Neural Networks Project 2022 - Question 2 Report

Kondylia Vergou, AM 4325

Panagiotis Vouzalis, AM 2653

Η συγγραφη του κωδικα εγινε σε περιβαλλον Windows

- Για να λειτουργησουν οι εντολες που αναφερονται παρακατω σε περιβαλλον Linux πρεπει να αντικατασταθει το < ; > με < : >
- Τα dependencies της εφαρμογης μας βρισκονται στο φακελο `/jars`
- Αναλογα με τη διαρυθμιση της πλατφορμας που θα τρεξει η εφαρμογη, μπορει να χρειαστει να προστεθει στο τελος του αρχειου προς εκτελεση το επιθεμα `.java` αλλιως πιθανως θα εμφανιζεται ενα σφαλμα τυπου `error: class not found on application class path: <.java file name>`

Instructions for Windows Platforms

1. Open terminal inside the directory `Neural Networks - Question 2`
2. Compile with: `javac -cp ".;jars/*" main/*.java`
3. Create a new dataset with: `java -cp ".;jars/*" main/DatasetGenerator` - This is not necessary
4. Run java application with: `java -cp ".;jars/*" main/Kmeans <dataset_name.txt> <number_of_clusters> <number_of_runs>`

- example: `java -cp ".;jars/*" main/Kmeans dataset.txt 9 20`

Instructions for Linux Platforms

1. Open terminal inside the directory `Neural Networks - Question 2`
2. Compile with: `javac -cp ".:jars/*" main/*.java`
3. Create a new dataset with: `java -cp ".:jars/*" main/DatasetGenerator` - This is not necessary
4. Run java application with: `java -cp ".:jars/*" main/Kmeans <dataset_name.txt> <number_of_clusters> <number_of_runs>`

- example: `java -cp ".:jars/*" main/Kmeans dataset.txt 9 20`

Για τις αναγκες της δευτερης ασκησης δημιουργησαμε το dataset.txt (παρεχεται και το dataset.csv) συμφωνα με τις απαιτησεις του ερωτηματος Σ2.
Ο κωδικας για τη δημιουργια του dataset βρισκεται στο αρχειο `/main/DatasetGenerator.java`

Code overview: /main/Kmeans.java

- Μ ομαδες - το Μ οριζεται απο το χρηστη:
`java -cp ".;jars/*" main/Kmeans.java <dataset_name.txt> <number_of_clusters> <number_of_runs>`

```java
public static void main (String[] args) {

    numberOfClusters = Integer.valueOf(args[1]);
```

- Φορτωση αρχειου παραδειγματων:

```java
// usage: use argument "new" to create a new dataset or use argument "<dataset_name.txt>" to import an existing .txt dataset
DatasetGenerator dataGen = new DatasetGenerator(args[0]);

//DatasetGenerator dataGen = new DatasetGenerator("new"); // use if you want to create a new dataset
//DatasetGenerator dataGen = new DatasetGenerator("<dataset_name.txt>"); // use if you want to import an existing dataset

dataset = dataGen.getDataset();
//dataGen.exportDatasetCSV();
//dataGen.exportDatasetTXT();
```

- Data initialization - τυχαια επιλογη αρχικων centroids:

```java
// choose the center points of our clusters at random
private static void initializeData() {

    ArrayList<Integer> indexesOfCenterPoints = new ArrayList<>();
    int indexOfCenterPoint;

    for (int i = 0; i < numberOfClusters; i++) {

        ArrayList<Point> cluster = new ArrayList<>();
        clusters.add(cluster);
        indexOfCenterPoint = randomGenerator.nextInt(dataset.size());

        // ensure we dont pick the same center point twice
        while (indexesOfCenterPoints.contains(indexOfCenterPoint)) {

            indexOfCenterPoint = randomGenerator.nextInt(dataset.size());
        }

        indexesOfCenterPoints.add(indexOfCenterPoint);
        centerPoints.add(dataset.get(indexOfCenterPoint));
    }
}
```

- Υπολογισμος του σφαλματος ομαδοποιησης (ευκλειδια αποσταση) για καθε point και τοποθετηση του point στο συνολο με τη μικροτερη αποσταση απο το κεντρο της αντιστοιχης ομαδας:

```java
do {

    // compute euclidean distance between a point and each cluster center, for every point
    for (Point point : dataset) {

        ArrayList<Float> distancesFromCenters = new ArrayList<>();

        // compute euclidean distance between a point and each cluster center
        for (Point centerPoint: centerPoints) {

            float distanceFromCluster = (float) Math.sqrt(Math.pow((point.getX() - centerPoint.getX()), 2) + Math.pow((point.getY() - centerPoint.getY()), 2));
            distancesFromCenters.add(distanceFromCluster);
        }

        // add the point to the cluster that minimizes the distance from its respective cluster center
        clusters.get(distancesFromCenters.indexOf(Collections.min(distancesFromCenters))).add(point);
        pointFromCenterDistances.get(distancesFromCenters.indexOf(Collections.min(distancesFromCenters))).add(Collections.min(distancesFromCenters));
    }
```

