MachineLearning_PA#3_Assignment#2_Fuzzy_C-Means

November 30, 2021

Dataset: Cluster Dataset In this assignment we have to implement standard version of Fuzzy c-means(FCM)

Importing the necessary libraries

[1]:
```
import numpy as np
import matplotlib.pyplot as plot
import matplotlib.animation as animation
import random
import pickle
from math import sqrt
```

ggplot style sheet

[2]:
```
plot.style.use('ggplot')
```

Fuzzy CMeans

[3]:
```
class FuzzyCMeans(object):
    centroids = []
    assgns = []
    clusterMeans = []
    bestCentroids = []
    bestAssgns = []
    worstCentroids = []
    worstAssgns = []
    worstSquareErrors = 0
    allSquareErrors = []
    bestSquareErrors = 10000
    bestFound = 0
    m = 2
    currentSquareError = 0
    epsilon = .001
    plotcolors = ['r', 'g', 'b', 'm', 'c', 'k', 'y','b',
                  'g', 'r', 'c', 'm', 'y', 'k','b', 'g',
                  'r', 'c', 'm', 'y', 'k','b', 'g', 'r',
                  'c', 'y', 'm', 'k','b', 'g', 'r', 'c',
                  'm', 'y', 'k','b', 'g', 'r', 'c', 'm',
                  'y', 'b','k', 'g', 'r', 'c', 'm', 'y',
```

```
                'k','g']
    def squareErrors(self):
        print("All squareErrors sums:")
        print(self.allSquareErrors)
        print("Best squareError sum:")
        print(self.bestSquareErrors)

    def assessSquareErrors(self):
        sqError = 0
        self.allSquareErrors.append(self.currentSquareError)
        if self.currentSquareError < self.bestSquareErrors:
          self.bestFound +=1
          print("New best square error:", self.bestFound)
          self.bestSquareErrors = self.currentSquareError
          self.bestCentroids = self.centroids
          self.bestAssgns = self.assgns

    def constRandomWeights(self, vectors, clusters):
        self.assgns = []
        for i in range(len(vectors)):
          x = random.sample(range(0,1000),clusters)
          x = np.array(x)
          sums = x.sum()
          x = x/sums
          self.assgns.append(list(x))
    #random weights
    def randomizeWeights(self, vectors, clusters):
        self.assgns = []
        k = 0
        arr = []
        cur_sum = 0
        cost = 0
        distances = []
        for i in range(len(vectors)):
          for j in range(clusters):
            cost = self.costFunction(vectors[i],self.centroids[j])
            distances.append(cost)
            cur_sum += cost
          for j in range(clusters):
            k = distances[j]/cur_sum
            arr.append(k)
          self.assgns.append(arr)
          cur_sum = 0
          arr = []
    #calculate the centroids
    def calcCentroids(self,vectors,clusters):
        xC = 0
```

```
        yC = 0
        memberWeight = 0
        x1 = 0
        y1 = 0
        x2 = 0
        y2 = 0
        for i in range(clusters):
          for j in range(len(vectors)):
            memberWeight = self.assgns[j][i]**self.m
            if memberWeight == 0:
              memberWeight = 0.00001
            x1 += (memberWeight * vectors[j][0])
            y1 += (memberWeight * vectors[j][1])
            x2 += memberWeight
            y2 += memberWeight
          xC = x1/x2
          yC = y1/y2
          self.centroids[i] = [xC,yC]
          x1 = 0
          y1 = 0
          x2 = 0
          y2 = 0
    # calculate teh membership
    def calcMembership(self,vectors,clusters):
        stop = False
        numeratorCost = 0
        denominatorCost = 0
        cur_sum = 0
        temp = self.currentSquareError
        self.currentSquareError = 0
        for i in range(len(vectors)):
          for j in range(clusters):
            numeratorCost = self.costFunction(vectors[i], self.centroids[j])
            for k in range(clusters):
              denominatorCost = self.costFunction(vectors[i], self.centroids[k])
              if denominatorCost == 0:
                denominatorCost = 0.00001
              cur_sum += (numeratorCost/denominatorCost)**(2/(self.m-1))
            if cur_sum == 0:
              cur_sum = 0.00001

            self.assgns[i][j] = 1/cur_sum
            self.currentSquareError += (self.assgns[i][j] * numeratorCost)
            cur_sum = 0
        if (temp - self.currentSquareError <= self.epsilon):
           stop = True
        return stop
```

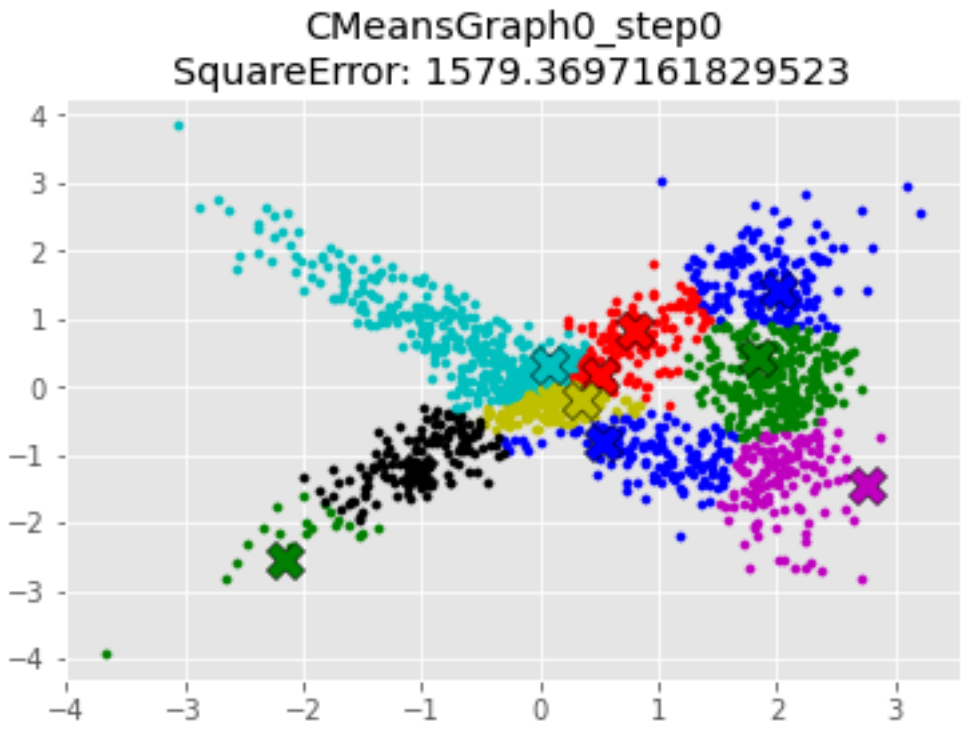
```
    def costFunction(self, point, centroid):
        return sqrt((point[0]-centroid[0])**2 + (point[1]-centroid[1])**2)

    #select the centroids
    def selectCentroids(self, centroids, upperBound, vectors):
        self.centroids = []
        indexes = [] # make an array of random indices
        indexes = random.sample(range(0,upperBound), centroids)
        for i in indexes:
            self.centroids.append(vectors[i])
    #garph
    def showGraph(self, vectors, steps, iteration):
        vectors = np.array(vectors)
        localCentroids = np.array(self.centroids)
        x,y = vectors.T
        figure = plot.figure()
        fig = figure.add_subplot(1,1,1)
        count = 0
        for num in range(len(vectors)):
            x,y = vectors[num].T
            highest = 0
            for i in range(len(self.centroids)):
              if self.assgns[num][i] > highest:
                highest = self.assgns[num][i]
                color = i
            fig.scatter(x,y, s = 10, c = self.plotcolors[color])
        for k in localCentroids:
            x,y = k.T
            fig.scatter(x,y, s=200, c=self.plotcolors[count], marker='X',␣
↪edgecolors='k')
            count = count+1
        plot.
↪title("CMeansGraph"+str(iteration)+"_step"+str(steps)+"\n"+"SquareError: " +␣
↪str(self.currentSquareError))
        plot.savefig("CMeansGraph"+str(iteration)+"_step"+str(steps)+".png")
        plot.show()
        fig.clear()
        return figure
    #best Results
    def bestResults(self):
       self.centroids = self.bestCentroids
       self.assgns = self.bestAssgns

    #Cmeans Classifier
    def CMeansClassifier(self, iters, clusters, vectors, steps):
        for i in range(iters):
```

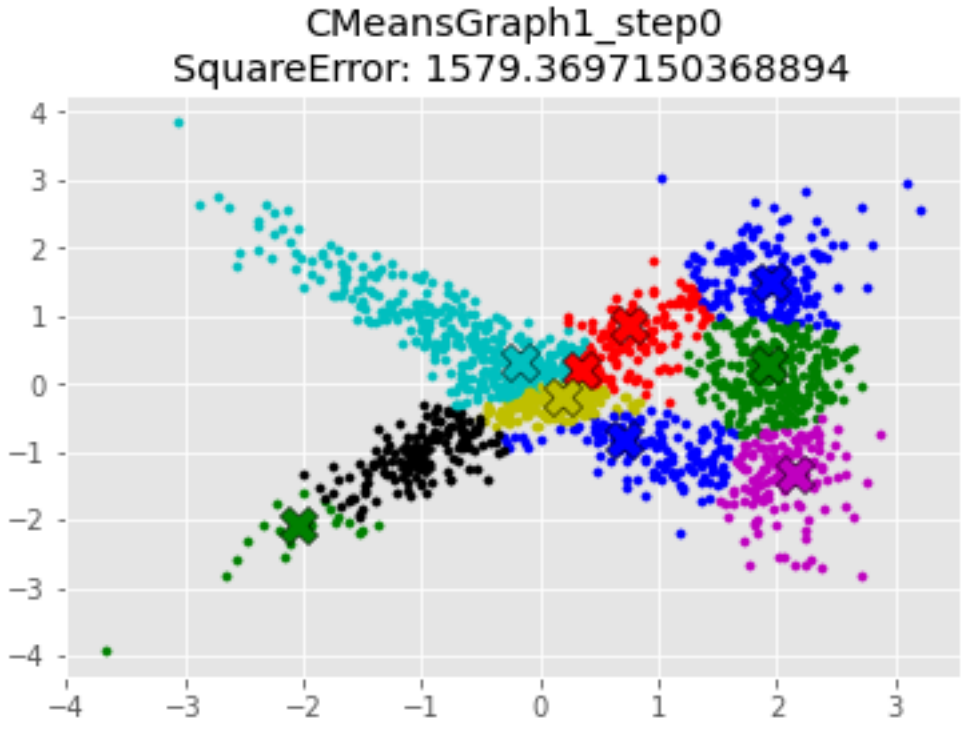
```
            stop = False
            self.selectCentroids(clusters, len(vectors), vectors)
            self.randomizeWeights(vectors, clusters)
            for j in range(steps):
                stop = False
                stop = self.calcMembership(vectors, clusters)
                self.calcCentroids(vectors,clusters)
                graph = self.showGraph(vectors, i, j)
                if(stop == True and j >= 25):
                    print("Epsilon reached, stopping early")
                    break
            self.assessSquareErrors()
        self.squareErrors()
        self.bestResults()

        print("The best graph is:")
        graph = self.showGraph(vectors, 0, "best")
        return graph
```

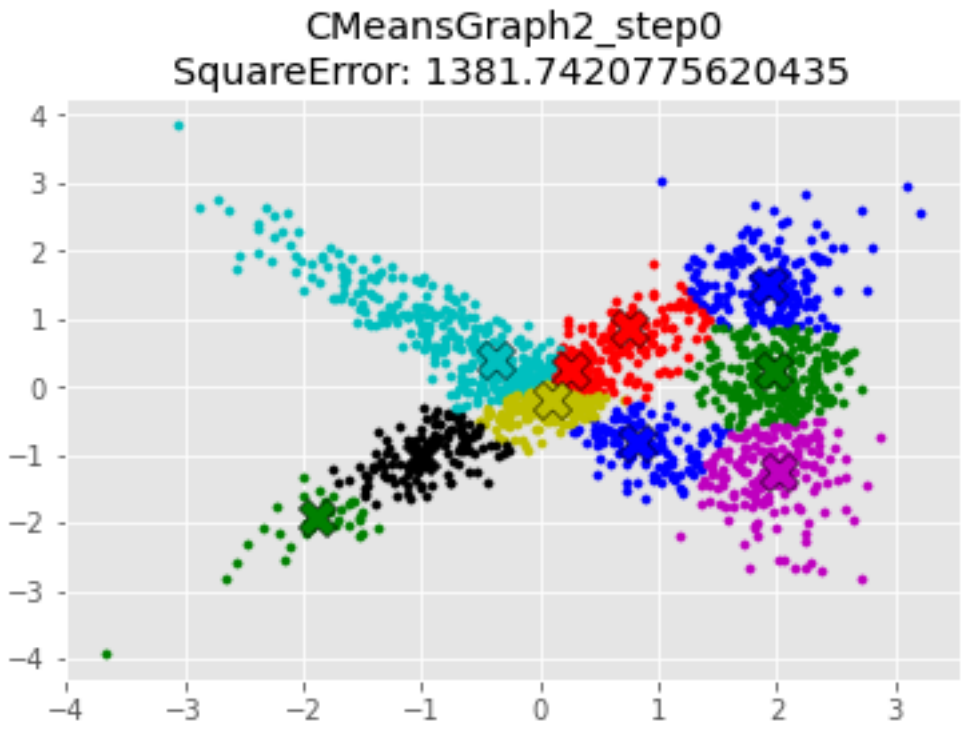
[4]:
```
#define the main class
def main():
    classifier = FuzzyCMeans()
    data = open("/Users/sriharshithaayyalasomayajula/Desktop/Machine␣
 ↪Learning_PSU/Program_3/Dataset/cluster_dataset.txt",'r')
    data = data.readlines()
    graphs = []
    vectors = []
    for i in data:
        vectors.append(i.split())
        for i in range(len(vectors)):
            for j in range(len(vectors[i])):
                vectors[i][j] = float(vectors[i][j])

    clusters = input("How many clusters do you want to generate? (Choose␣
 ↪between 1-50?)")
    iterations = input("How many iterations do you want to run?")

    steps = 100
    graph = classifier.
 ↪CMeansClassifier(int(iterations),int(clusters),vectors,steps)
```

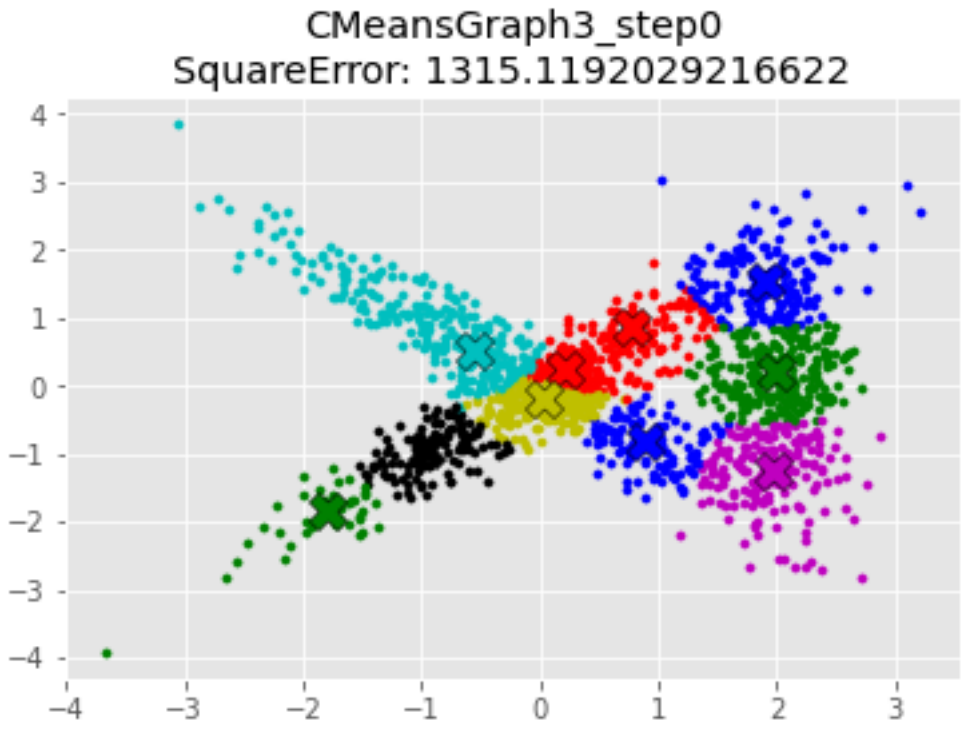
[5]:
```
#Calling the main fucntion
main()
```

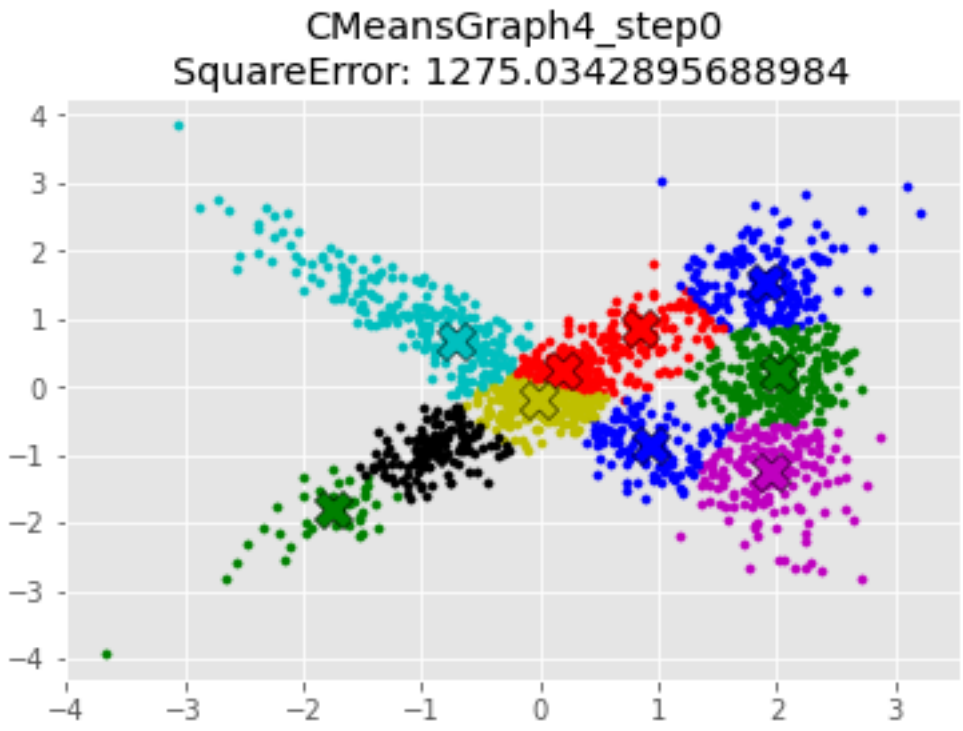
```
How many clusters do you want to generate? (Choose between 1-50?)10
How many iterations do you want to run?2
```

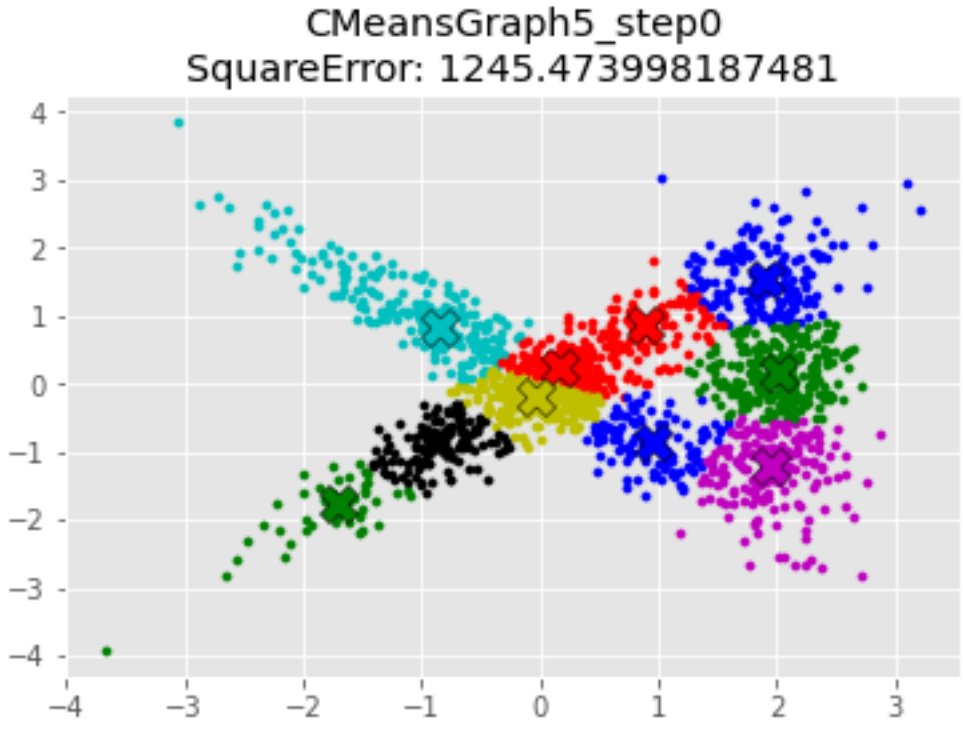

CMeansGraph0_step0
SquareError: 1579.3697161829523

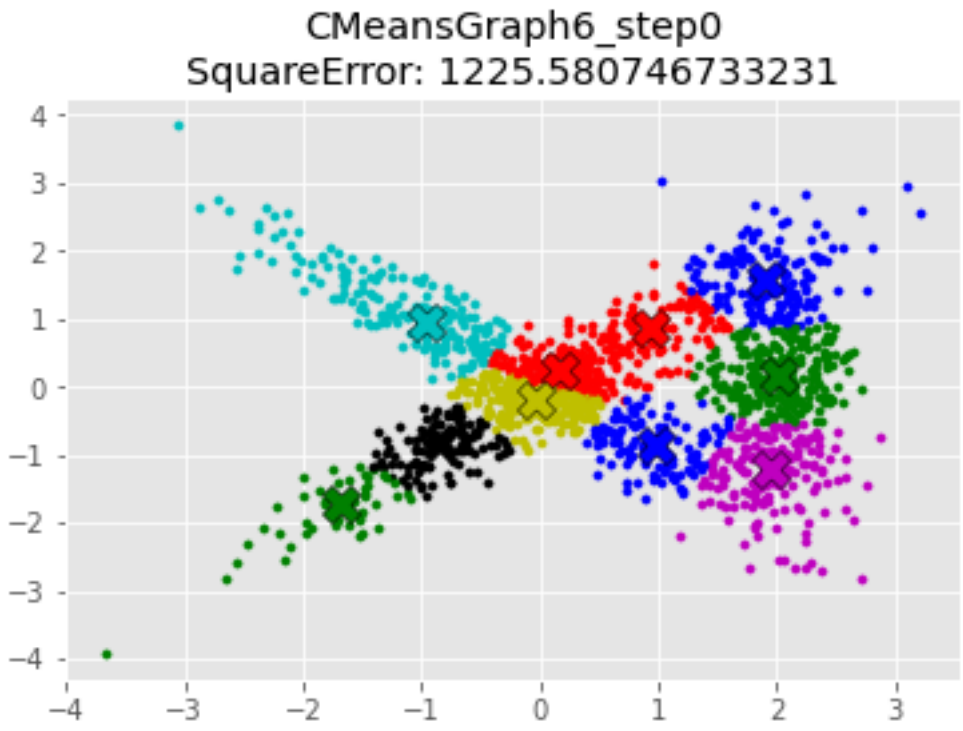

CMeansGraph1_step0
SquareError: 1579.3697150368894

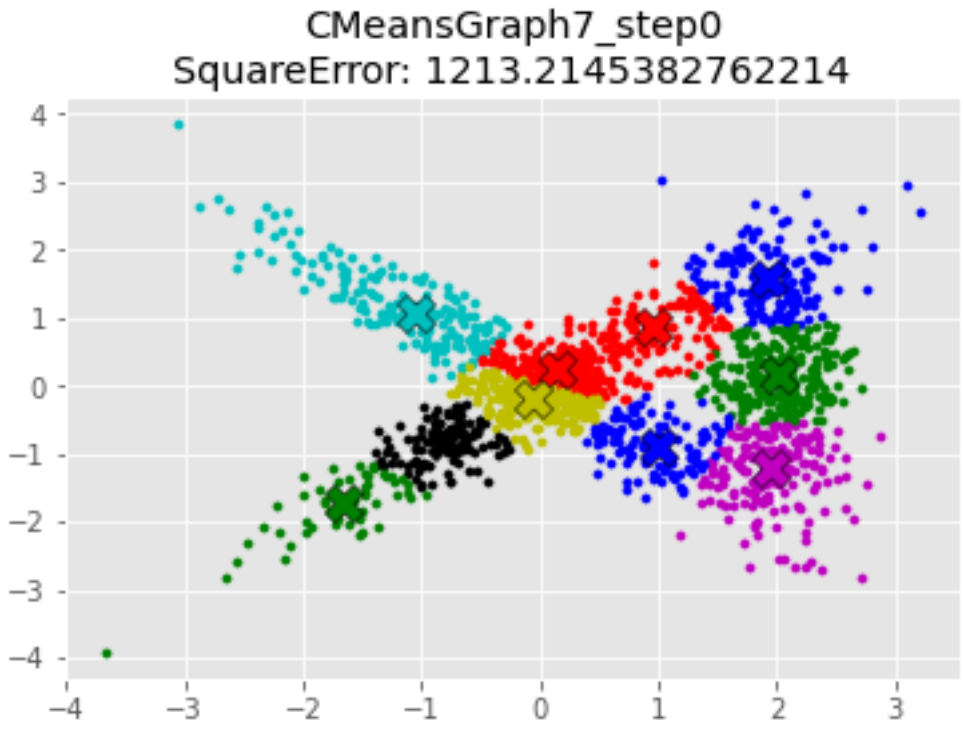

CMeansGraph2_step0
SquareError: 1381.7420775620435
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

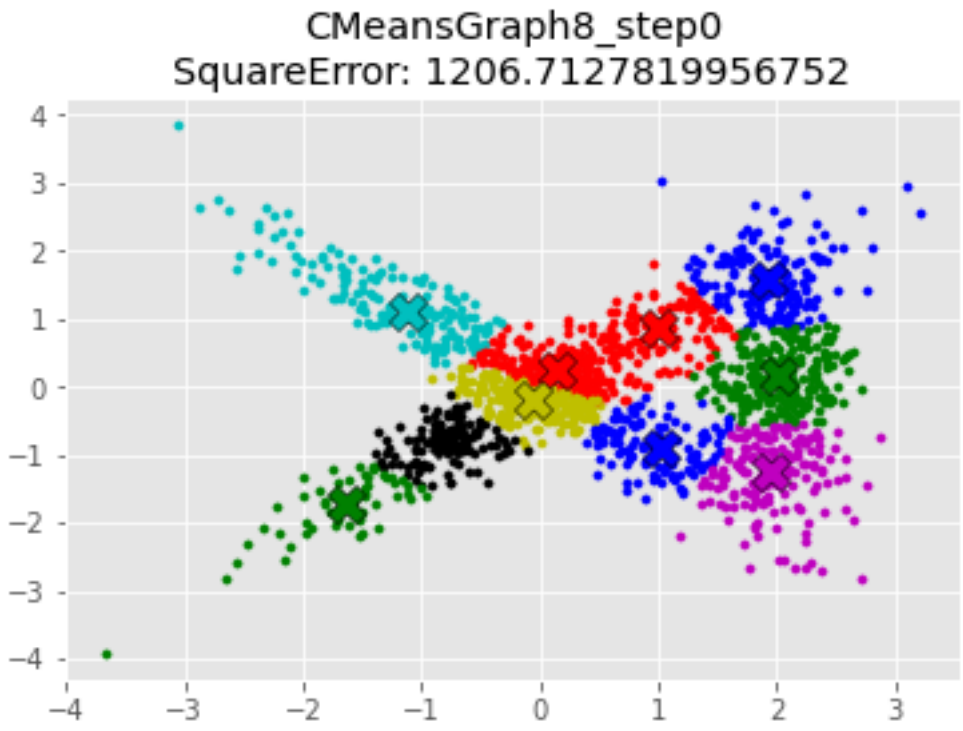

CMeansGraph3_step0
SquareError: 1315.1192029216622
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

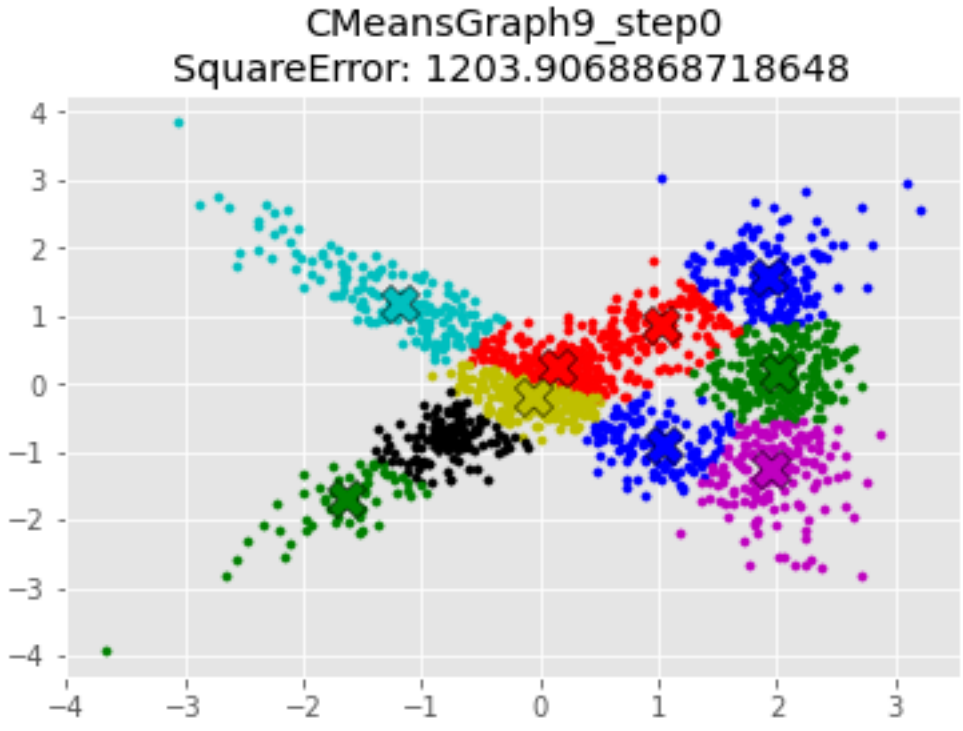

CMeansGraph4_step0
SquareError: 1275.0342895688984

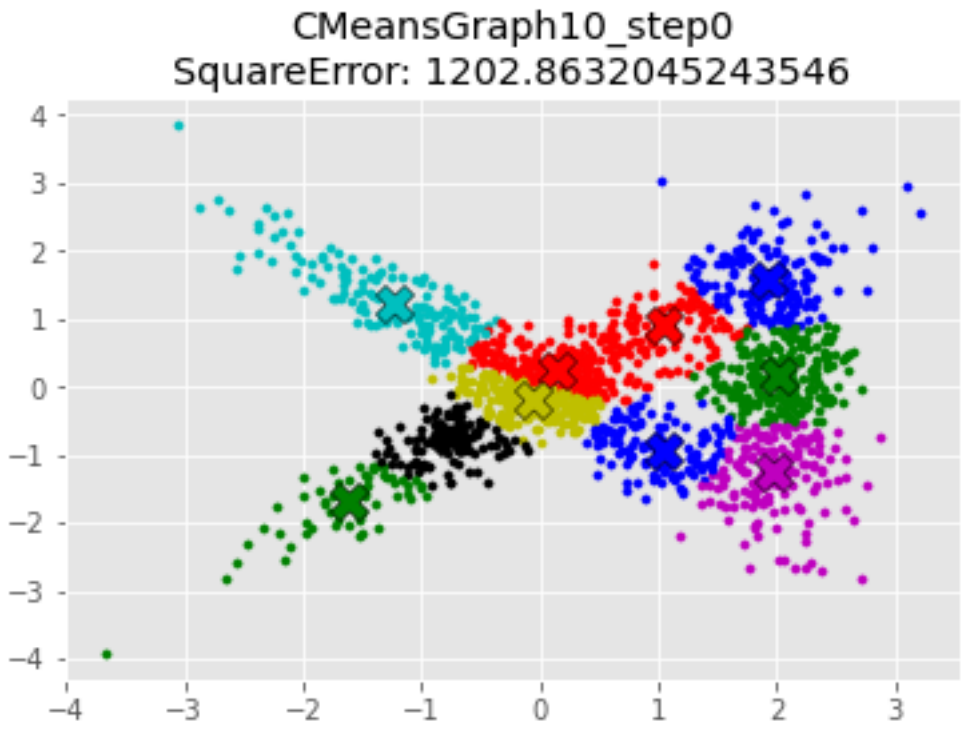

CMeansGraph5_step0
SquareError: 1245.473998187481

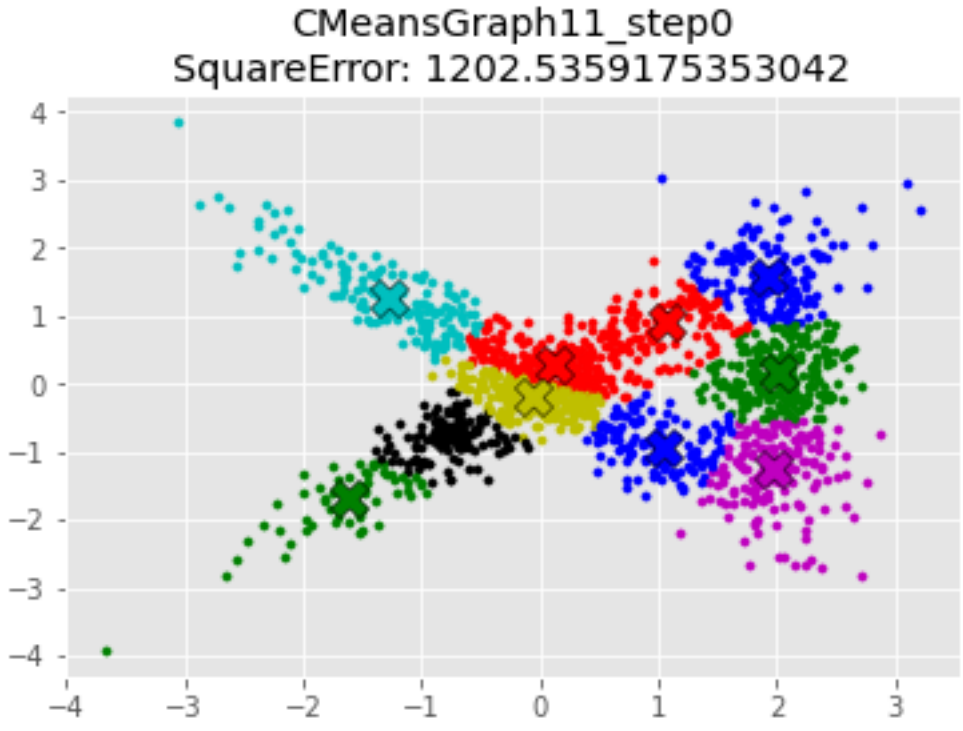

CMeansGraph6_step0
SquareError: 1225.580746733231
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

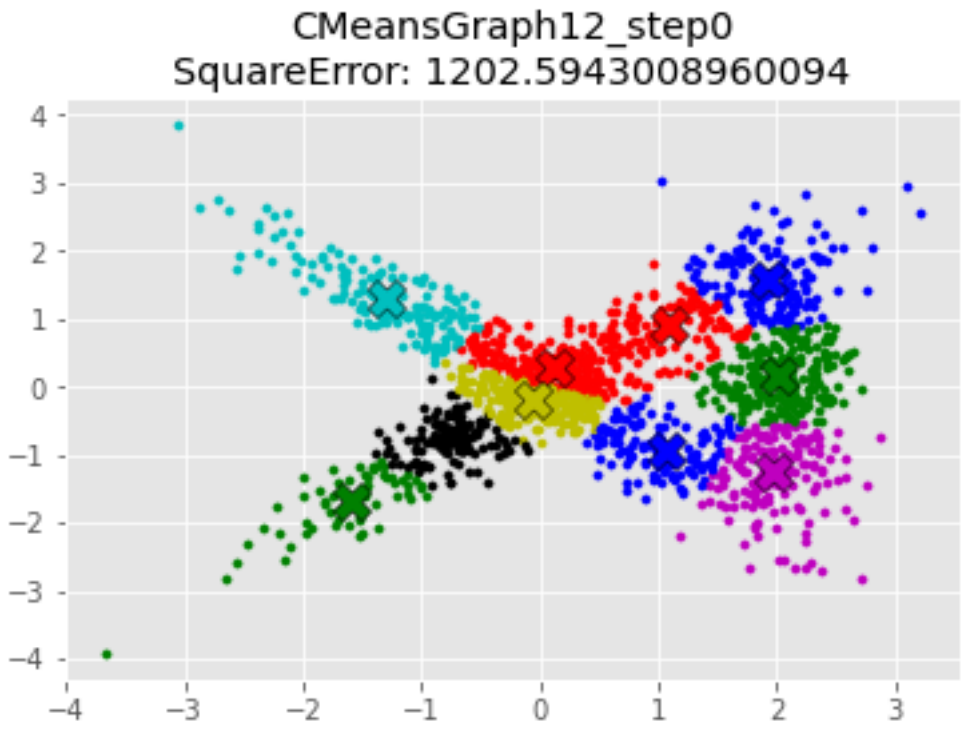

CMeansGraph7_step0
SquareError: 1213.2145382762214
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

