《数据挖掘》课程作业

郭慕尧

July 8, 2019

1 选用算法

对于 seeds_dataset 数据集的数据聚类采用 K-means 算法
算法流程:

1. 将文件中的数据读入到 dataset 列表中，通过 len(dataset[0]) 来获取数据维数，在测试样例中是四维
2. 产生聚类的初始位置。首先扫描数据，获取每一维数据分量中的最大值和最小值，然后在这个区间上随机产生一个值，循环 k 次 (k 为所分的类别), 这样就产生了聚类初始中心（k 个）
3. 按照最短距离（欧式距离）原则将所有样本分配到 k 个聚类中心中的某一个，这步操作的结果是产生列表 assigments，可以通过 Python 中的 zip 函数整合成字典。注意到原始聚类中心可能不在样本中，因此可能出现分配的结果出现某一个聚类中心点集合为空，此时需要结束，提示“随机数产生错误，需要重新运行”，以产生合适的初始中心。
4. 计算各个聚类中心的新向量，更新距离，即每一类中每一维均值向量。然后再进行分配，比较前后两个聚类中心向量是否相等，若不相等则进行循环，否则终止循环，进入下一步。
5. 将结果输出到文件和屏幕中

2 Python 代码

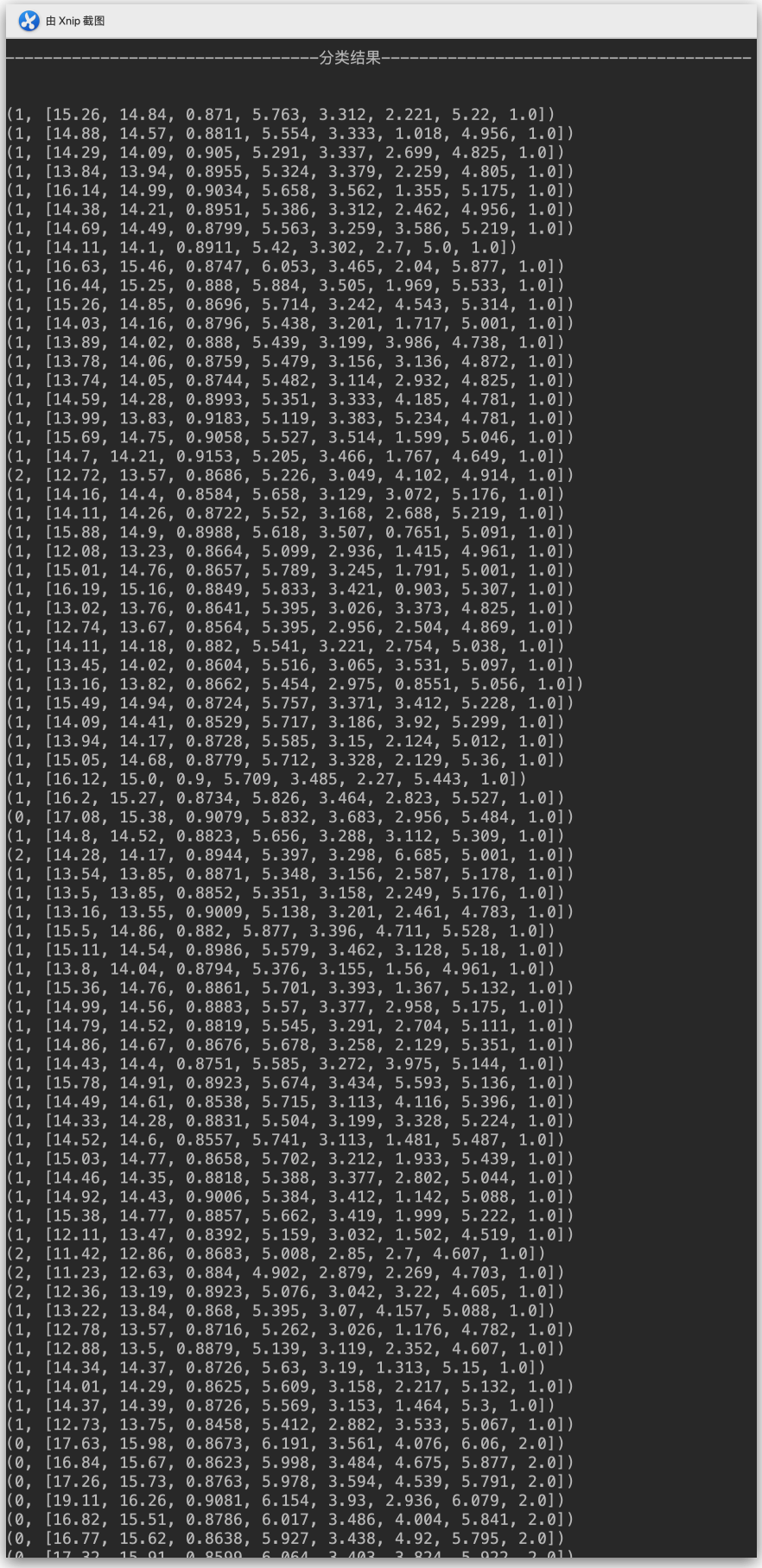
```python
from collections import defaultdict
from random import uniform
from math import sqrt


def read_points():
    dataset = []
    with open('seeds_dataset.txt', 'r') as file:
        for line in file:
            if line == '\n':
                continue
            dataset.append(list(map(float, line.split())))
        file.close()
        return dataset


def write_results(listResult, dataset, k):
    with open('result.txt', 'a') as file:
        for kind in range(k):
            file.write("CLASSINFO:%d\n" % (kind + 1))
            for j in listResult[kind]:
                file.write('%d\n' % j)
            file.write('\n')
        file.write('\n\n')
        file.close()


def point_avg(points):
    dimensions = len(points[0])
    new_center = []
    for dimension in range(dimensions):
        sum = 0
        for p in points:
            sum += p[dimension]
        new_center.append(float("%.8f" % (sum / float(len(points)))))
    return new_center
```

```
def update_centers(data_set, assignments, k):
    new_means = defaultdict(list)
    centers = []
    for assignment, point in zip(assignments, data_set):
        new_means[assignment].append(point)
    for i in range(k):
        points = new_means[i]
        centers.append(point_avg(points))
    return centers


def assign_points(data_points, centers):
    assignments = []
    for point in data_points:
        shortest = float('inf')
        shortest_index = 0
        for i in range(len(centers)):
            value = distance(point, centers[i])
            if value < shortest:
                shortest = value
                shortest_index = i
        assignments.append(shortest_index)
    if len(set(assignments)) < len(centers):
        print("\n产生随机数错误，请重新运行程序\n")
        exit()