- Υπολογισμος των νεων κεντρων ως προς το μεσο ορο των στοιχειων καθε cluster:

```java
// set the new cluster center with respect to the mean value of the cluster points, for every cluster
for (ArrayList<Point> cluster : clusters) {

    int clusterIndex = clusters.indexOf(cluster);
    //System.out.println("\nCluster " + clusterIndex + ": Center = " + centerPoints.get(clusterIndex).toString());
    myWriter.write("\nCluster " + clusterIndex + ": Center = " + centerPoints.get(clusterIndex).toString() + "\n");

    float sumX = 0;
    float sumY = 0;

    int displayLimit = 0;
    for (Point point : cluster) {

        //System.out.print(point.toString() + " ");
        myWriter.write(point.toString() + " ");
        sumX += point.getX();
        sumY += point.getY();

        displayLimit++;
        if (displayLimit == 10) {

            myWriter.write("\n");
            displayLimit = 0;
        }

    }
    //System.out.println();
    //System.out.println("SumX of cluster " + clusterIndex + ": " + sumX);
    //System.out.println("SumY of cluster " + clusterIndex + ": " + sumY);
    //System.out.println("Average of cluster " + clusterIndex + ": " + sumX / cluster.size() + "," + sumY / cluster.size());

    myWriter.write("\n");
    //myWriter.write("SumX of cluster " + clusterIndex + ": " + sumX + "\n");
    //myWriter.write("SumY of cluster " + clusterIndex + ": " + sumY + "\n");
    //myWriter.write("Average of cluster " + clusterIndex + ": " + sumX / cluster.size() + "," + sumY / cluster.size() + "\n");


    // set the new cluster center
    centerPoints.set(clusterIndex, new Point(sumX / cluster.size(), sumY / cluster.size()));
    //System.out.println("Cluster " + clusterIndex + ": New center = " + centerPoints.get(clusterIndex).toString());
    myWriter.write("Cluster " + clusterIndex + ": New center = " + centerPoints.get(clusterIndex).toString() + "\n");
}
```

- Ελεγχος τερματισμου βασει της συνθηκης "Εχουν μεταβληθει τα διανυσματα w_j μεταξυ 2 επαναληψεων?":

```java
// are the w_j vectors different between 2 runs?
// if yes then start a new iteration
// if no then end the run
for (int k = 0; k < pointFromCenterDistances.size(); k++) {

    for (int l = 0; l < pointFromCenterDistances.get(k).size(); l++) {

        if (previousPointFromCenterDistances.get(k).get(l).equals(pointFromCenterDistances.get(k).get(l))) {

            areWvectorsDifferent = false;
            break;
        }
    }

    if (!areWvectorsDifferent) {

        break;
    }
}

// if the w_j vectors are different then prepare for a new iteration
if (areWvectorsDifferent) {

    previousPointFromCenterDistances = pointFromCenterDistances;
}

clusters_of_every_run.add(clusters);
centers_of_every_run.add(centerPoints);

// clear the clusters after every iteration
if (i != (numberOfRuns-1)) {

    for (ArrayList<Point> cluster : clusters) {

        cluster.clear();
    }
}

// if the w_j vectors are different between 2 runs then start a new iteration, else end
} while (areWvectorsDifferent);
```

- Υπολογισμος του συνολικου σφαλματος ομαδοποιησης (ευκλειδια αποσταση) για ολα τα clusters:

```java
// compute the total dispersion by adding the dispersions of each cluster
float totalDispersion = 0;
for (int d = 0; d < clusters.size(); d++) {

    float clusterDispersion = 0;
    for (Float clusterDistances : pointFromCenterDistances.get(d)) {
        clusterDispersion += clusterDistances;
    }

    totalDispersion += clusterDispersion;
}

// ArrayList<Float> totalDispersions holds the total cluster dispersion for each run
totalDispersions.add(totalDispersion);

//System.out.println("\nRun " + (i+1) + " dispersion = " + totalDispersion);
myWriter.write("\nRun " + (i+1) + " dispersion = " + totalDispersion + "\n");
```

- 20 runs και αποθηκευση της λυσης με το ελαχιστο σφαλμα ομαδοποιησης:
`java -cp ".;jars/*" main/Kmeans.java <dataset_name.txt> <number_of_clusters> <number_of_runs>`

```java
// keep the run with the minimum dispersion
if (totalDispersion < MIN_DISPERSION) {

    //System.out.print((i+1) + " " + totalDispersion + " " + MIN_DISPERSION + "\n");
    MIN_DISPERSION = totalDispersion;
    run_of_MIN_DISPERSION = (i+1);
    clusters_MIN_DISPERSION = clusters_of_every_run.get(i);
    centerPoints_MIN_DISPERSION = centers_of_every_run.get(i);
}
}
```