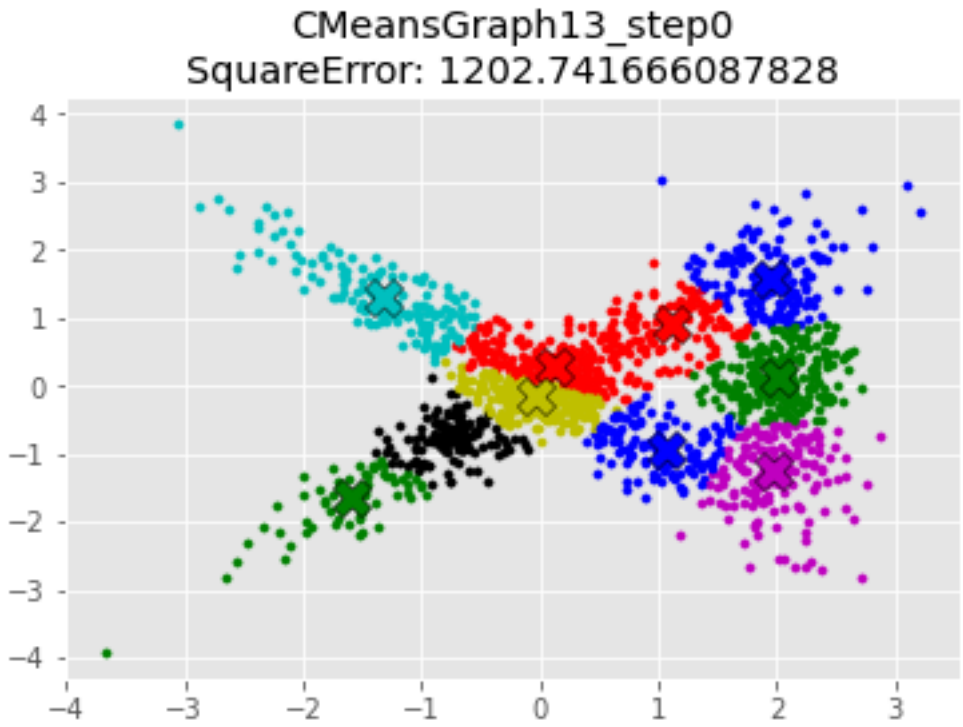

CMeansGraph8_step0
SquareError: 1206.7127819956752

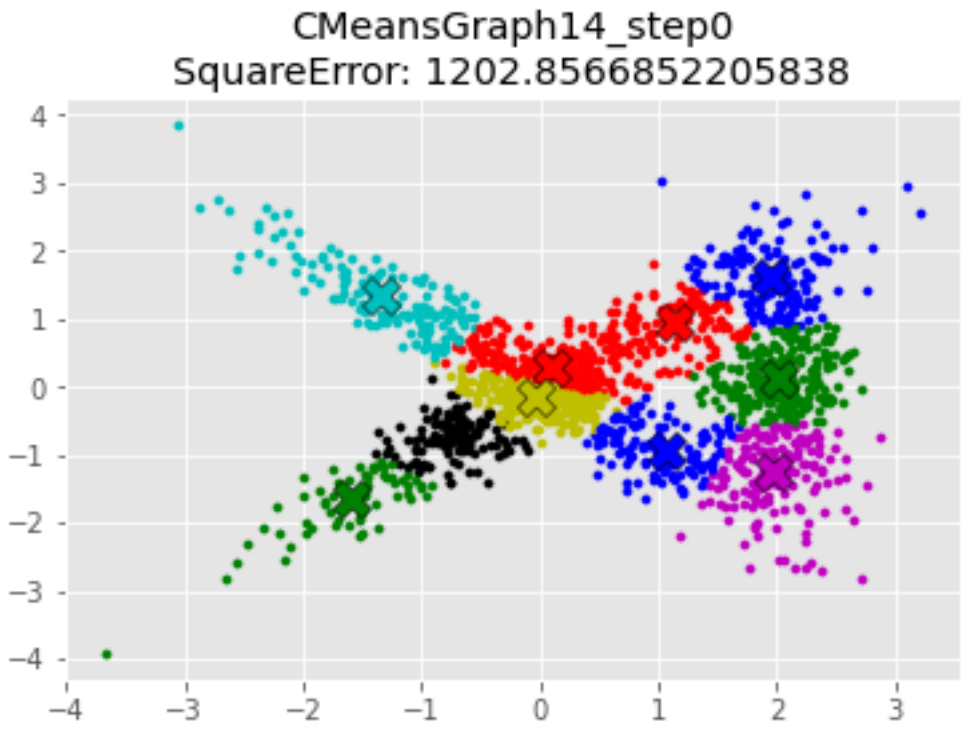

CMeansGraph9_step0
SquareError: 1203.9068868718648

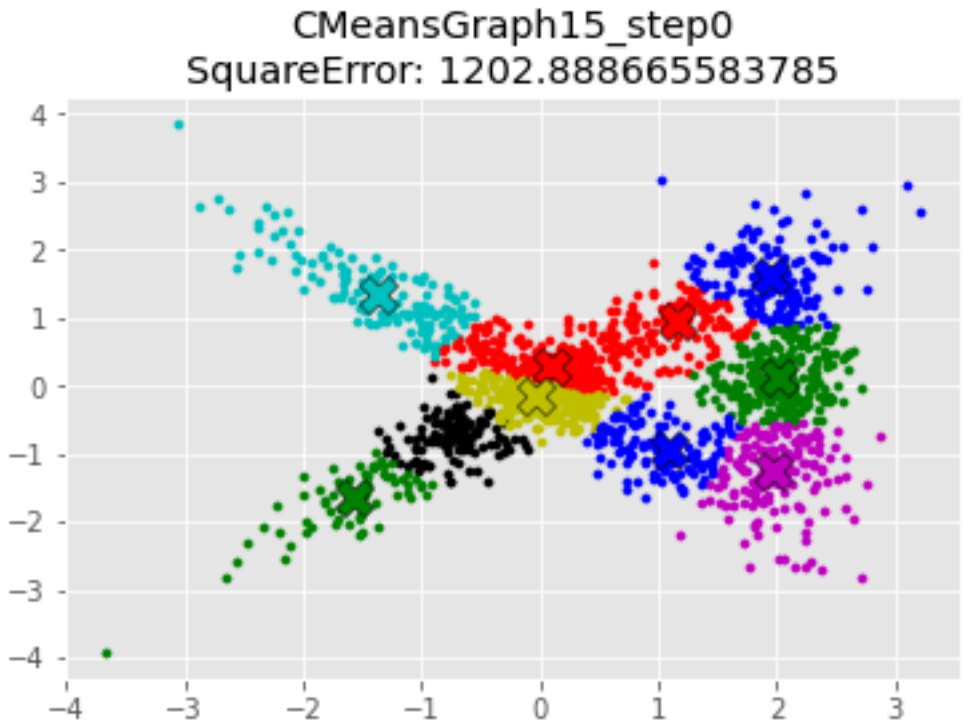

CMeansGraph10_step0
SquareError: 1202.8632045243546

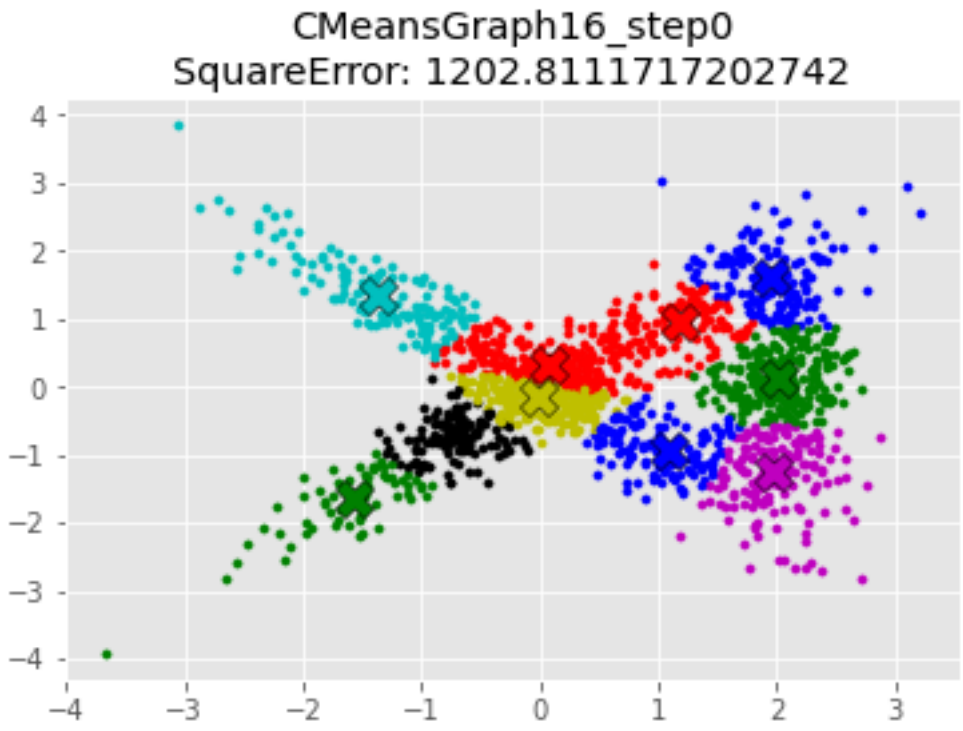

CMeansGraph11_step0
SquareError: 1202.5359175353042

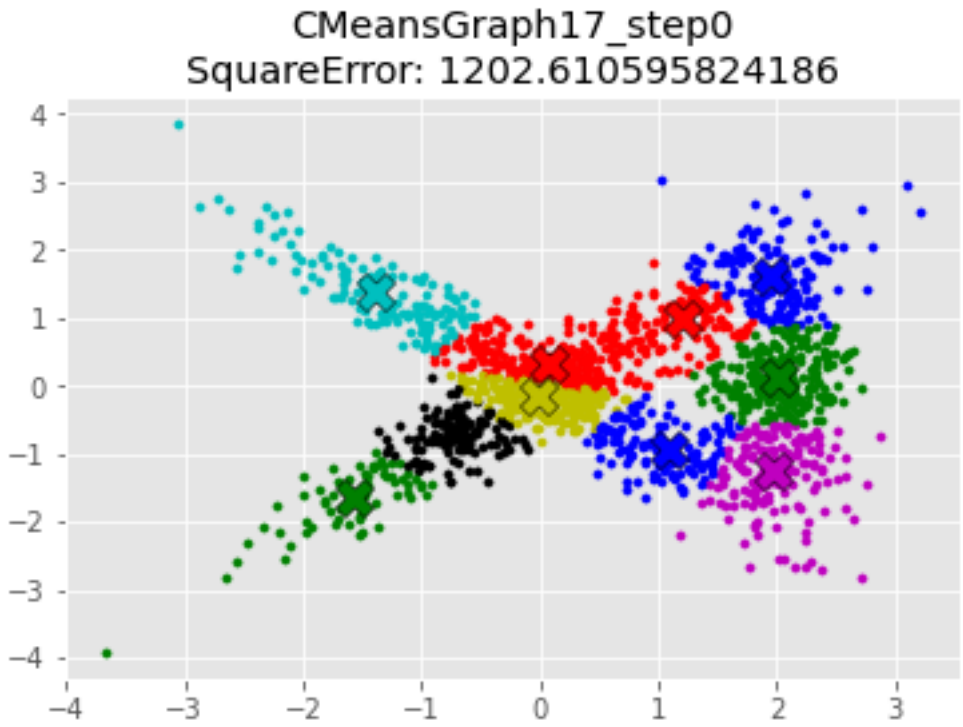

CMeansGraph12_step0
SquareError: 1202.5943008960094
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

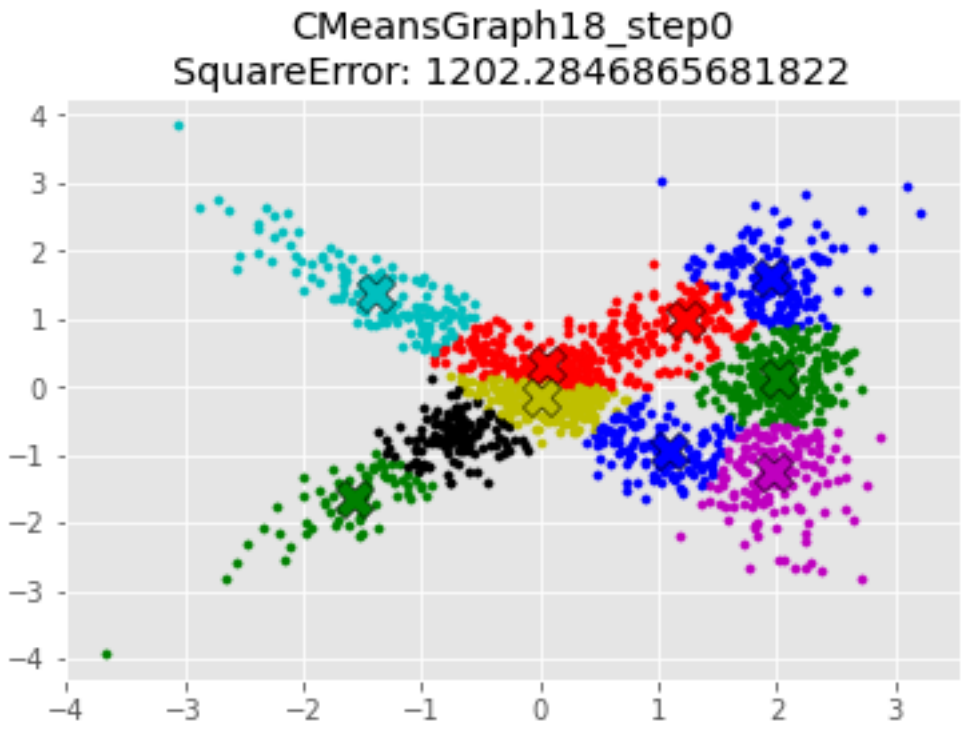

CMeansGraph13_step0
SquareError: 1202.741666087828
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

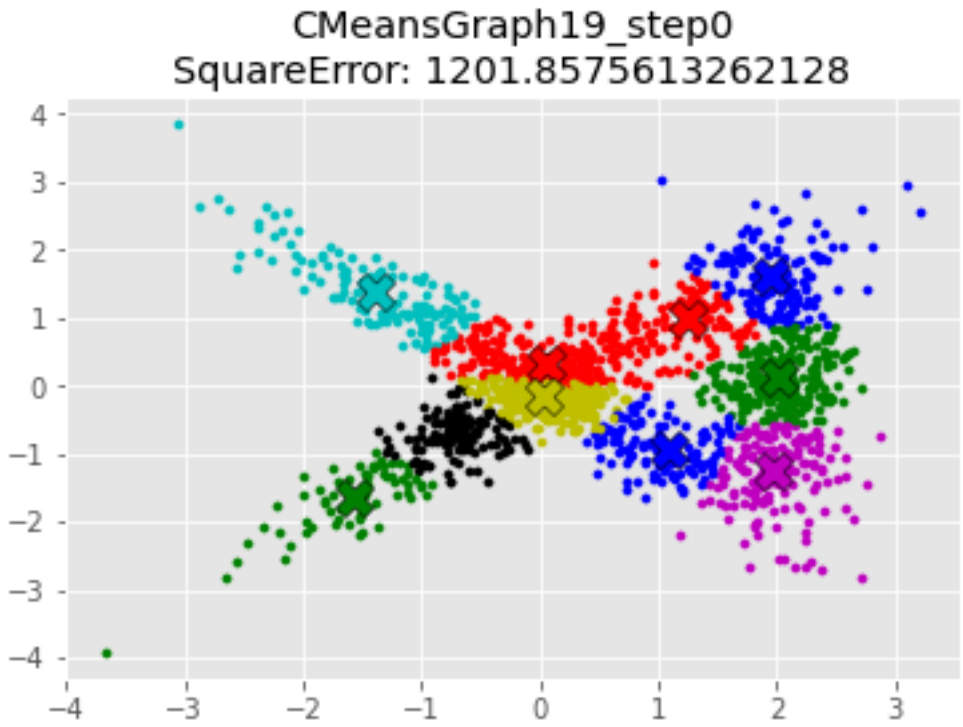

CMeansGraph14_step0
SquareError: 1202.8566852205838
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

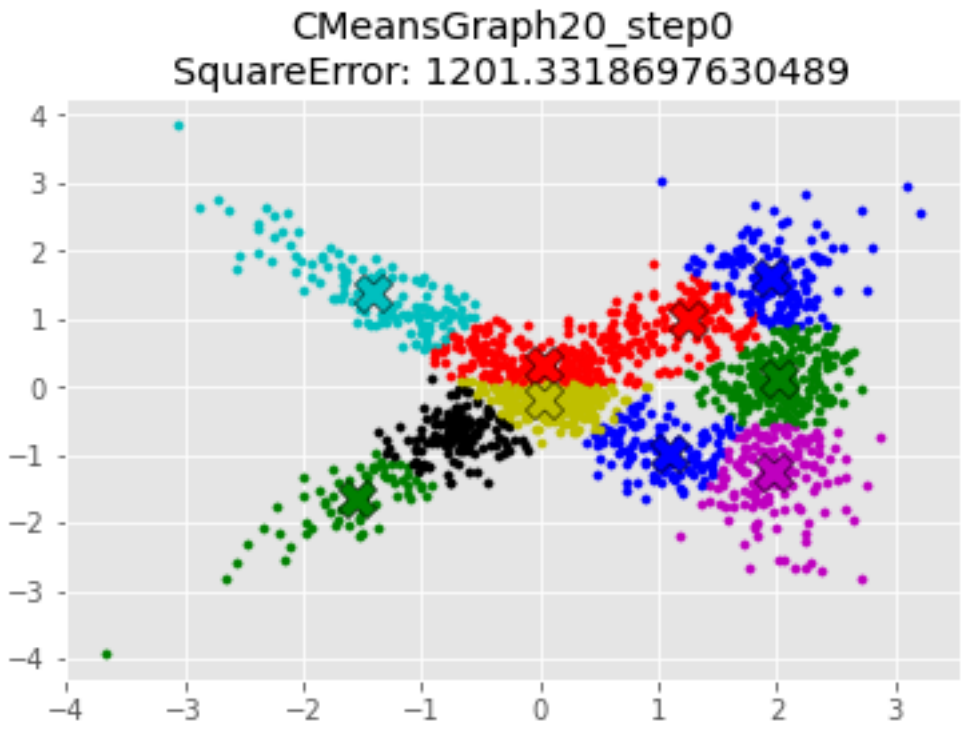

CMeansGraph15_step0
SquareError: 1202.888665583785
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

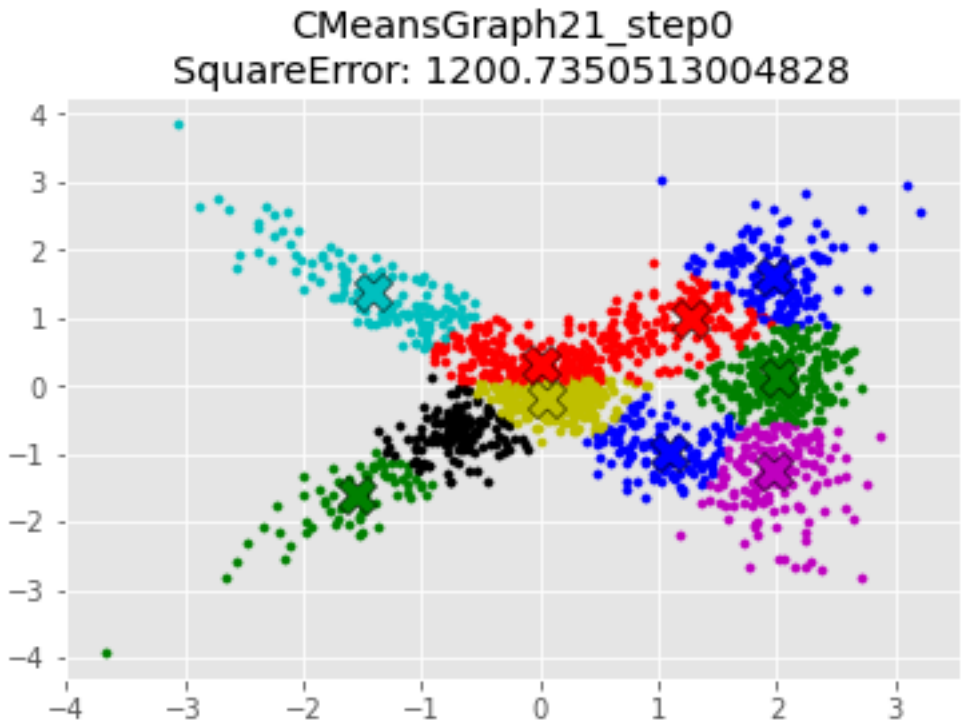

CMeansGraph16_step0
SquareError: 1202.8111717202742
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

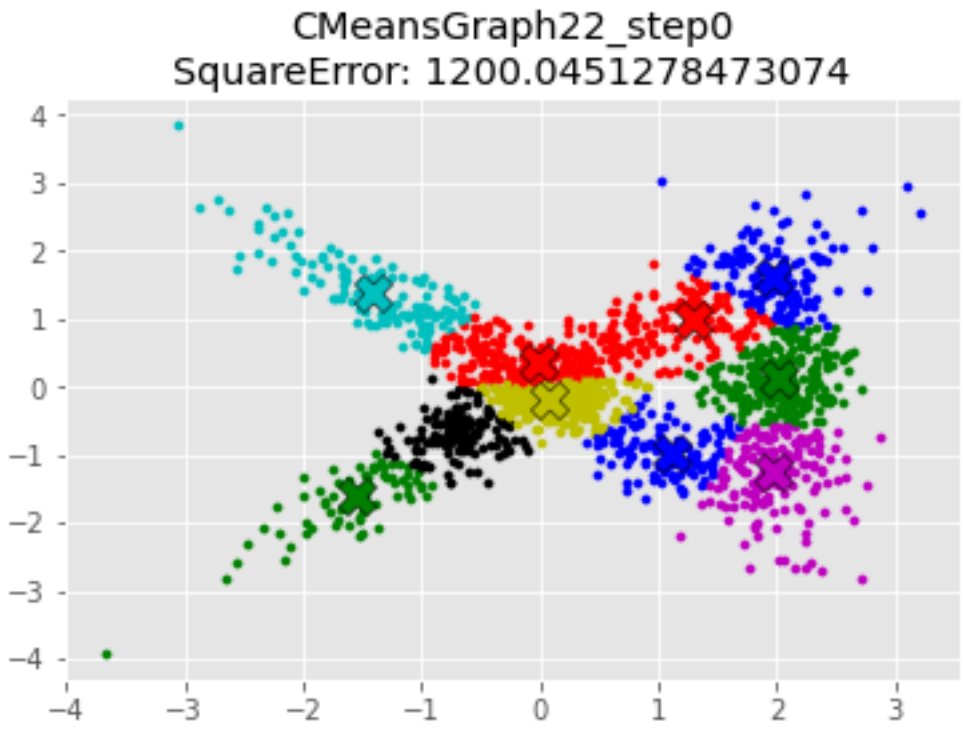

CMeansGraph17_step0
SquareError: 1202.610595824186
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

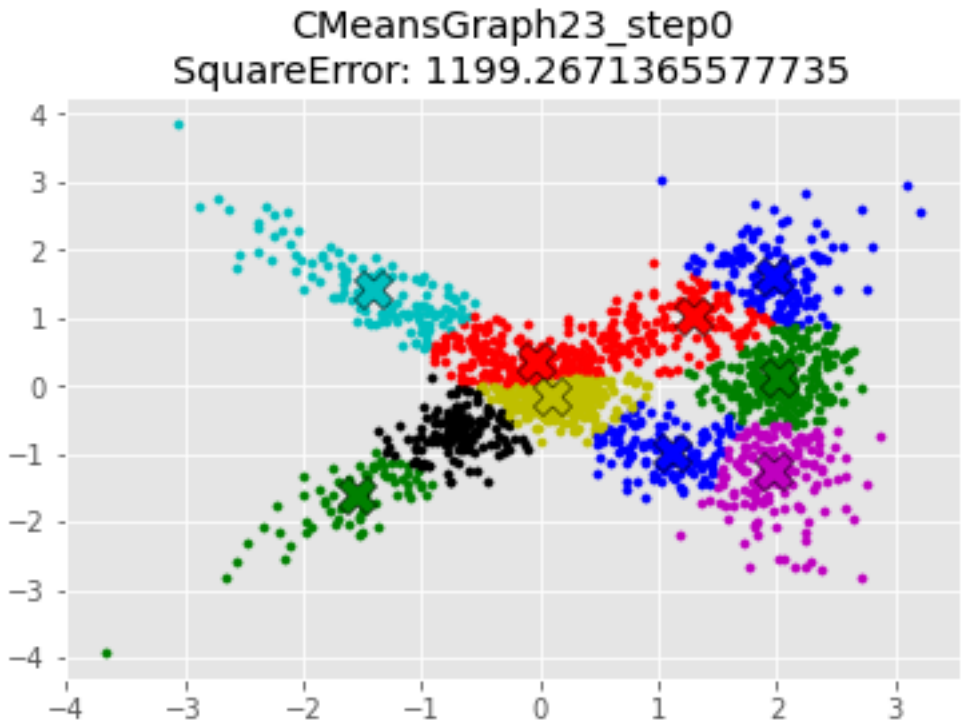

CMeansGraph18_step0
SquareError: 1202.2846865681822
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

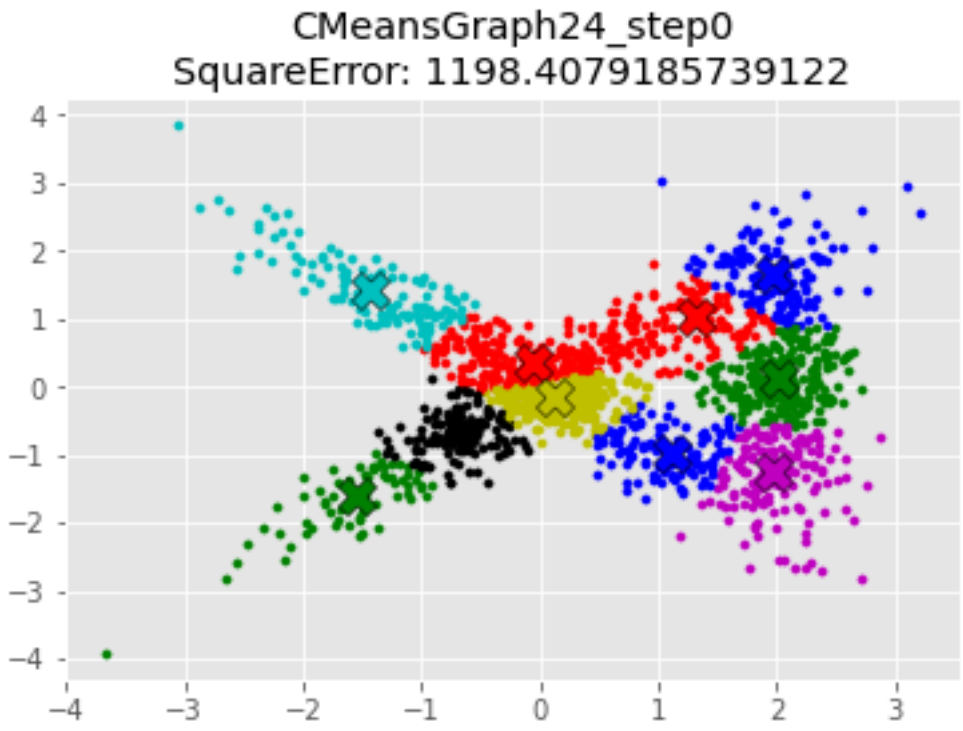

CMeansGraph19_step0
SquareError: 1201.8575613262128
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

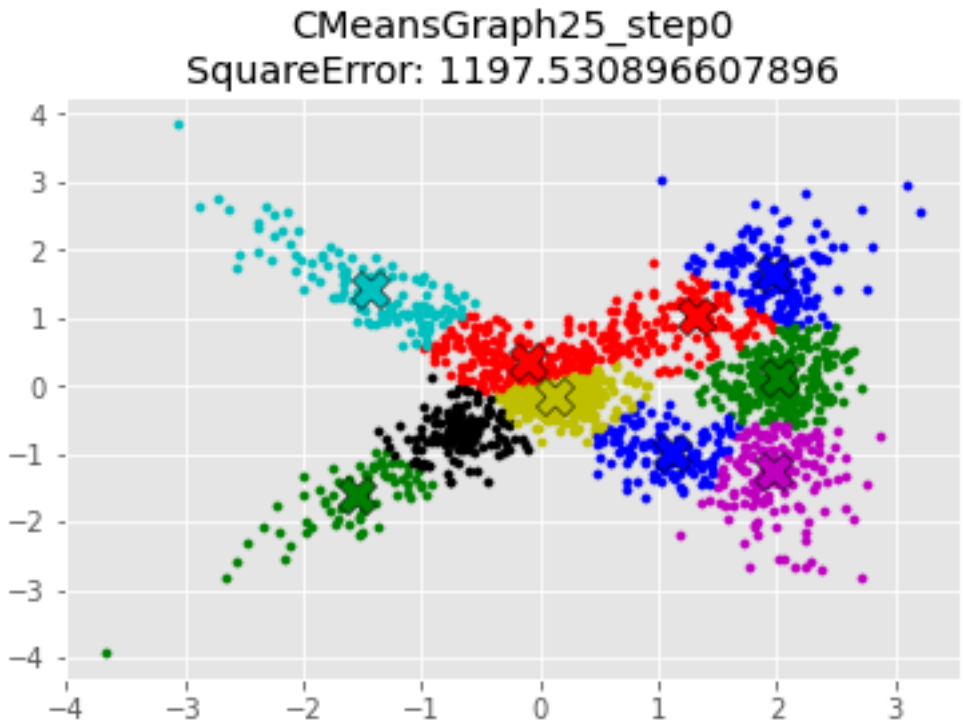

CMeansGraph20_step0
SquareError: 1201.3318697630489
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

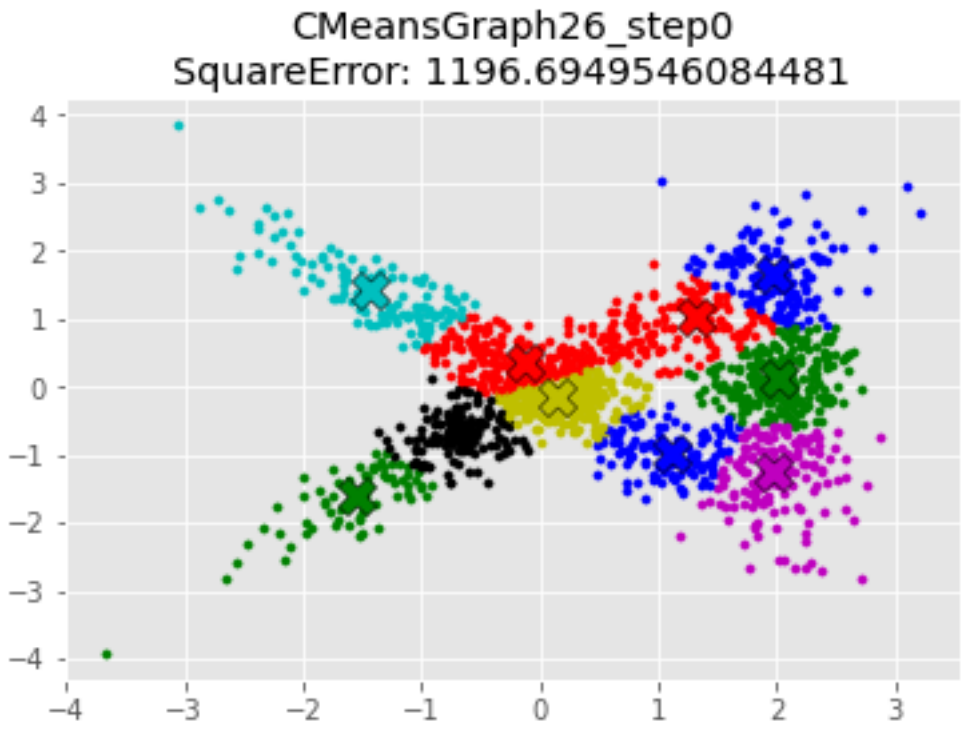

CMeansGraph21_step0
SquareError: 1200.7350513004828
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

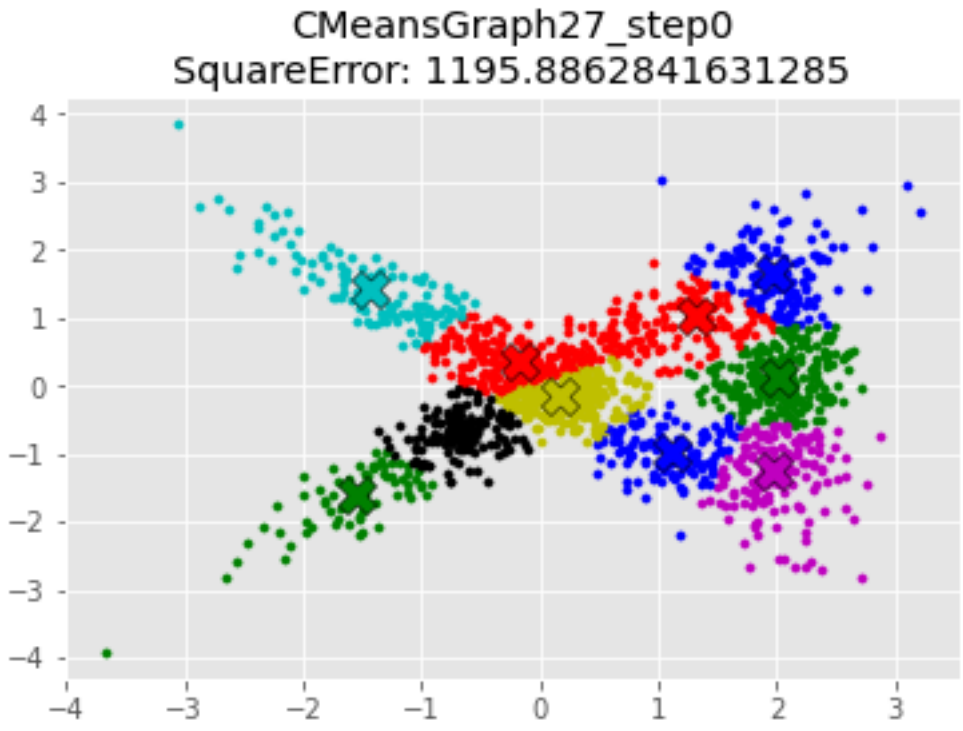

CMeansGraph22_step0
SquareError: 1200.0451278473074
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

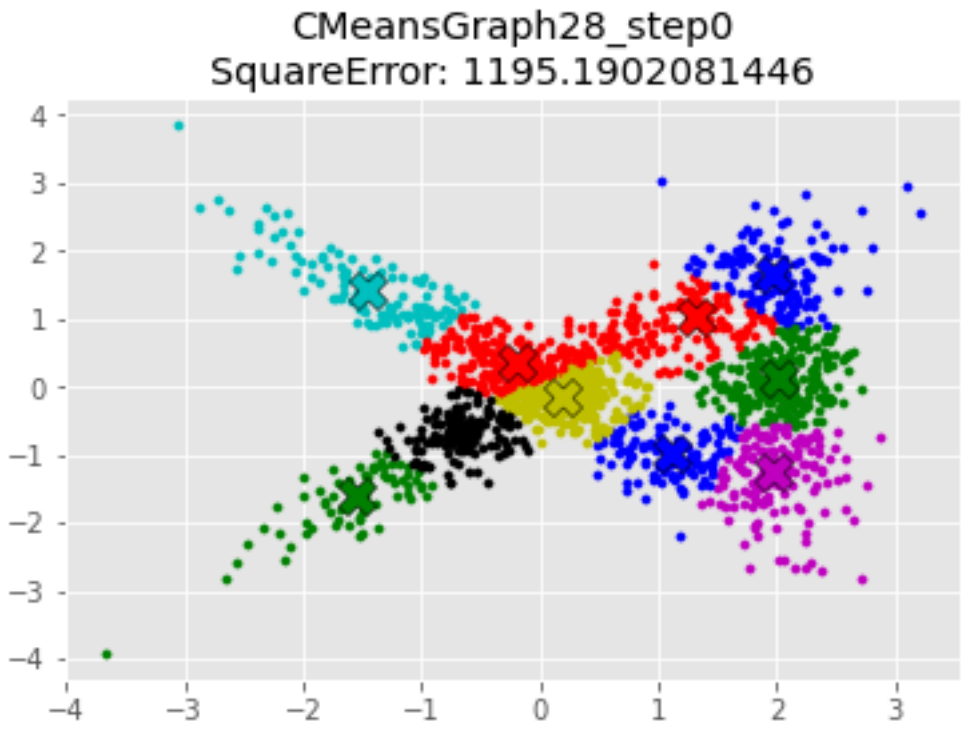

CMeansGraph23_step0
SquareError: 1199.2671365577735
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

