X,Y)
[xx v]=min(D');

%
for i=1:K
    ind=find(v == i);
    Y_new(i,:) = mean(X( ind,:));
end
```

Halting

- change = mean(mean(abs(Y-Y_new)))
- Halting condition

  chang < ep

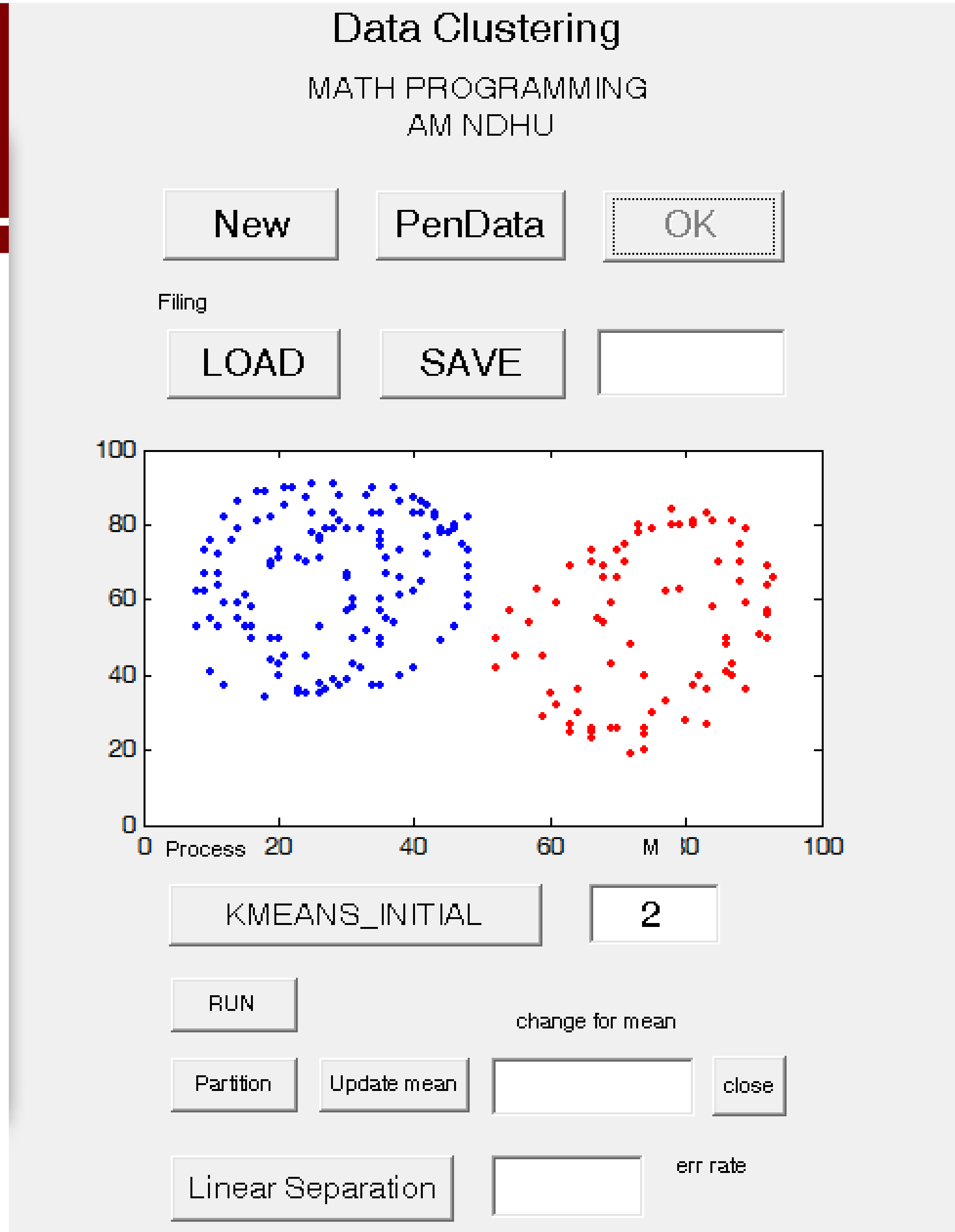
my_kmeans

my_kmeans.m

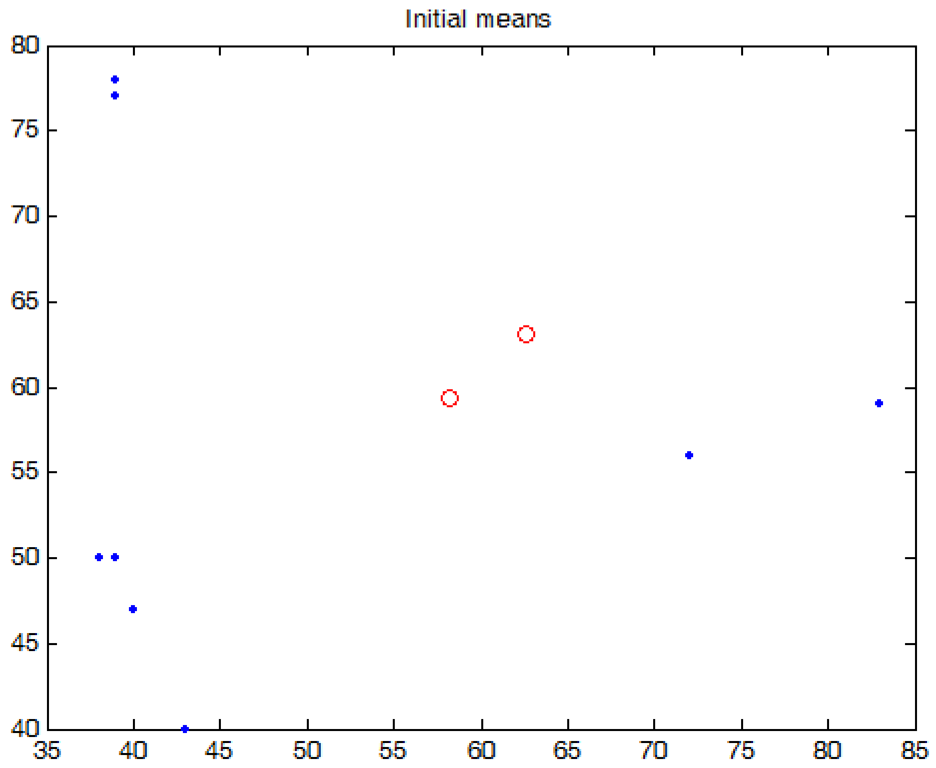
```
load data_9.mat
>> plot(X(:,1),X(:,2),'.');
>> Y=my_kmeans(X,10);
>> hold on
>> plot(Y(:,1),Y(:,2),'or');
```

ClusteringTest2.fig
ClusteringTest2.m

A version that is able to show stepwise execution of partitioning and updating of the kmeans algorithm


Data Clustering
MATH PROGRAMMING
AM NDHU
New
PenData
OK
Filing
LOAD
SAVE
100
80
60
40
20
0
0
Process
20
40
60
M
100
KMEANS_INITIAL
2
RUN
change for mean
Partition
Update mean
close
Linear Separation
err rate

Initialization



X =

39	78
39	77
43	40
40	47
38	50
72	56
83	59
39	50

Y =

49.0603	57.2121
49.1061	57.2084

Cross distance

X =

39	78
39	77
43	40
40	47
38	50
72	56
83	59
39	50

Y =

49.0603	57.2121
49.1061	57.2084