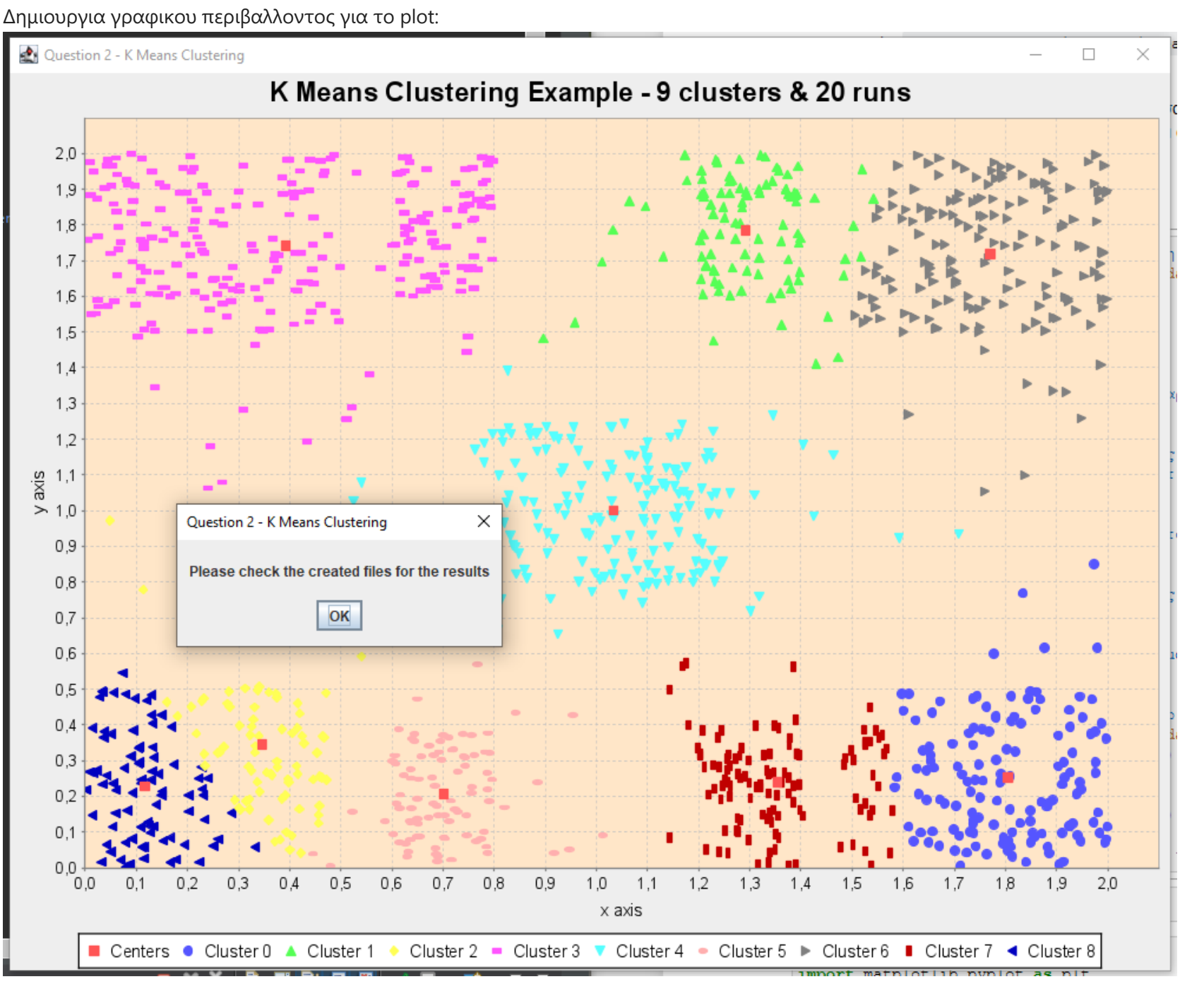
- Αποθηκευση τελικων συντεταγμενων των centroids και του σφαλματος ομαδοποιησης στο αρχειο final_centroids_dispersions (παραδειγμα για 9 clusters και 20 runs):

```
---------- K means clustering example ----------
Number of clusters = 9
Number of runs = 20

---------- Final centroid coordinates ----------
Cluster 1 center = 1.0416359,0.986985
Cluster 2 center = 1.6022335,0.24239336
Cluster 3 center = 1.2923727,1.799663
Cluster 4 center = 1.7235991,1.5450246
Cluster 5 center = 0.64948756,1.8053868
Cluster 6 center = 0.3978845,0.25267282
Cluster 7 center = 0.39896592,1.3871365
Cluster 8 center = 0.17588888,1.7561957
Cluster 9 center = 1.7966483,1.8517332

---------- Dispersions across all runs ----------
Run 1 dispersion = 540.8662
Run 2 dispersion = 384.42502
Run 3 dispersion = 288.98016
Run 4 dispersion = 261.02002
Run 5 dispersion = 259.6171
Run 6 dispersion = 259.16824
Run 7 dispersion = 258.86148
Run 8 dispersion = 258.8769
Run 9 dispersion = 258.70828
Run 10 dispersion = 258.57513
Run 11 dispersion = 258.53192
Run 12 dispersion = 258.57935
Run 13 dispersion = 258.6511
Run 14 dispersion = 258.70074
Run 15 dispersion = 258.77325
Run 16 dispersion = 258.51773
Run 17 dispersion = 258.43878
Run 18 dispersion = 258.0747
Run 19 dispersion = 257.83115
Run 20 dispersion = 257.83008

Minimum dipersion occured on run 20 = 257.83008
```

