Πολυεπίπεδο Perceptron-Αλγόριθμος K-Means.

Τμήμα Μηχανικών Η/Υ Πανεπιστήμιο Ιωαννίνων 2021-2022

Μάθημα: Υπολογιστική Νοημοσύνη

Υπεύθυνος Καθηγητής: Λύκας Αριστείδης

Άσκηση 2:

Εντολή μεταγλώττισης για την παραγωγή του εκτελέσιμου αρχείου:

Ο αλγόριθμος k-means υλοποιήθηκε στο αρχείο p3.c και η μεταγλώττισή του γίνεται με την εξής εντολή:
gcc -o p3 p3.c -lm

Έπειτα, μπορούμε να τρέξουμε το εκτελέσιμο με την εντολή:
./p3

Το αρχείο δεδομένων του k-means είναι το "DatasetII". Η εντολή "createDatasetII" στην main του κώδικα έχει μπει σε σχόλια, ώστε να μη χρειάζεται να τρέχει κάθε φορά, όμως ο κώδικάς της έχει συμπεριληφθεί στο αρχείο πηγαίου κώδικα και έχουμε παραδώσει και ενδεικτικό αρχείο "DatasetII", ώστε να γίνει η εκτέλεση του αρχείου p3.

Κατασκευή Συνόλου Δεδομένων 2:

Αρχικά, ξεκινήσαμε με την κατασκευή του συνόλου δεδομένων που θα χρησιμοποιηθεί για τον αλγόριθμο K-means. Υλοποιήθηκε η συνάρτηση `void createDatasetII()` η οποια καλείται στην αρχή της main μια φορά, παράγει το αρχείο "DatasetII" απο το οποίο θα πάρουμε τα παραδείγματα. Στην συνέχεια μπαίνει σε κατάσταση σχολιασμού, αν ο χρήστης επιθυμεί να φτιάξει ένα καινούργιο σύνολο δεδομένων μπορεί να διαγράψει το σχολιασμό.

```
void createDatasetII(){
//Create the Dataset 2.
    int i=0;
    FILE *fp;
    fp = fopen("DatasetII","w");//Make the file of the Dataset
    srand(time(NULL));
    for (i=0;i<150;i++){
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (125 + 1 - 75) + 75)/100, (double)((rand() % (125 + 1 - 75) + 75))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (50 + 1 - 0) + 0)/100, (double)((rand() % (50 + 1 - 0) + 0))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (50 + 1 - 0) + 0)/100, (double)((rand() % (200 + 1 - 150) + 150))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (200 + 1 - 150) + 150)/100, (double)((rand() % (50 + 1 - 0) + 50))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (200 + 1 - 150) + 150)/100, (double)((rand() % (200 + 1 - 150) + 150))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (20 + 1 - 0) + 0)/100, (double)((rand() % (20 + 1 - 0) + 0))/100);
    }
    for (i=0;i<75;i++){
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (80 + 1 - 60) + 60)/100, (double)((rand() % (40 + 1 - 0) + 0))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (80 + 1 - 60) + 60)/100, (double)((rand() % (200 + 1 - 160) + 160))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (140 + 1 - 120) + 120)/100, (double)((rand() % (40 + 1 - 0) + 0))/100);
        fprintf(fp,"%1.3f %1.3f\n", (double)(rand() % (140 + 1 - 120) + 120)/100, (double)((rand() % (200 + 1 - 160) + 160))/100);
    }
    fclose(fp);
```

#define και καθορισμός πινάκων:

Στην αρχή του προγράμματος έχουν γίνει define και έχουν οριστεί οι μεταβλητές και οι πίνακες που θα χρειαστούμε για την υλοποίηση του αλγορίθμου K-means.

```c
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <math.h>

#define M 9 //Teams that will be created by K-Means.
#define dimension 2 //dimension of the points of the dataset.
#define datasetValues 1200 //How many points on the dataset.

float datasetArray[datasetValues][dimension];//The values of the Dataset stored in an array.
float centersArray[M][dimension];//The center for every Team.
int whichTeam[datasetValues];//We will store here,the theam that  belongs every value on the dataset.
int teamChanged = 1;//flag to count the team changes.


void createDatasetII();
void createDatasetArray();
void createFirstCenters();
void Kmeans();
void calculateSfalmaOmadopoihshs();
```