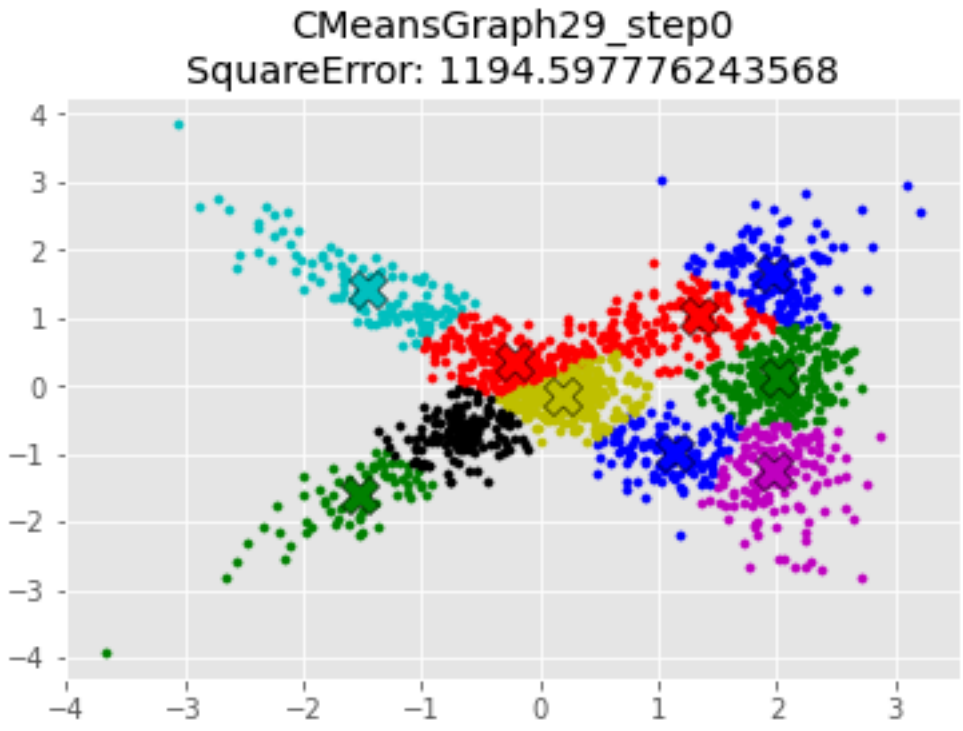

CMeansGraph24_step0
SquareError: 1198.4079185739122
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

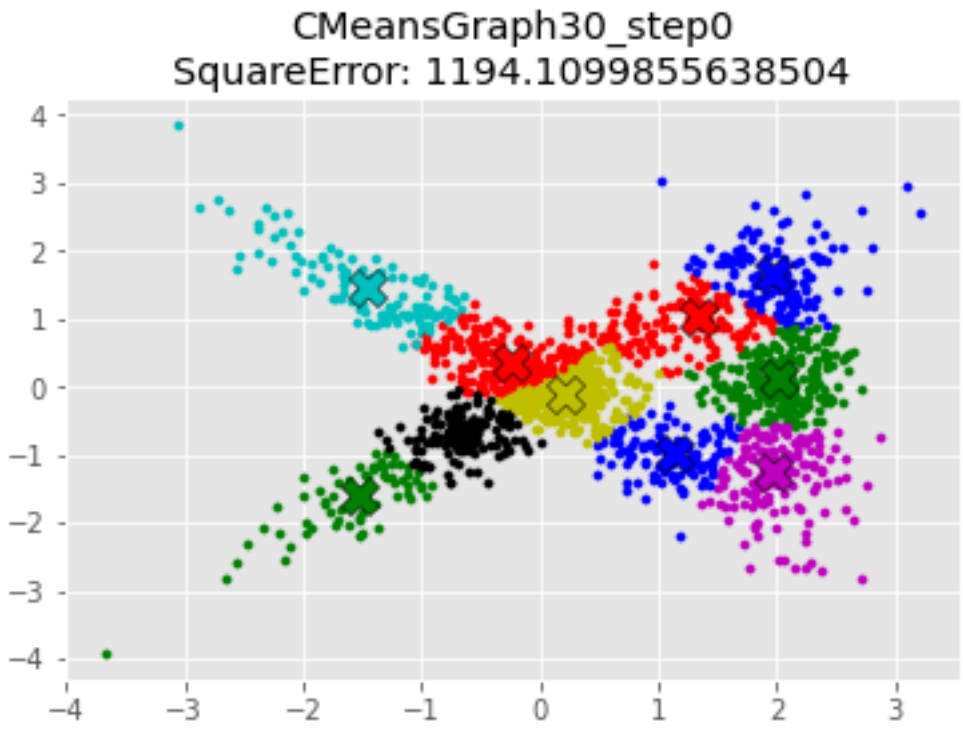

CMeansGraph25_step0
SquareError: 1197.530896607896
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

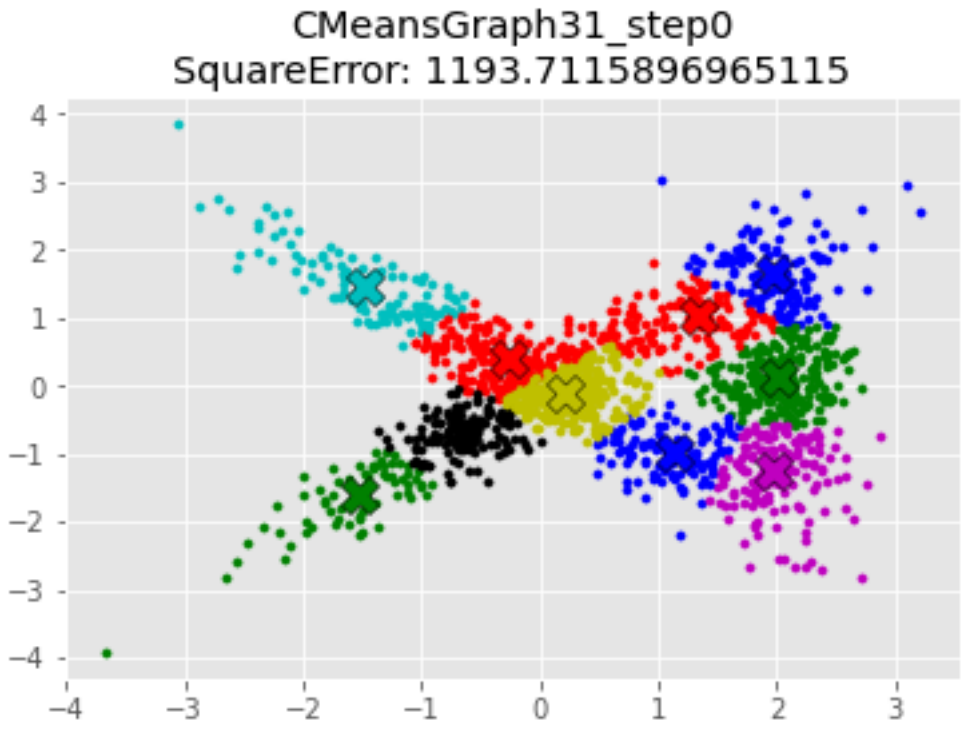

CMeansGraph26_step0
SquareError: 1196.6949546084481

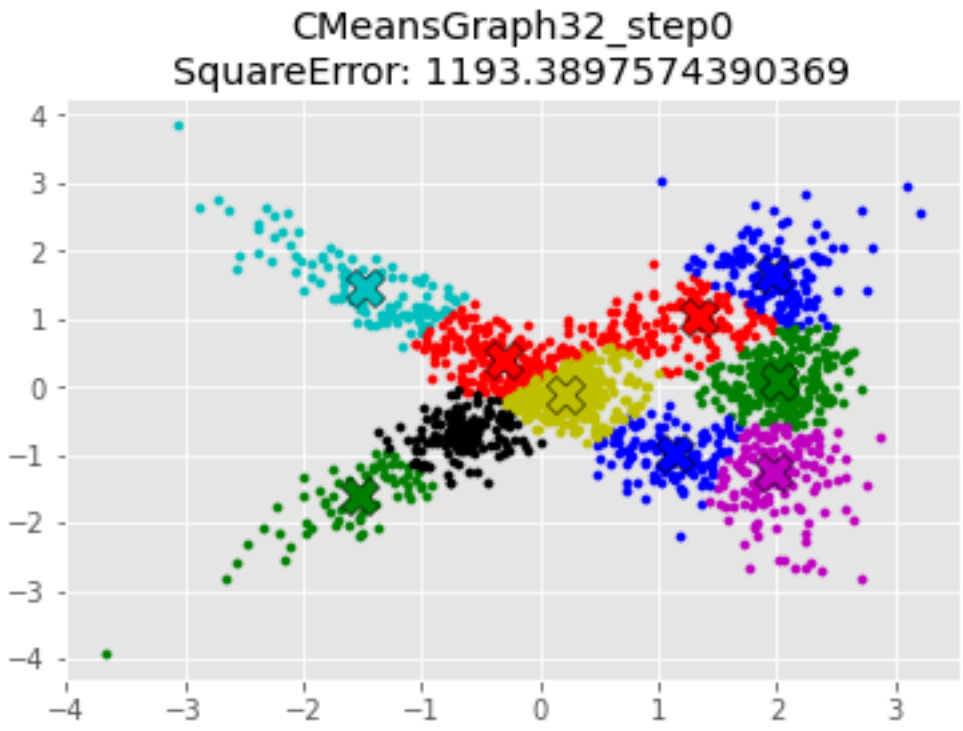

CMeansGraph27_step0
SquareError: 1195.8862841631285

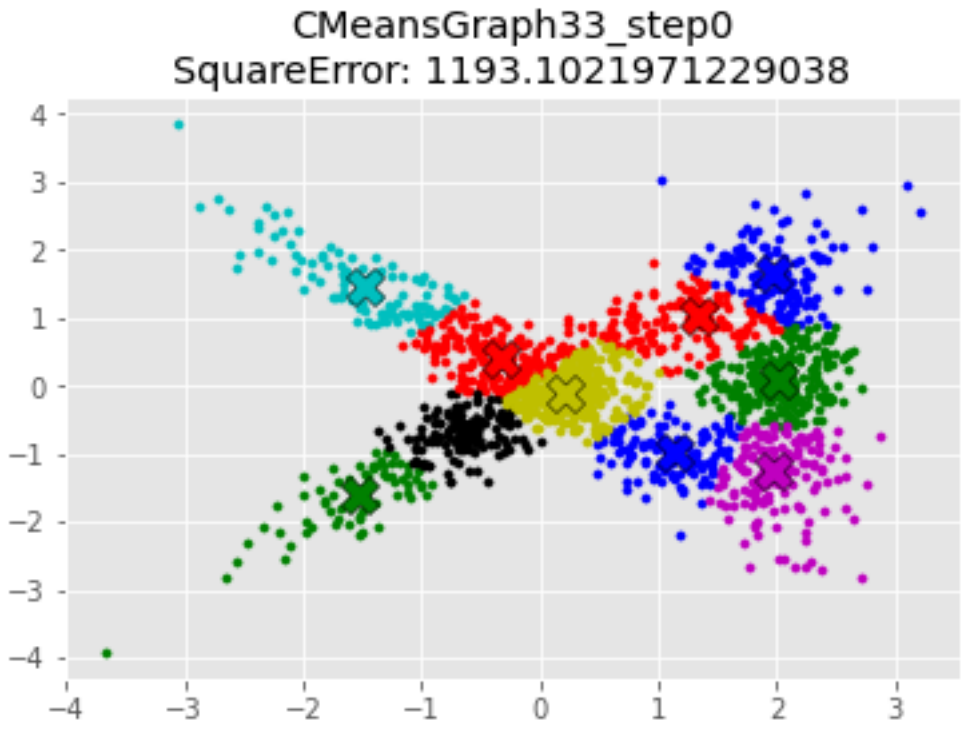

CMeansGraph28_step0
SquareError: 1195.1902081446
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

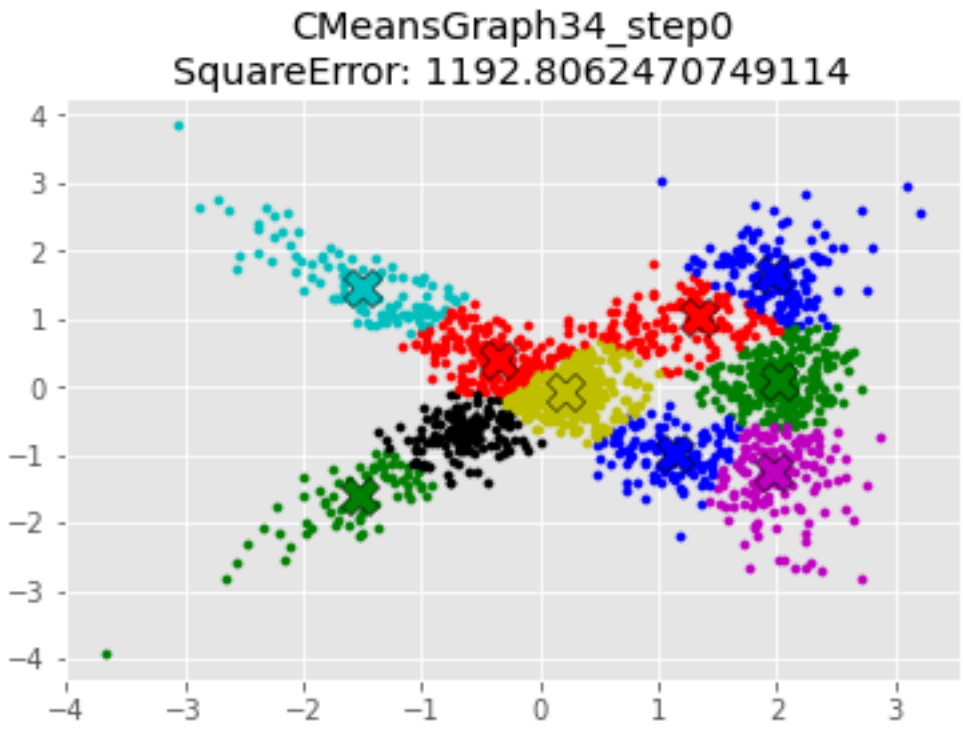

CMeansGraph29_step0
SquareError: 1194.597776243568
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

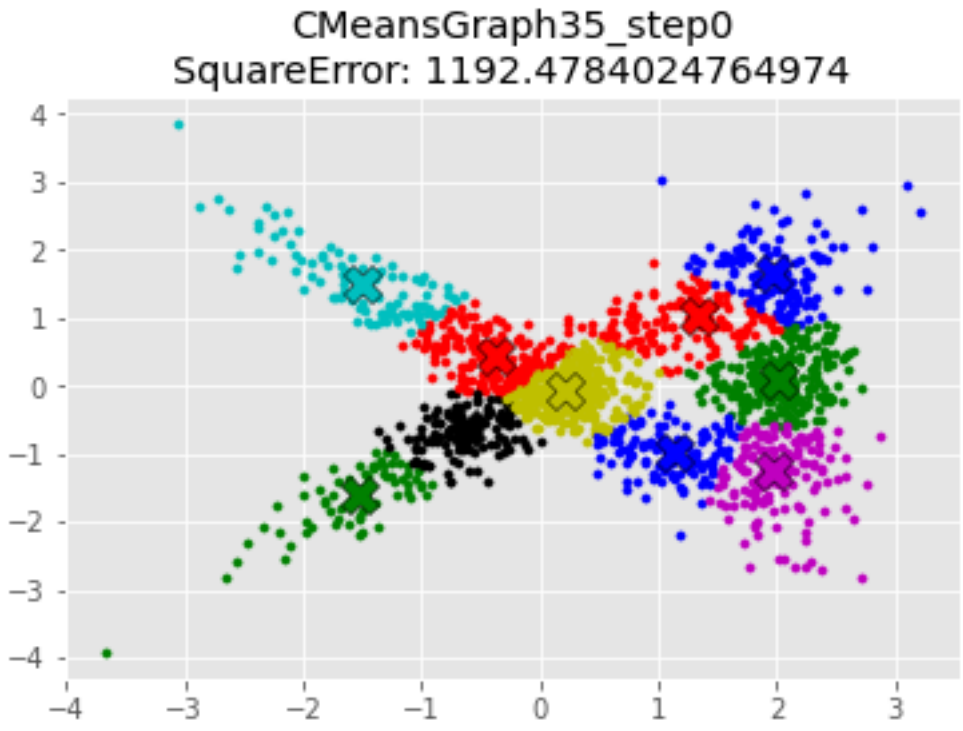

CMeansGraph30_step0
SquareError: 1194.1099855638504
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

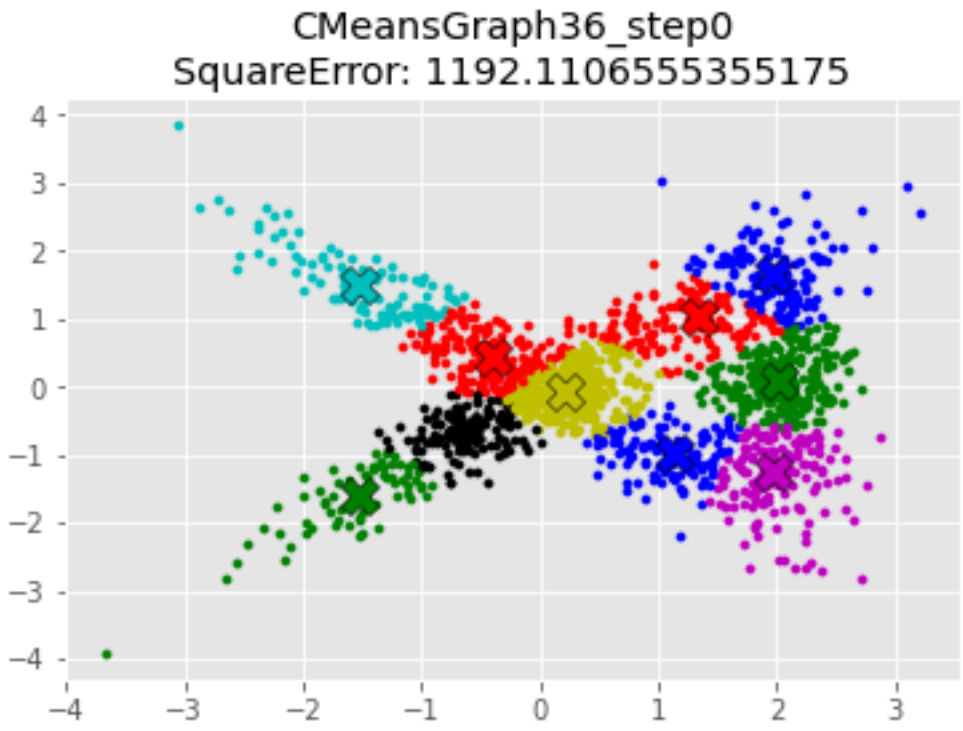

CMeansGraph31_step0
SquareError: 1193.7115896965115
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

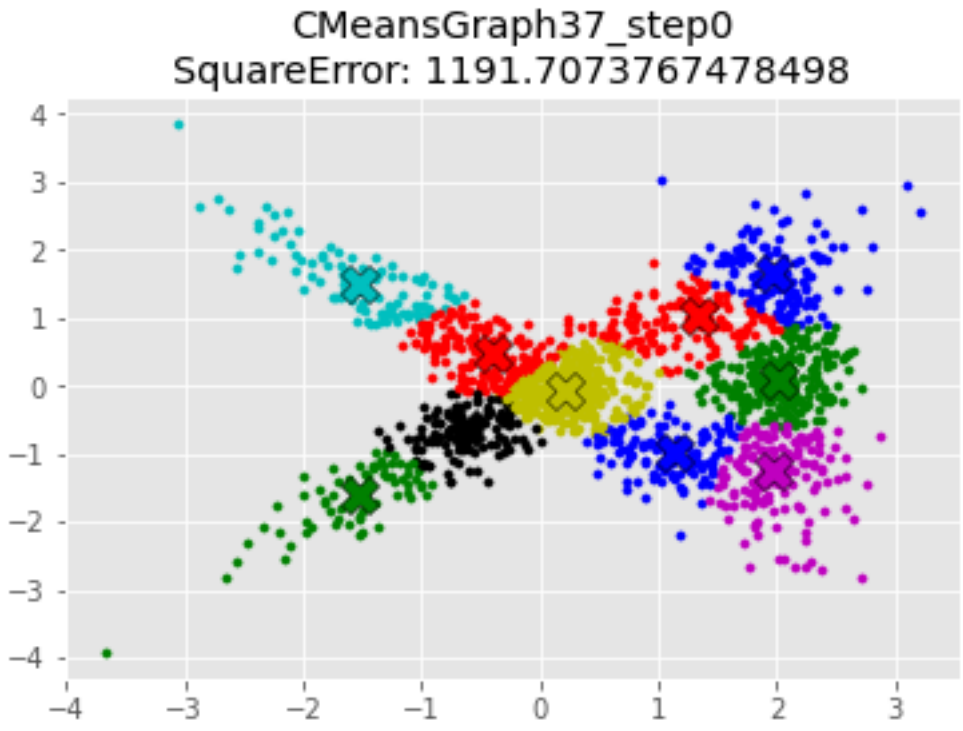

CMeansGraph32_step0
SquareError: 1193.3897574390369

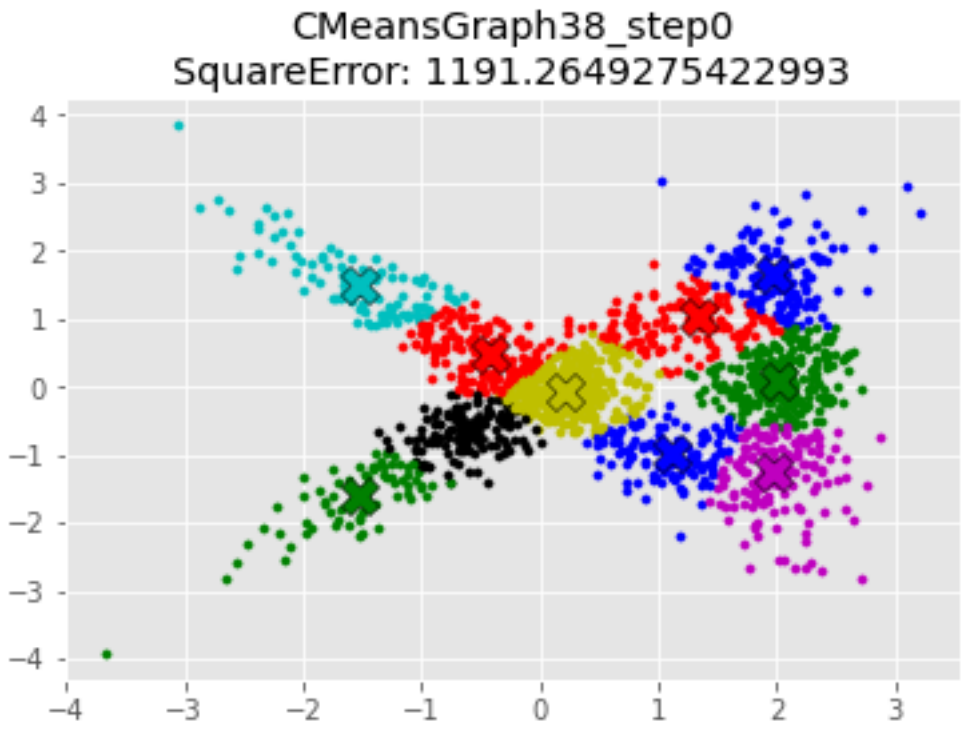

CMeansGraph33_step0
SquareError: 1193.1021971229038

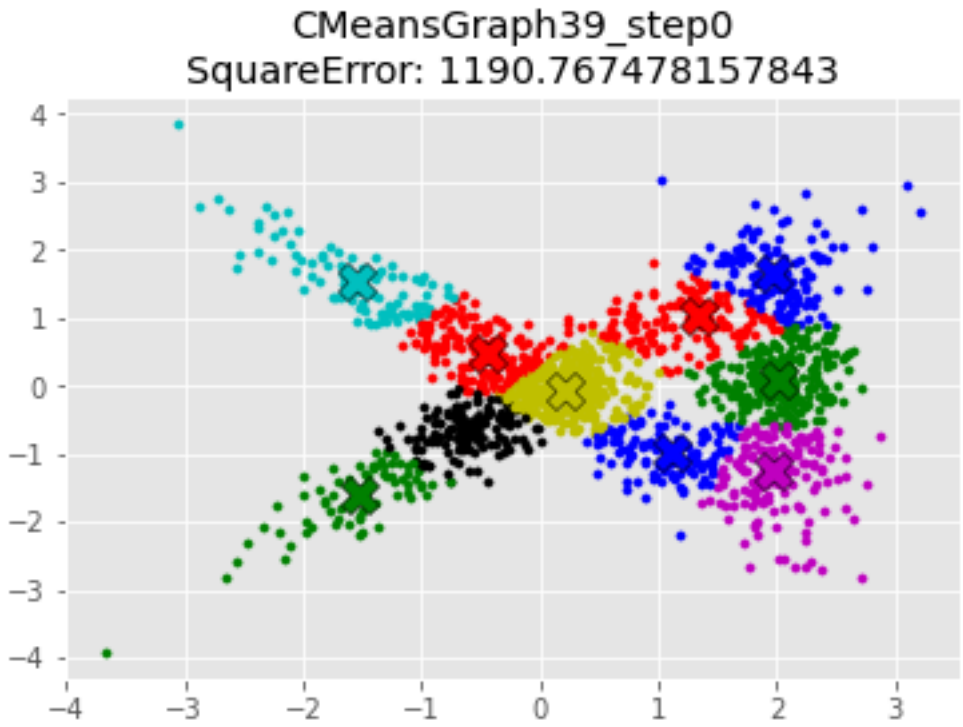

CMeansGraph34_step0
SquareError: 1192.8062470749114

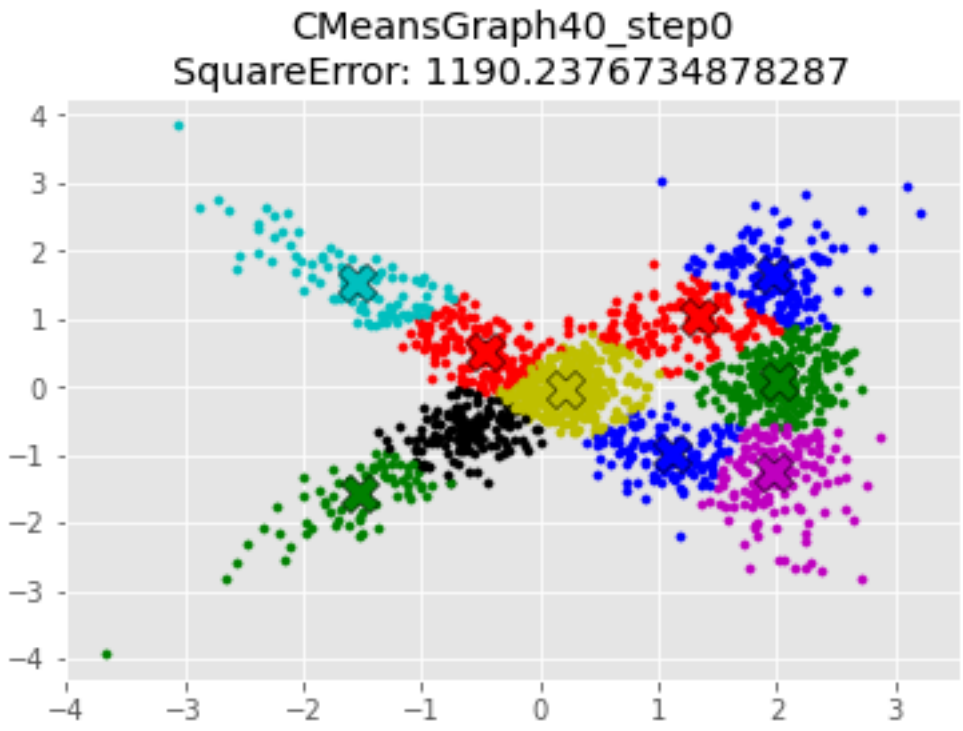

CMeansGraph35_step0
SquareError: 1192.4784024764974

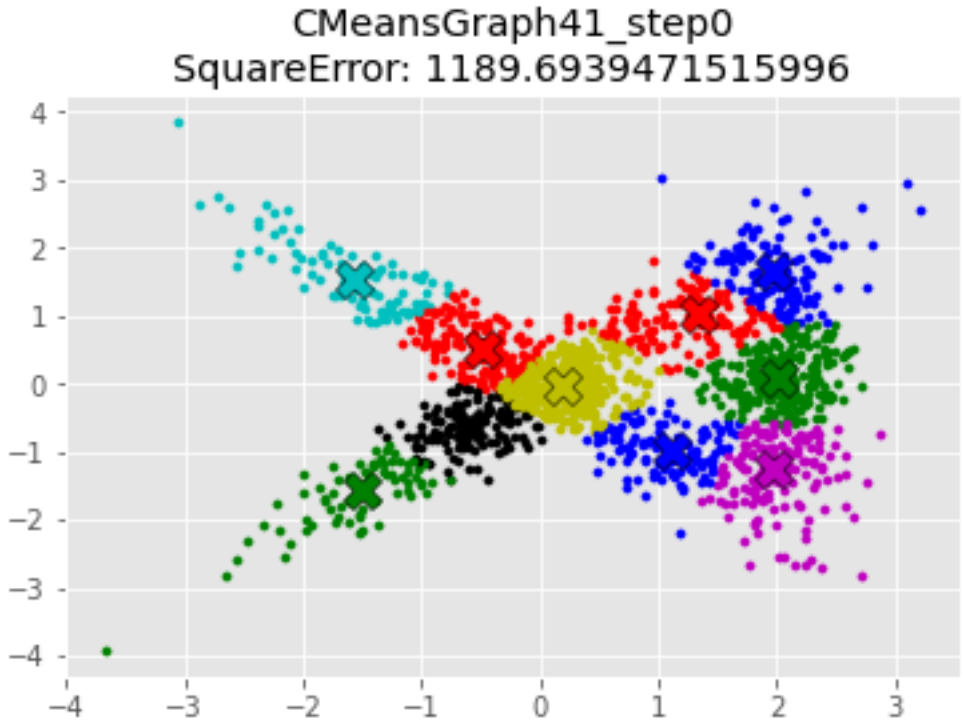

CMeansGraph36_step0
SquareError: 1192.1106555355175

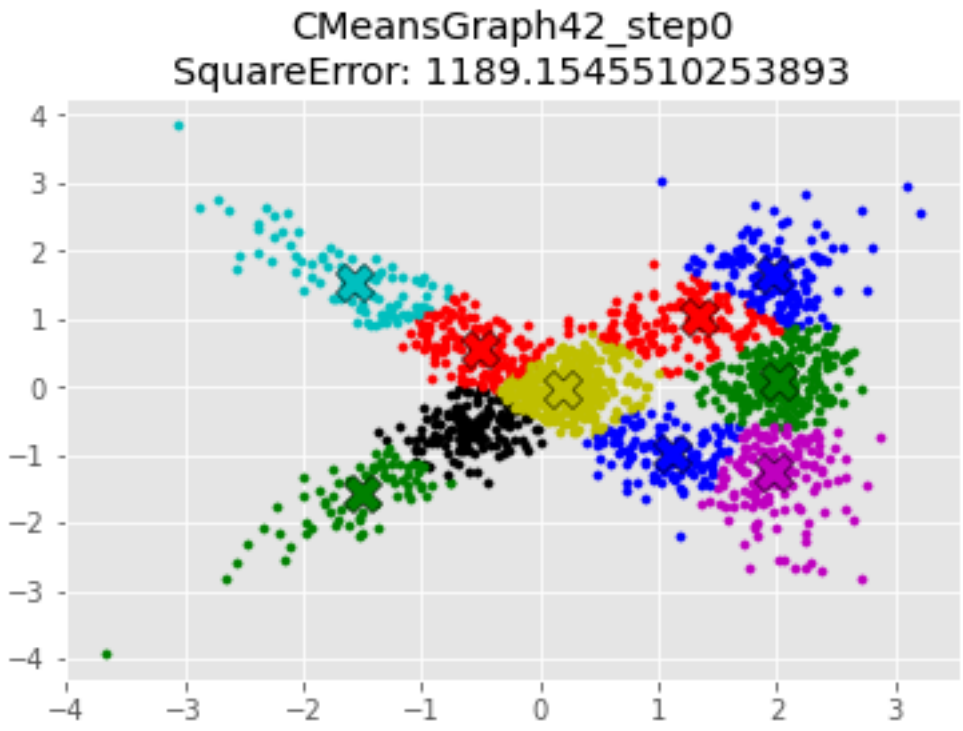

CMeansGraph37_step0
SquareError: 1191.7073767478498

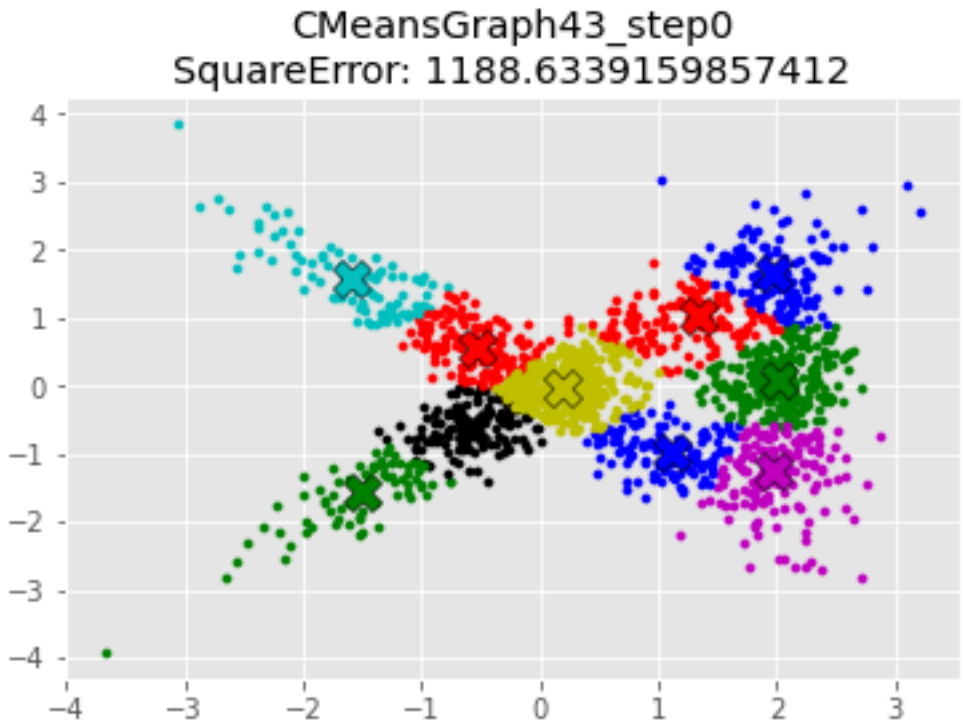

CMeansGraph38_step0
SquareError: 1191.2649275422993
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

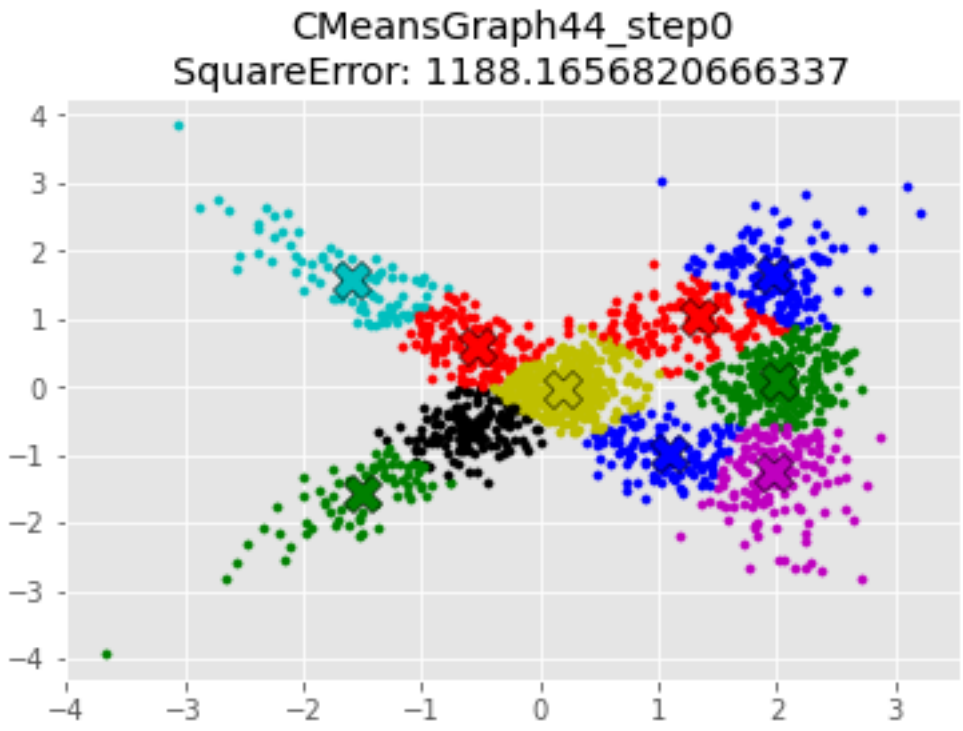

CMeansGraph39_step0
SquareError: 1190.767478157843
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