    return assignments


def distance(a, b):
    dimention = len(a)
    sum = 0
    for i in range(dimention):
        sq = (a[i] - b[i]) ** 2
        sum += sq
    return sqrt(sum)


def generate_k(data_set, k):
    centers = []
    dimentions = len(data_set[0])
    min_max = defaultdict(int)
    for point in data_set:
        for i in range(dimentions):
            value = point[i]
            min_key = 'min_%d' % i
            max_key = 'max_%d' % i
            if min_key not in min_max or value < min_max[min_key]:
                min_max[min_key] = value
            if max_key not in min_max or value > min_max[max_key]:
                min_max[max_key] = value
    for j in range(k):
        rand_point = []
        for i in range(dimentions):
            min_val = min_max['min_%d' % i]
            max_val = min_max['max_%d' % i]
            tmp = float("%.8f" % (uniform(min_val, max_val)))
            rand_point.append(tmp)
        centers.append(rand_point)
    return centers


def k_means(dataset, k):
    k_points = generate_k(dataset, k)
    assignments = assign_points(dataset, k_points)
    old_assignments = None
    while assignments != old_assignments:
        new_centers = update_centers(dataset, assignments, k)
        old_assignments = assignments
        assignments = assign_points(dataset, new_centers)
    result = list(zip(assignments, dataset))
    print('\n\n---------------------------------分类结果-------------------------------------\n\n'
                                                )
    for out in result:
        print(out, end='\n')
```

```
    print('\n\n---------------------------------标号简记-------------------------------------\n\n'
                                                    )
    listResult = [[] for i in range(k)]
    count = 0
    for i in assignments:
        listResult[i].append(count)
        count = count + 1
    write_results(listResult, dataset, k)
    for kind in range(k):
        print("第%d类数据有:" % (kind + 1))
        print('{')
        count = 0
        for j in listResult[kind]:
            print(j, end=', ')
            count = count + 1
            if count % 25 == 0:
                print('\n')
        print('}\n')
        print("集合大小为",count,"\n\n")


def main():
    dataset = read_points()
    k_means(dataset, 3)


if __name__ == "__main__":
    main()
```