```
D = cross_distances(X,Y);
```

D =

23.0943	23.1176
22.1984	22.2226
18.2478	18.2596
13.6519	13.6797
13.2039	13.2404
22.9717	22.9257
33.9868	33.9412
12.3783	12.4135

```
[xx v]=min(D');
```

v =

1 1 1 1 1 2 2 1

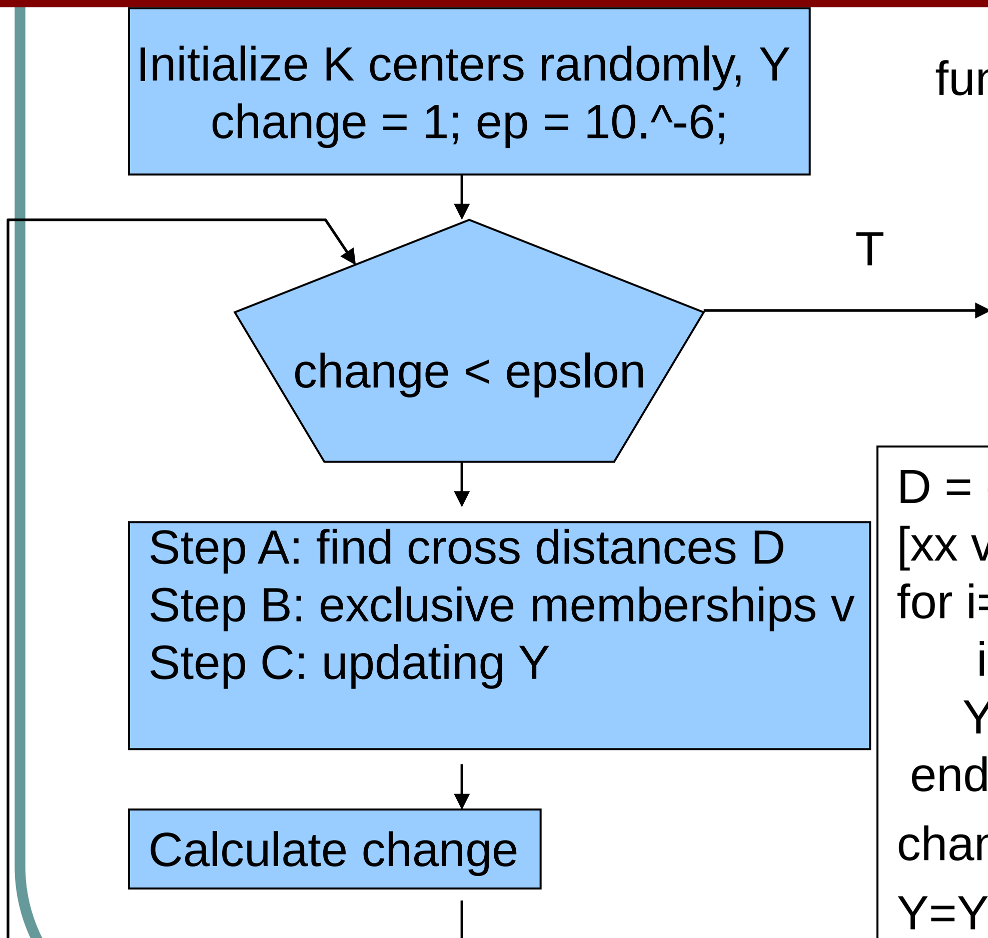
Partition & Update

ind =

1 1 1 1 1 2 2 1

```
for i=1:M
    ind=find(v == i);
   Y_new(i,:) =mean(X(ind,:));
end
```

function Y=my_kmeans(X,M)




```
D = cross_distances(X,Y);
[xx v]=min(D');
for i=1:M
      ind=find(v == i);
     Y_new(i,:) =mean(X(ind,:));
 end
change = mean(mean(abs(Y-Y_new)));
Y=Y_new;
```

k-means clustering - Wikipedia

Description

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, the *k*-means clustering aims to partition the *n* observations into *k* sets ($k < n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

History

The term "*k*-means" was first used by James MacQueen in 1967,[1] though the idea goes back to Hugo Steinhaus in 1956.[2] The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published until 1982.[3]

Standard Algorithm

Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).

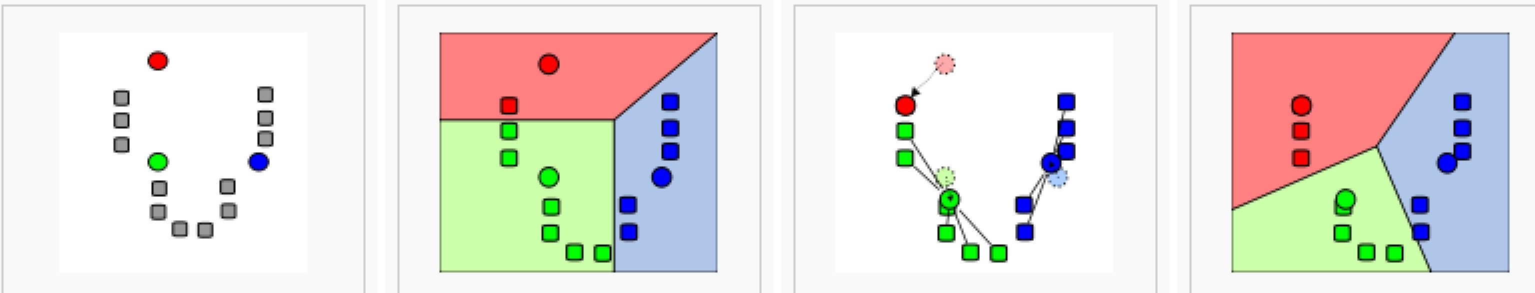
$$S_i^{(t)} = \left\{ \mathbf{x}_j : \big\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \big\| \leq \big\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \big\| \text{ for all } i^* = 1, \ldots, k \right\}$$

Update step: Calculate the new means to be the centroid of the observations in the cluster.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

The algorithm is deemed to have converged when the assignments no longer change.

Demonstration of the standard algorithm



1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).

2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3) The centroid of each of the *k* clusters becomes the new means.

4) Steps 2 and 3 are repeated until convergence has been reached.

Advanced topics based on K-means

- Classification
- Function approximation
- Density estimation