CSCI964 Computational Intelligence: Lab #5

Mei Wangzhihui 2019124044

Task 1

```
   '''
@Author: maywzh
@Date: 2020-03-21 14:14:43
@FilePath: /ji_coursenotes/2020spring/CSCI964/lab5-0323/src/kmeans.py
'''
import numpy as np
import random
import re
import matplotlib.pyplot as plt


def loadDataSet(filename):
   """数据读取."""
   dataSet = np.loadtxt(filename, delimiter=',', usecols=(0, 1, 2, 3))
   return dataSet


def initCentroids(dataSet, k):
   """初始化k个簇心"""
   dataSet = list(dataSet)
   return random.sample(dataSet, k)


def getCentroids(clusterDict):
   """重新计算k个簇心"""
   centroidList = []
   for key in clusterDict.keys():
      centroid = np.mean(clusterDict[key], axis=0)
      centroidList.append(centroid)
   return centroidList


def calcuDistance(vec1, vec2):
   """计算两点间的欧式距离"""
   return np.sqrt(np.sum(np.square(vec1 - vec2)))


def getMSE(centroidList, clusterDict):
   """计算各簇集合间的均方误差 将簇类中各个点与簇心的距离累加求和"""
   sum = 0.0
   for key in clusterDict.keys():
      vec1 = centroidList[key]
      distance = 0.0
      for item in clusterDict[key]:
         vec2 = item
         distance += calcuDistance(vec1, vec2)
      sum += distance
   return sum


def minifyDistance(dataSet, centroidList):
```