3 聚类结果

```
-----------------------------标号简记-----------------------------------


第1类数据有:
{
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25,

26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,

53, 54, 55, 56, 57, 58, 59, 63, 64, 65, 66, 67, 68, 69, 100, 122, 124, 132, 133, 134, 135, 137, 138, 139, }

集合大小为 74

第2类数据有:
{
19, 39, 60, 61, 62, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159,

160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184,

185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209,

}

集合大小为 75

第3类数据有:
{
37, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,

94, 95, 96, 97, 98, 99, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,

120, 121, 123, 125, 126, 127, 128, 129, 130, 131, 136, }

集合大小为 61
```


由 Xnip 截图
----------分类结果----------
(1, [15.26, 14.84, 0.871, 5.763, 3.312, 2.221, 5.22, 1.0])
(1, [14.88, 14.57, 0.8811, 5.554, 3.333, 1.018, 4.956, 1.0])
(1, [14.29, 14.09, 0.905, 5.291, 3.337, 2.699, 4.825, 1.0])
(1, [13.84, 13.94, 0.8955, 5.324, 3.379, 2.259, 4.805, 1.0])
(1, [16.14, 14.99, 0.9034, 5.658, 3.562, 1.355, 5.175, 1.0])
(1, [14.38, 14.21, 0.8951, 5.386, 3.312, 2.462, 4.956, 1.0])
(1, [14.69, 14.49, 0.8799, 5.563, 3.259, 3.586, 5.219, 1.0])
(1, [14.11, 14.1, 0.8911, 5.42, 3.302, 2.7, 5.0, 1.0])
(1, [16.63, 15.46, 0.8747, 6.053, 3.465, 2.04, 5.877, 1.0])
(1, [16.44, 15.25, 0.888, 5.884, 3.505, 1.969, 5.533, 1.0])
(1, [15.26, 14.85, 0.8696, 5.714, 3.242, 4.543, 5.314, 1.0])
(1, [14.03, 14.16, 0.8796, 5.438, 3.201, 1.717, 5.001, 1.0])
(1, [13.89, 14.02, 0.888, 5.439, 3.199, 3.986, 4.738, 1.0])
(1, [13.78, 14.06, 0.8759, 5.479, 3.156, 3.136, 4.872, 1.0])
(1, [13.74, 14.05, 0.8744, 5.482, 3.114, 2.932, 4.825, 1.0])
(1, [14.59, 14.28, 0.8993, 5.351, 3.333, 4.185, 4.781, 1.0])
(1, [13.99, 13.83, 0.9183, 5.119, 3.383, 5.234, 4.781, 1.0])
(1, [15.69, 14.75, 0.9058, 5.527, 3.514, 1.599, 5.046, 1.0])
(1, [14.7, 14.21, 0.9153, 5.205, 3.466, 1.767, 4.649, 1.0])
(2, [12.72, 13.57, 0.8686, 5.226, 3.049, 4.102, 4.914, 1.0])
(1, [14.16, 14.4, 0.8584, 5.658, 3.129, 3.072, 5.176, 1.0])
(1, [14.11, 14.26, 0.8722, 5.52, 3.168, 2.688, 5.219, 1.0])
(1, [15.88, 14.9, 0.8988, 5.618, 3.507, 0.7651, 5.091, 1.0])
(1, [12.08, 13.23, 0.8664, 5.099, 2.936, 1.415, 4.961, 1.0])
(1, [15.01, 14.76, 0.8657, 5.789, 3.245, 1.791, 5.001, 1.0])
(1, [16.19, 15.16, 0.8849, 5.833, 3.421, 0.903, 5.307, 1.0])
(1, [13.02, 13.76, 0.8641, 5.395, 3.026, 3.373, 4.825, 1.0])
(1, [12.74, 13.67, 0.8564, 5.395, 2.956, 2.504, 4.869, 1.0])
(1, [14.11, 14.18, 0.882, 5.541, 3.221, 2.754, 5.038, 1.0])
(1, [13.45, 14.02, 0.8604, 5.516, 3.065, 3.531, 5.097, 1.0])
(1, [13.16, 13.82, 0.8662, 5.454, 2.975, 0.8551, 5.056, 1.0])