- Δημιουργια plot μεσω του jFreeChart:

```java
// panel generation
public Kmeans(String windowTitle, int numberOFClusters, int numberOfRuns, String dirString) {

    super(windowTitle);

    // create jFreeChart dataset
    XYDataset dataset = createJFreeChartDataset();

    // create scatter plot
    String plotName = "K Means Clustering Example - " + numberOFClusters + " clusters & " + numberOfRuns + " runs";
    JFreeChart chart = ChartFactory.createScatterPlot(
            plotName, "x axis", "y axis", dataset, PlotOrientation.VERTICAL, true, true, false);

    // change background color
    XYPlot plot = (XYPlot)chart.getPlot();
    plot.setBackgroundPaint(new Color(255, 230, 200));

    // create panel
    ChartPanel panel = new ChartPanel(chart);
    setContentPane(panel);

    // export plot to png
    try {

        File imageFile = new File("./" + dirString + "/" + numberOFClusters + "clusters_ " + numberOfRuns + "runs.png");
        OutputStream out = new BufferedOutputStream(new FileOutputStream(imageFile));
        BufferedImage image = chart.createBufferedImage(1000, 800);
        ChartUtilities.writeBufferedImageAsPNG(out, image);

    } catch (IOException e) {

        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}

// jFreeChart dataset generation
private XYDataset createJFreeChartDataset() {

    XYSeriesCollection dataset = new XYSeriesCollection();

    XYSeries centerSeries = new XYSeries("Centers");

    for (Point point : centerPoints_MIN_DISPERSION) {

        centerSeries.add(point.getX(), point.getY());
    }

    dataset.addSeries(centerSeries);

    int i = 0;
    for (ArrayList<Point> cluster : clusters_MIN_DISPERSION) {

        XYSeries clusterSeries = new XYSeries("Cluster " + i);
        for (Point point : cluster) {

            clusterSeries.add(point.getX(), point.getY());
        }

        i++;
        dataset.addSeries(clusterSeries);
    }

    return dataset;
}
```

- Δημιουργια γραφικου περιβαλλοντος για το plot:



- Αποθηκευση ολων των παραγομενων αρχειων για καθε στο root directory της εφαρμογης, καθε run στο δικο του directory:

Desktop › Neural Networks - Question 2

Name
.ipynb_checkpoints
jars
main
output_3clusters_20runs
output_5clusters_20runs
output_7clusters_20runs
output_9clusters_20runs
output_11clusters_20runs
output_13clusters_20runs
output_15clusters_20runs
output_17clusters_20runs
output_19clusters_20runs
output_21clusters_20runs
dataset.csv
dataset.txt
dispersionAgainstClusters.txt
Neural Networks - Question 2.ipynb
README.txt
S2.jpg
YN_askhsh2021_22.pdf

- Παραδειγμα output αρχειων για 9 clusters και 20 runs:
  - Στα αρχεια details_runXX.txt παρεχονται τα centroids, τα clusters, και η ελαχιστη διασπορα για καθε run
  - Αποθηκευση του plot ως .png στο directory με τα outputs

Desktop › Neural Networks - Question 2 › output_9clusters_20runs

Name	Date modified
9clusters_ 20runs.png	03/01/2022 03:37
details_run1.txt	03/01/2022 03:37
details_run2.txt	03/01/2022 03:37
details_run3.txt	03/01/2022 03:37
details_run4.txt	03/01/2022 03:37
details_run5.txt	03/01/2022 03:37
details_run6.txt	03/01/2022 03:37
details_run7.txt	03/01/2022 03:37
details_run8.txt	03/01/2022 03:37
details_run9.txt	03/01/2022 03:37
details_run10.txt	03/01/2022 03:37
details_run11.txt	03/01/2022 03:37
details_run12.txt	03/01/2022 03:37
details_run13.txt	03/01/2022 03:37
details_run14.txt	03/01/2022 03:37
details_run15.txt	03/01/2022 03:37
details_run16.txt	03/01/2022 03:37
details_run17.txt	03/01/2022 03:37
details_run18.txt	03/01/2022 03:37
details_run19.txt	03/01/2022 03:37
details_run20.txt	03/01/2022 03:37
final_centroids_dispersions.txt	03/01/2022 03:37
pd_minDispersion_9clusters_20runs.txt	03/01/2022 03:37

Μεταβολη του σφαλματος ομαδοποιησης αναλογα τον αριθμο των clusters

In [1]:

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('dispersionAgainstClusters.txt')
df
```

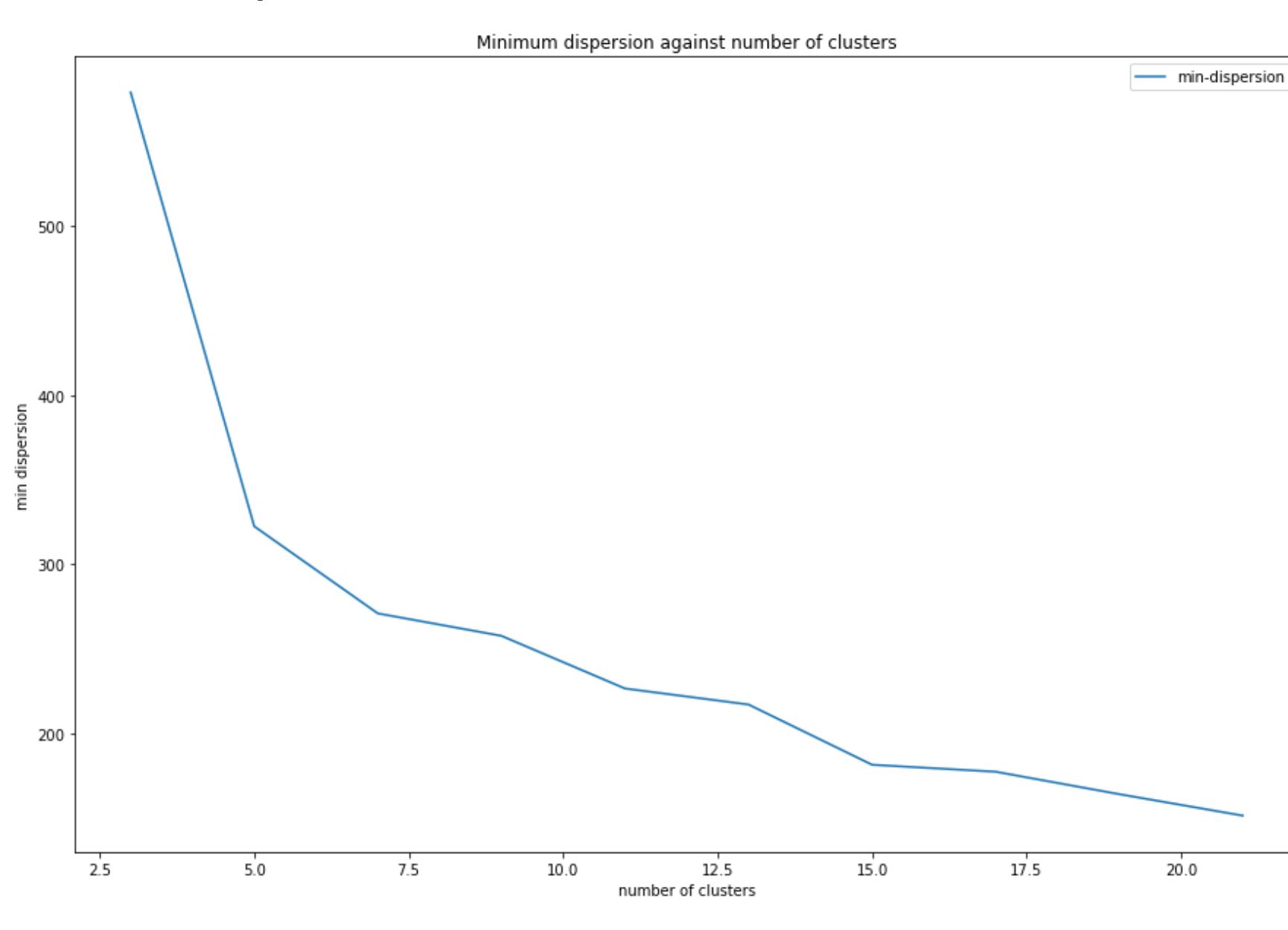
Out[1]:

	clusters	min-dispersion
0	3	578.31714
1	5	322.40290
2	7	271.04608
3	9	257.83008
4	11	226.77328
5	13	217.31060
6	15	181.79158
7	17	177.68852
8	19	164.44003
9	21	151.79976

In [2]:

```python
# line plot of min dispersion against clusters
ax = df.plot(x = 'clusters', y = 'min-dispersion', figsize = (15, 10))
ax.set_title('Minimum dispersion against number of clusters')
ax.set_xlabel("number of clusters")
ax.set_ylabel("min dispersion")
```

Out[2]: `Text(0, 0.5, 'min dispersion')`



Μπορει να χρησιμοποιηθει το σφαλμα ομαδοποιησης για να εκτιμησουμε τον πραγματικο αριθμο των cluster?

Σε γενικες γραμμες ειναι δυνατον να **εκτιμησουμε** τον αριθμο των cluster εαν κανουμε plot το σφαλμα ομαδοποιησης ως προς τις διαφορετικες τιμες του M (αριθμος cluster).

Συνηθως, οσο ο αριθμος M αυξανεται, η αποσταση μεταξυ των points σε καθε cluster μειωνεται.
Ωστοσο δεν μπορουμε να διαλεξουμε μια αυθαιρετα μεγαλη τιμη του M, διοτι ετσι καταληγουμε σε ενα απο δυο πιθανα σεναρια:

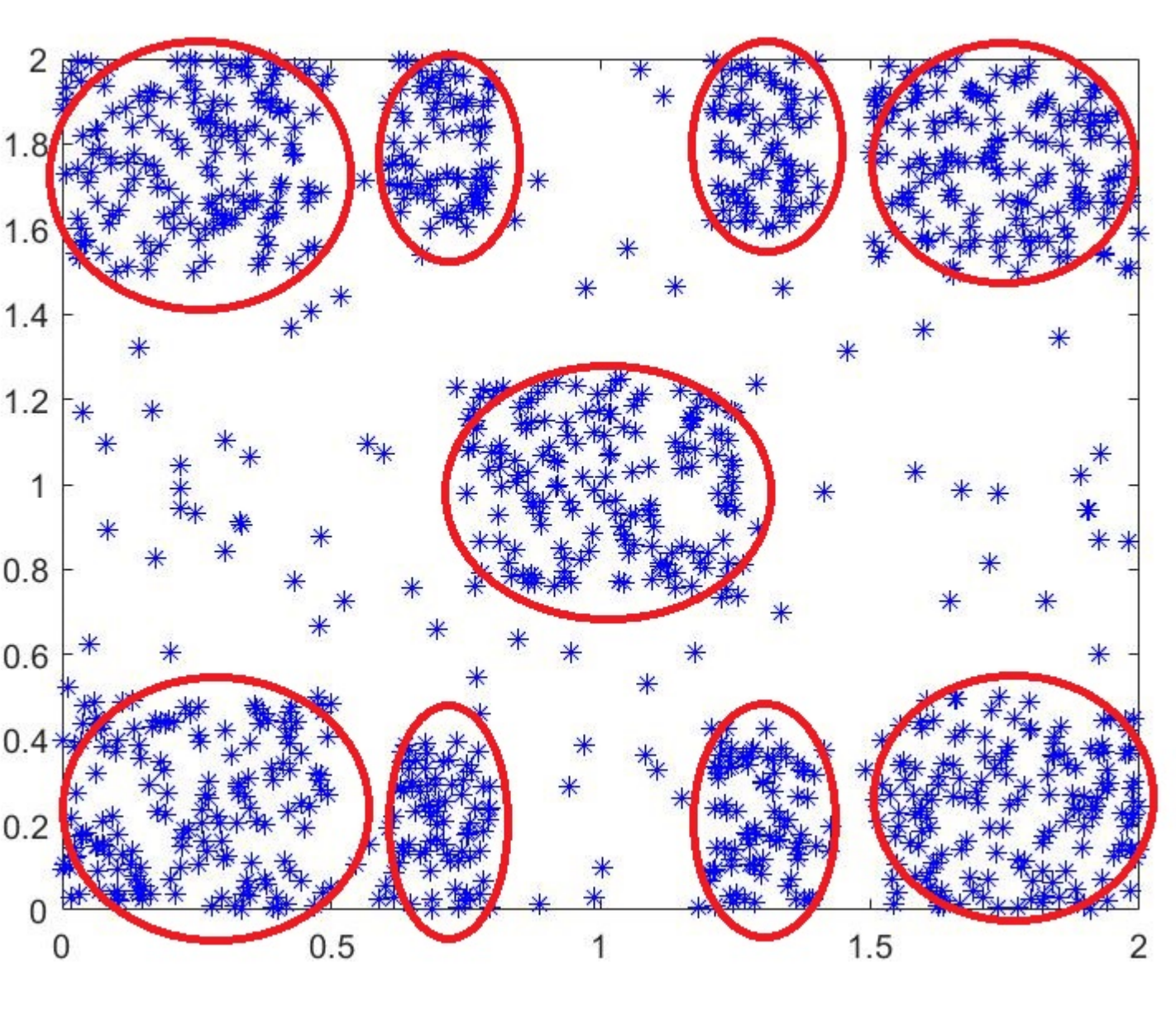
1. Ειτε θα υπαρχουν πολλα μικρα cluster με λιγα points
2. Ειτε θα υπαρχουν πολλα cluster με ενα μοναχα point.

Either way, η ολη ιδεα της "ομαδοποιησης" αποδυναμωνεται οσο μεγαλωνει το M, για αυτο το λογο θα πρεπει να διαλεξουμε μια "λογικη" τιμη για το M.