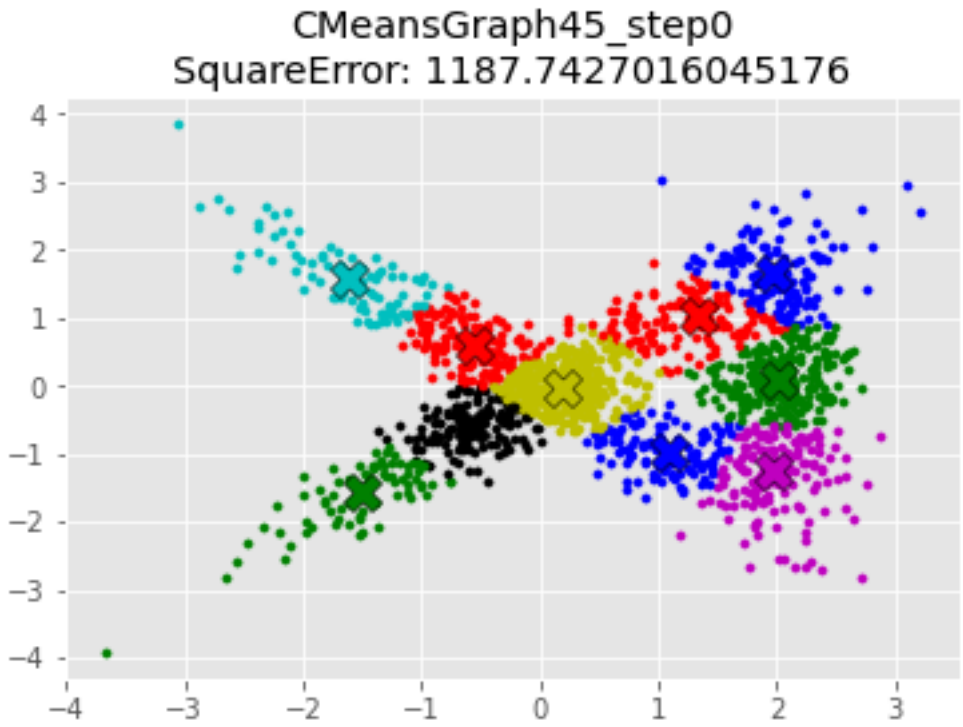

CMeansGraph40_step0
SquareError: 1190.2376734878287
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

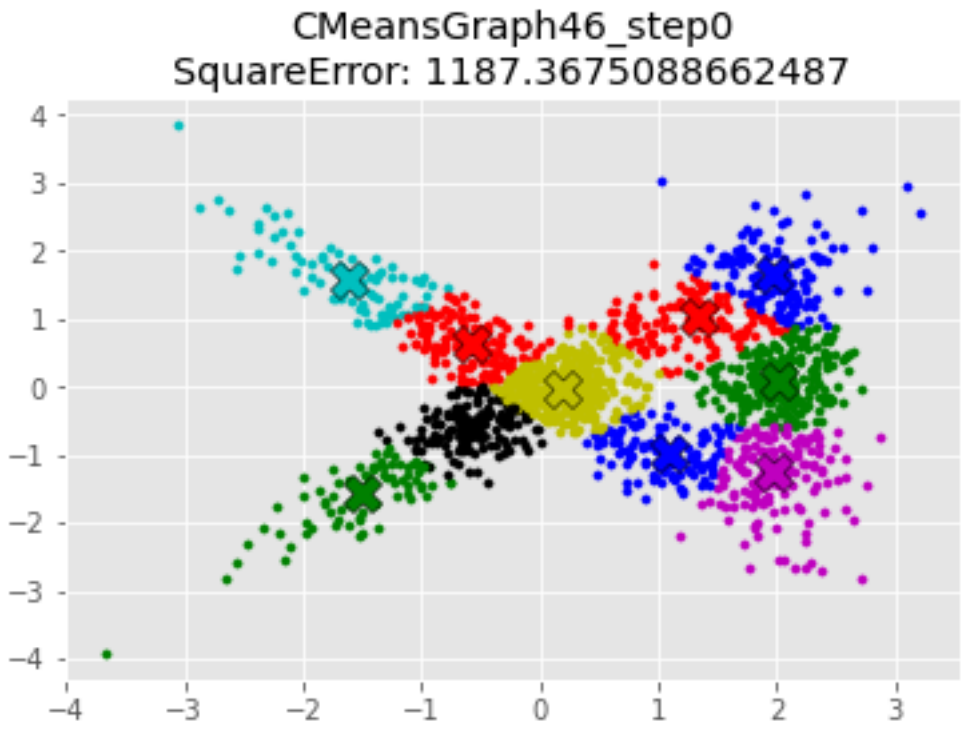

CMeansGraph41_step0
SquareError: 1189.6939471515996
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

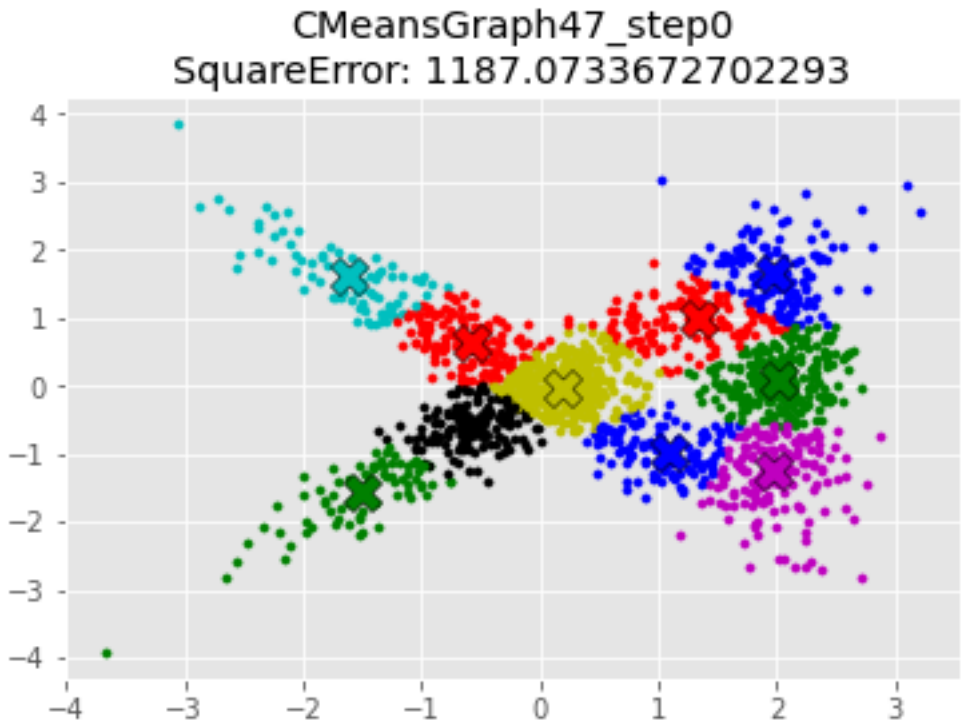

CMeansGraph42_step0
SquareError: 1189.1545510253893
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

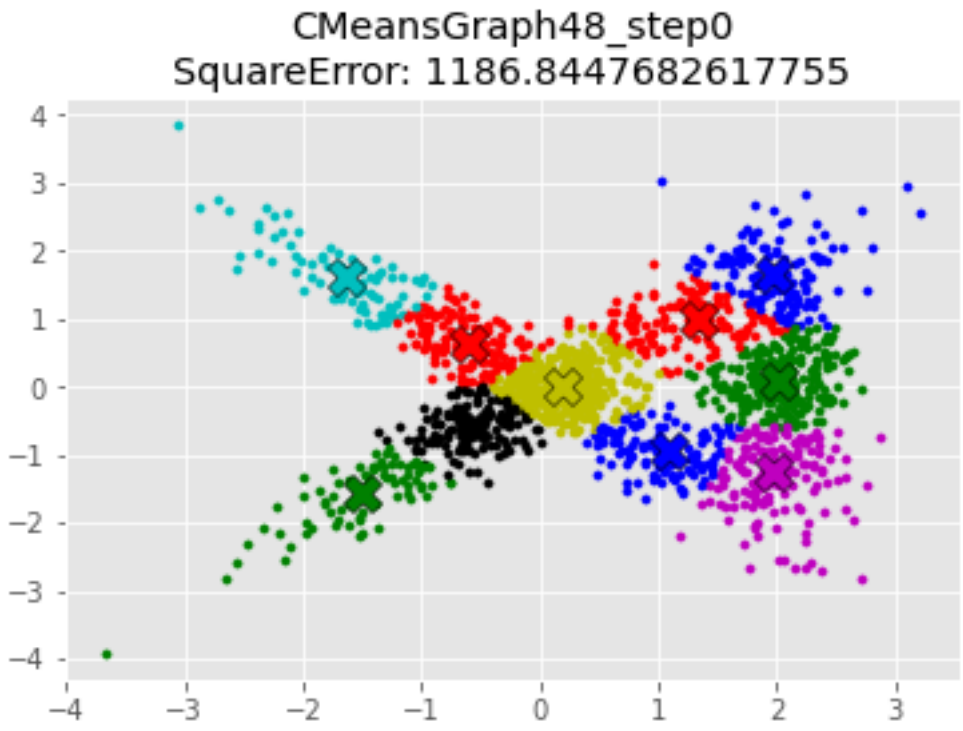

CMeansGraph43_step0
SquareError: 1188.6339159857412
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

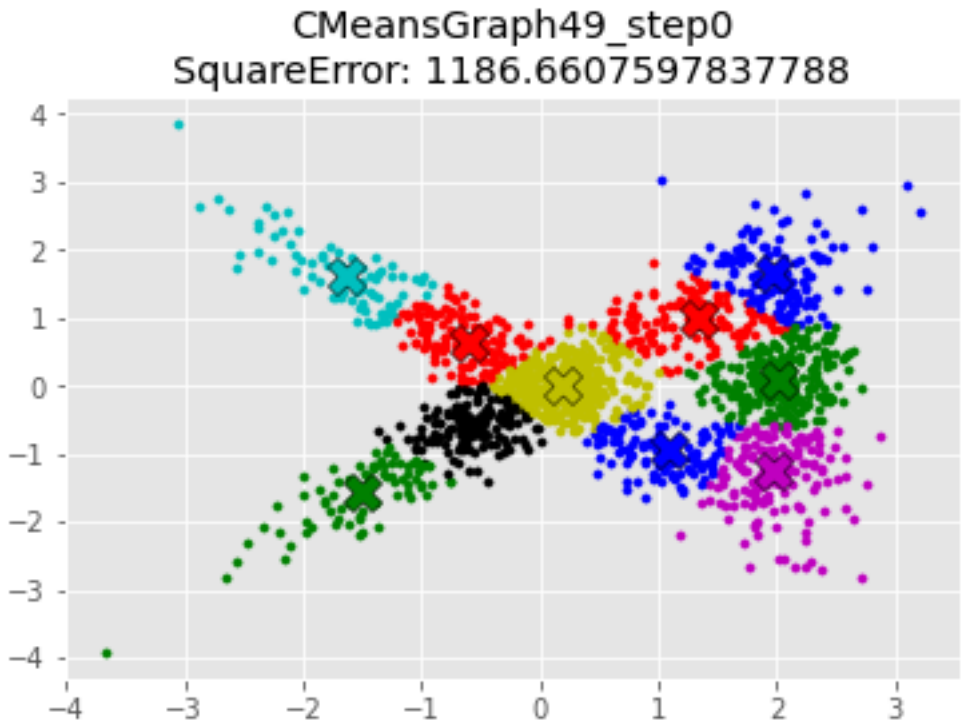

CMeansGraph44_step0
SquareError: 1188.1656820666337
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

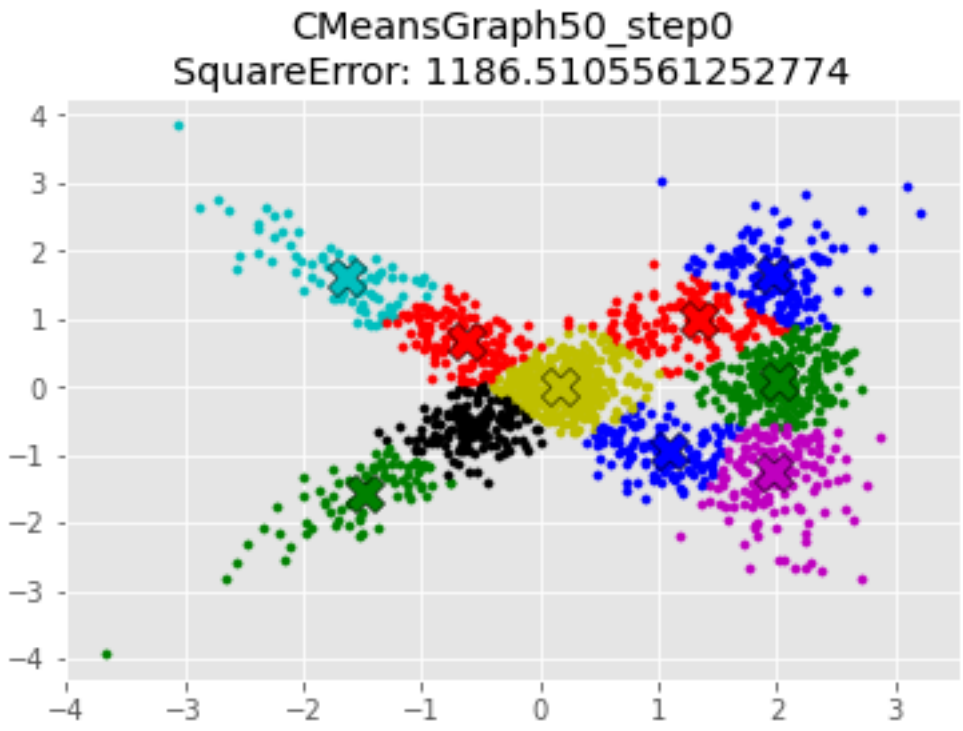

CMeansGraph45_step0
SquareError: 1187.7427016045176
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

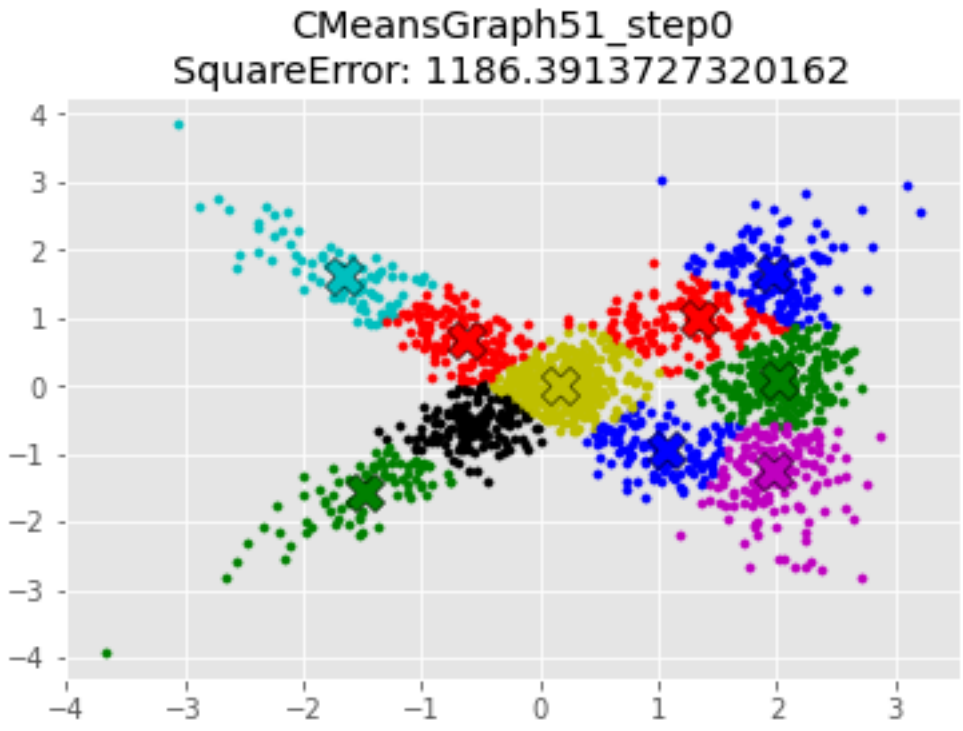

CMeansGraph46_step0
SquareError: 1187.3675088662487
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

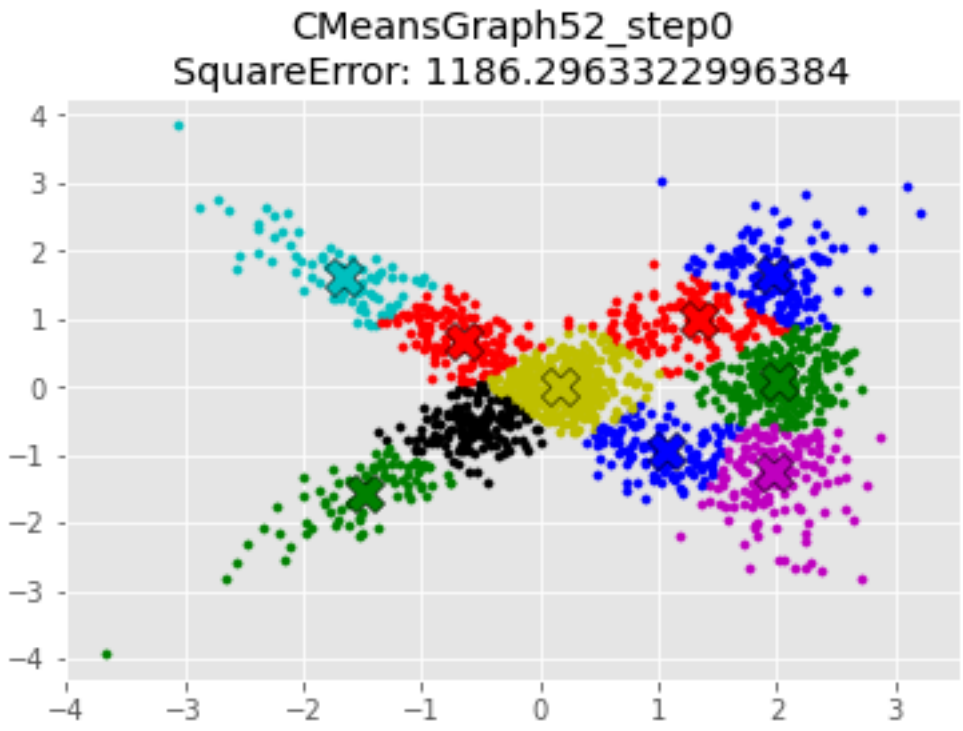

CMeansGraph47_step0
SquareError: 1187.0733672702293
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

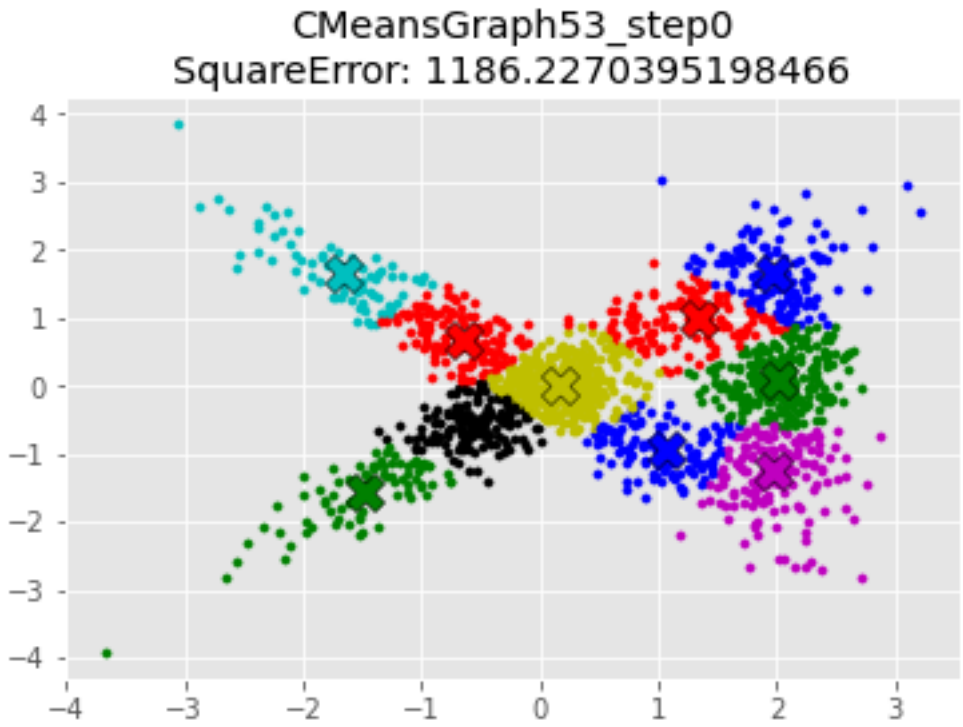

CMeansGraph48_step0
SquareError: 1186.8447682617755
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

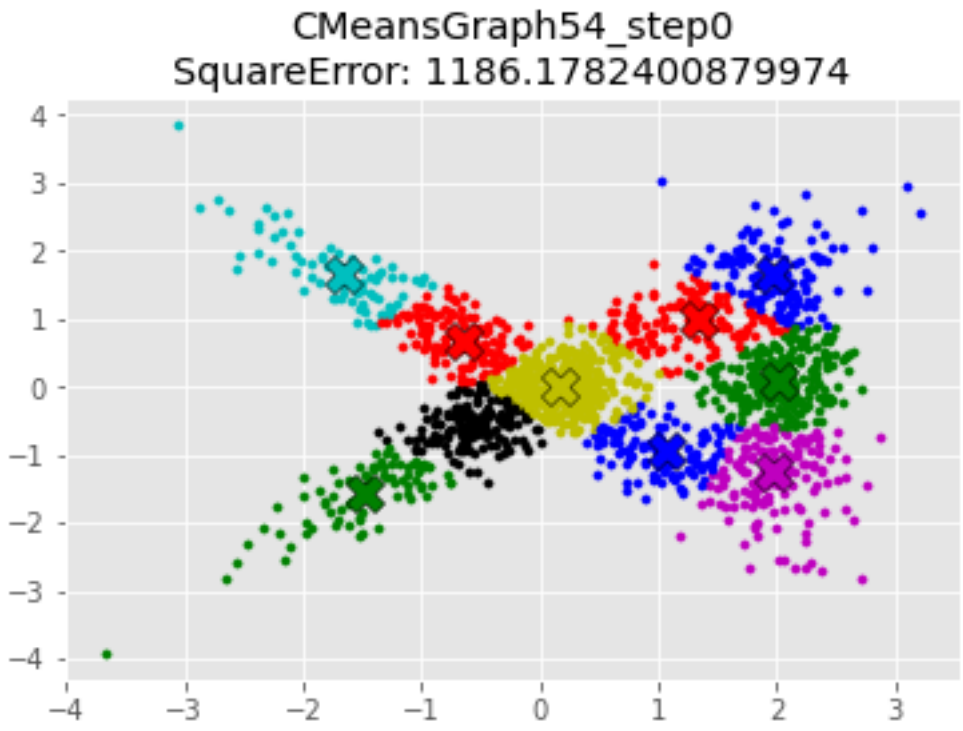

CMeansGraph49_step0
SquareError: 1186.6607597837788
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

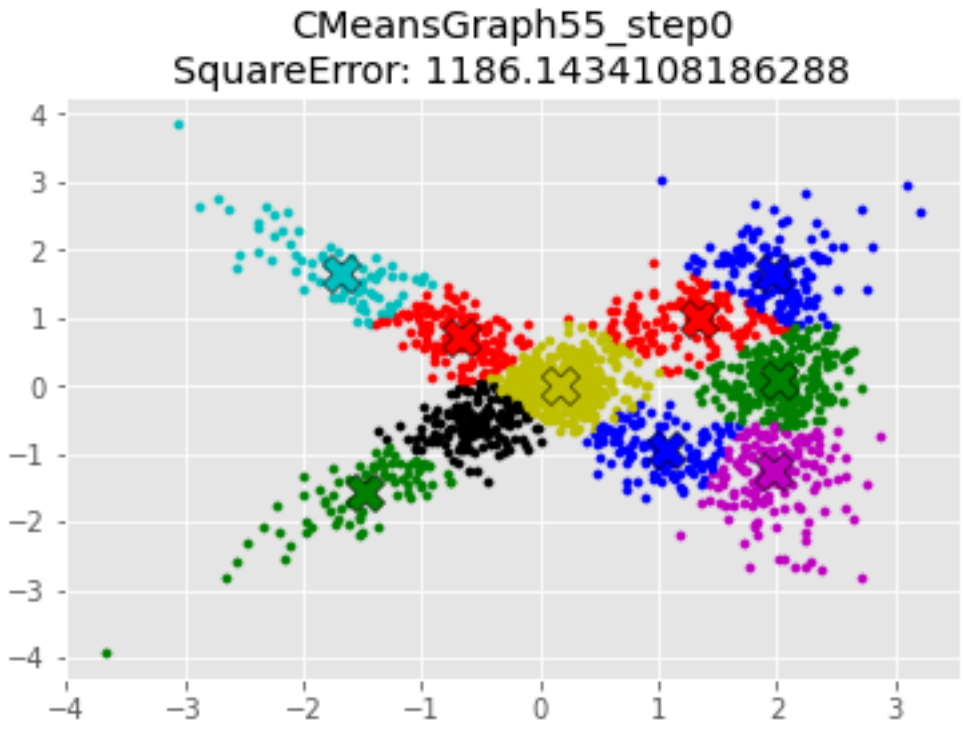

CMeansGraph50_step0
SquareError: 1186.5105561252774

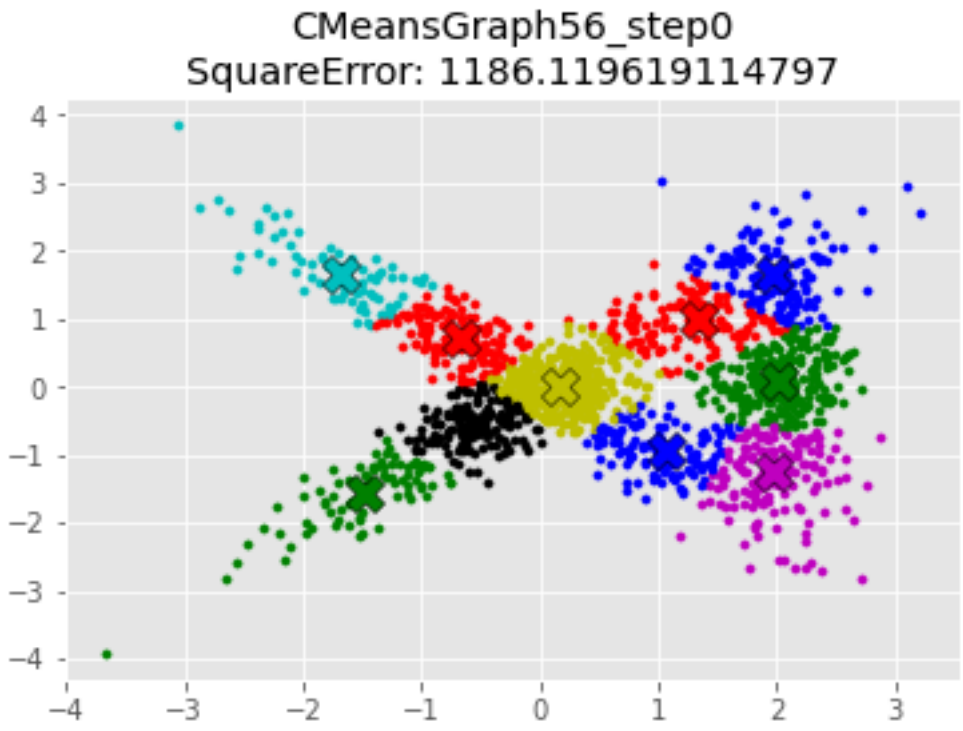

CMeansGraph51_step0
SquareError: 1186.3913727320162

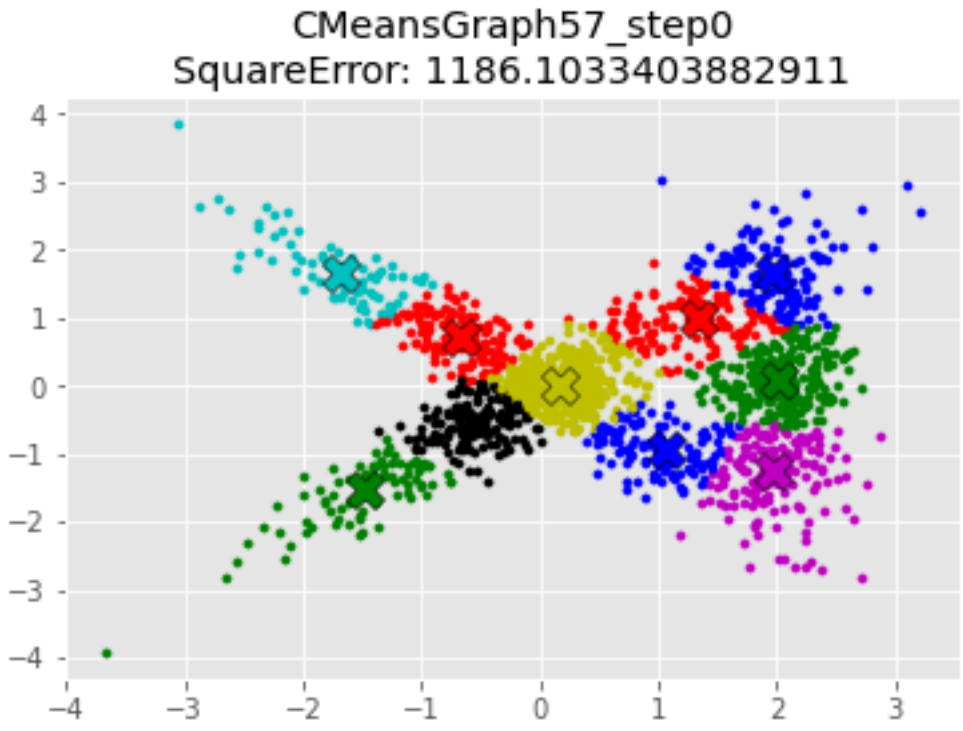

CMeansGraph52_step0
SquareError: 1186.2963322996384
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

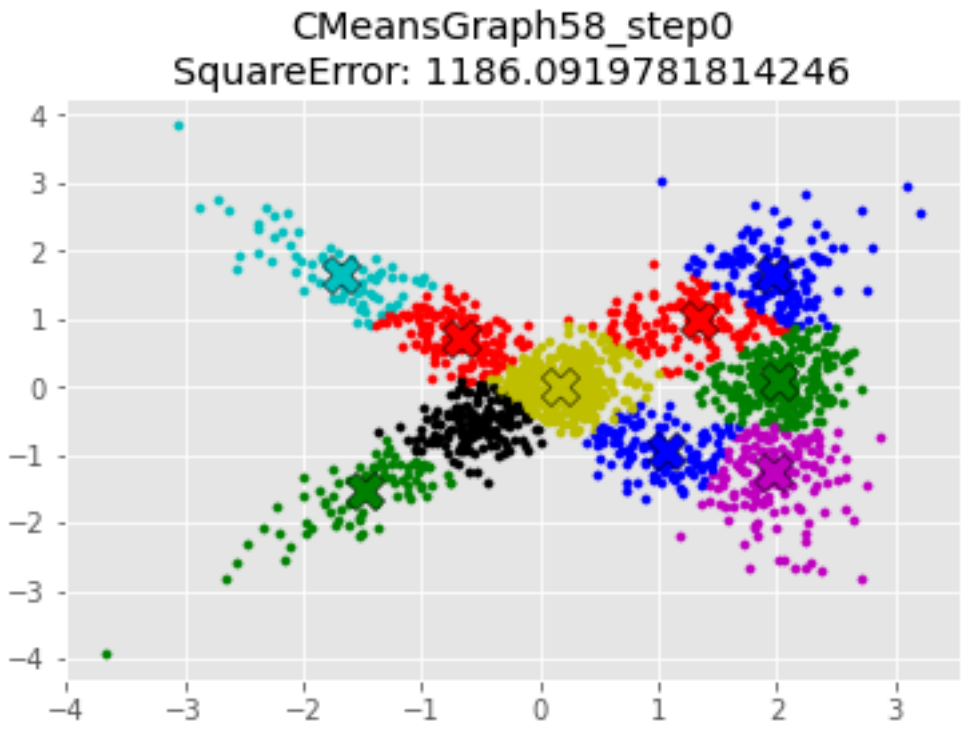

CMeansGraph53_step0
SquareError: 1186.2270395198466
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

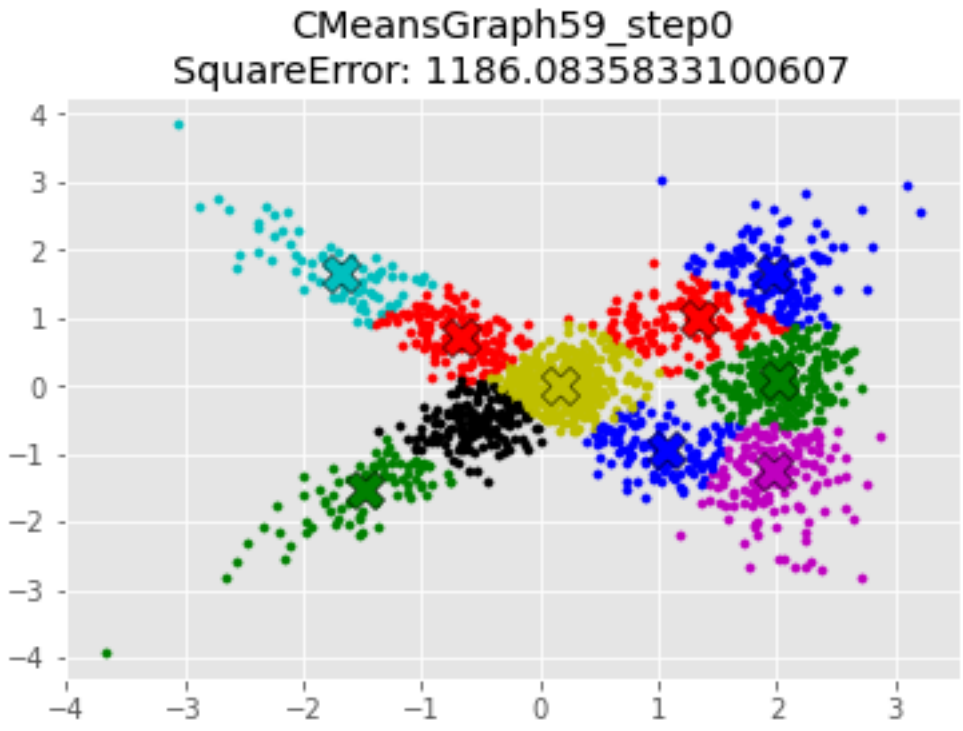

CMeansGraph54_step0
SquareError: 1186.1782400879974
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

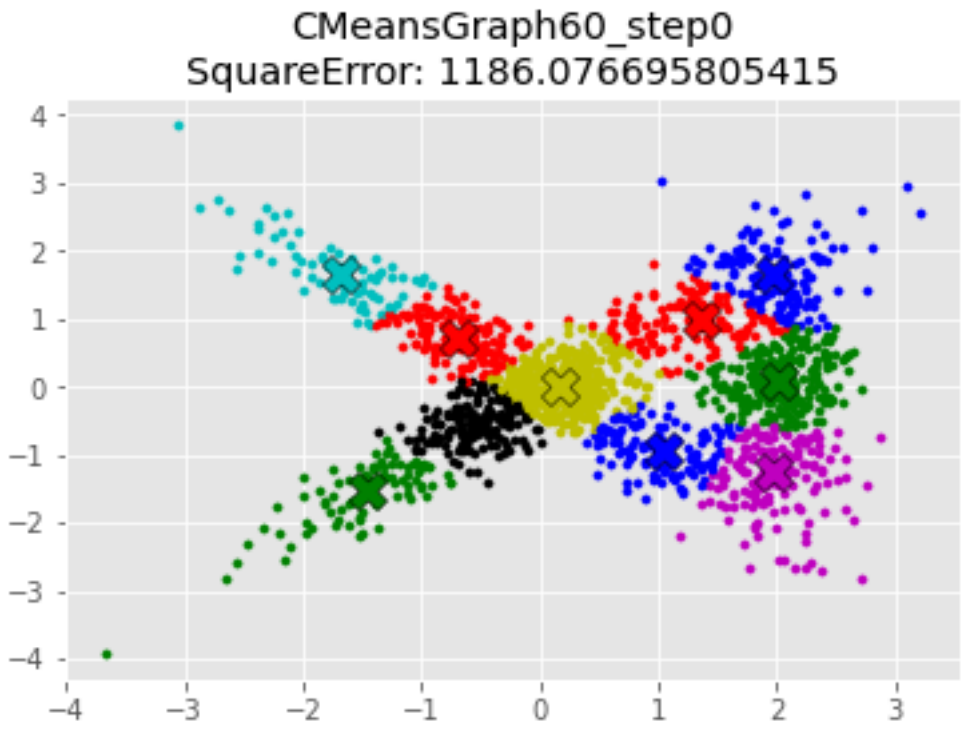

CMeansGraph55_step0
SquareError: 1186.1434108186288
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