Κατασκευή Πίνακα συνόλου δεδομένων:

Υλοποιήσαμε την συνάρτηση `void createDatasetArray()` προκειμένου να πάρουμε τα παραδείγματα από τα αρχείο και να τα αποθηκεύσουμε στον καθολικό πίνακα datasetArray.

```c
void createDatasetArray(){
    //Create an array with all the values of the datasetII.
    FILE *fp;
    int i,j;
    fp=fopen("DatasetII","r");
    for (i=0;i<datasetValues;i++){
        for (j=0;j<dimension;j++){
            fscanf(fp,"%f",&datasetArray[i][j]);
        }
    }
    fclose(fp);
}
```

Ο πίνακας αυτός έχει αρχικοποιηθεί στην αρχή του προγράμματος ως

```c
float datasetArray[datasetValues][dimension];//The values of the Dataset stored in an array.
```

Κατασκευή Πίνακα τυχαίων κέντρων:

Για να αναπαραστήσουμε την θέση των κέντρων κάθε ομάδας χρησιμοποιούμε τον πίνακα centersArray έναν πίνακα με M γραμμές δηλαδή όσο το πλήθος των ομάδων και dimension στήλες δηλαδή όσο η διάσταση των παραδειγμάτων.

```
void createFirstCenters(){
    //Choose which points of the given Dataset will be the first random centers.
    int i=0,j=0,randomChoice;

    srand(time(NULL));
    for (i=0;i<M;i++){
        randomChoice = rand()%datasetValues;
        for (j=0;j<dimension;j++){
            centersArray[i][j]=datasetArray[randomChoice][j];
        }
    }
}
```

K-means:

Ο αλγόριθμος K-means αναπαριστάται μέσω μία συνάρτησεις και ενός do while loop στην συνάρτηση main().

```
int main(){
    //createDatasetII();
    createDatasetArray();
    createFirstCenters();
    do{//while loop until no point changes team.
        teamChanged =0;
        Kmeans();
    }while(teamChanged);
    calculateSfalmaOmadopoihshs();
}
```

Η καθολική μεταβλητή teamChanged λειτουργεί σαν flag προκειμένου να ξέρουμε αν έχει γίνει αλλαγή σε κάποιο παράδειγμα με απώτερο σκοπό να αναγνωρίσουμε τον τερματισμό του αλγορίθμου. Η μεταβλητή αυτή τροποποιείται μέσα στην συνάρτηση Kmeans().

Η συνάρτηση `void Kmeans();`

Για κάθε παράδειγμα, υπολογίζει την μικρότερη απόσταση του απο τα υπάρχων κέντρα και ανάλογα απο ποιο κέντρο την βρήκε καθορίζει την ομάδα του παραδείγματος αυτού. Ακόμα, γίνεται έλεγχος αν η ομάδα που ανήκε το παράδειγμα η οποία είναι αποθηκευμένη στον πίνακα whichTeam[datasetValues] είναι ίδια με την ομάδα που έχει την ελάχιστη απόσταση και αν δεν ισχύει αυτη η συνθήκη η καθολική μεταβλητή teamChanged ενημερώνεται σε 1 , όπως και ο πίνακας whichTeam με την ομάδα που έχει επιλεχθεί για το στοιχείο.

```
void Kmeans(){
    //Mach every point in a team.
    int i,j,k;
    float newTeam,min,sum;
    int howManyInTeam[M] = {0};
    float sumArray[M][dimension] = {0};
    for (i=0;i<datasetValues;i++){//for every point.
        min=10000000;
        for (j=0;j<M;j++){//for every team-center. Find the Distance of the point from every center
            sum=0;
            for (k=0;k<dimension;k++){
                sum += pow(centersArray[j][k]-datasetArray[i][k],2);
            }
            if(sqrt(sum)<min){
                min=sqrt(sum);
                newTeam = j;
            }
        }
        if(whichTeam[i]!=newTeam){
            teamChanged=1;
            whichTeam[i]=newTeam;
        }
```