Ενας τροπος για να το κανουμε αυτο ειναι να χρησιμοποιησουμε την ιδεα του "γονατου" στο plot και να επιλεξουμε μια τιμη για την οποια το σφαλμα ομαδοποιησης μειωνεται δραματικα.

Στο παραπανω plot παρατηρουμε ενα γονατο στο διαστημα $M = 5$ εως $M = 9$.
Ειδικοτερα, στην τιμη $M = 5$ η τιμη του σφαλματος μειωνεται εντονα.
Στην τιμη $M = 9$ μειωνεται λιγοτερο εντονα.
Στην τιμη $M = 7$ δεν μειωνεται τοσο εντονα οσο στις αλλες δυο.

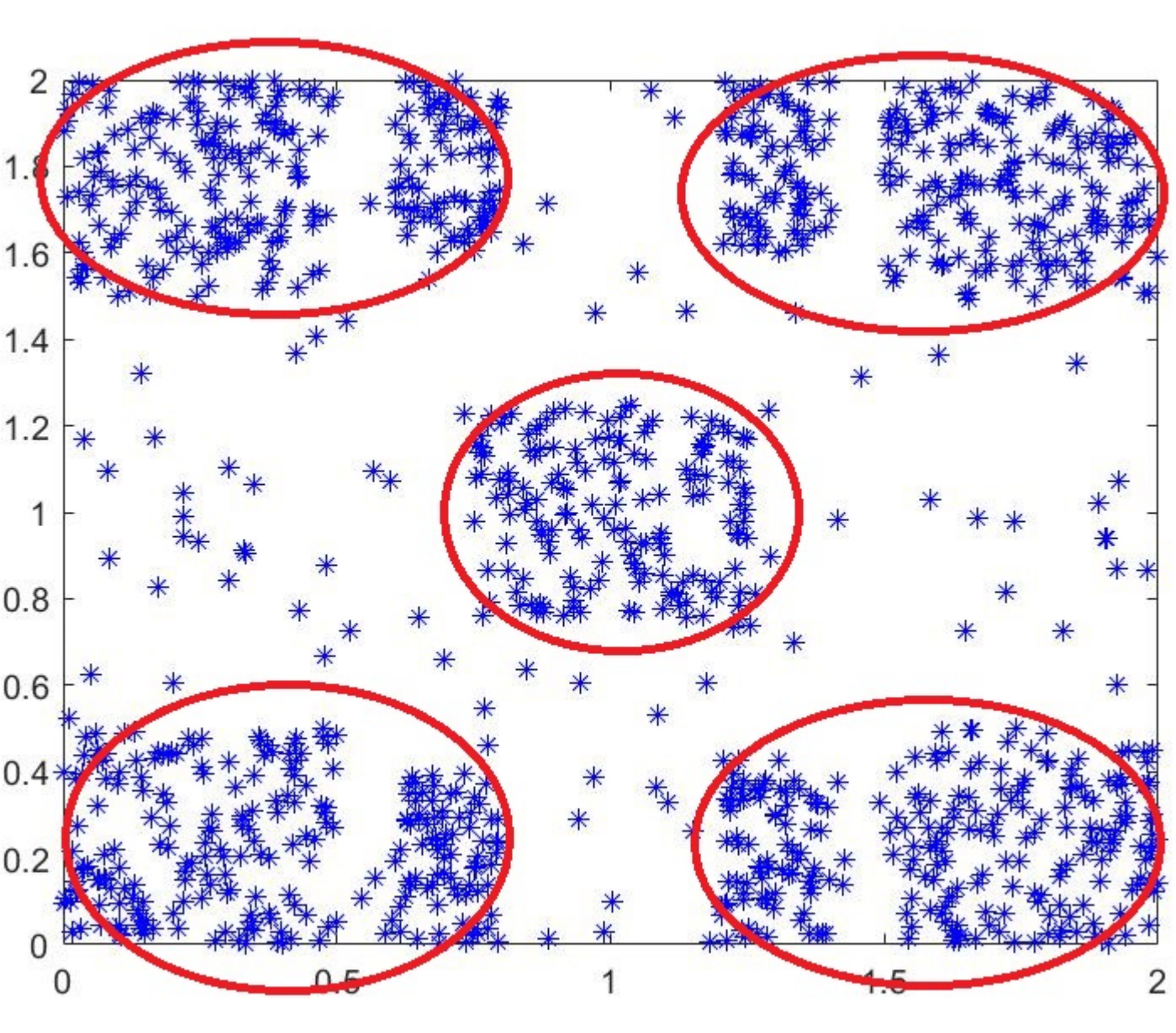
Αρα οποιαδηποτε απο αυτες τις τρεις τιμες μπορει να χρησιμοποιηθει για την εκτιμηση του αριθμου των clusters, με πιθανοτερες επιλογες ειτε την τιμη $M = 5$ ειτε την τιμη $M = 9$.

Εαν **δεν** ειχαμε την πρωτερη γνωση πως ο πραγματικος αριθμος των clusters ειναι 9, **τοτε πιθανοτατα θα επιλεγαμε την τιμη 5** λογω της εντονης πτωσης του σφαλματος.
Η εκφωνηση μας λεει πως ο πραγματικος αριθμος των clusters ειναι 9. Ας το ερευνησουμε:

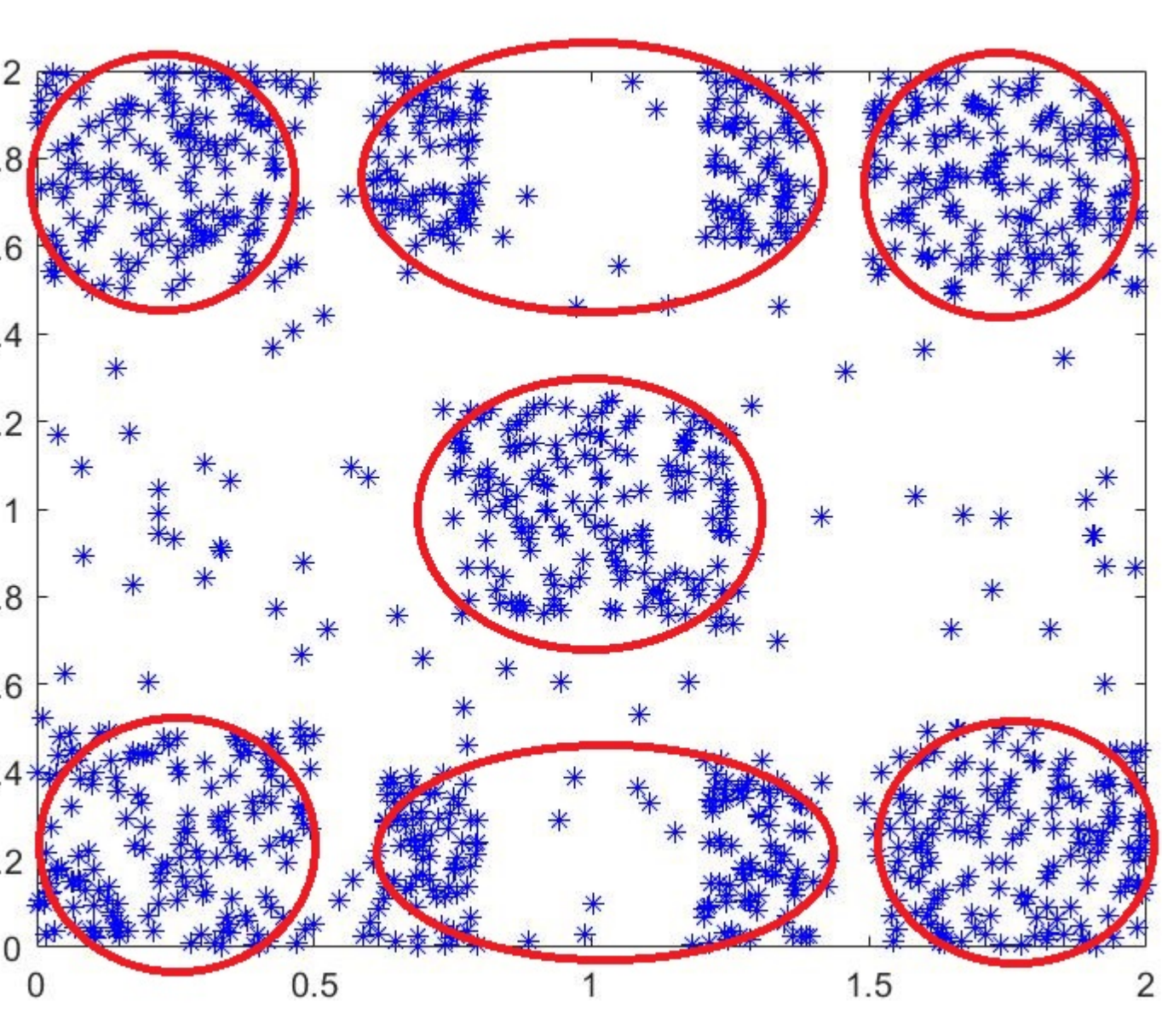