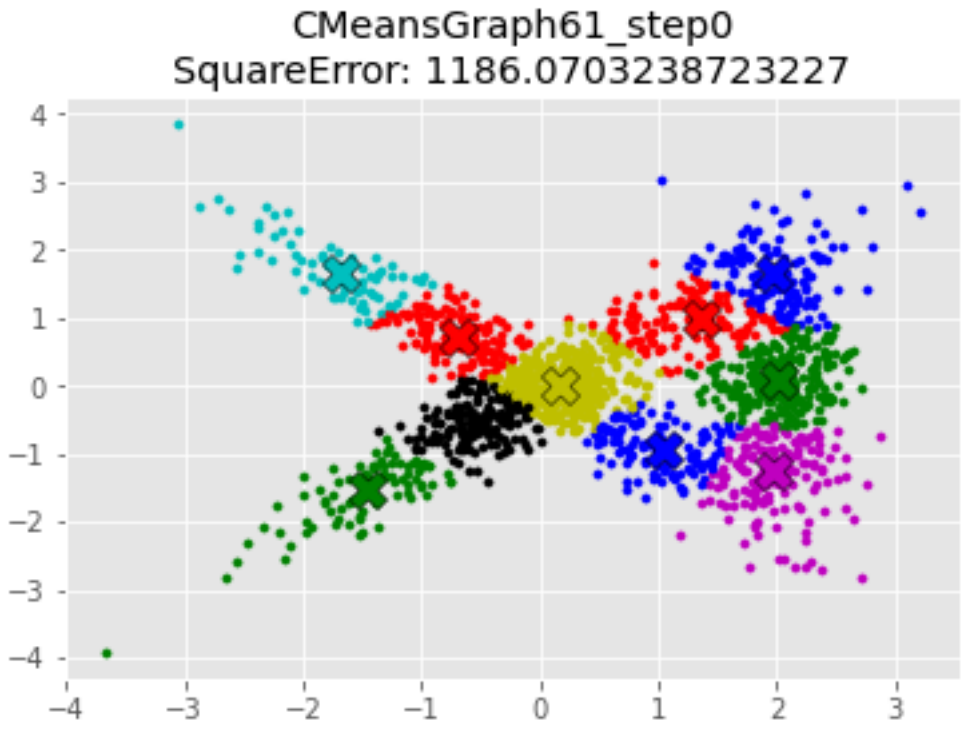

CMeansGraph56_step0
SquareError: 1186.119619114797

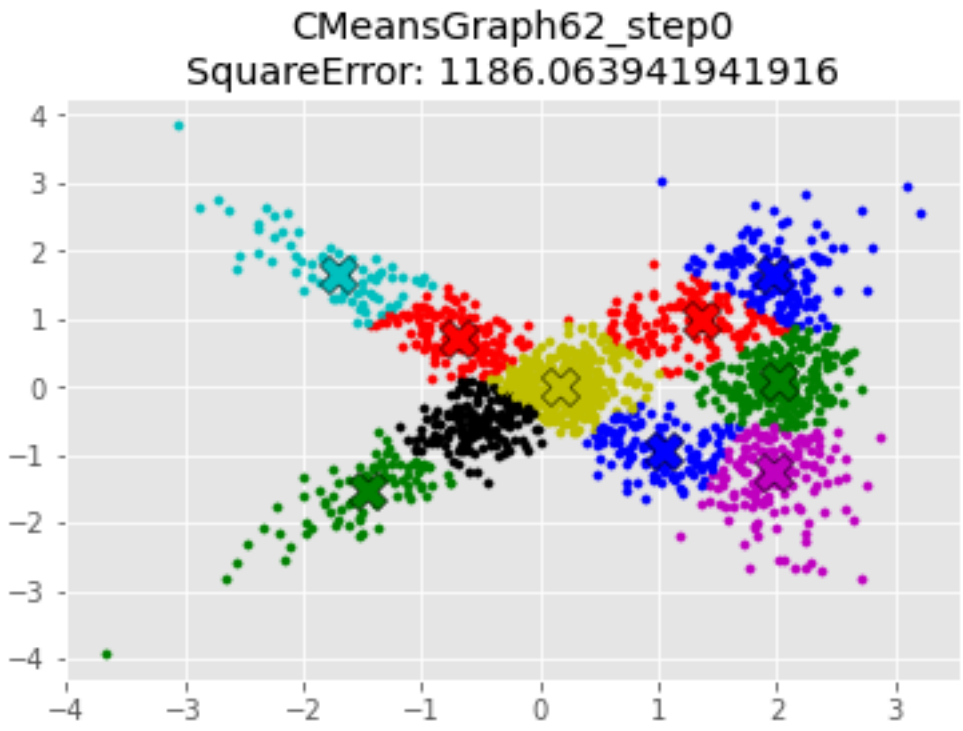

CMeansGraph57_step0
SquareError: 1186.1033403882911

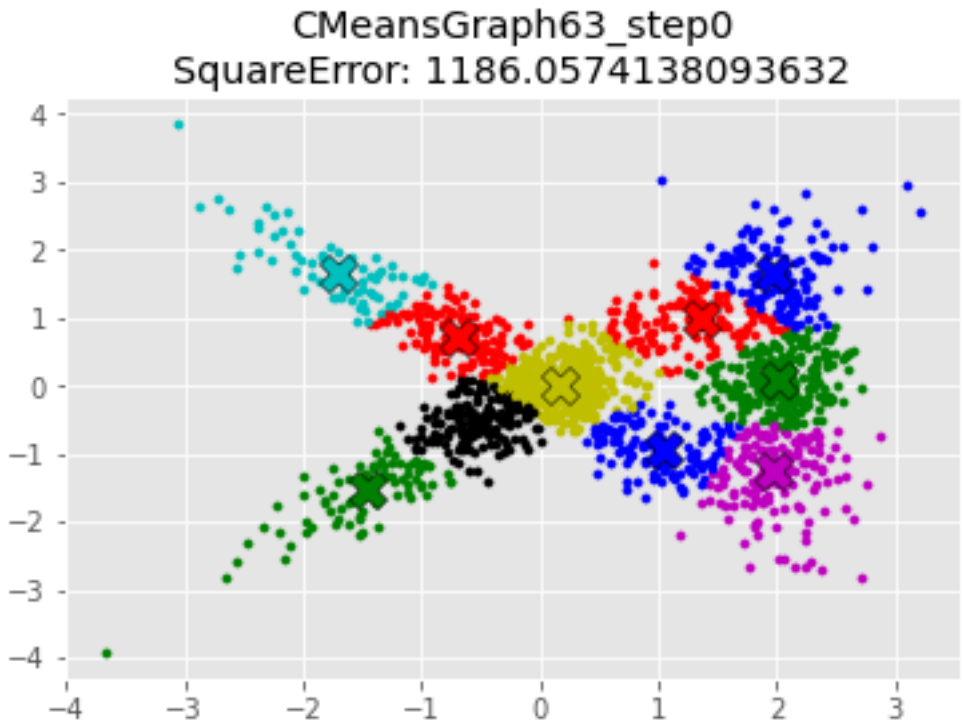

CMeansGraph58_step0
SquareError: 1186.0919781814246

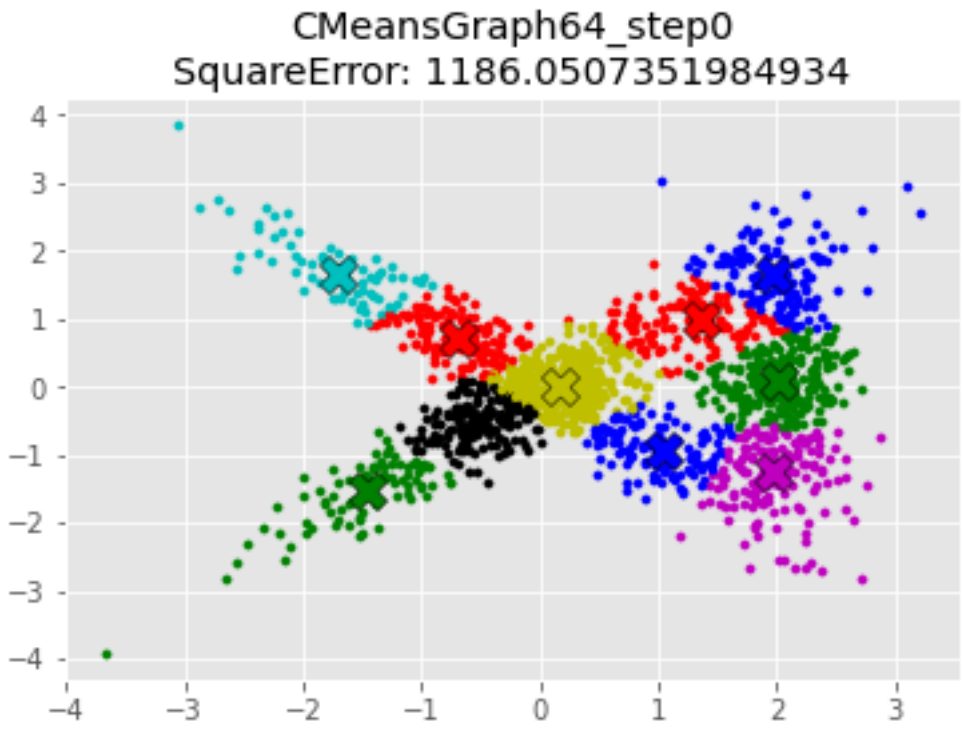

CMeansGraph59_step0
SquareError: 1186.0835833100607

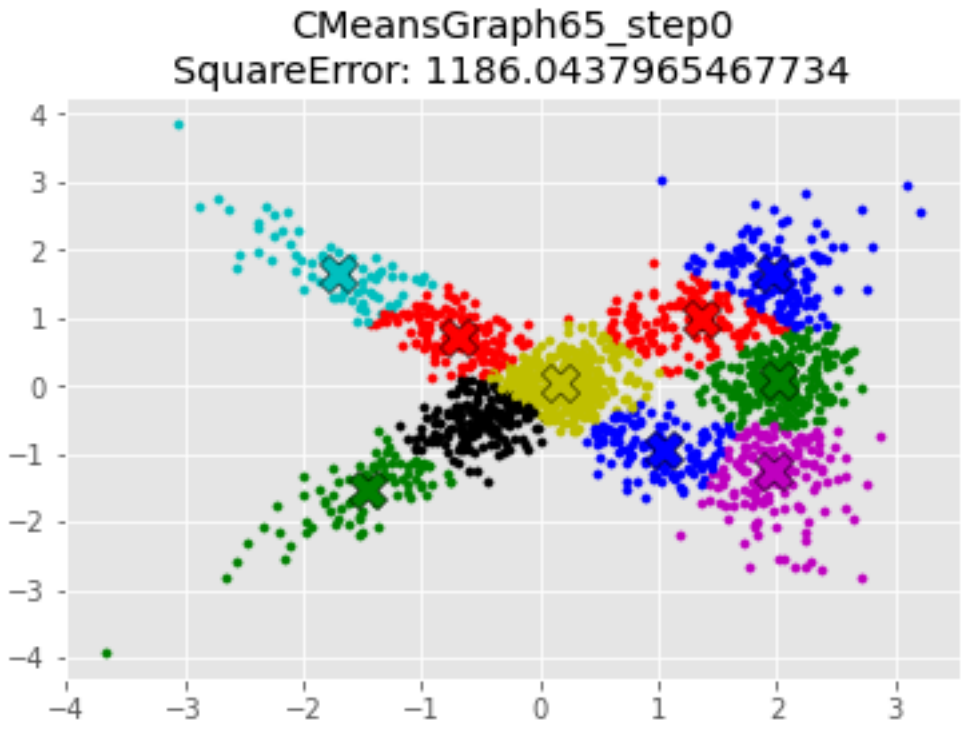

CMeansGraph60_step0
SquareError: 1186.076695805415
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

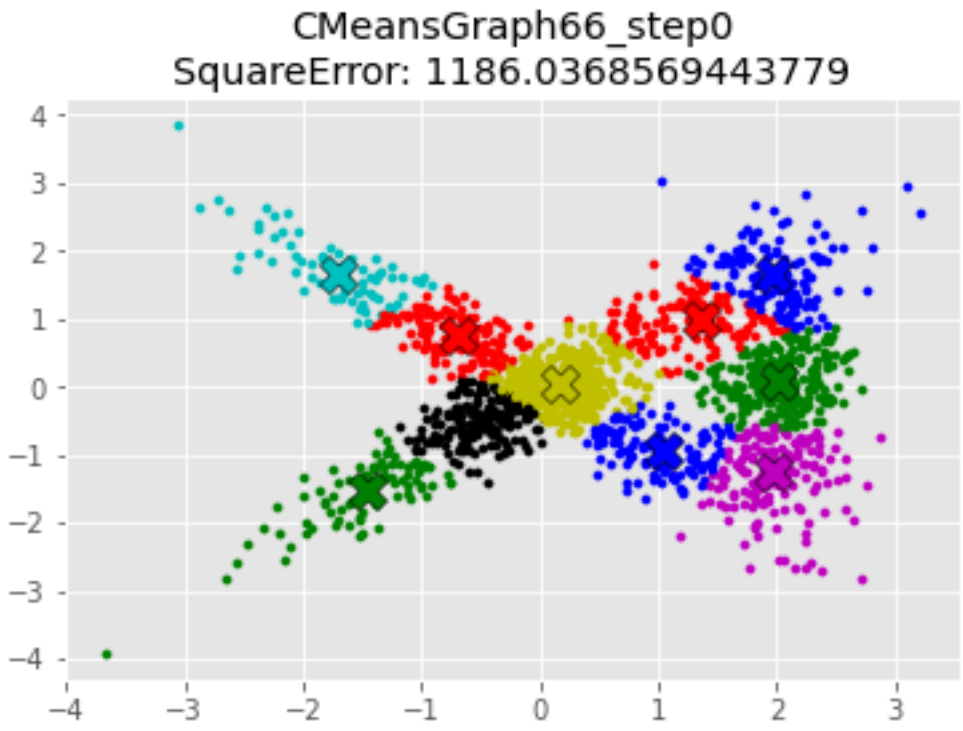

CMeansGraph61_step0
SquareError: 1186.0703238723227
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

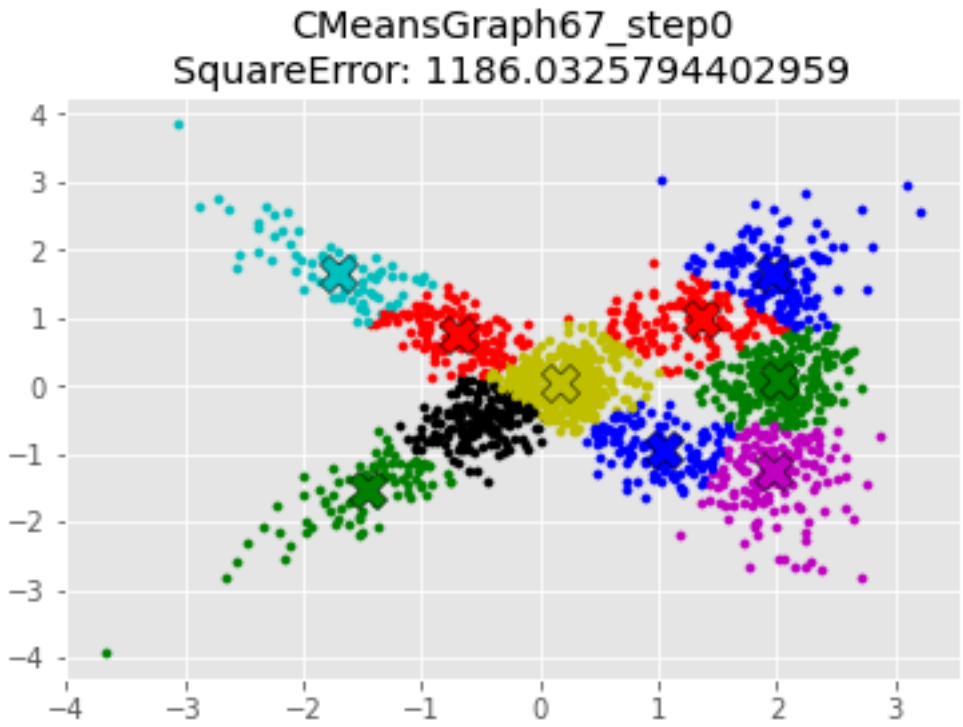

CMeansGraph62_step0
SquareError: 1186.063941941916
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

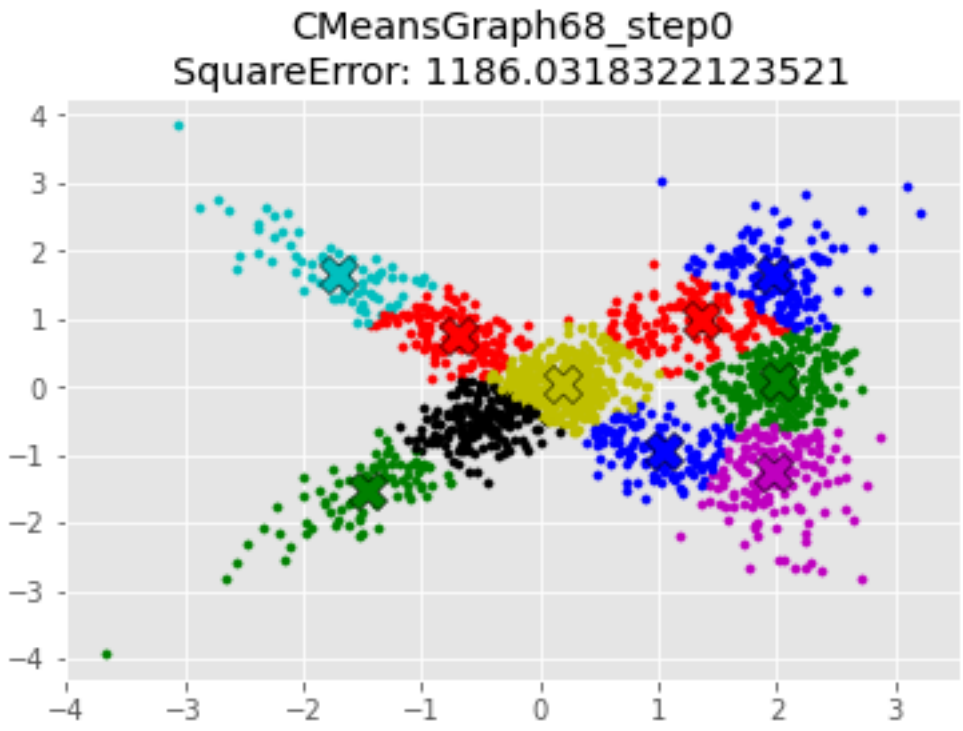

CMeansGraph63_step0
SquareError: 1186.0574138093632
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3


CMeansGraph64_step0
SquareError: 1186.0507351984934
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

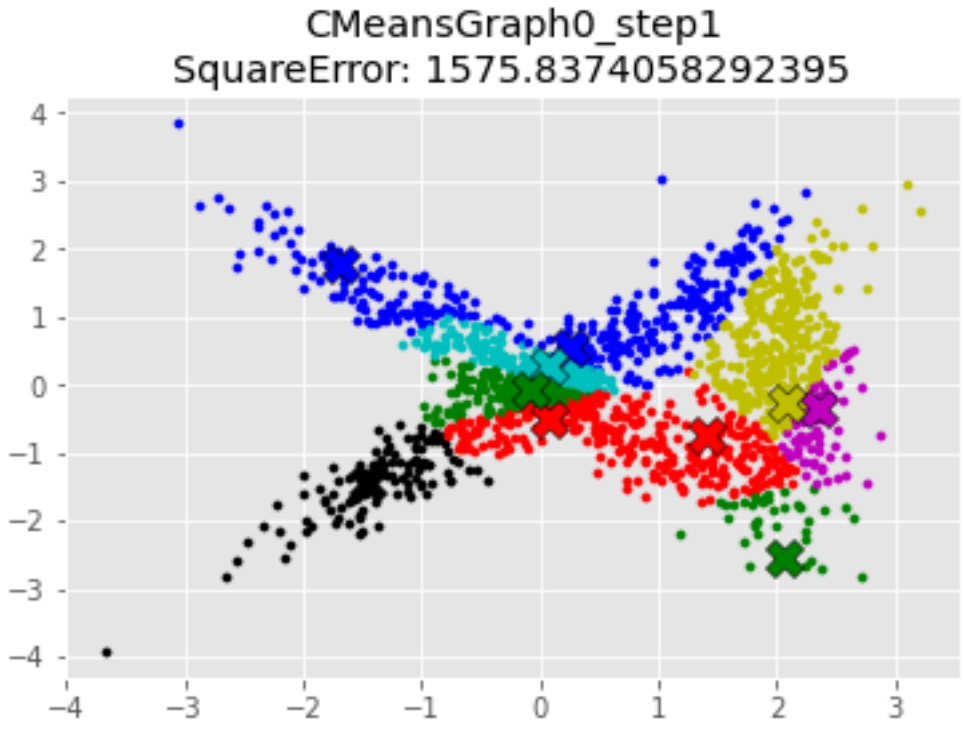

CMeansGraph65_step0
SquareError: 1186.0437965467734
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

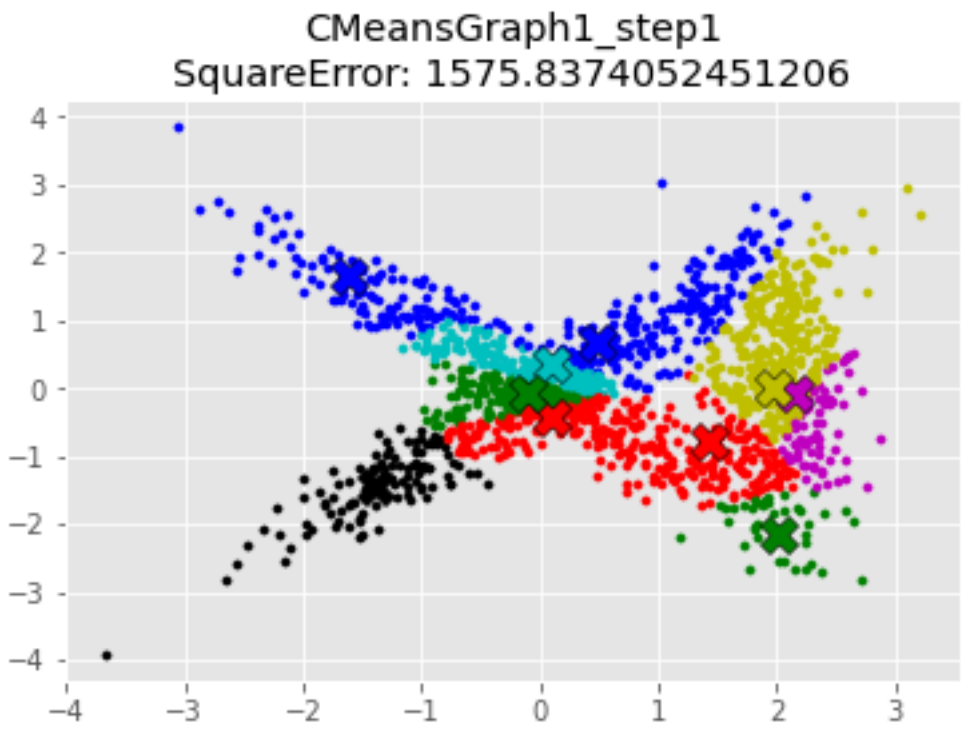

CMeansGraph66_step0
SquareError: 1186.0368569443779
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

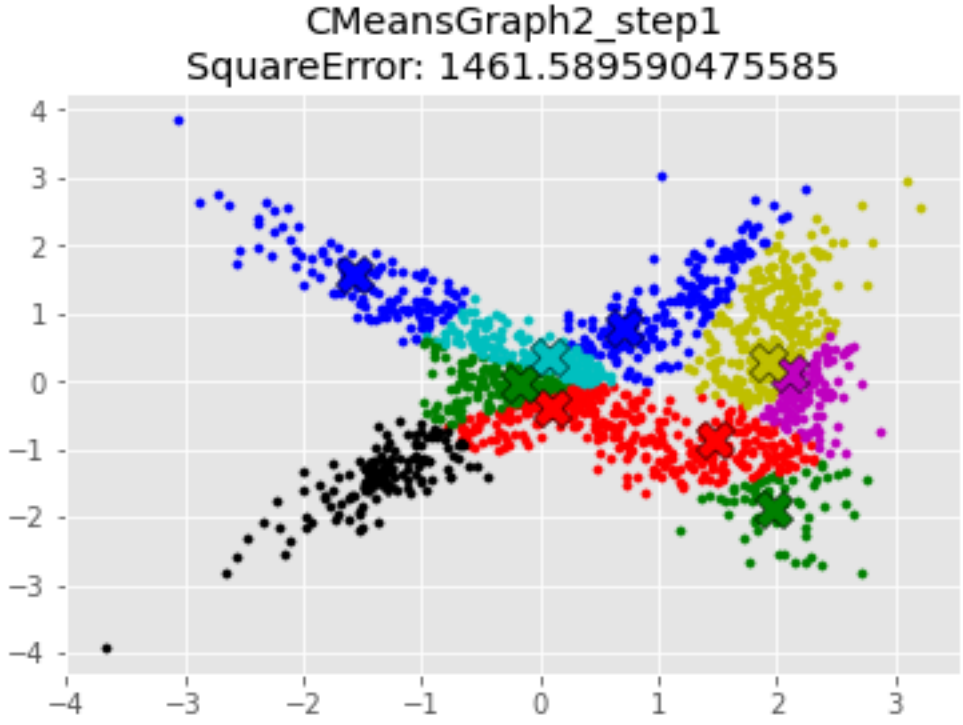

CMeansGraph67_step0
SquareError: 1186.0325794402959
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

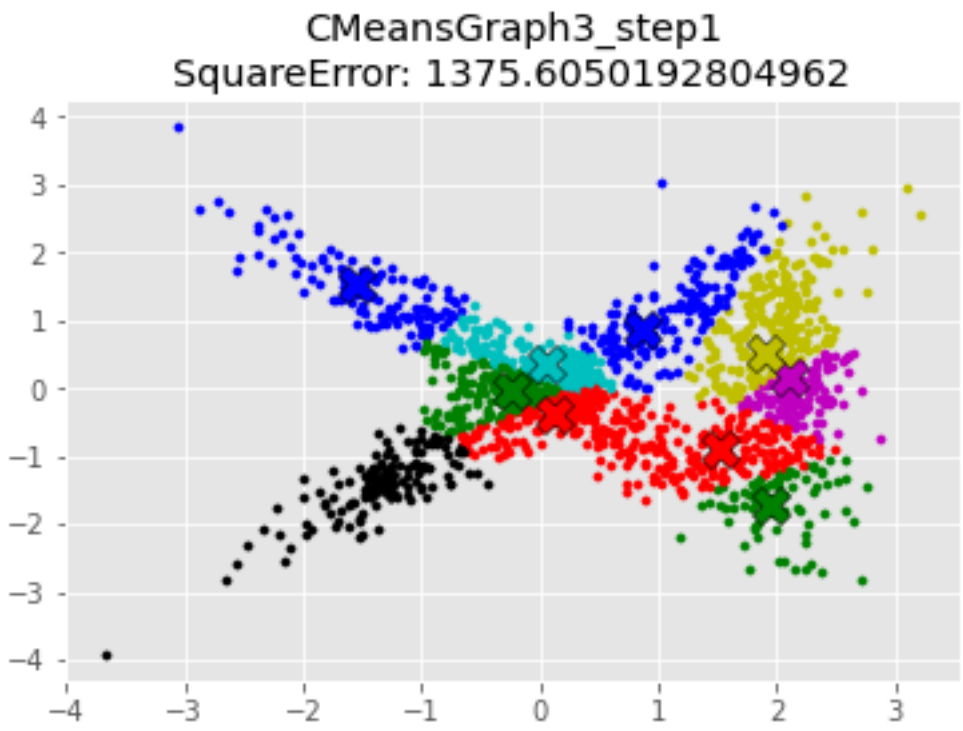


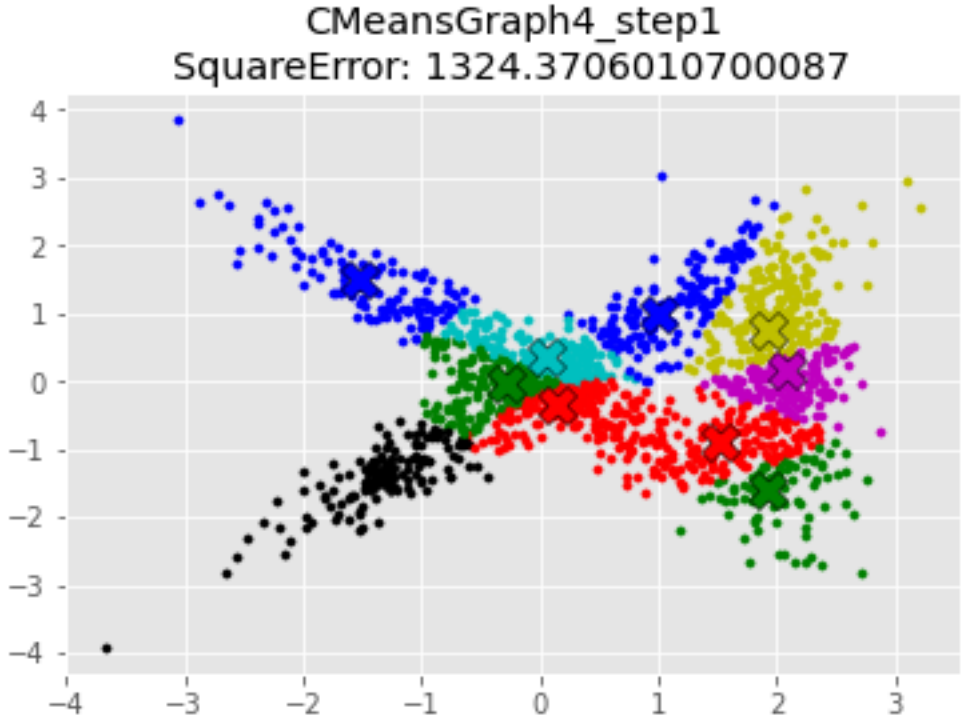

```
Epsilon reached, stopping early
New best square error: 1
```

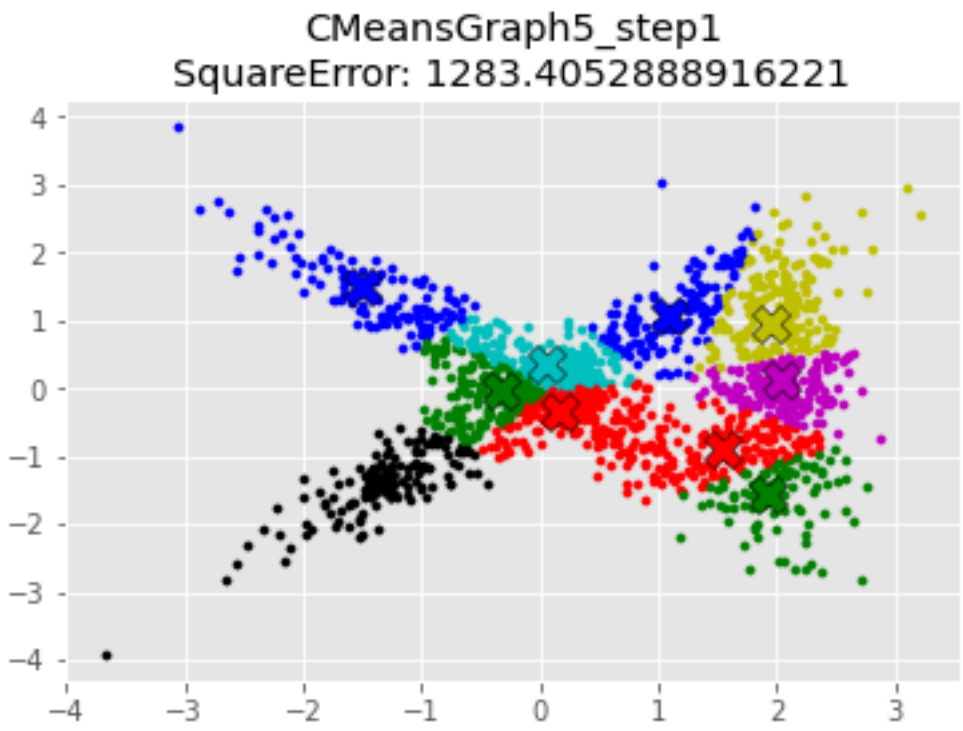


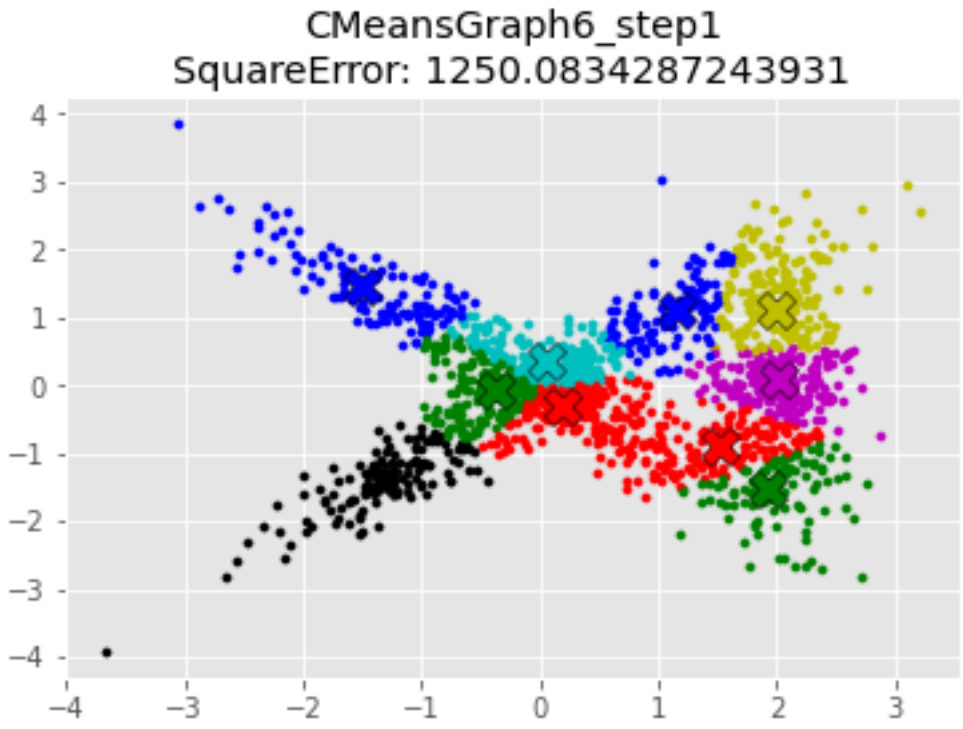

CMeansGraph1_step1
SquareError: 1575.8374052451206
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

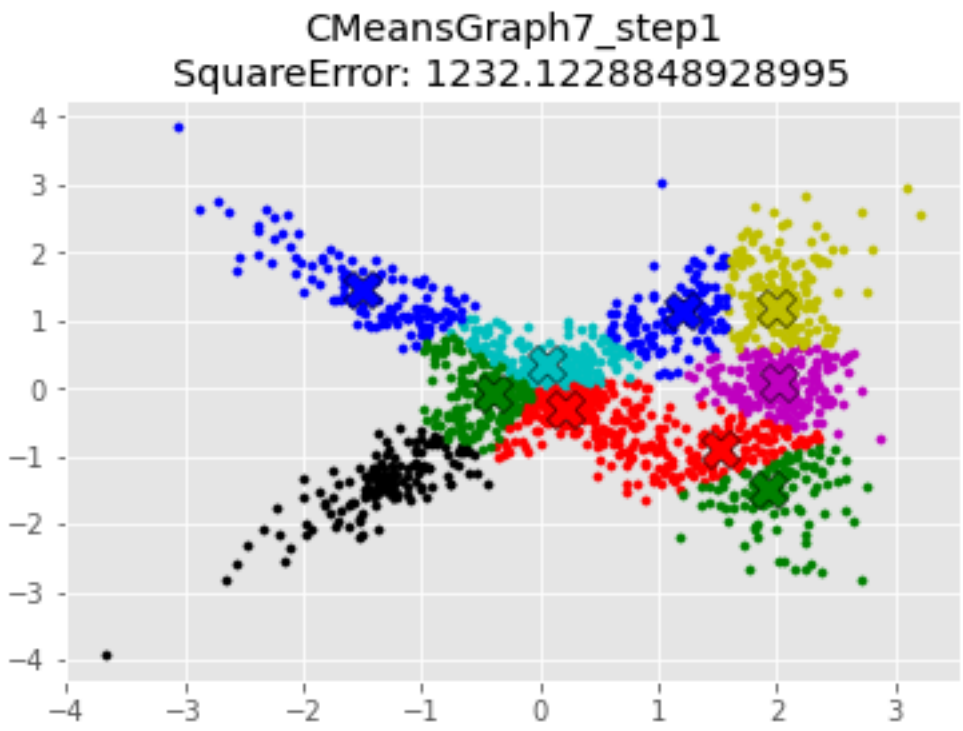

CMeansGraph2_step1
SquareError: 1461.589590475585
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

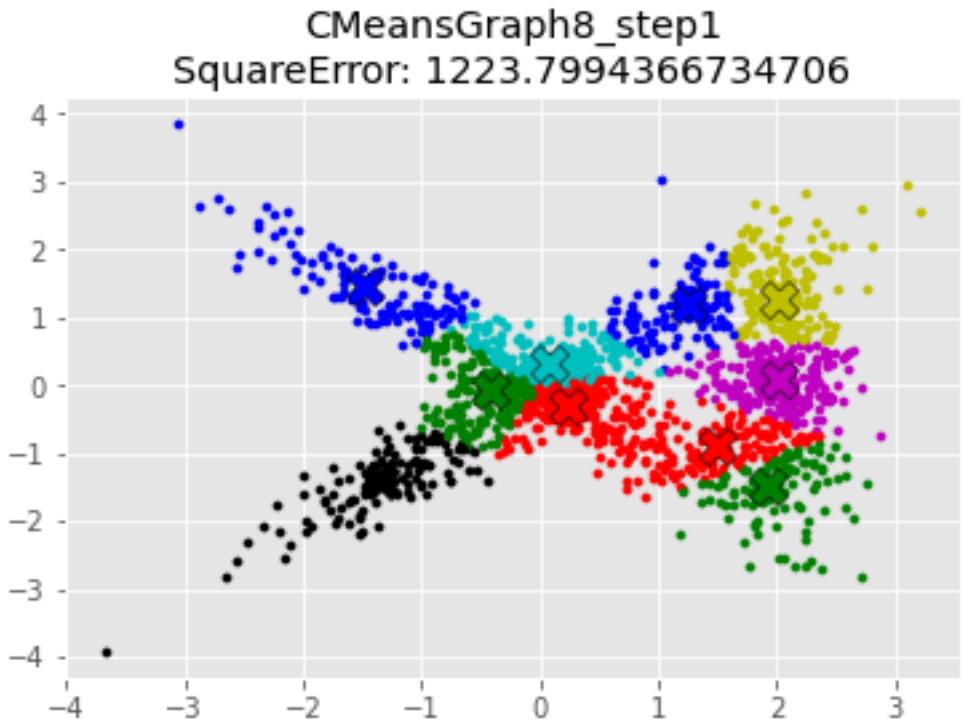

CMeansGraph3_step1
SquareError: 1375.6050192804962
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

