2.1

(1) 任取Z1 = X1 = (0, 0)T

(2) D11 = 0, D21 = , D31 = 2, D41 = , D51 = , D61 = ,

D71 = , D81 = , D91 = , D101 =

D71最大，X7与Z1最远，故Z2 = X7 = (5, 7)T，T = 1/2 | Z1 – Z2 | =

(3) D12 =, D22 = , D32 = , D42 = 2, D52 = , D62 = ,

D72 = 0, D82 = , D92 = , D102 =

在max{ min(Di1, Di2) } 中，D92最大且大于T，所以Z3 = X9 = (7, 3)T

(4) D13 = , D23 = , D33 = , D43 = , D53 = 5, D63 = ,

D73 = , D83 = 1, D93 = 0, D103 = 1

在max{ min(Di1, Di2, Di3) } 中，D31最大为2，但小于T，故寻找聚类中心结束，共有Z1，Z2，Z3三个聚类中心

(5) 按最近距离将10个模式样本分到三个类别中，有

w1: { X1, X2, X3 } w2: { X4, X5, X6, X7 } w3: { X8, X9, X10 }

2.2

由于伪代码不便于测试和验证所写算法正确性，故实现了基于C++语言的K-均值算法，测试用例为课本K-均值算法小节中的例题。

#include <iostream>

#include <set>

#include <vector>

#include <cmath>

**using** **namespace** std;

#define DIS (0.001)

**int** main() {

vector< pair<**double** , **double**> > points;

*//读入样本点*

points.emplace\_back(make\_pair(0.0f, 0.0f));

points.emplace\_back(make\_pair(1.0f, 0.0f));

points.emplace\_back(make\_pair(0.0f, 1.0f));

points.emplace\_back(make\_pair(1.0f, 1.0f));

points.emplace\_back(make\_pair(2.0f, 1.0f));

points.emplace\_back(make\_pair(1.0f, 2.0f));

points.emplace\_back(make\_pair(2.0f, 2.0f));

points.emplace\_back(make\_pair(3.0f, 2.0f));

points.emplace\_back(make\_pair(6.0f, 6.0f));

points.emplace\_back(make\_pair(7.0f, 6.0f));

points.emplace\_back(make\_pair(8.0f, 6.0f));

points.emplace\_back(make\_pair(6.0f, 7.0f));

points.emplace\_back(make\_pair(7.0f, 7.0f));

points.emplace\_back(make\_pair(8.0f, 7.0f));

points.emplace\_back(make\_pair(9.0f, 7.0f));

points.emplace\_back(make\_pair(7.0f, 8.0f));

points.emplace\_back(make\_pair(8.0f, 8.0f));

points.emplace\_back(make\_pair(9.0f, 8.0f));

points.emplace\_back(make\_pair(8.0f, 9.0f));

points.emplace\_back(make\_pair(9.0f, 9.0f));

*//确定参数值*

**int** N = points.size();

**int** K = 2;

vector< pair<**double**, **double**> > centers;

vector< multiset < pair<**double** , **double**> > > cluster;

*//随机选取第一次迭代的中心*

centers.emplace\_back(points[0]);

centers.emplace\_back(points[1]);

**while** (**true**){

*//清空聚类*

cluster = {};

**for**(**int** i = 0; i < K; i++){

multiset < pair<**double**, **double**> > set = {};

cluster.emplace\_back(set);

}

*//对每一个点*

**for**(**int** i = 0; i < N; i++){

*//计算与当前每个中心的距离*

**double** x = points[i].first;

**double** y = points[i].second;

**double** xCenter = centers[0].first;

**double** yCenter = centers[0].second;

pair<**int**, **int**> shortestLength = make\_pair(0, (x - xCenter)\*(x - xCenter) + (y - yCenter) \* (y - yCenter)) ;

**for**(**int** j = 1; j < K; j++){

xCenter = centers[j].first;

yCenter = centers[j].second;

**double** tempLength = (x - xCenter)\*(x - xCenter) + (y - yCenter) \* (y - yCenter);

**if**(tempLength < shortestLength.second){

shortestLength = make\_pair(j, tempLength);

}

}

*//将当前点归为最近聚类*

cluster[shortestLength.first].insert(make\_pair(x, y));

}

*//对每一个聚类，计算新的中心，若改变，则重复此前步骤*

**bool** isChanged = **false**;

**for**(**int** i = 0; i < K; i++){

**double** sumX = 0;

**double** sumY = 0;

**for**(**auto** it = cluster[i].begin(); it != cluster[i].end(); it++){

sumX += it->first;

sumY += it->second;

}

**double** newX = sumX/cluster[i].size();

**double** newY = sumY/cluster[i].size();

**if**( (abs(centers[i].first - newX) > DIS) || (abs(centers[i].second - newY) > DIS)){

isChanged = **true**;

centers[i] = make\_pair(newX, newY);

}

}

**if**(!isChanged)

**break**;

}

**for**(**int** i = 0; i < K; i++){

cout << "第" << i+1 << "个聚类中心为(" << centers[i].first << ", " << centers[i].second << ")" << endl;

cout << "该聚类包括：" << endl;

**for**(**auto** it = cluster[i].begin(); it != cluster[i].end(); it++){

cout << "(" << it->first << ", " << it->second <<")" << endl;

}

cout << endl;

}

**return** 0;

}

执行该程序，输出结果如下