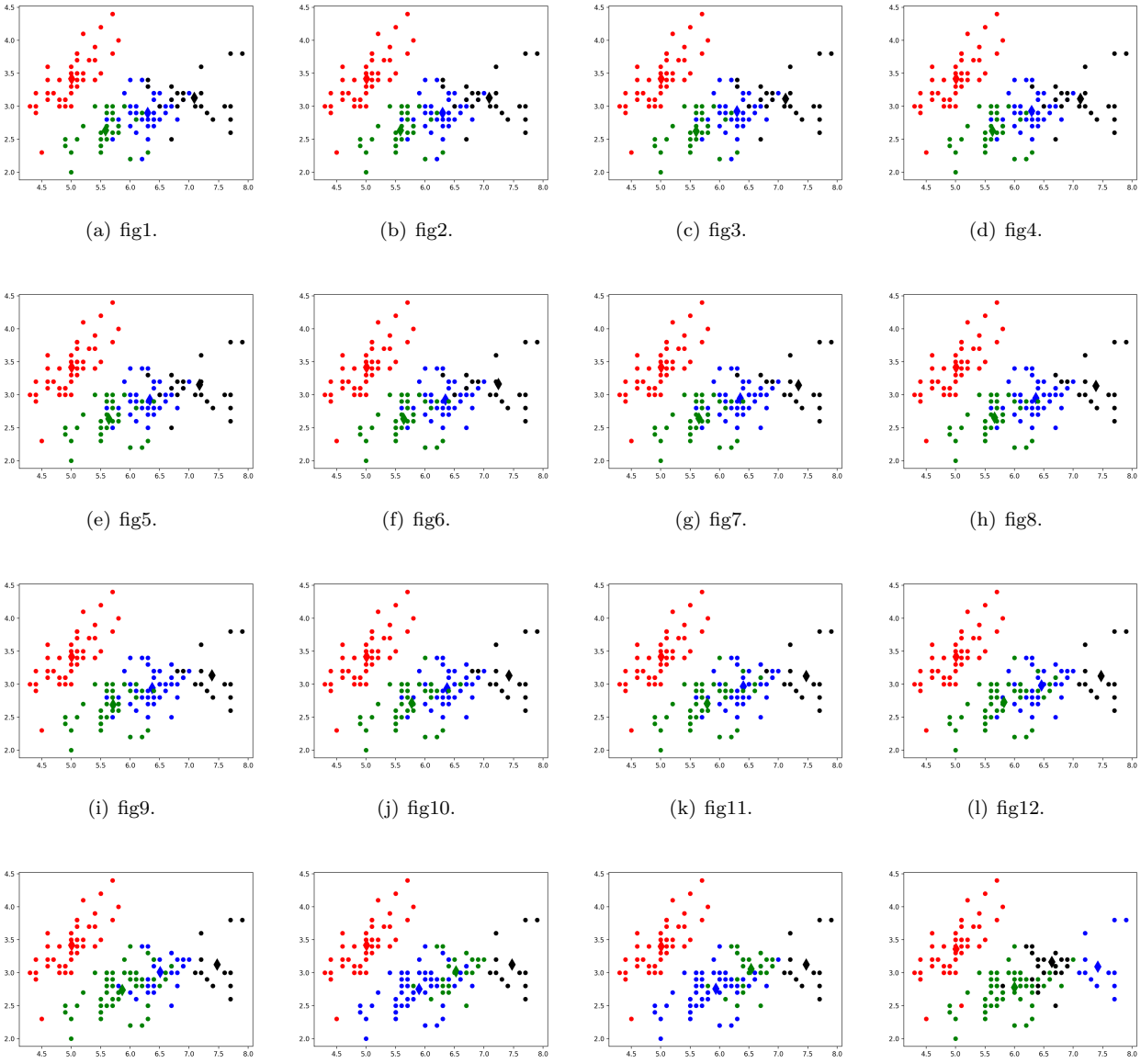
Problem 1 continued on next page...

```
    """
    基于选定k个簇心，重新聚簇
    """
    clusterDict = dict()
    k = len(centroidList)
    for item in dataSet:
        vec1 = item
        flag = -1
        minDis = float("inf")
        for i in range(k):
            vec2 = centroidList[i]
            distance = calcuDistance(vec1, vec2)
            if distance < minDis:
                minDis = distance
                flag = i # 循环结束时， flag保存与当前item最近的蔟标记
        if flag not in clusterDict.keys():
            clusterDict.setdefault(flag, [])
        clusterDict[flag].append(item) # 加入相应的类别中
    return clusterDict # 不同的类别


def showCluster(centroidList, clusterDict):
    """可视化聚类结果
    """
    colorMark = ['or', 'ob', 'og', 'ok', 'oy', 'ow'] #
        不同簇类标记，o表示圆形，另一个表示颜色
    centroidMark = ['dr', 'db', 'dg', 'dk', 'dy', 'dw']

    for key in clusterDict.keys():
        plt.plot(centroidList[key][0], centroidList[key]
                 [1], centroidMark[key], markersize=12) # 质心点
        for item in clusterDict[key]:
            plt.plot(item[0], item[1], colorMark[key])
    plt.show()


def test_k_means():
    dataSet = loadDataSet("../data/iris.txt")
    centroidList = initCentroids(dataSet, 4)
    clusterDict = minifyDistance(dataSet, centroidList)
    newMSE = getMSE(centroidList, clusterDict)
    oldMSE = 1 # 当两次聚类的误差小于某个值是，说明质心基本确定。

    times = 2
    while abs(newMSE - oldMSE) >= 0.000001:
        centroidList = getCentroids(clusterDict)
        clusterDict = minifyDistance(dataSet, centroidList)
        oldMSE = newMSE
        newMSE = getMSE(centroidList, clusterDict)
        times += 1
        showCluster(centroidList, clusterDict)
```

```
if __name__ == '__main__':
    test_k_means()
```

Task 2

The process of K-means on Iris dataset



(a) fig1. (b) fig2. (c) fig3. (d) fig4.

(e) fig5. (f) fig6. (g) fig7. (h) fig8.

(i) fig9. (j) fig10. (k) fig11. (l) fig12.

(m) fig13. (n) fig14. (o) fig15. (p) fig16.

Figure 1: Clustering of Iris dataset