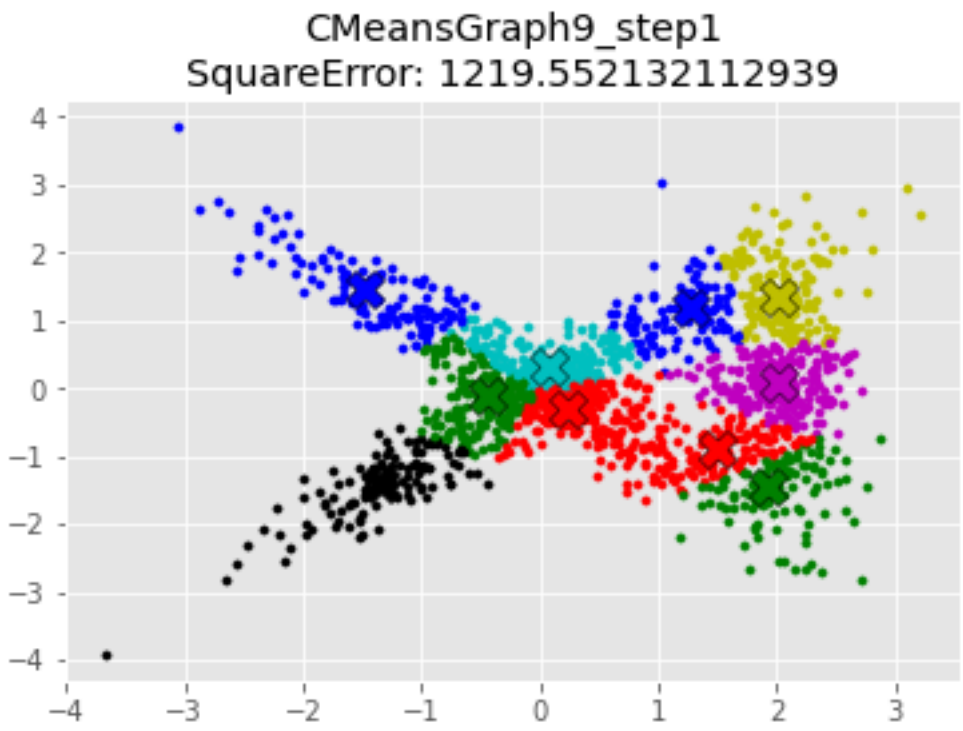

CMeansGraph4_step1
SquareError: 1324.3706010700087
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

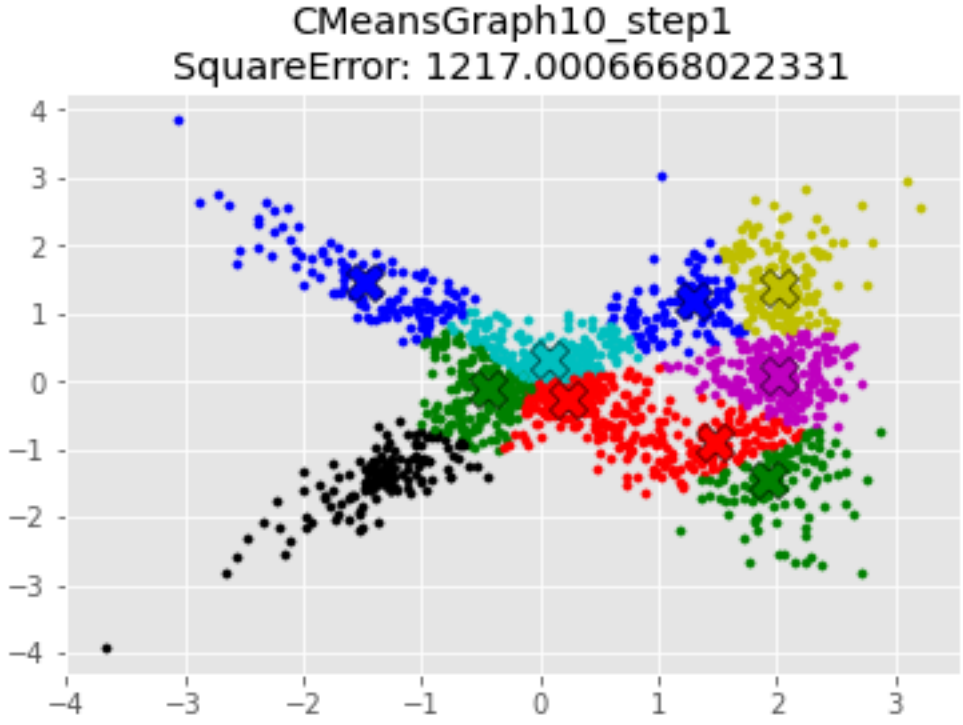

CMeansGraph5_step1
SquareError: 1283.4052888916221

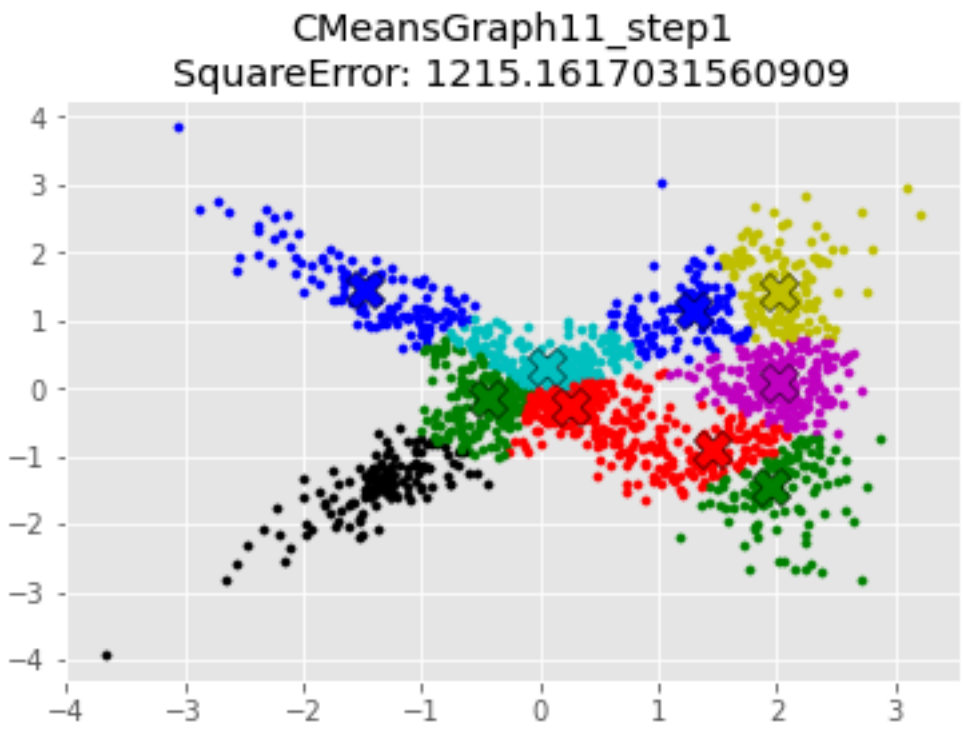

CMeansGraph6_step1
SquareError: 1250.0834287243931

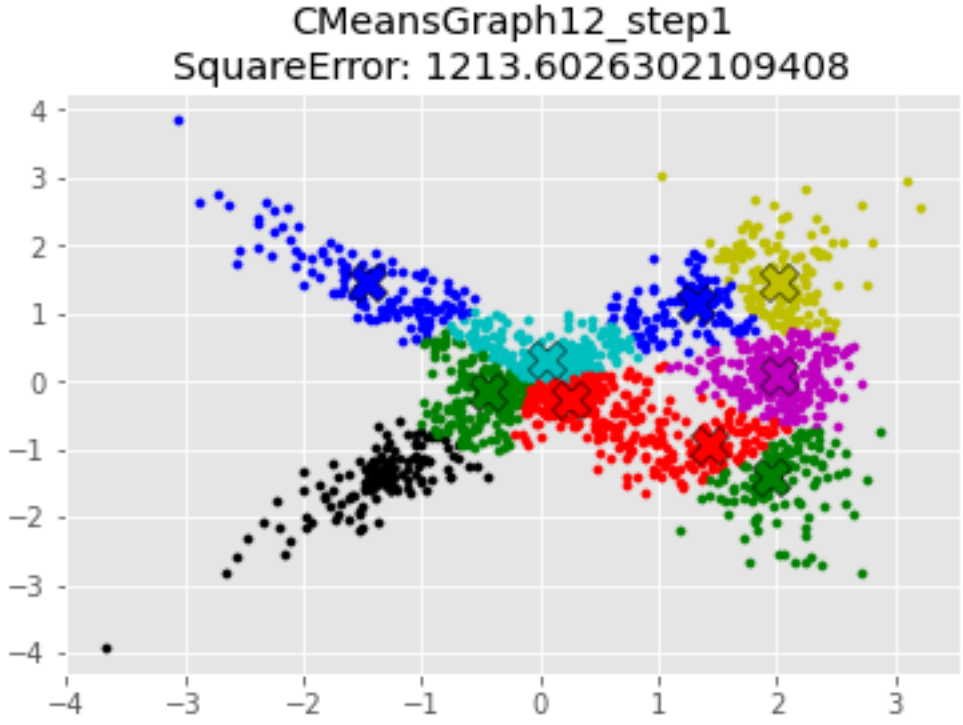

CMeansGraph7_step1
SquareError: 1232.1228848928995
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

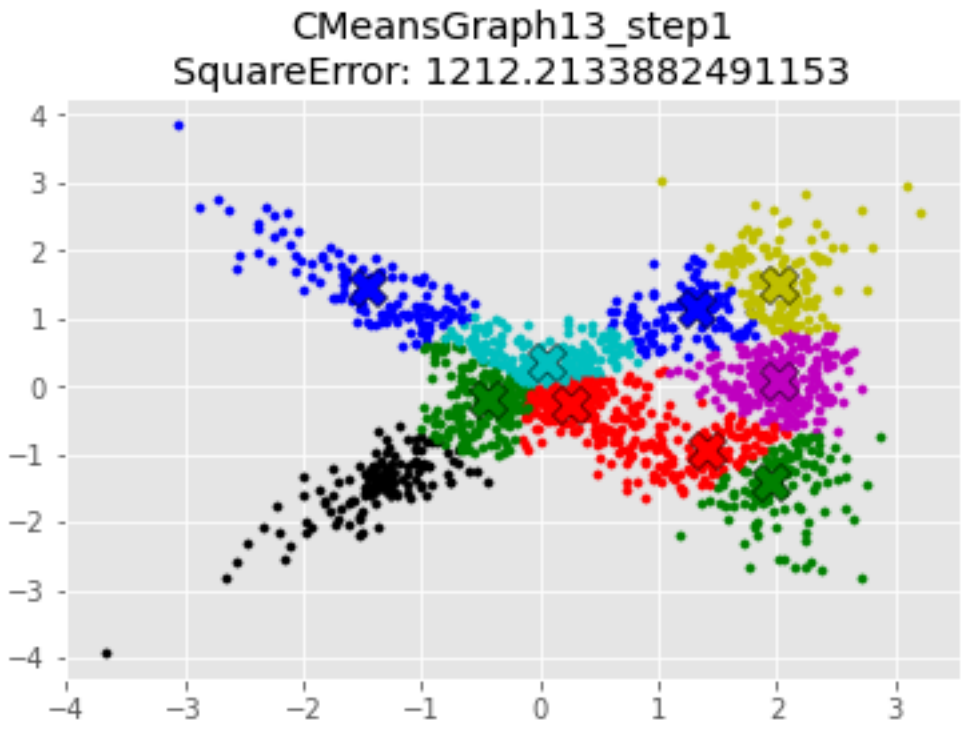

CMeansGraph8_step1
SquareError: 1223.7994366734706
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

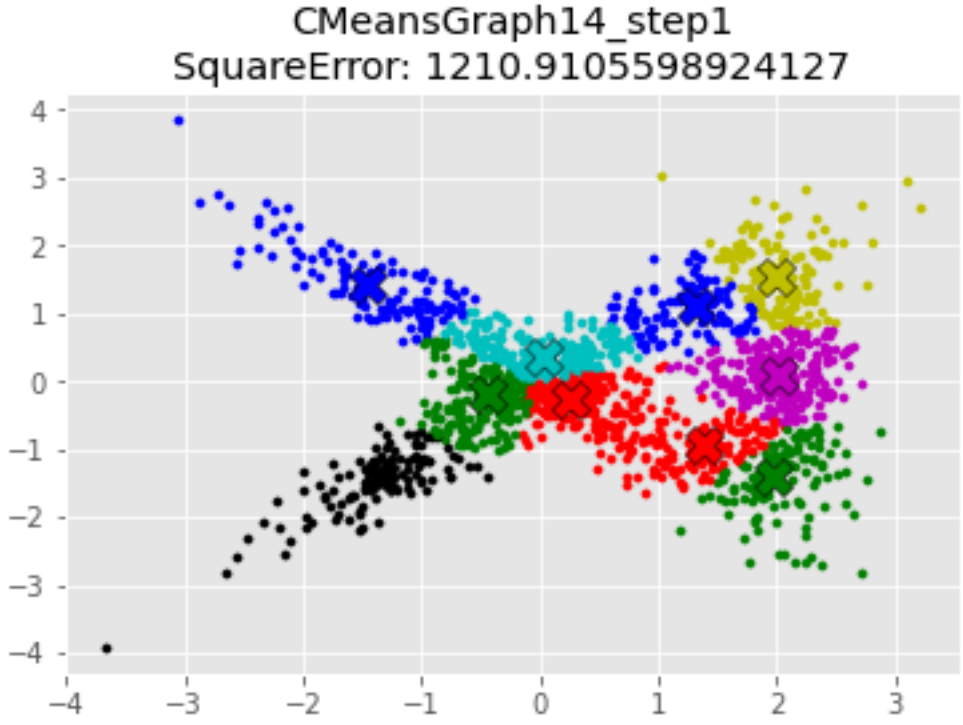

CMeansGraph9_step1
SquareError: 1219.552132112939
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

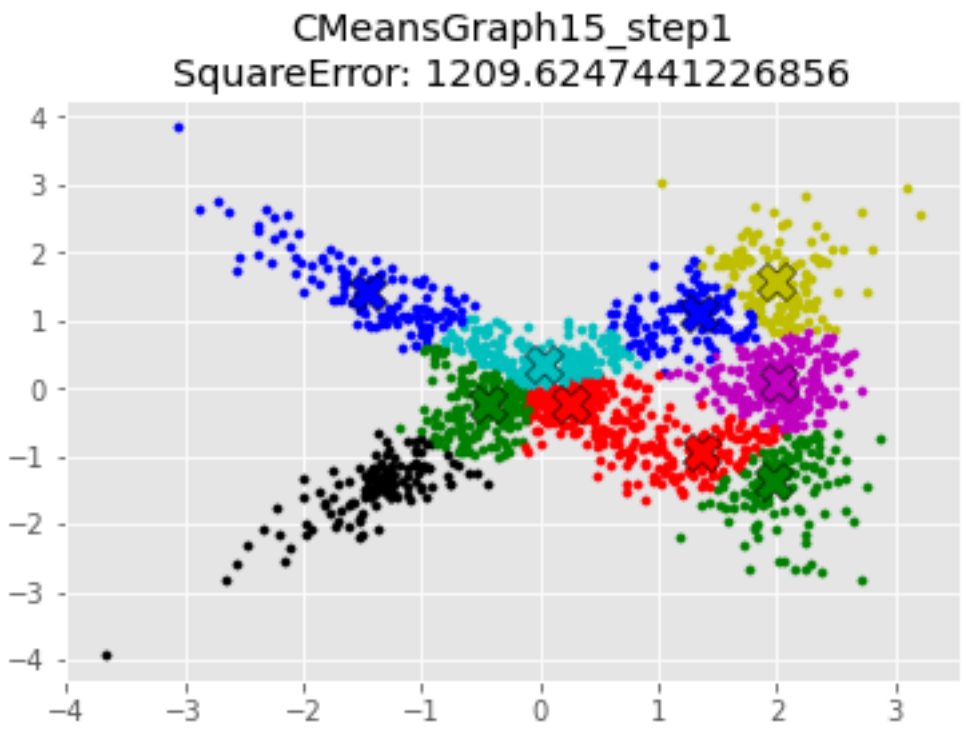

CMeansGraph10_step1
SquareError: 1217.0006668022331
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

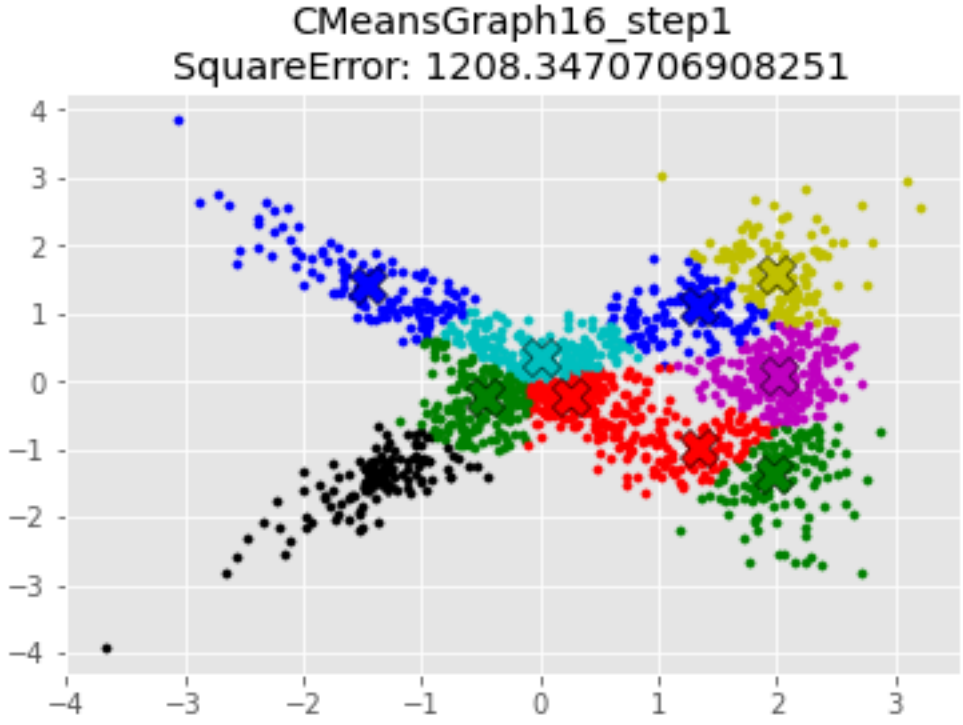

CMeansGraph11_step1
SquareError: 1215.1617031560909
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

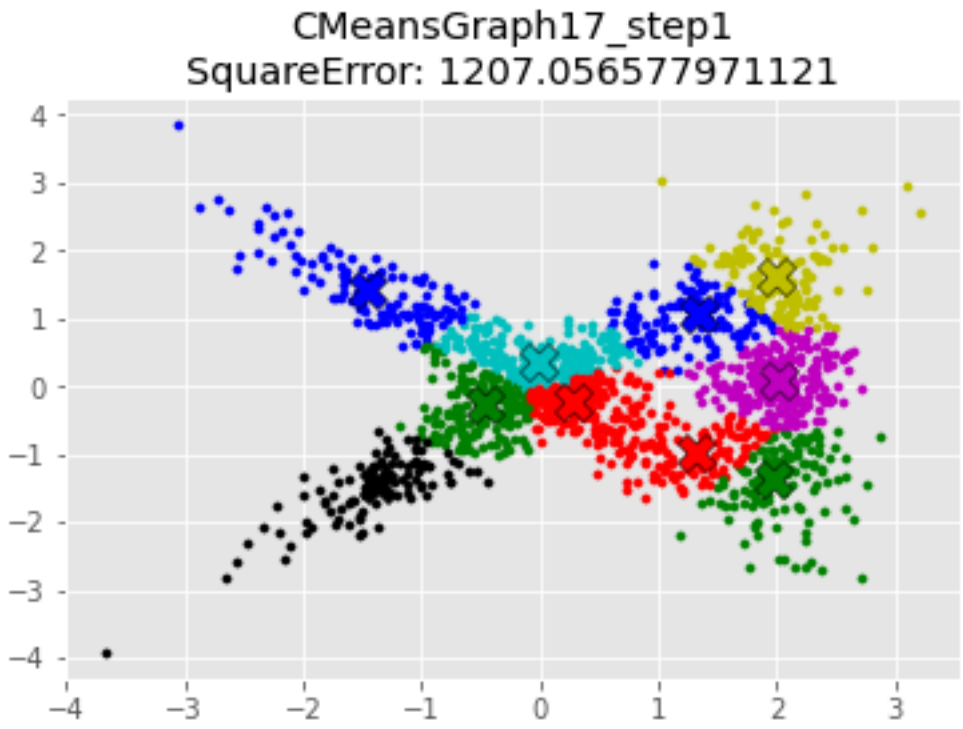

CMeansGraph12_step1
SquareError: 1213.6026302109408
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

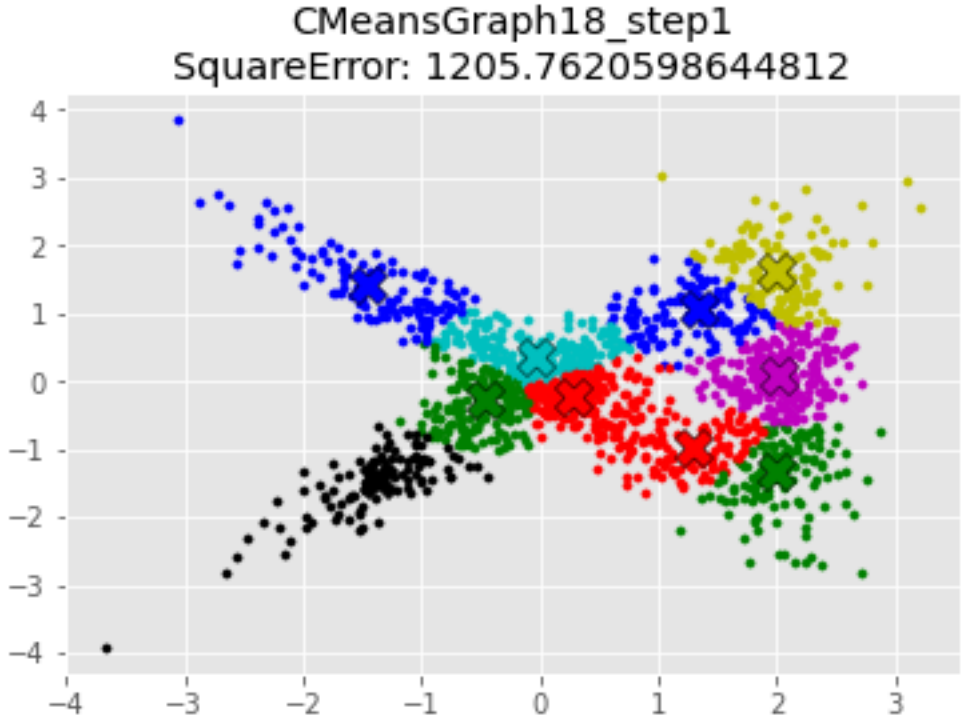

CMeansGraph13_step1
SquareError: 1212.2133882491153
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

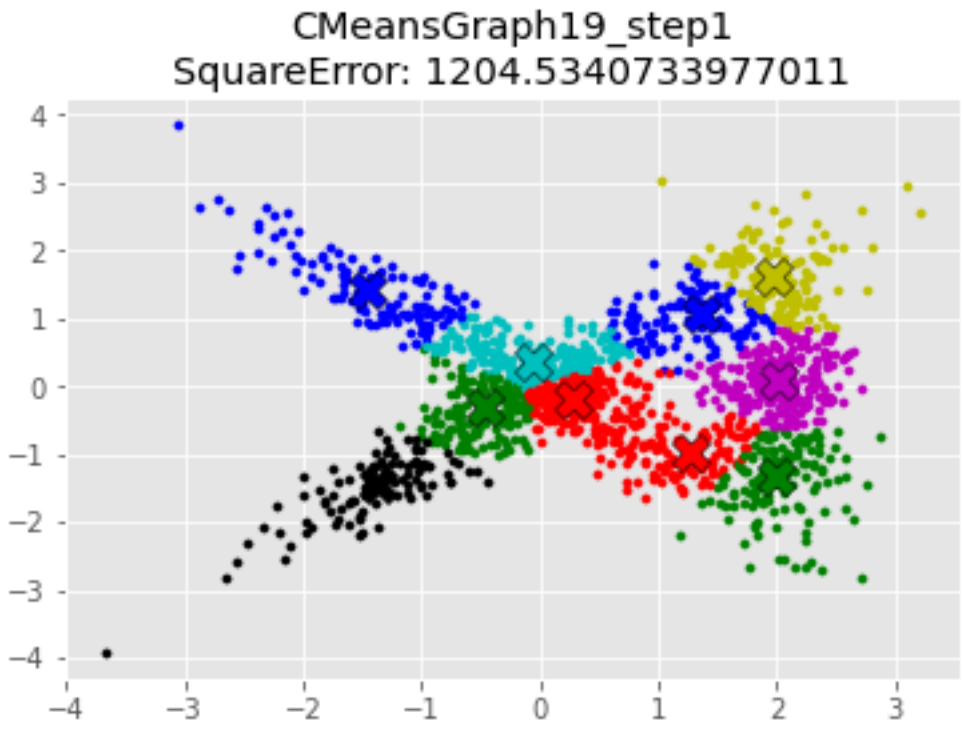

CMeansGraph14_step1
SquareError: 1210.9105598924127
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

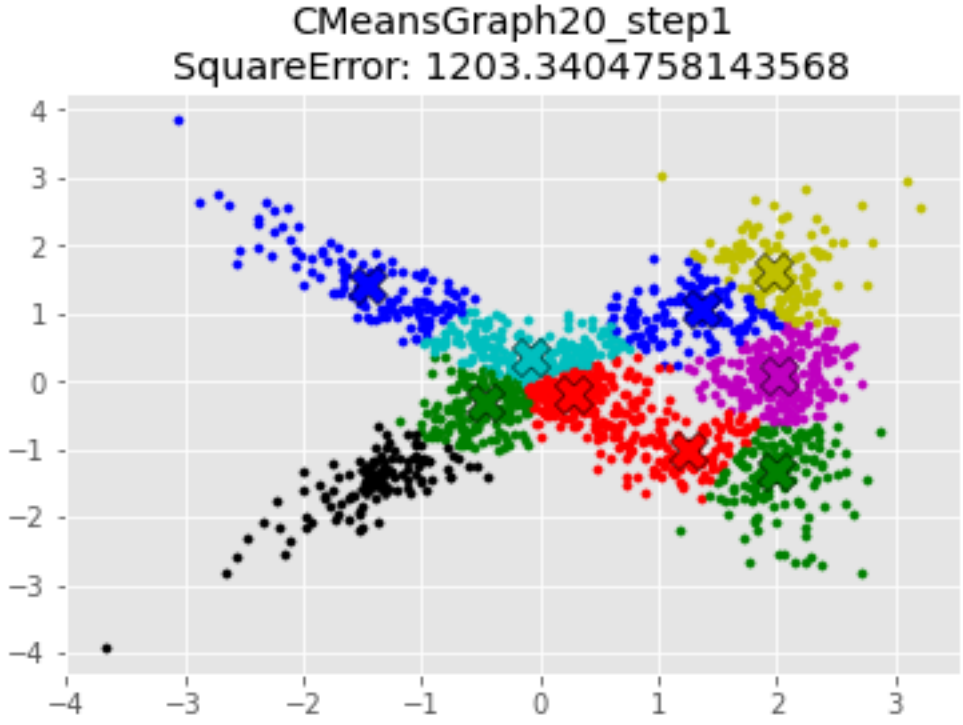

CMeansGraph15_step1
SquareError: 1209.6247441226856
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

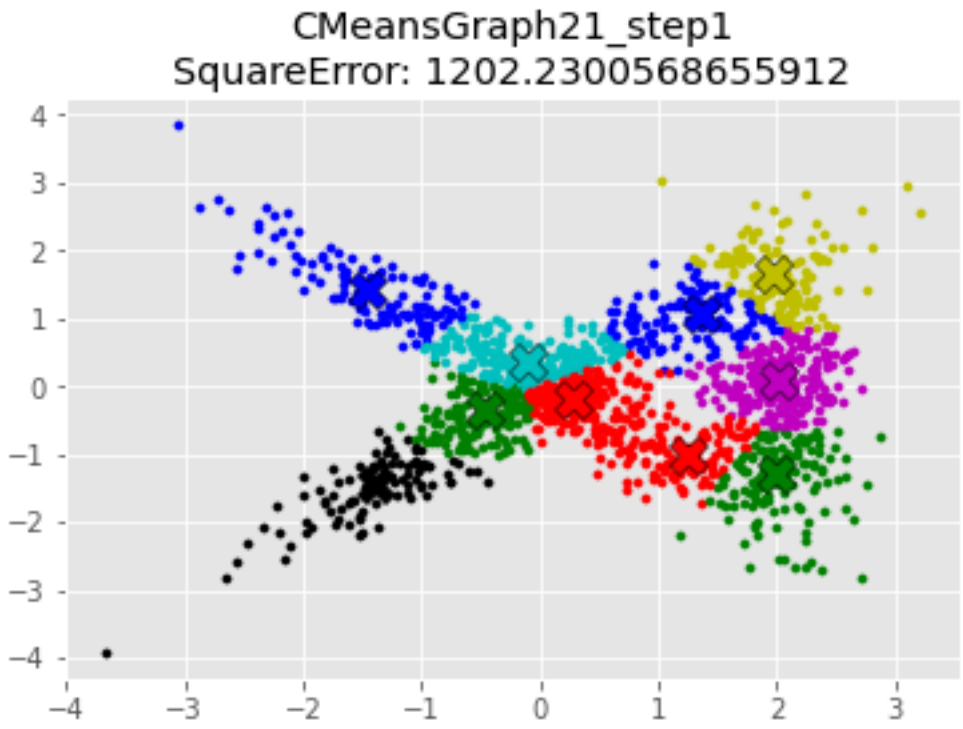

CMeansGraph16_step1
SquareError: 1208.3470706908251
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

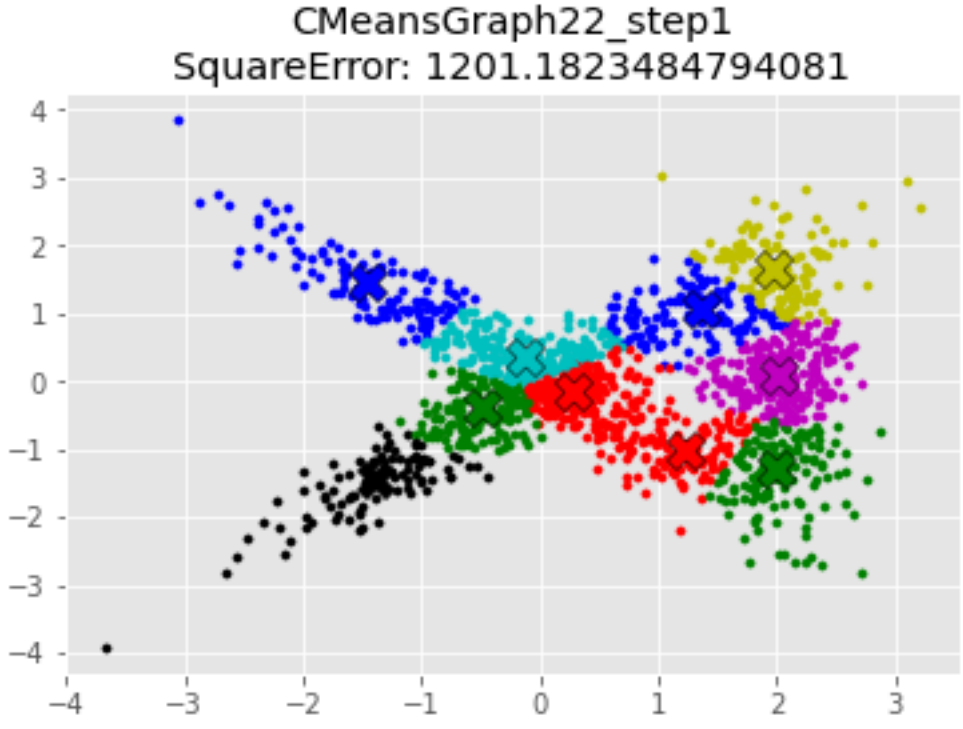

CMeansGraph17_step1
SquareError: 1207.056577971121
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

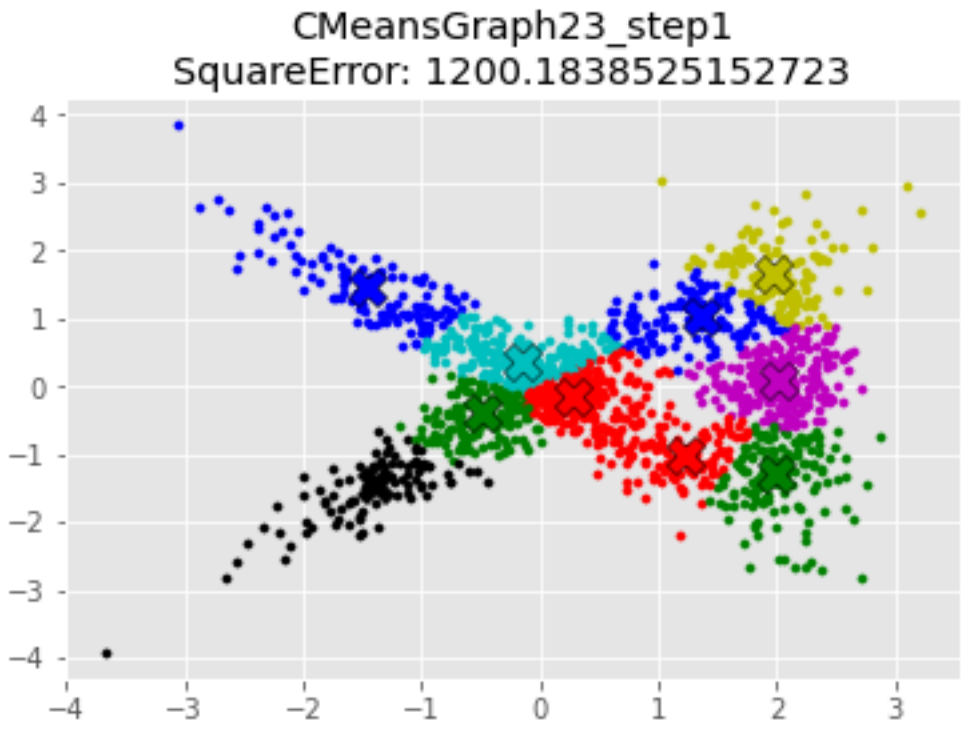

CMeansGraph18_step1
SquareError: 1205.7620598644812
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