Παρατηρουμε πως υπαρχουν πραγματι 9 clusters στο dataset.

Επιχειρηματολογωντας για την ιδεα της υπαρξης 5 cluster, θα μπορουσε κανεις να πει πως υπαρχουν 5 cluster και να παρουσιασει το παρακατω plot:



Με τον ιδιο τροπο, θα μπορουσε κανεις να πει επισης πως υπαρχουν 7 clusters, παρουσιαζοντας το παρακατω plot:



Μονο και μονο παρατηρωντας το παραπανω plot μπορουμε να πουμε με σιγουρια πως ο αριθμος 7 δεν ειναι καλη εκτιμηση του αριθμου των cluster.

Conclusion

Εαν δε γνωριζουμε τον πραγματικο αριθμο των cluster τοτε μια πολυ καλη εκτιμηση αυτου του αριθμου ειναι να επιλεξουμε ειτε την τιμη $M = 5$ ειτε την τιμη $M = 9$ που ειναι και οι τιμες στα ακρα του "γονατου".

Απο τα παραπανω plots βλεπουμε πως και οι δυο τιμες ειναι "σωστες", ομως απο την εκφωνηση γνωριζουμε εκ των προτερων πως η τιμη $M = 9$ ειναι η περισσοτερο σωστη. Εν τελει, ολα εξαρτωνται απο το ποσο ελαστικοι ειμαστε οσον αφορα την "ομοιογενεια" των cluster που θελουμε να φτιαξουμε (μικροτερη ομοιογενεια $\rightarrow$ 5 clusters, μεγαλυτερη ομοιογενεια $\rightarrow$ 9 clusters).