Επίσης,υπολογίζει την θέση που θα έχουν τα νέα κέντρα. Αρχικά, αρχικοποιούνται δύο πίνακες ο πίνακας howManyInTeam[M] ο οποίος κρατάει το πλήθος των στοιχείων που βρίσκονται σε κάθε ομάδα και ο πίνακας sumArray[M][dimension] ο οποίος κρατάει το άθροισμα των συντεταγμένων των παραδειγμάτων κάθε ομάδας. Έπειτα διατρέχεται κάθε παράδειγμα και γεμίζουν με τα κατάλληλα στοιχεία οι παραπάνω πίνακες προκειμένου να βρούμε το νέο εκπρόσωπο-Κέντρο με διαίρεση του SumArray και του howManyInTeam.

```
    }//find new centers
    for (i =0;i<datasetValues;i++){
        howManyInTeam[whichTeam[i]]++;
        for (j=0;j<dimension;j++){
            sumArray[whichTeam[i]][j] +=   datasetArray[i][j];
        }
    }
    for (i =0;i<M;i++){
        for (j=0;j<dimension;j++){
            centersArray[i][j]= sumArray[i][j] / howManyInTeam[i];
        }
    }
}
```

Η συνάρτηση Kmeans() καλείται μέσα στην main() εώς ότου να μην αλλάξει κανένα παράδειγμα ομάδα.

Τέλος, υπολογίζεται το σφάλμα ομαδοποίησης μέσω της συνάρτησης.

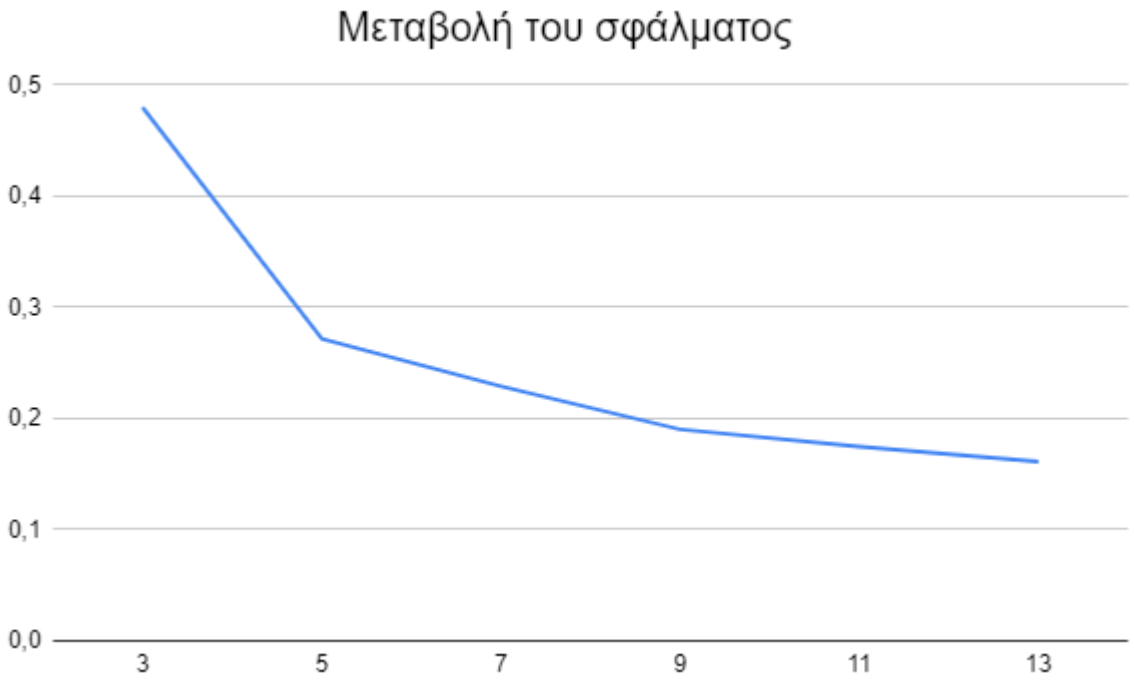
```
void calculateSfalmaOmadopoihshs()
```