(1, [15.49, 14.94, 0.8724, 5.757, 3.371, 3.412, 5.228, 1.0])
(1, [14.09, 14.41, 0.8529, 5.717, 3.186, 3.92, 5.299, 1.0])
(1, [13.94, 14.17, 0.8728, 5.585, 3.15, 2.124, 5.012, 1.0])
(1, [15.05, 14.68, 0.8779, 5.712, 3.328, 2.129, 5.36, 1.0])
(1, [16.12, 15.0, 0.9, 5.709, 3.485, 2.27, 5.443, 1.0])
(1, [16.2, 15.27, 0.8734, 5.826, 3.464, 2.823, 5.527, 1.0])
(0, [17.08, 15.38, 0.9079, 5.832, 3.683, 2.956, 5.484, 1.0])
(1, [14.8, 14.52, 0.8823, 5.656, 3.288, 3.112, 5.309, 1.0])
(2, [14.28, 14.17, 0.8944, 5.397, 3.298, 6.685, 5.001, 1.0])
(1, [13.54, 13.85, 0.8871, 5.348, 3.156, 2.587, 5.178, 1.0])
(1, [13.5, 13.85, 0.8852, 5.351, 3.158, 2.249, 5.176, 1.0])
(1, [13.16, 13.55, 0.9009, 5.138, 3.201, 2.461, 4.783, 1.0])
(1, [15.5, 14.86, 0.882, 5.877, 3.396, 4.711, 5.528, 1.0])
(1, [15.11, 14.54, 0.8986, 5.579, 3.462, 3.128, 5.18, 1.0])
(1, [13.8, 14.04, 0.8794, 5.376, 3.155, 1.56, 4.961, 1.0])
(1, [15.36, 14.76, 0.8861, 5.701, 3.393, 1.367, 5.132, 1.0])
(1, [14.99, 14.56, 0.8883, 5.57, 3.377, 2.958, 5.175, 1.0])
(1, [14.79, 14.52, 0.8819, 5.545, 3.291, 2.704, 5.111, 1.0])
(1, [14.86, 14.67, 0.8676, 5.678, 3.258, 2.129, 5.351, 1.0])
(1, [14.43, 14.4, 0.8751, 5.585, 3.272, 3.975, 5.144, 1.0])
(1, [15.78, 14.91, 0.8923, 5.674, 3.434, 5.593, 5.136, 1.0])
(1, [14.49, 14.61, 0.8538, 5.715, 3.113, 4.116, 5.396, 1.0])
(1, [14.33, 14.28, 0.8831, 5.504, 3.199, 3.328, 5.224, 1.0])
(1, [14.52, 14.6, 0.8557, 5.741, 3.113, 1.481, 5.487, 1.0])
(1, [15.03, 14.77, 0.8658, 5.702, 3.212, 1.933, 5.439, 1.0])
(1, [14.46, 14.35, 0.8818, 5.388, 3.377, 2.802, 5.044, 1.0])
(1, [14.92, 14.43, 0.9006, 5.384, 3.412, 1.142, 5.088, 1.0])
(1, [15.38, 14.77, 0.8857, 5.662, 3.419, 1.999, 5.222, 1.0])
(1, [12.11, 13.47, 0.8392, 5.159, 3.032, 1.502, 4.519, 1.0])
(2, [11.42, 12.86, 0.8683, 5.008, 2.85, 2.7, 4.607, 1.0])
(2, [11.23, 12.63, 0.884, 4.902, 2.879, 2.269, 4.703, 1.0])
(2, [12.36, 13.19, 0.8923, 5.076, 3.042, 3.22, 4.605, 1.0])
(1, [13.22, 13.84, 0.868, 5.395, 3.07, 4.157, 5.088, 1.0])
(1, [12.78, 13.57, 0.8716, 5.262, 3.026, 1.176, 4.782, 1.0])
(1, [12.88, 13.5, 0.8879, 5.139, 3.119, 2.352, 4.607, 1.0])
(1, [14.34, 14.37, 0.8726, 5.63, 3.19, 1.313, 5.15, 1.0])
(1, [14.01, 14.29, 0.8625, 5.609, 3.158, 2.217, 5.132, 1.0])
(1, [14.37, 14.39, 0.8726, 5.569, 3.153, 1.464, 5.3, 1.0])
(1, [12.73, 13.75, 0.8458, 5.412, 2.882, 3.533, 5.067, 1.0])
(0, [17.63, 15.98, 0.8673, 6.191, 3.561, 4.076, 6.06, 2.0])
(0, [16.84, 15.67, 0.8623, 5.998, 3.484, 4.675, 5.877, 2.0])
(0, [17.26, 15.73, 0.8763, 5.978, 3.594, 4.539, 5.791, 2.0])
(0, [19.11, 16.26, 0.9081, 6.154, 3.93, 2.936, 6.079, 2.0])
(0, [16.82, 15.51, 0.8786, 6.017, 3.486, 4.004, 5.841, 2.0])
(0, [16.77, 15.62, 0.8638, 5.927, 3.438, 4.92, 5.795, 2.0])