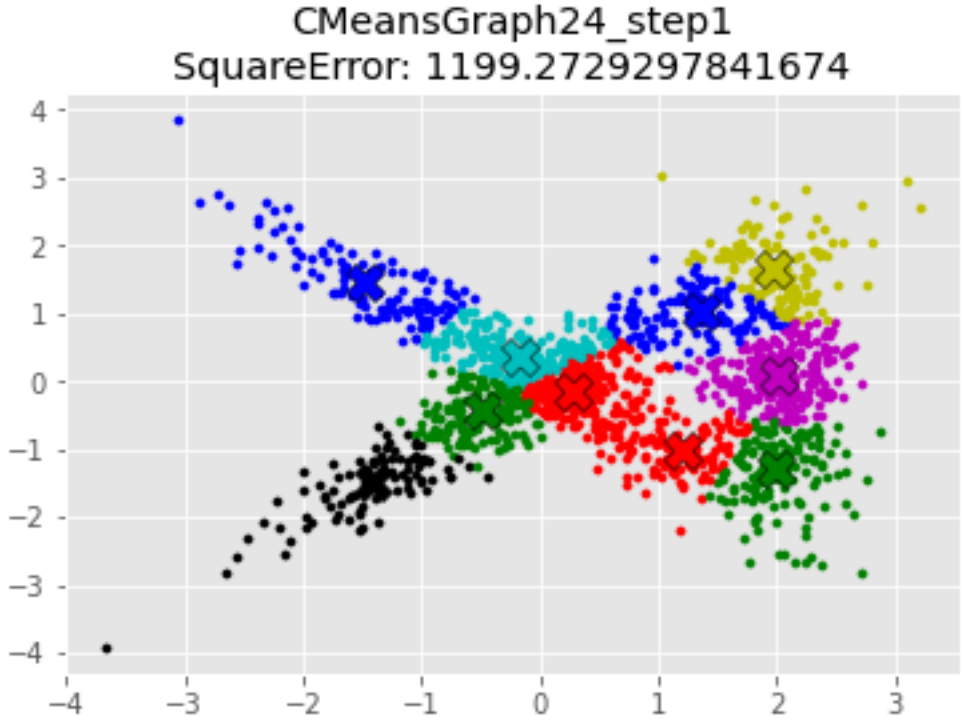

CMeansGraph19_step1
SquareError: 1204.5340733977011

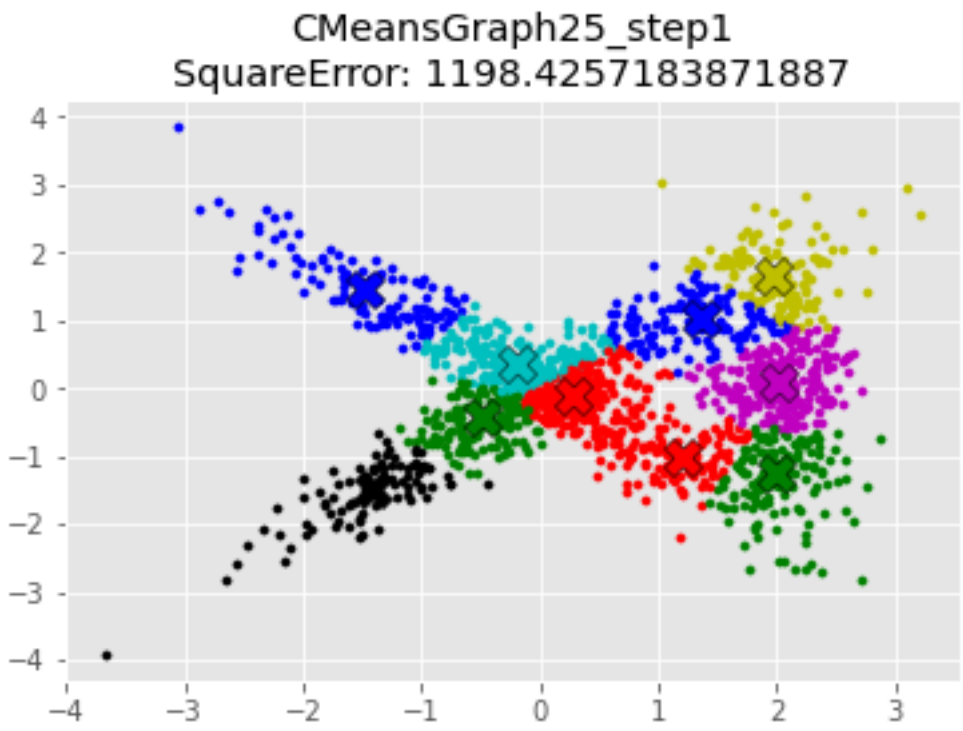

CMeansGraph20_step1
SquareError: 1203.3404758143568

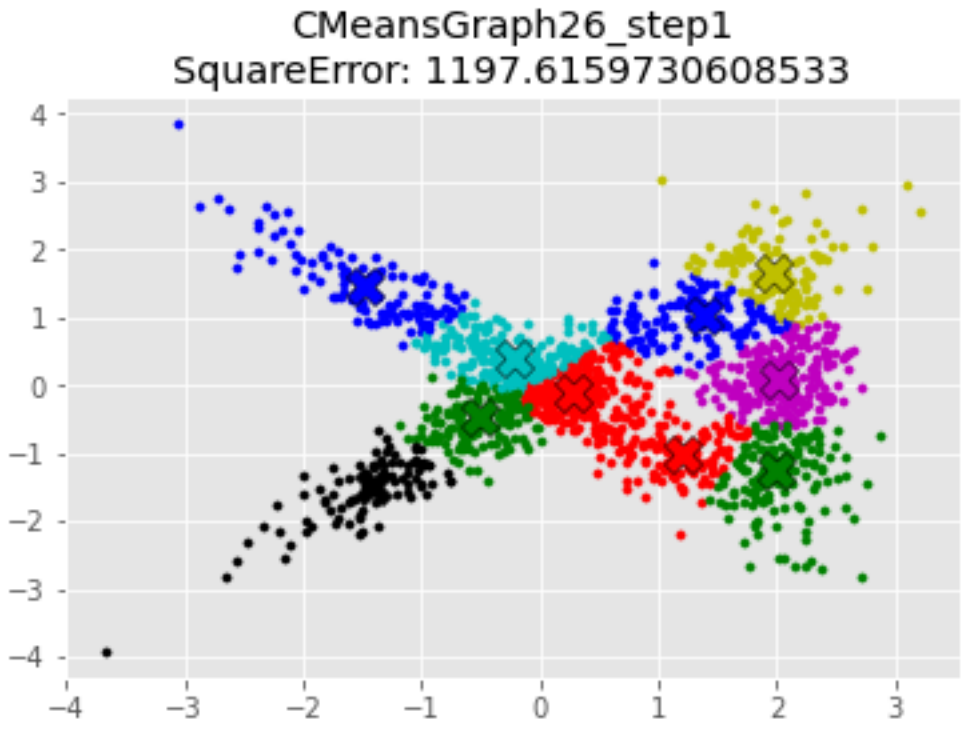

CMeansGraph21_step1
SquareError: 1202.2300568655912

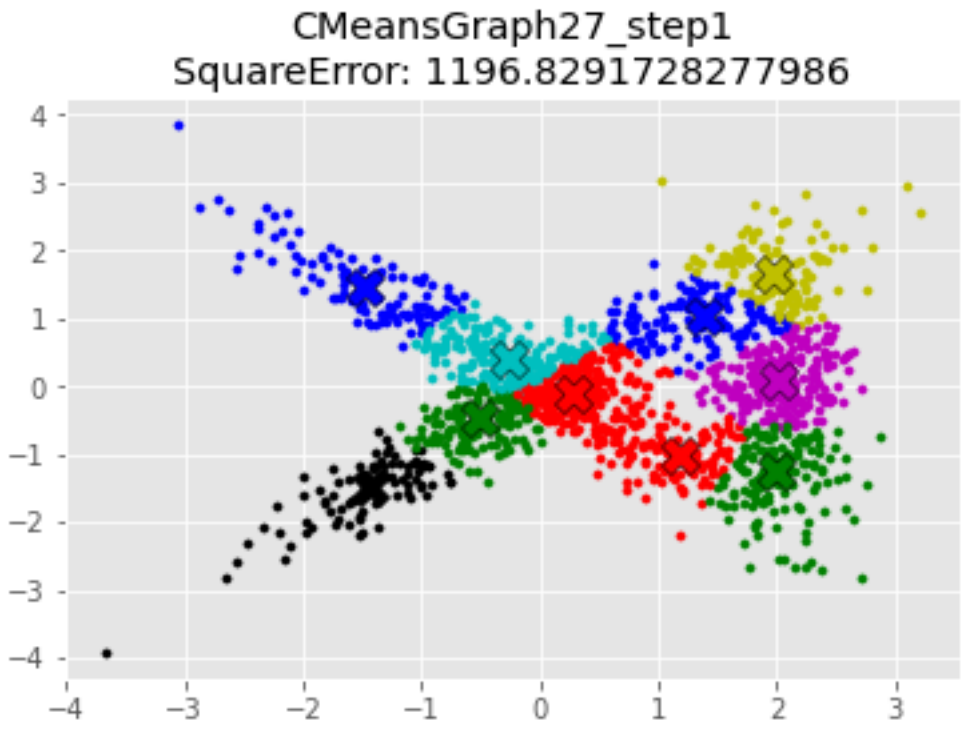

CMeansGraph22_step1
SquareError: 1201.1823484794081

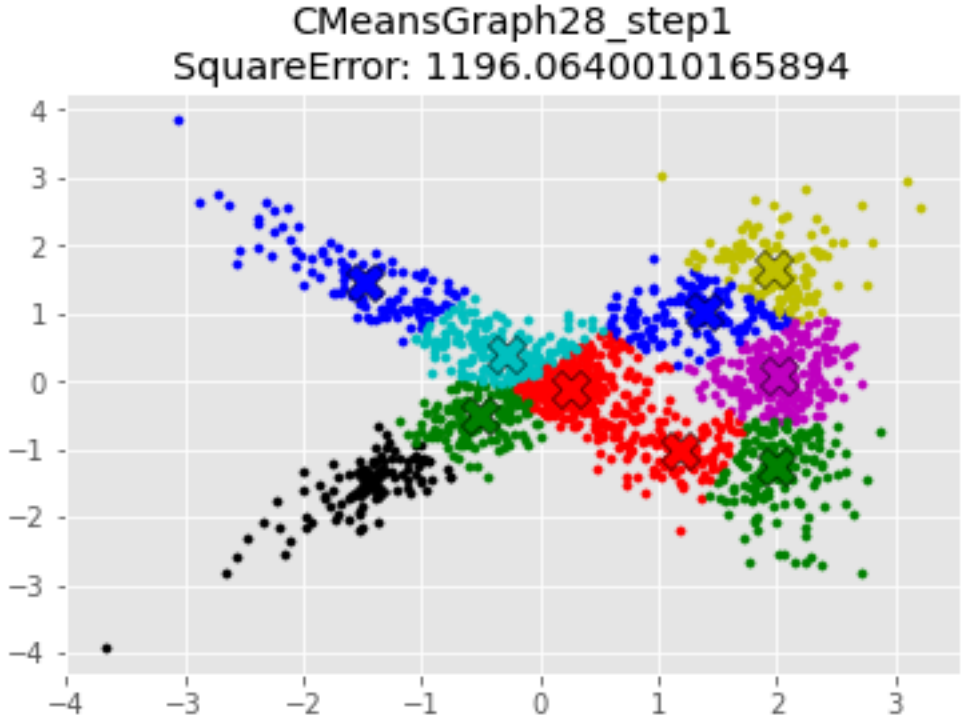

CMeansGraph23_step1
SquareError: 1200.1838525152723

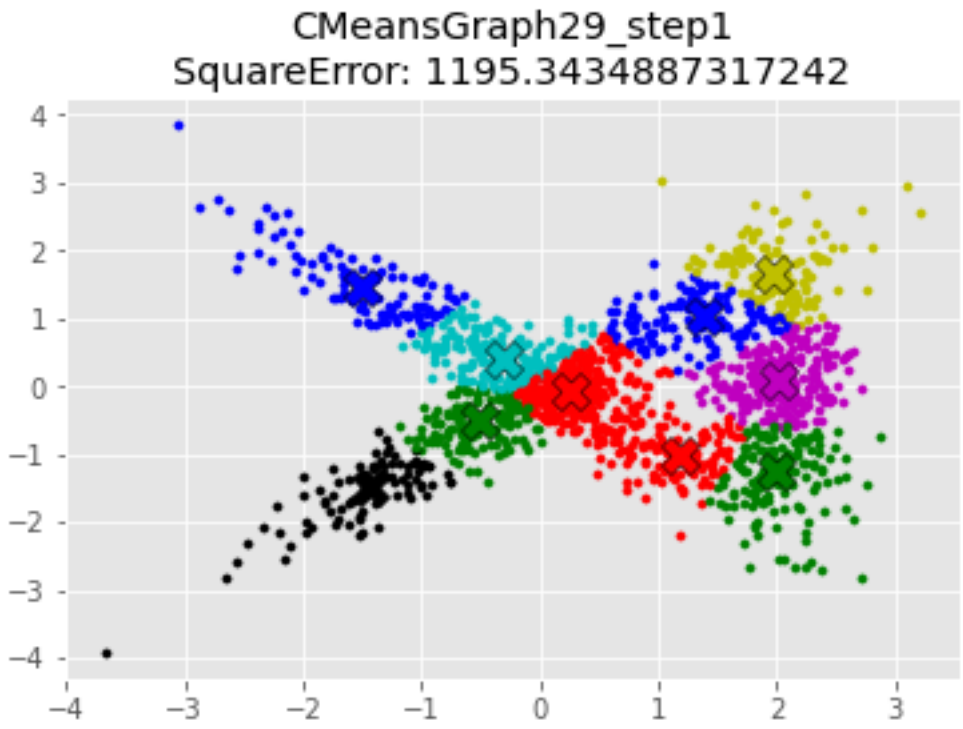

CMeansGraph24_step1
SquareError: 1199.2729297841674

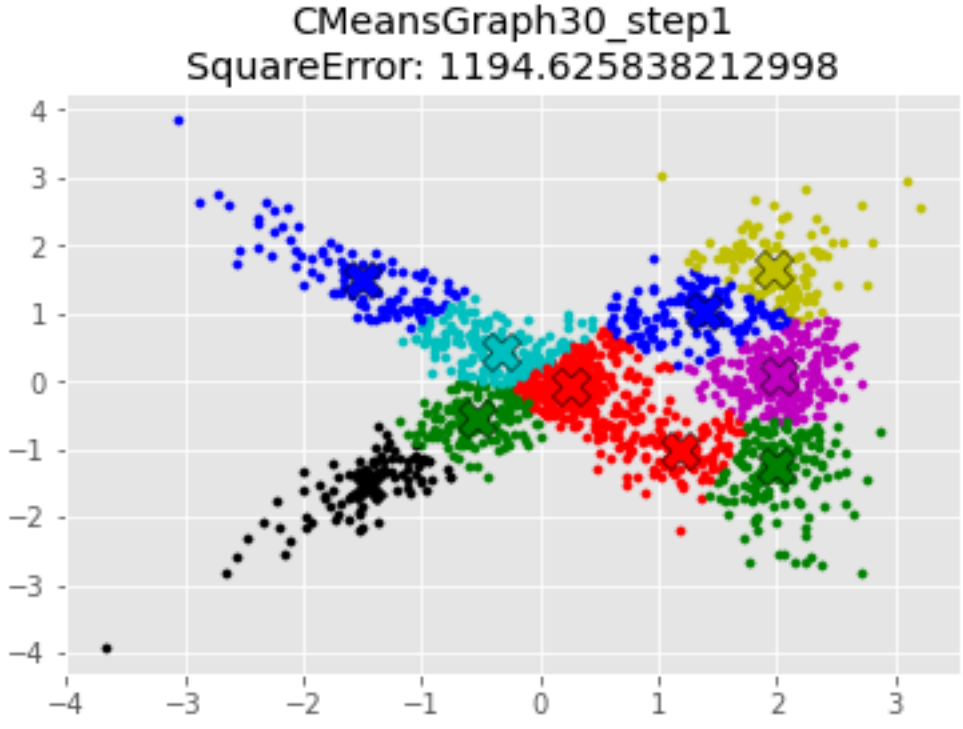

CMeansGraph25_step1
SquareError: 1198.4257183871887

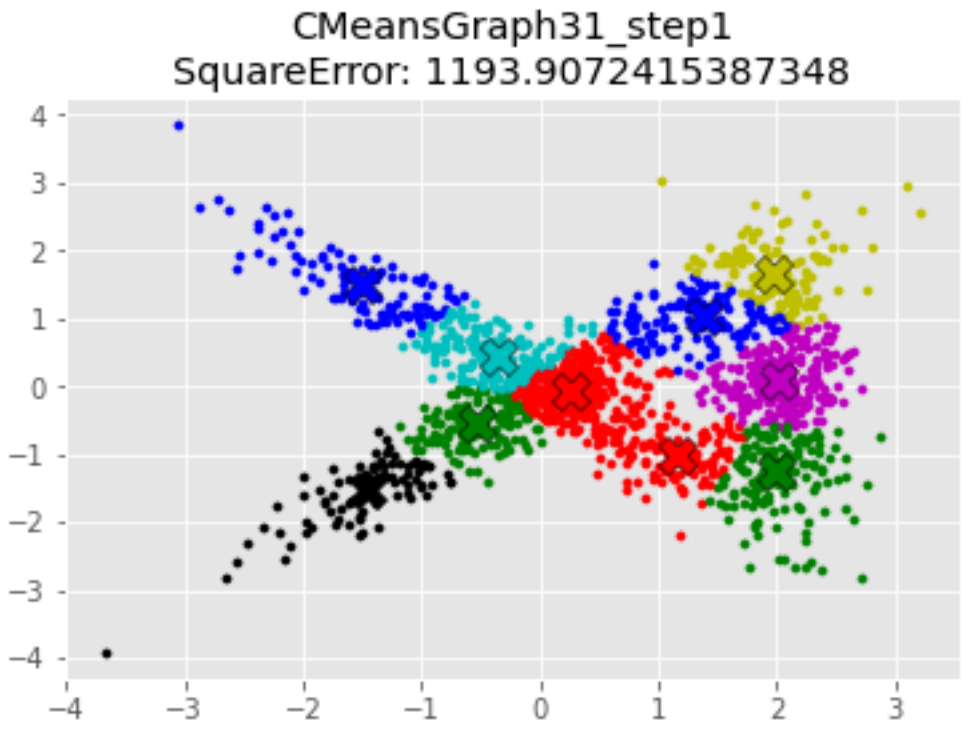

CMeansGraph26_step1
SquareError: 1197.6159730608533

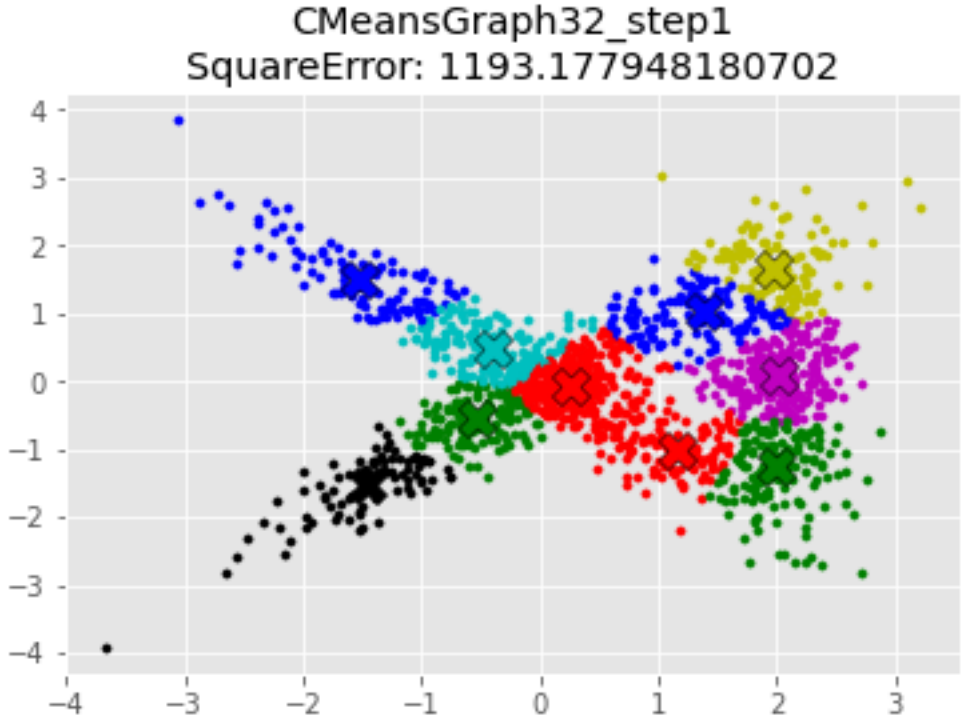

CMeansGraph27_step1
SquareError: 1196.8291728277986

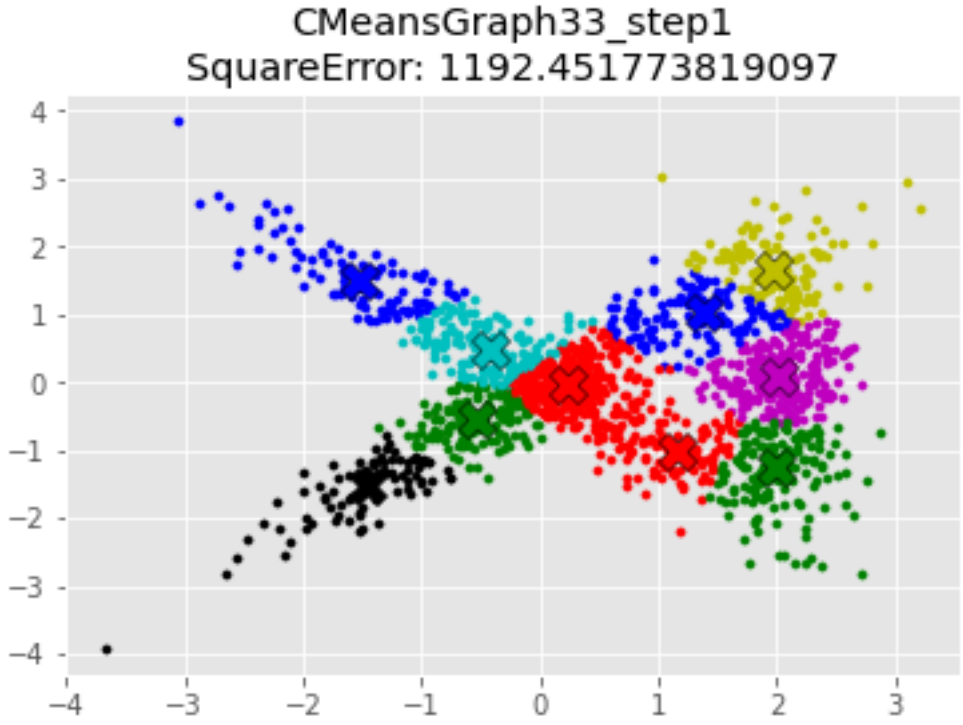

CMeansGraph28_step1
SquareError: 1196.0640010165894

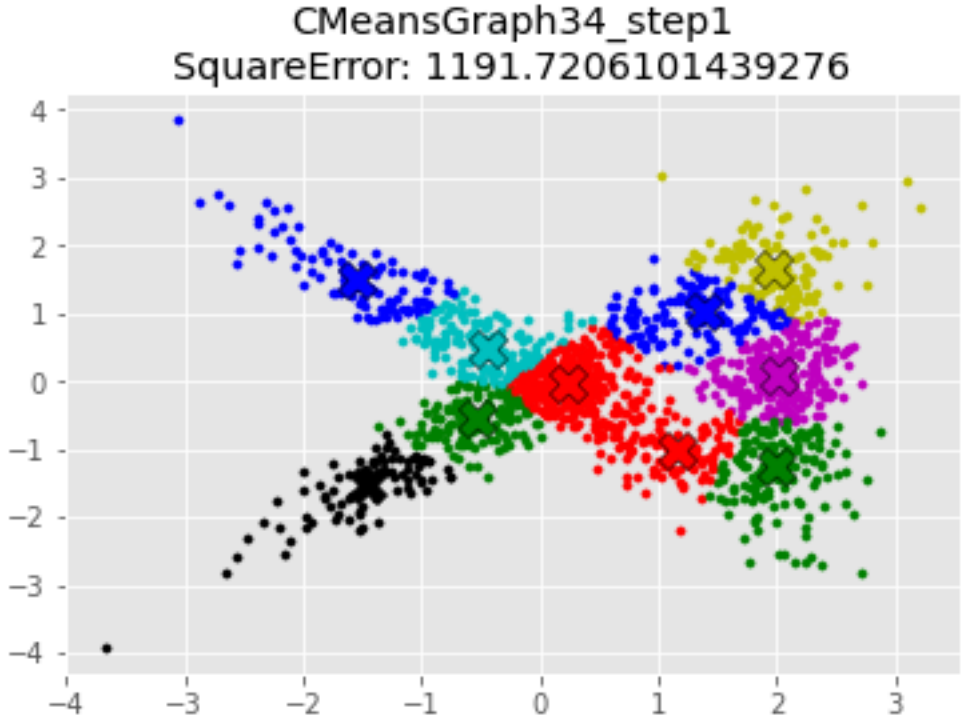

CMeansGraph29_step1
SquareError: 1195.3434887317242

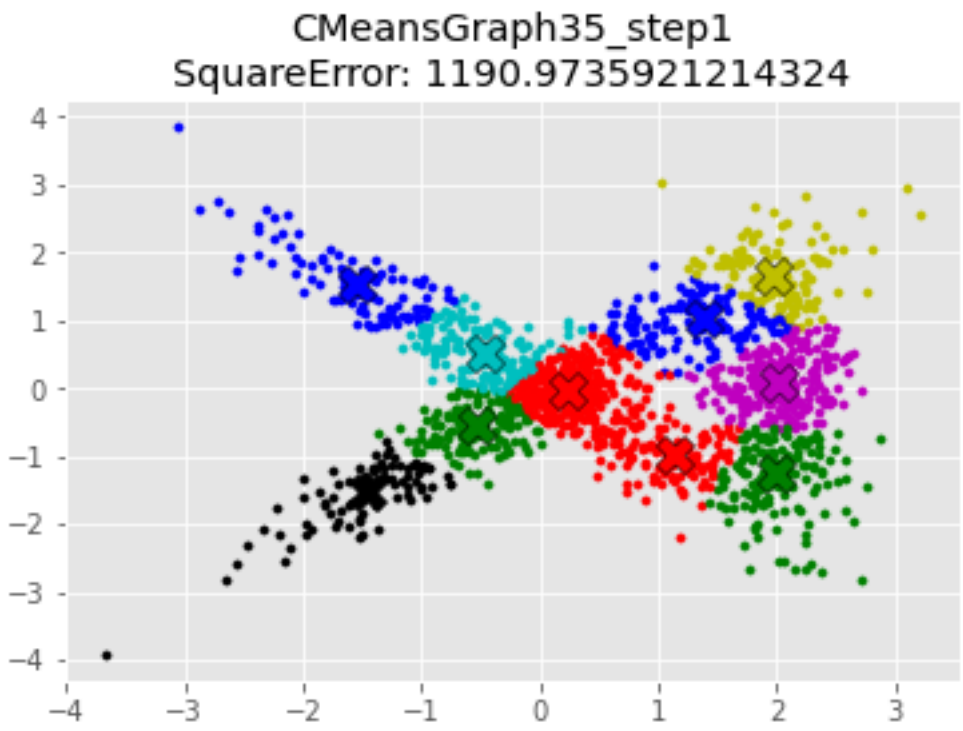

CMeansGraph30_step1
SquareError: 1194.625838212998

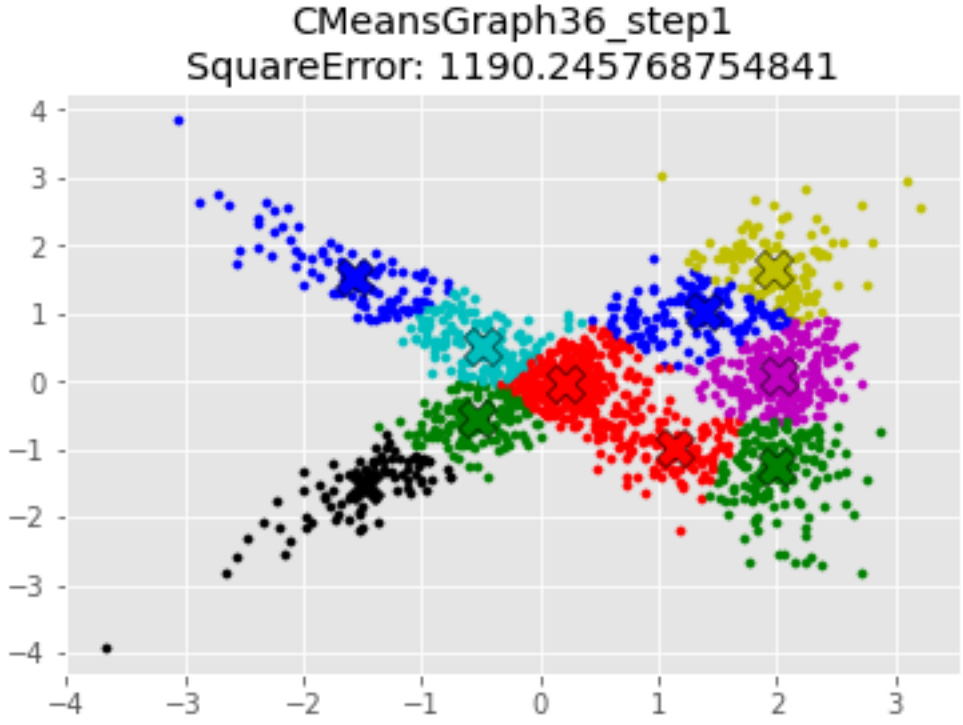

CMeansGraph31_step1
SquareError: 1193.9072415387348

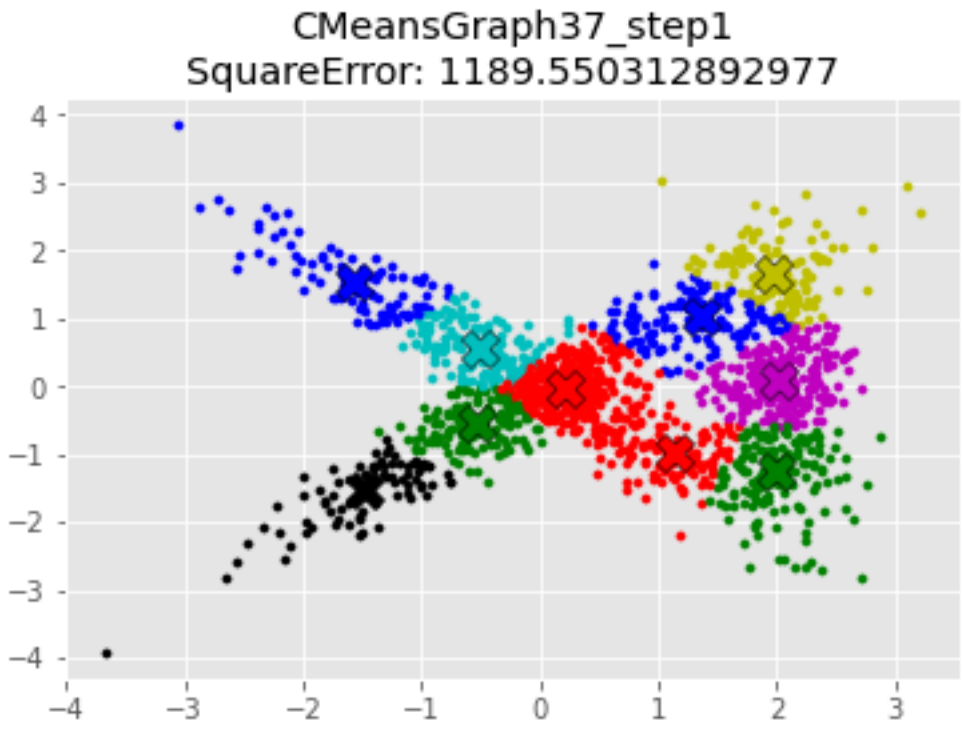

CMeansGraph32_step1
SquareError: 1193.177948180702

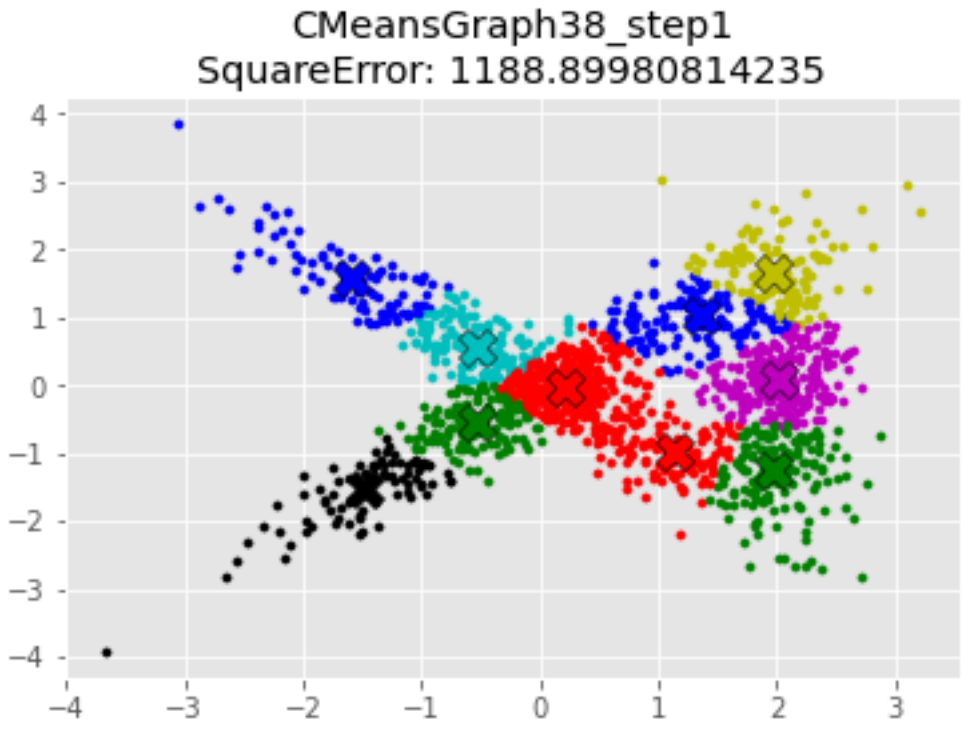

CMeansGraph33_step1
SquareError: 1192.451773819097

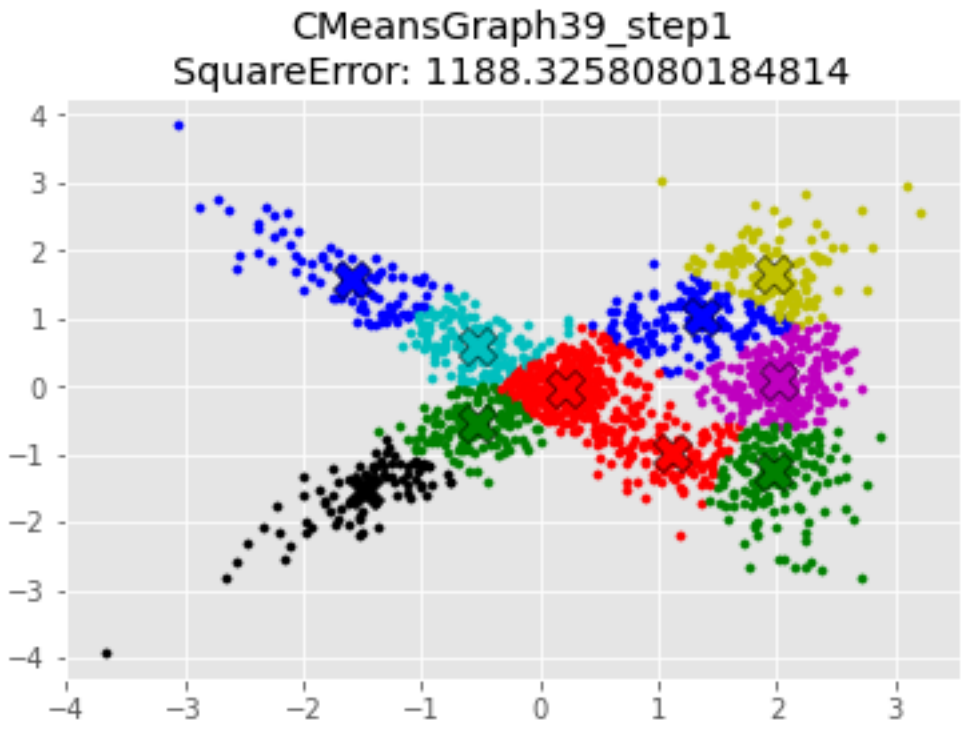

CMeansGraph34_step1
SquareError: 1191.7206101439276

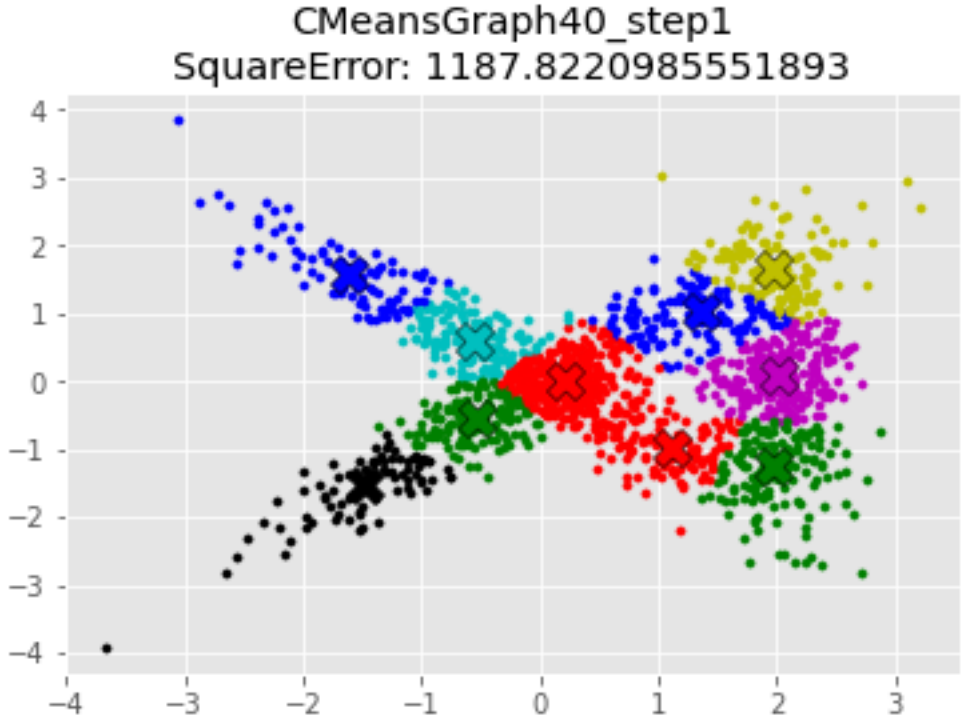

CMeansGraph35_step1
SquareError: 1190.9735921214324
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