Για κάθε παράδειγμα, υπολογίζεται η Ευκλείδεια απόσταση ||Xi-μκ||^2 από το κέντρο μk της ομάδας στην οποία ανήκει και αθροίζουμε τις αποστάσεις για όλα τα παραδείγματα xi. Τέλος, διαιρούμε το συνολικό σφάλμα που έχουμε συλλέξει με τον αριθμό των παραδειγμάτων.

```
void calculateSfalmaOmadopoihshs(){
    int i,j;
    float sfalma=0.0;
    for (i=0;i<datasetValues;i++){
        sfalma+=sqrt(pow(datasetArray[i][0]-centersArray[whichTeam[i]][0],2)+pow(datasetArray[i][1]-centersArray[whichTeam[i]][1],2));
    }
    sfalma = (float)sfalma/datasetValues;
    printf("Sfalma Omadopoihshs : %f\n",sfalma);
}
```

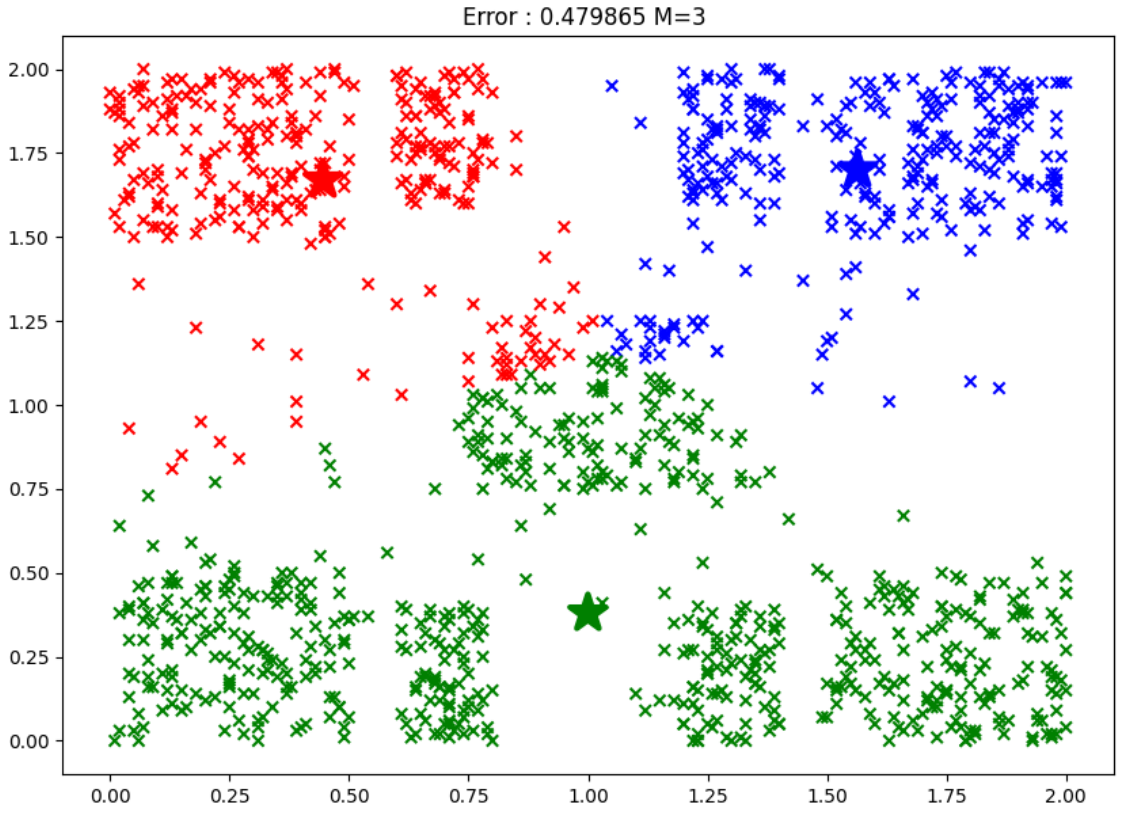
Πορίσματα:

Μετά από 20 εκτελέσεις του προγράμματος για καθένα από τα M=3,5,7,9,11,13 προέκυψε η γραφική παράσταση Σφάλματος-Αριθμού ομάδων

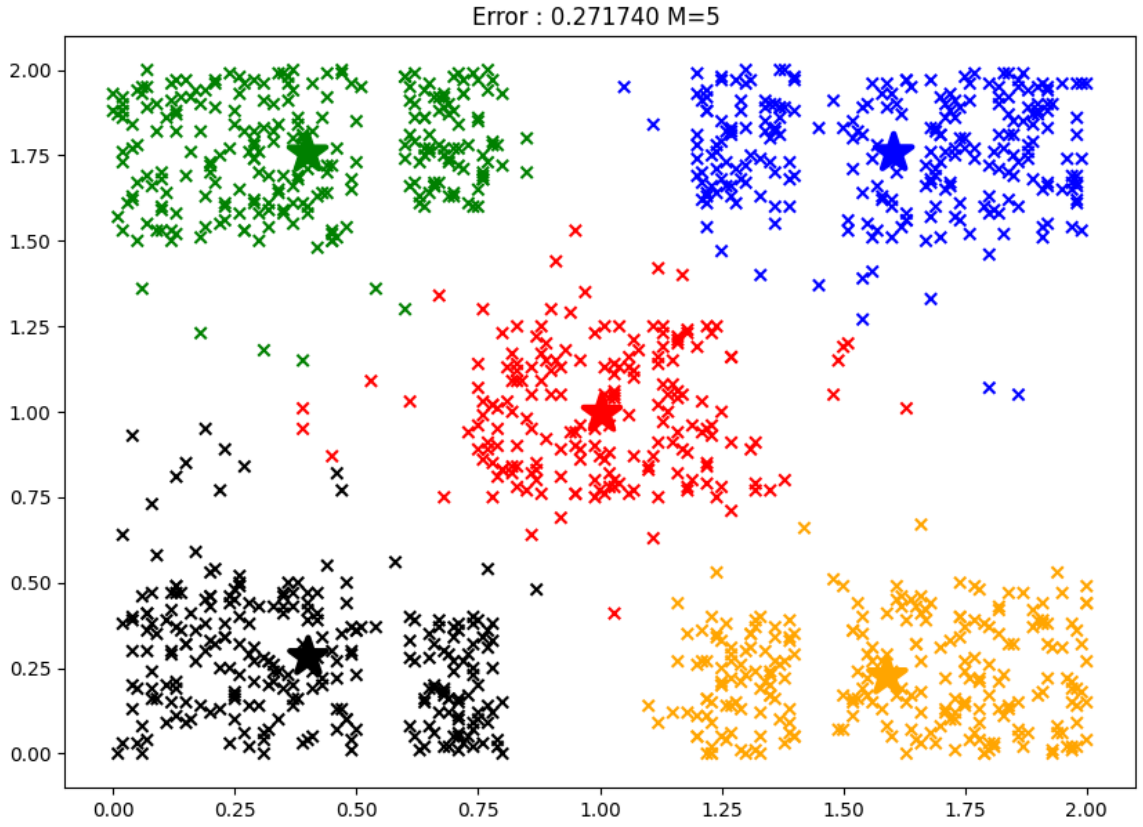


Απο την παραπάνω γραφική βλέπουμε ότι το σφάλμα ελαττώνεται μέχρι και για M=9 με μεγάλη κλίση ενώ για M=11,13 η μεταβολή που παρουσιάζεται στην καλύτερη περίπτωση είναι της τάξης του 10^(-3). Άρα μπορούμε με μεγάλη βεβαιότητα να συμπαιράνουμε ότι ο πραγματικός αριθμός των ομάδων ειναι 9. Το ίδιο μπορούμε να συμπεράνουμε και αν δούμε τις επόμενες γραφικές παραστάσεις.