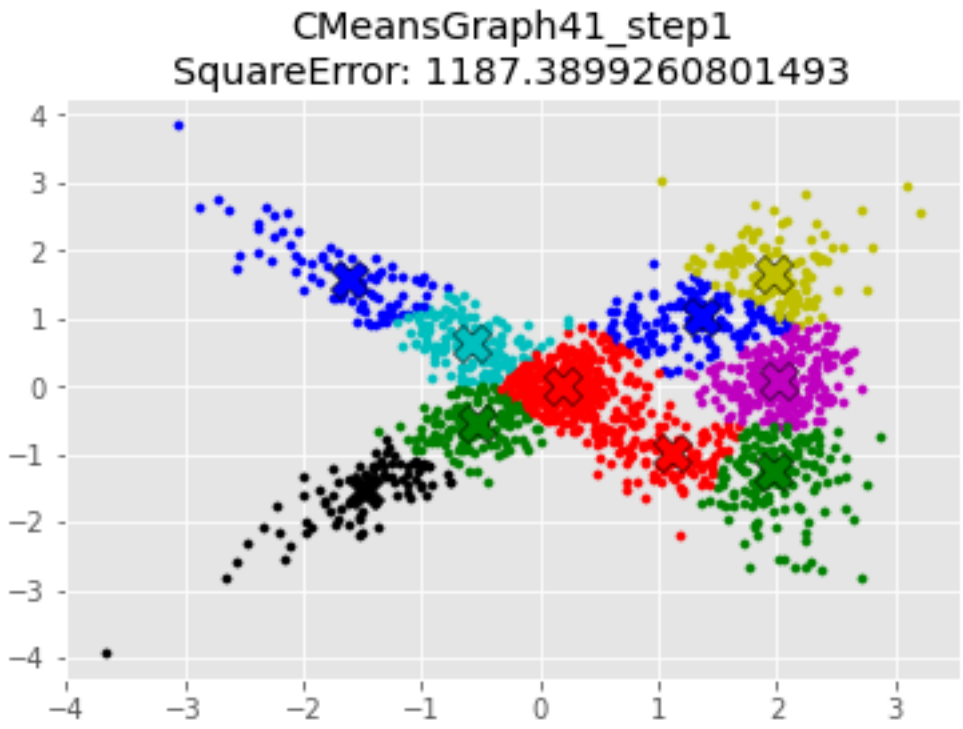

CMeansGraph36_step1
SquareError: 1190.245768754841
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

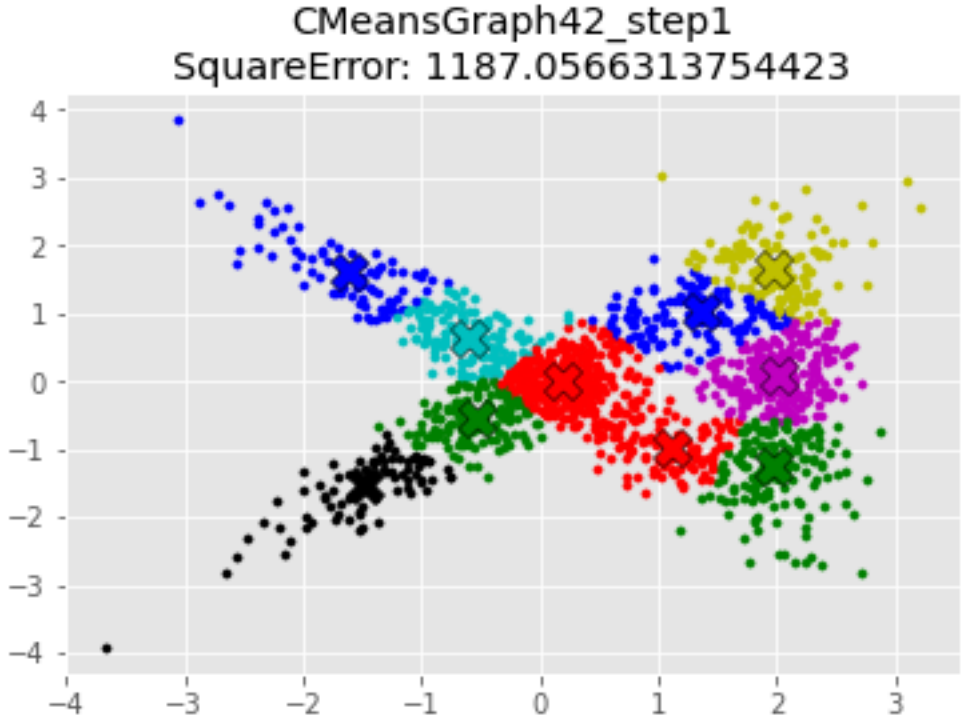

CMeansGraph37_step1
SquareError: 1189.550312892977
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

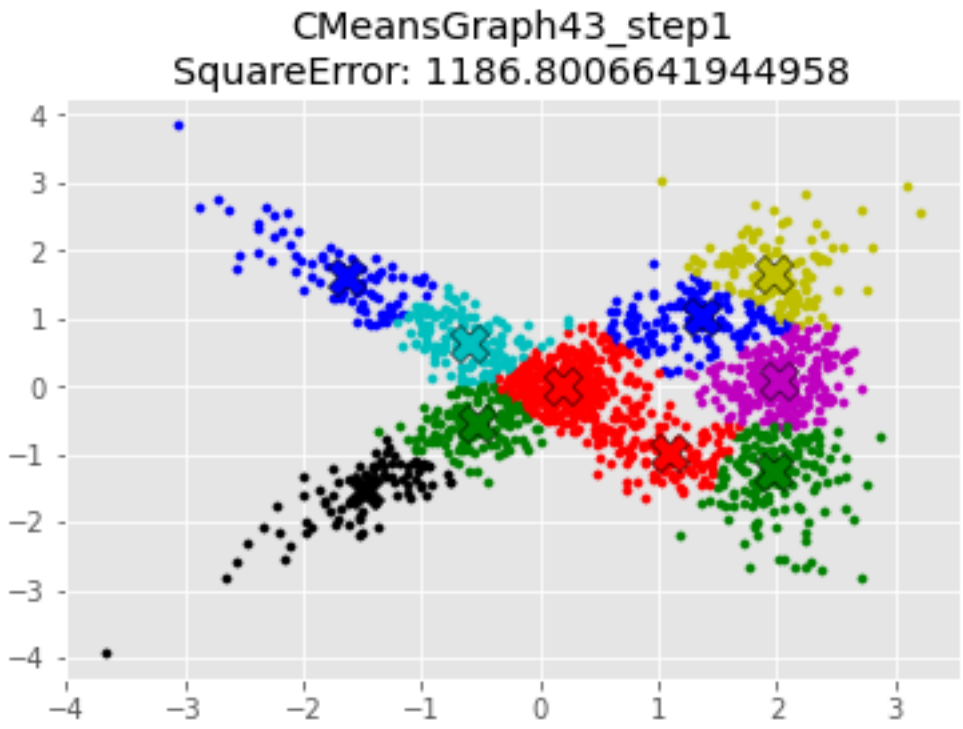

CMeansGraph38_step1
SquareError: 1188.89980814235
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

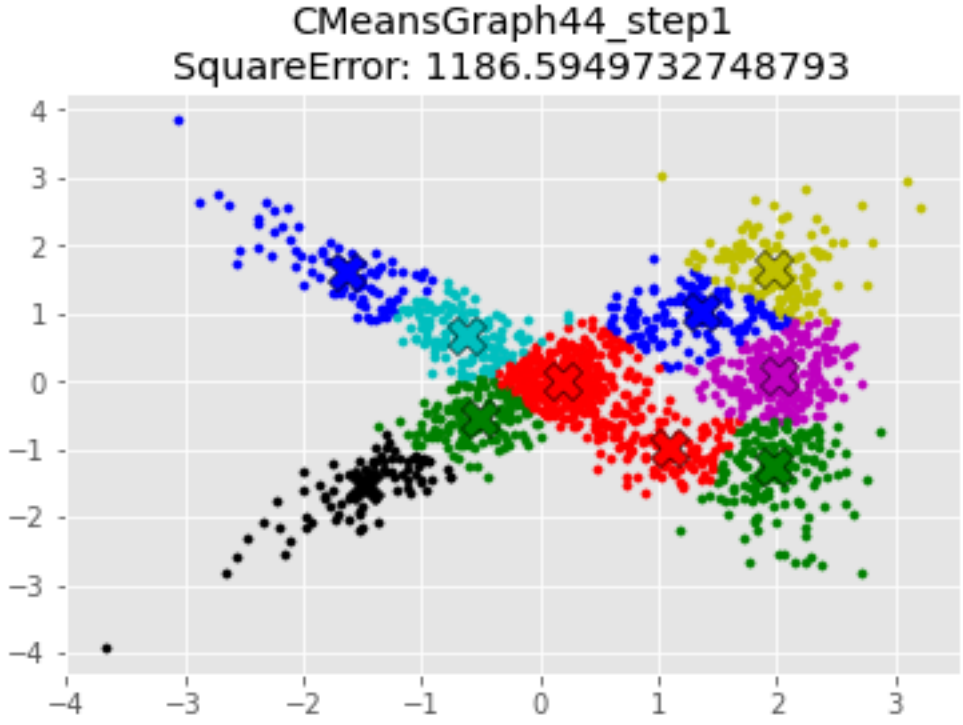

CMeansGraph39_step1
SquareError: 1188.3258080184814
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

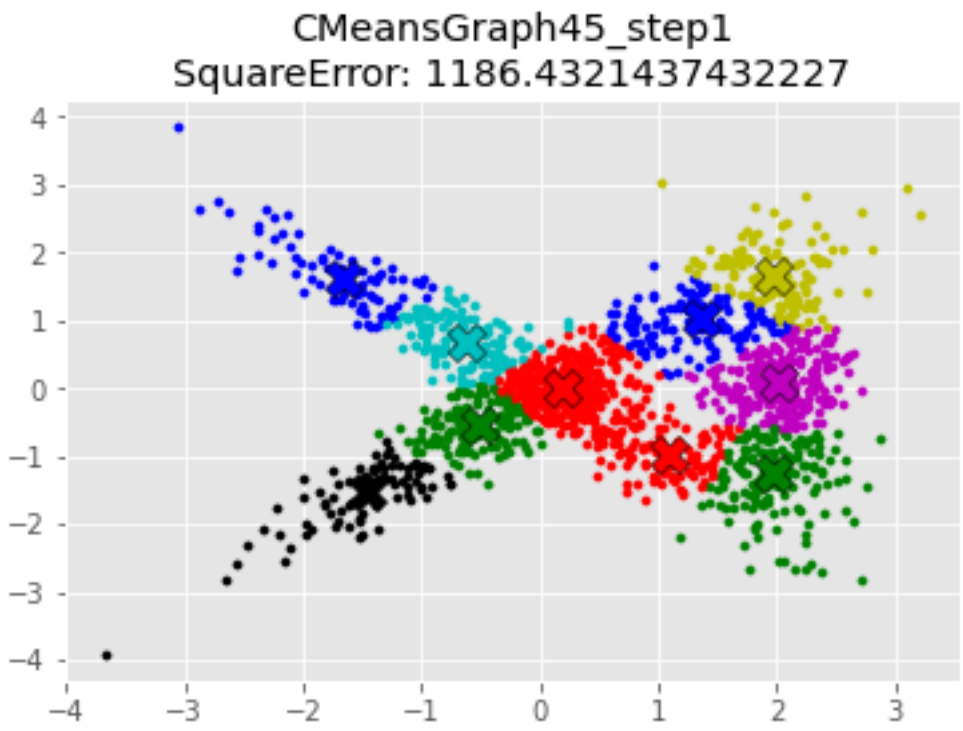

CMeansGraph40_step1
SquareError: 1187.8220985551893
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

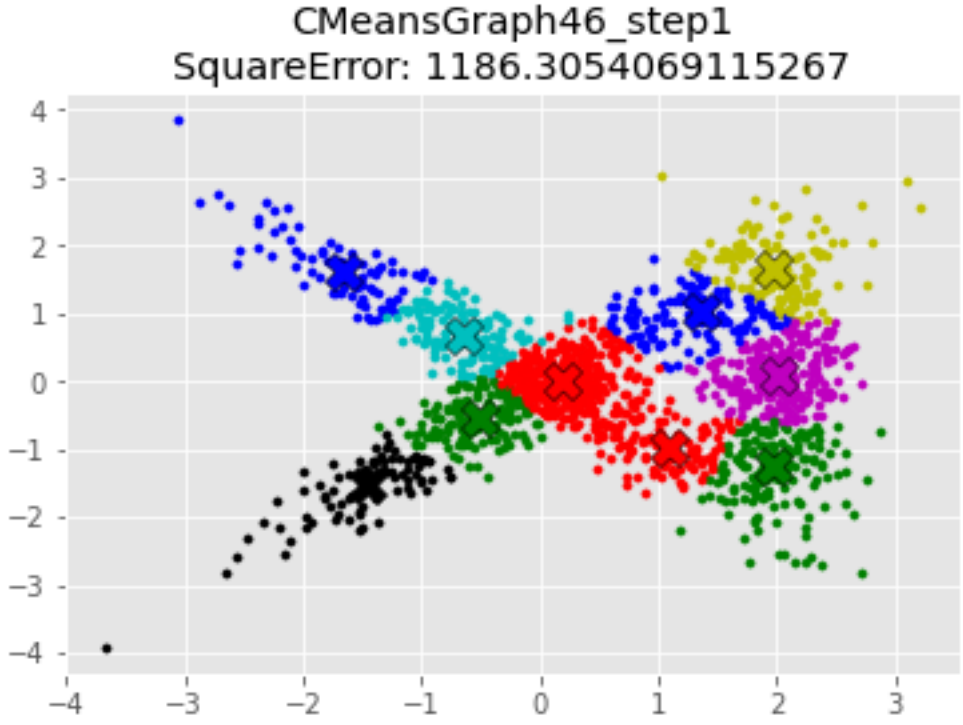

CMeansGraph41_step1
SquareError: 1187.3899260801493

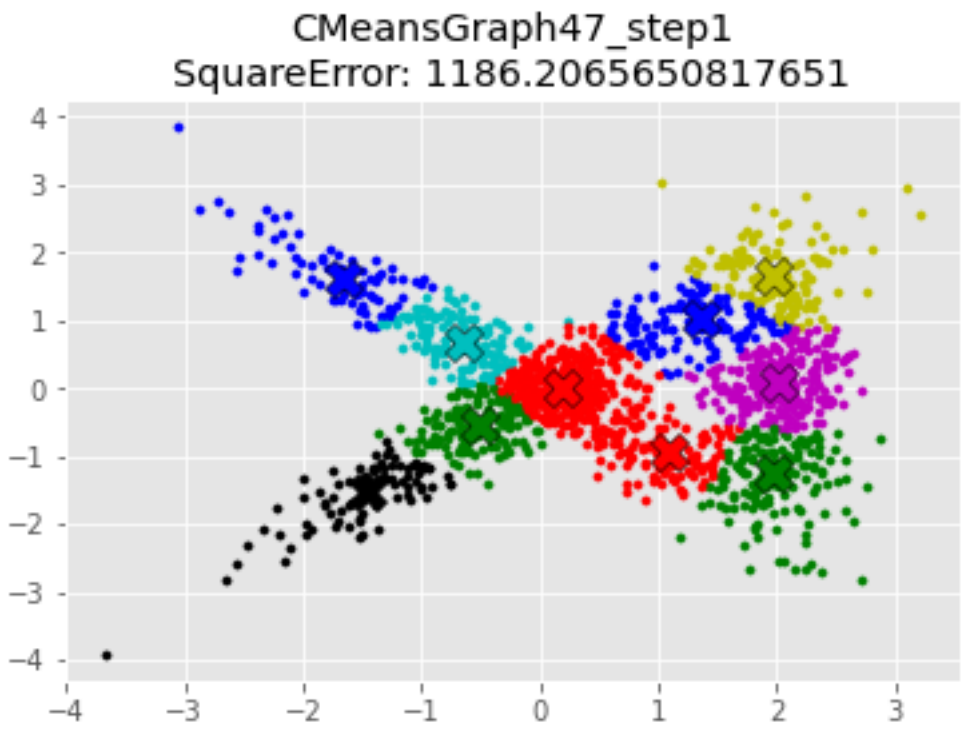

CMeansGraph42_step1
SquareError: 1187.0566313754423

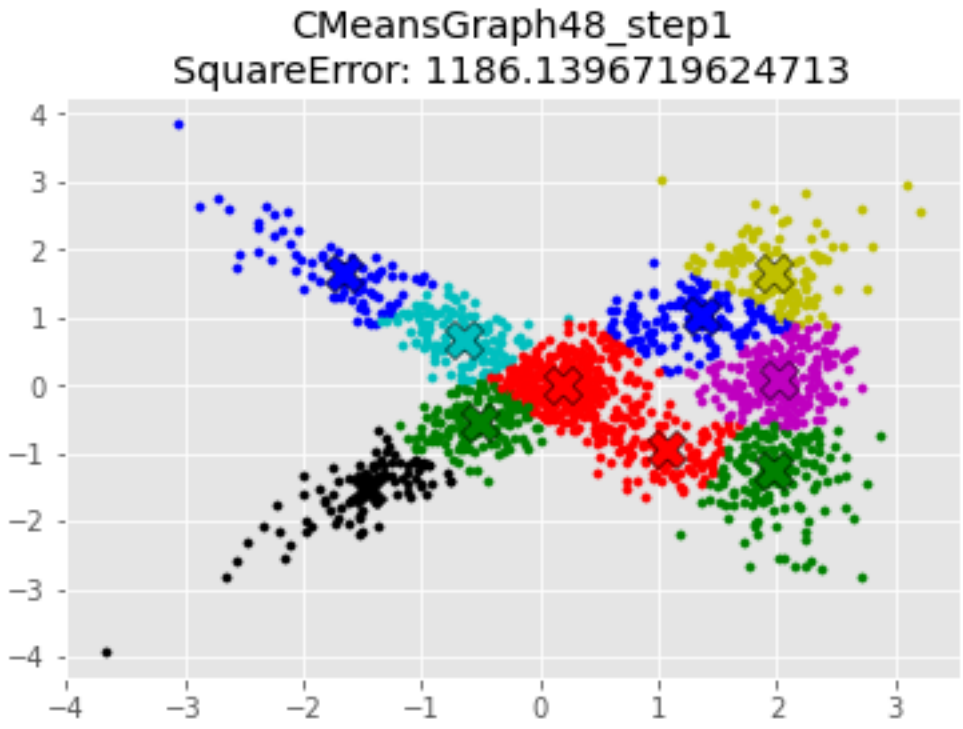

CMeansGraph43_step1
SquareError: 1186.8006641944958
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

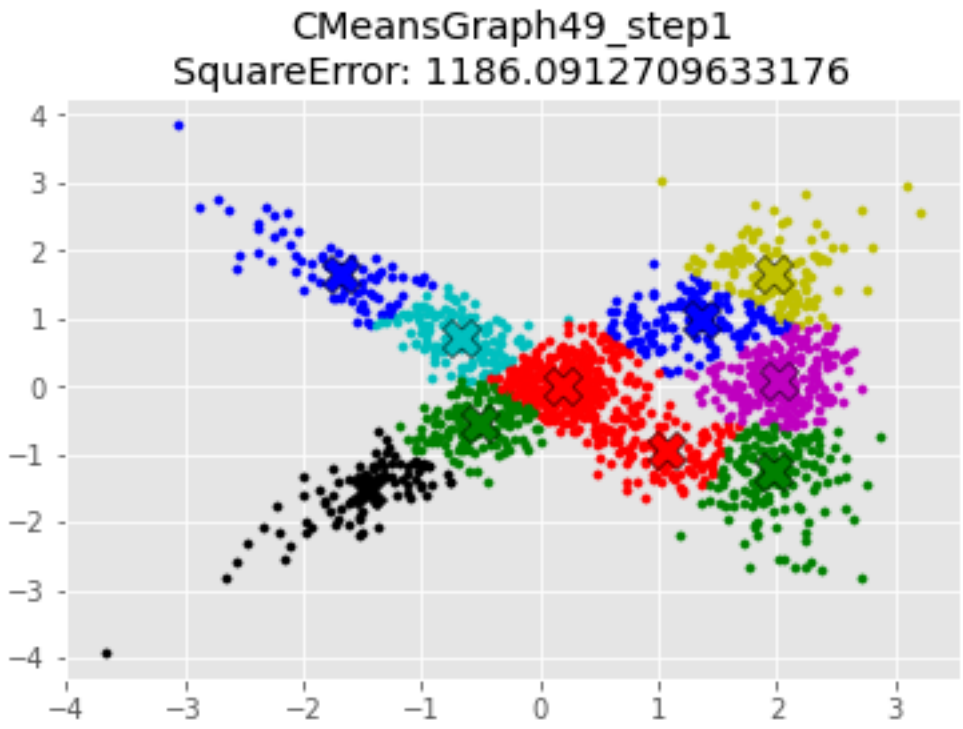

CMeansGraph44_step1
SquareError: 1186.5949732748793
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

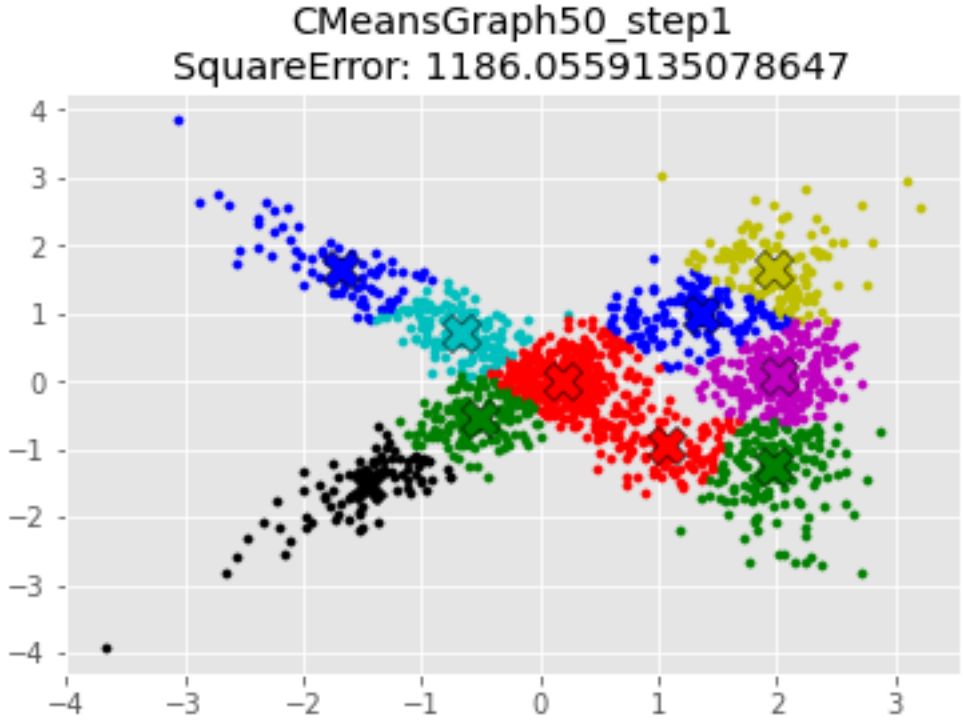

CMeansGraph45_step1
SquareError: 1186.4321437432227

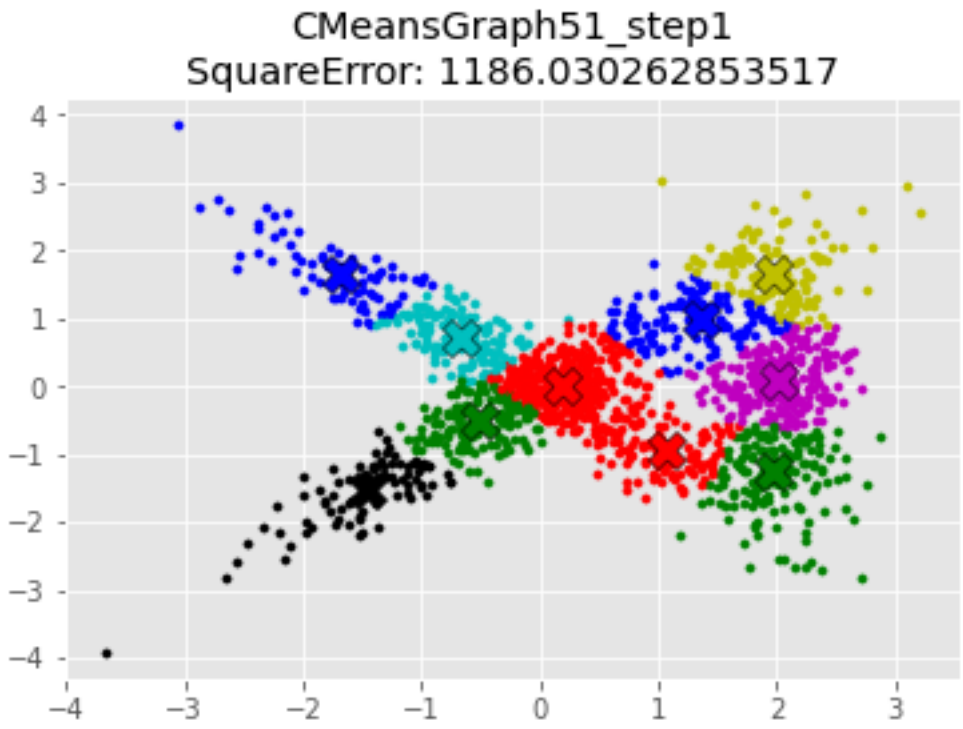

CMeansGraph46_step1
SquareError: 1186.3054069115267

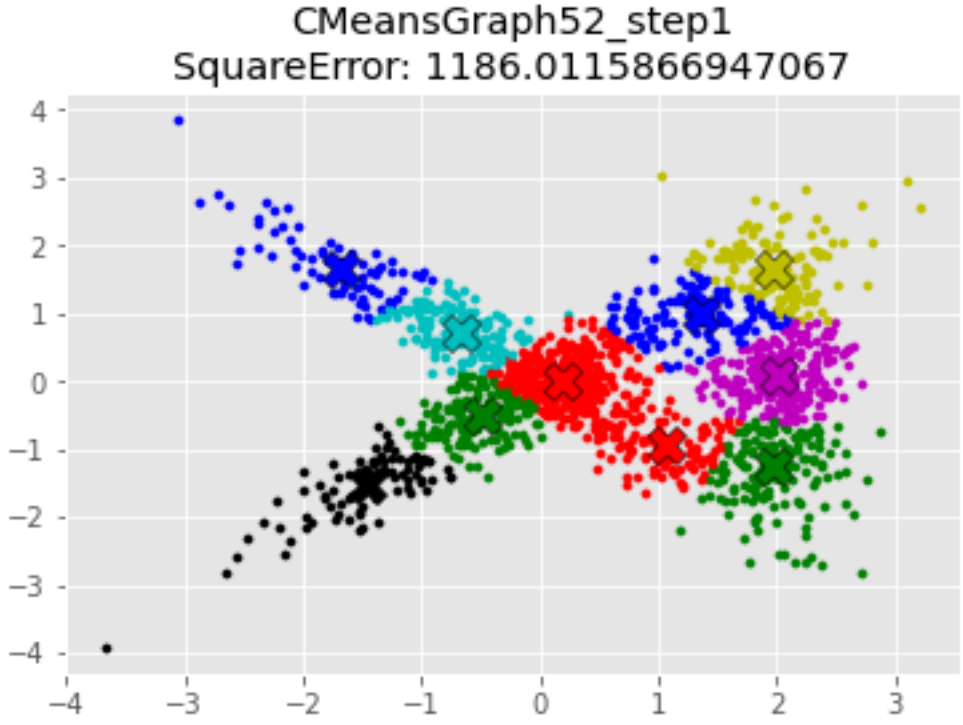

CMeansGraph47_step1
SquareError: 1186.2065650817651

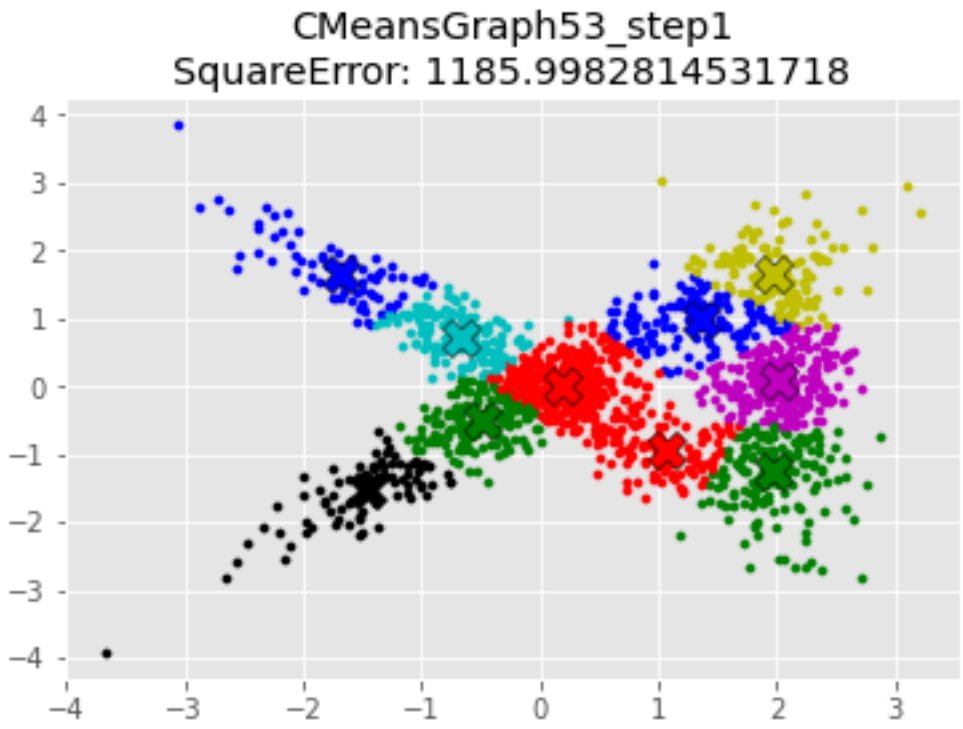

CMeansGraph48_step1
SquareError: 1186.1396719624713

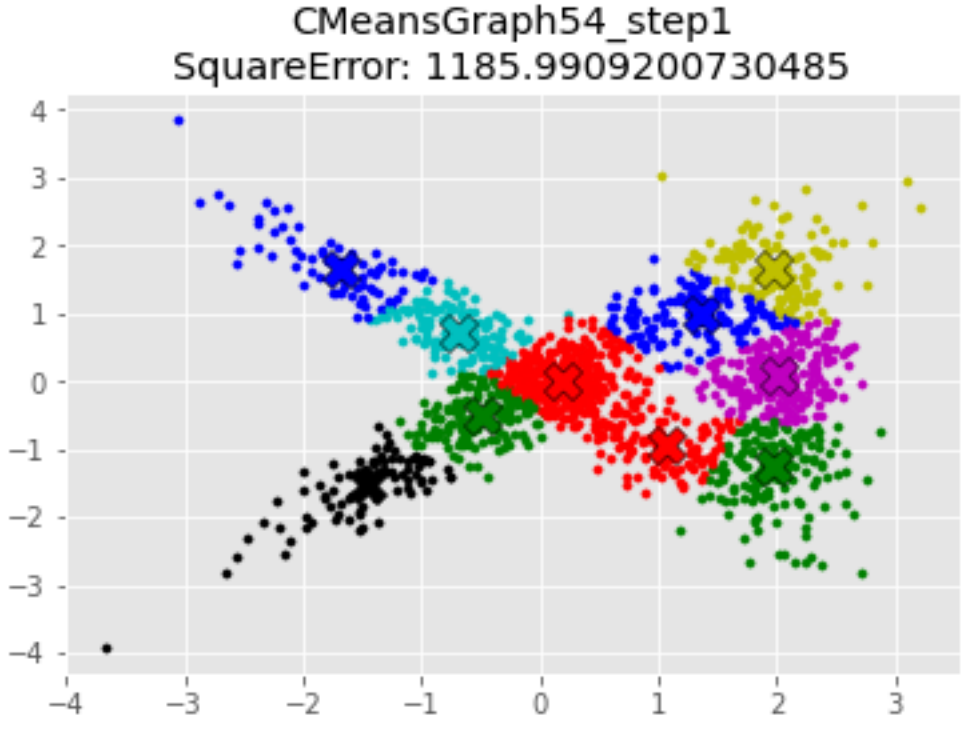

CMeansGraph49_step1
SquareError: 1186.0912709633176

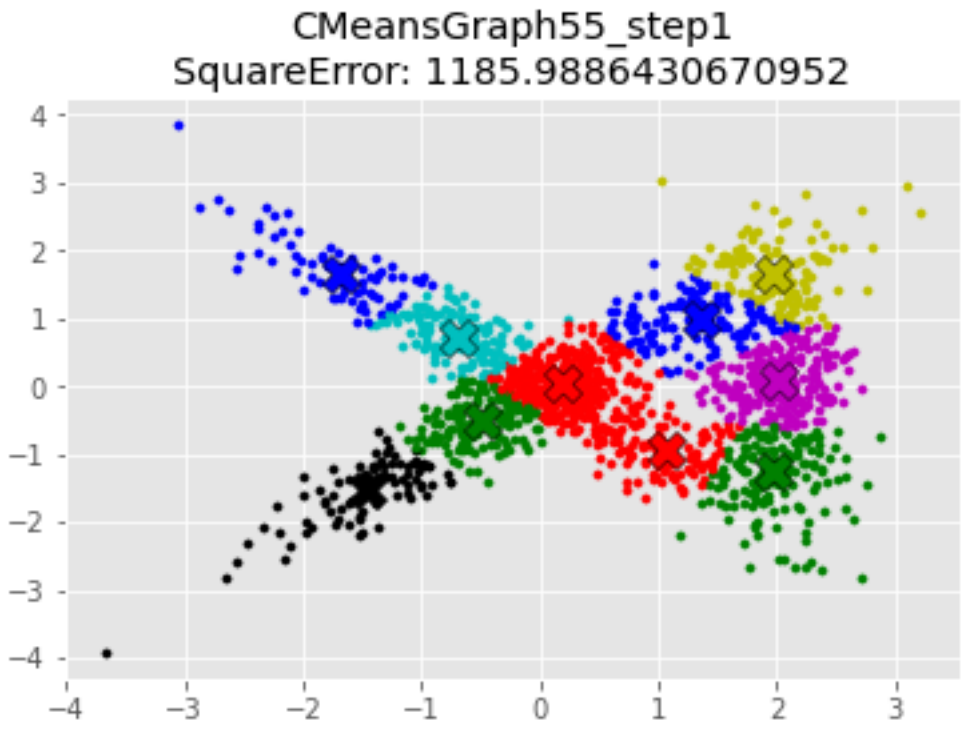

CMeansGraph50_step1
SquareError: 1186.0559135078647

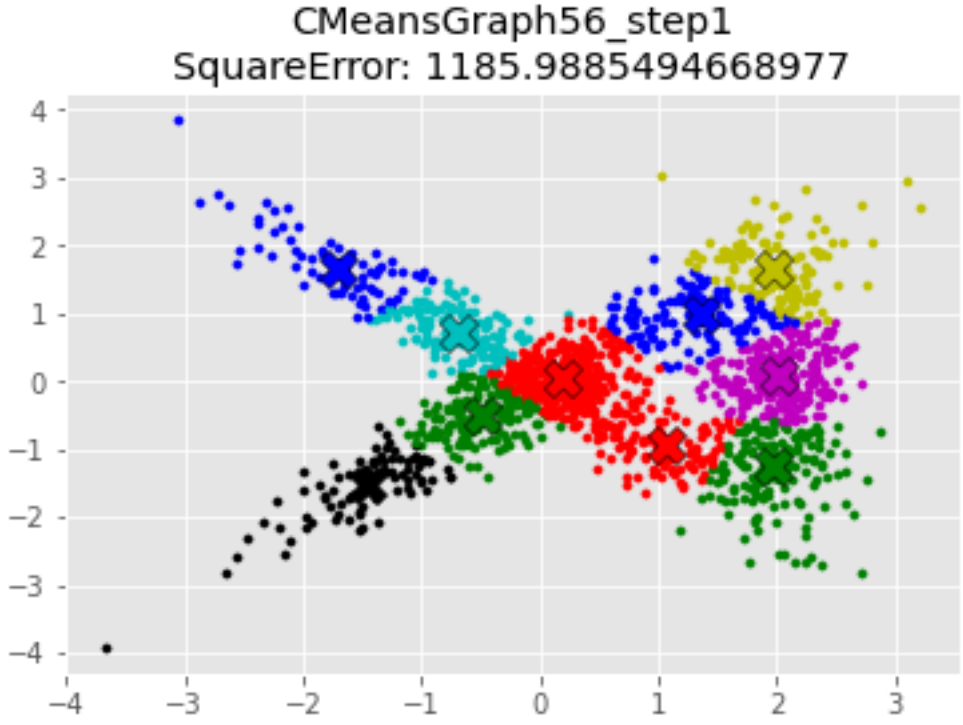

CMeansGraph51_step1
SquareError: 1186.030262853517
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3


CMeansGraph52_step1
SquareError: 1186.0115866947067
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3


CMeansGraph53_step1
SquareError: 1185.9982814531718
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3


CMeansGraph54_step1
SquareError: 1185.9909200730485
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3


CMeansGraph55_step1
SquareError: 1185.9886430670952
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3


CMeansGraph56_step1
SquareError: 1185.9885494668977
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

```
Epsilon reached, stopping early
New best square error: 2
All squareErrors sums:
[1186.0318322123521, 1185.9885494668977]
Best squareError sum:
1185.9885494668977
The best graph is:
```