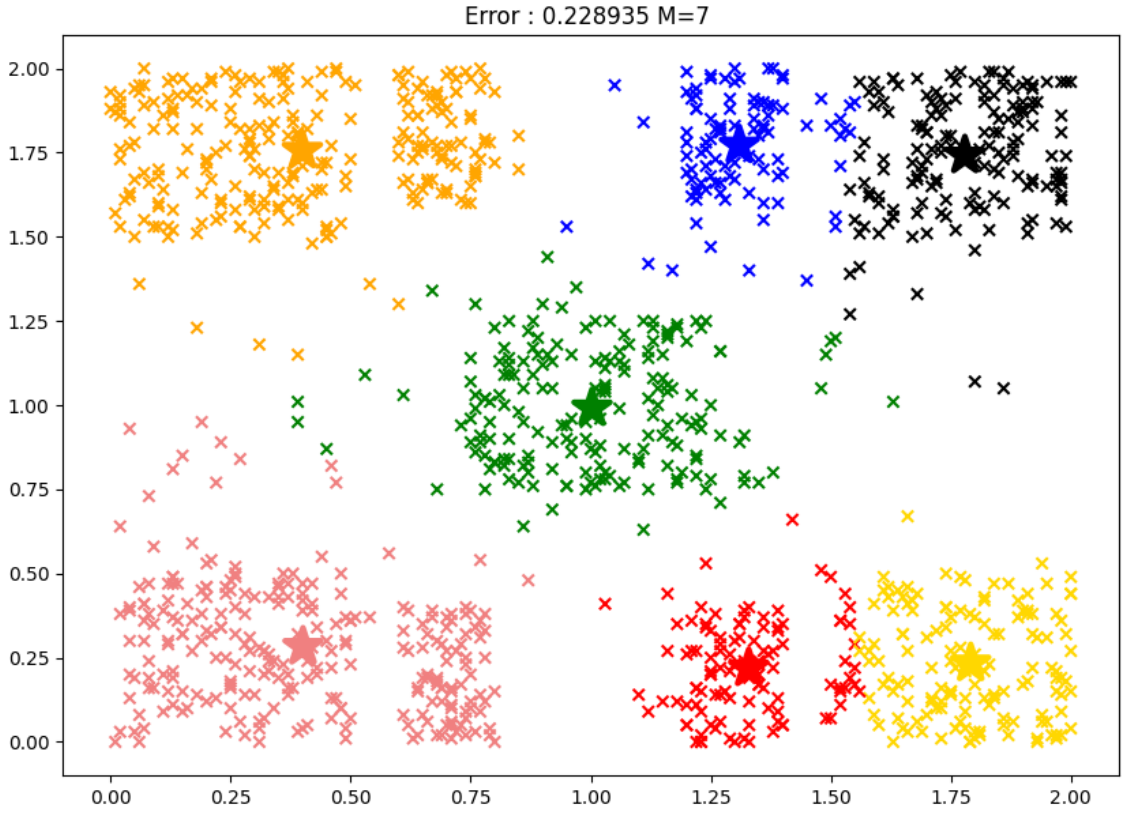
ΓΙΑ Μ = 3:


Error : 0.479865 M=3
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

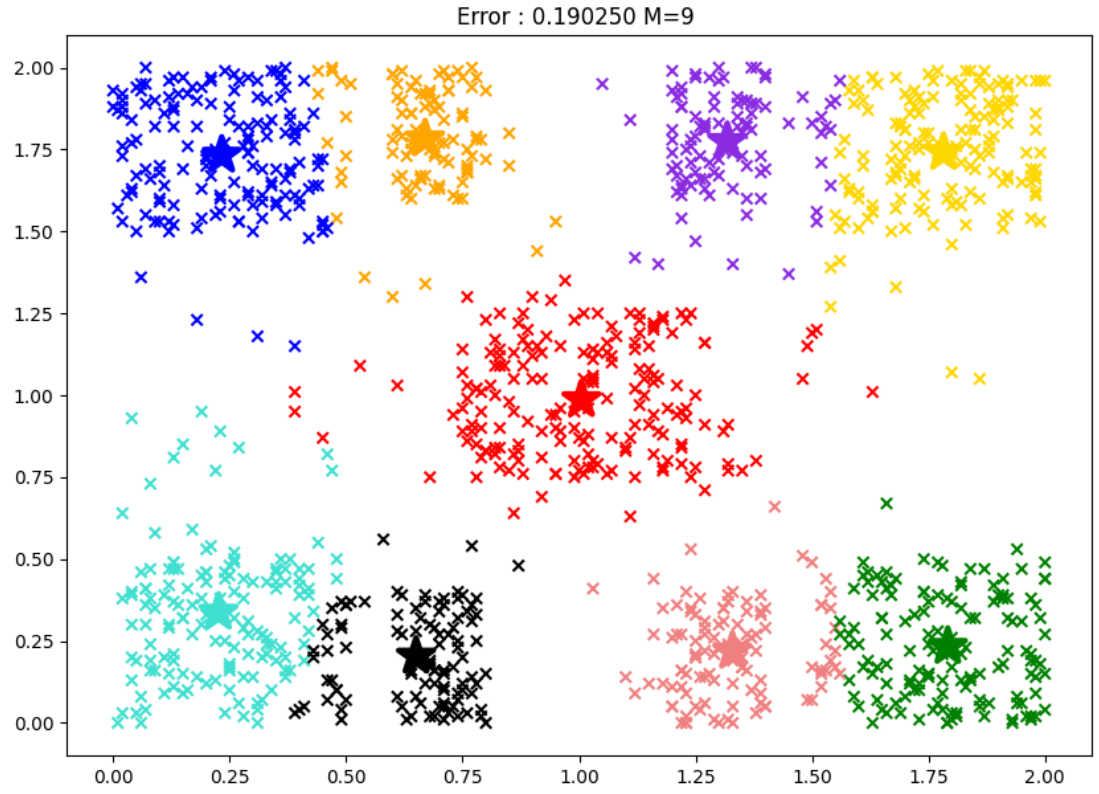
ΓΙΑ Μ = 5:


Error : 0.271740 M=5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

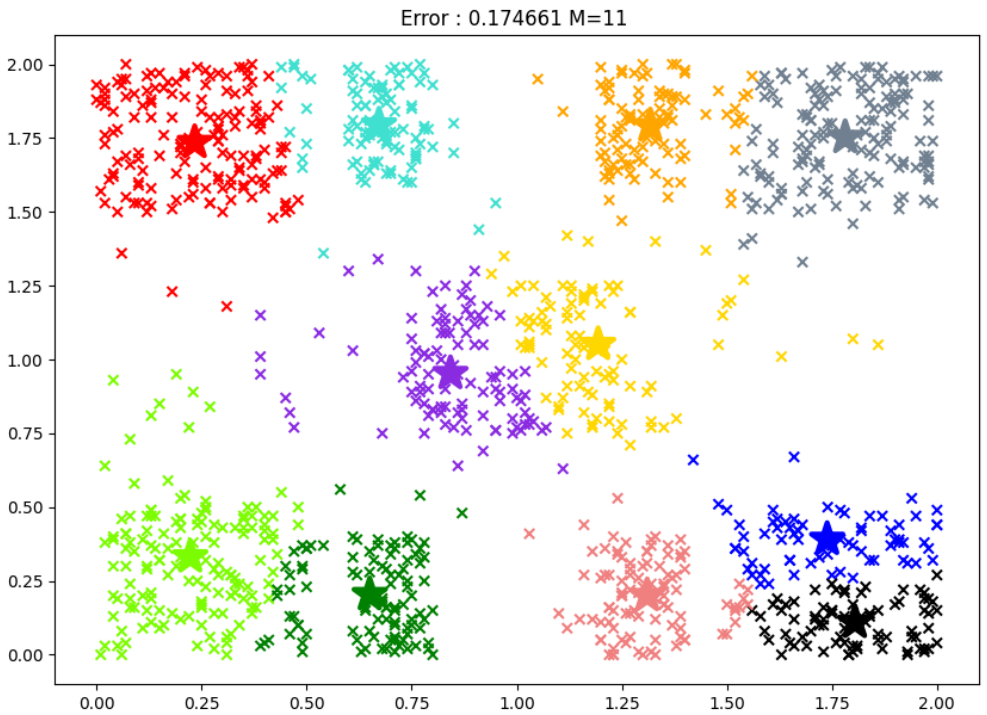
ΓΙΑ Μ = 7:


Error : 0.228935 M=7
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

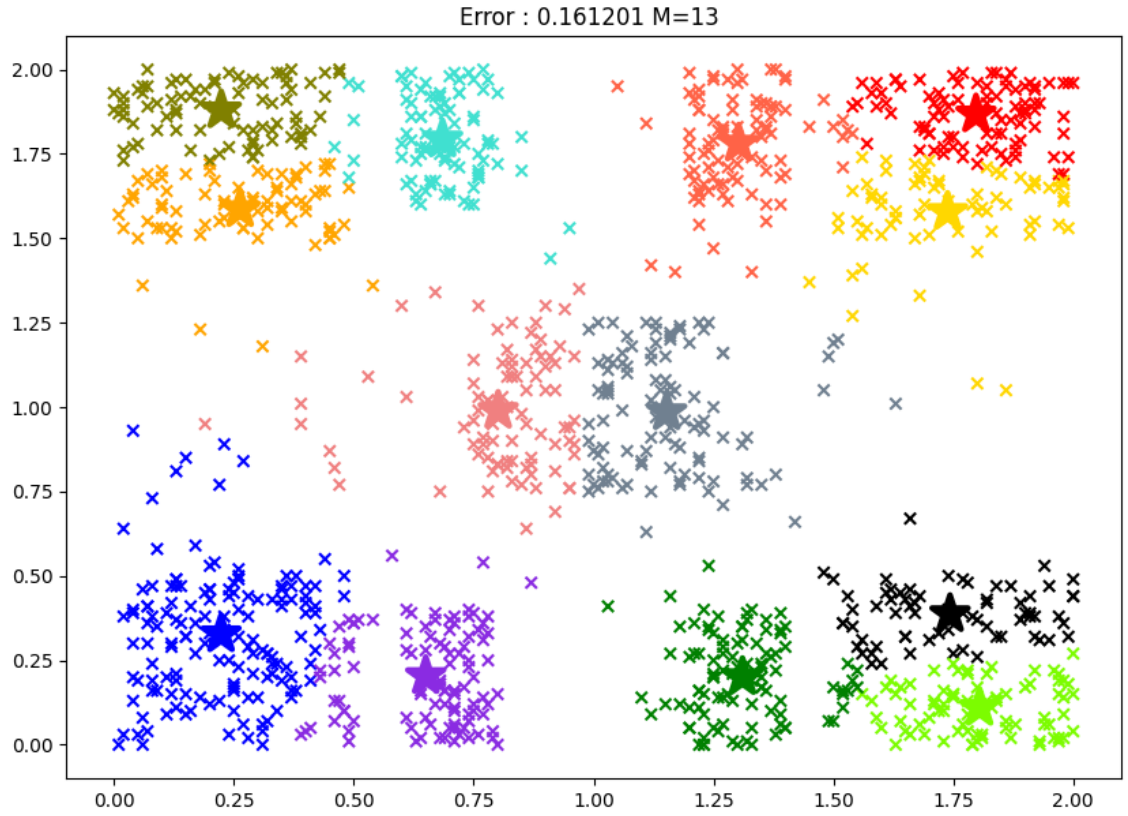
ΓΙΑ Μ = 9:



ΓΙΑ Μ = 11:



ΓΙΑ M=13:



Γίνεται κατανοητό πως δεν έχουν γίνει καινούργιες ομάδες,αλλά έχουν σπάσει ορισμένες σε μικρότερες υπο-ομάδες για M>=9, με αποτέλεσμα να μπορούμε να συμπεράνουμε ότι M=9.