**实验三 基于划分的聚类算法研究与实现**

**一、实验目的**

通过编程实现k均值聚类算法。

**二、实验设备和环境**

**Vscode下的gcc环境，c99以上**

**三、实验内容**

编程实现k均值聚类算法，并在数据集上完成测试，分别给出k=5、10、20时的聚类结果，在实验结果的基础上分析k均值及k中心点算法的优缺点。

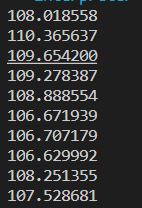
**四、实验过程、结果及分析**

1. 实验过程，关键代码，实验结果及分析。

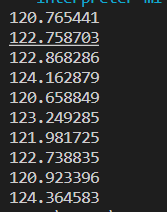
根据k-均值算法思路，随机选取k个向量作为初始的簇，循环地将所有向量分类并调整

所有簇的值，直到无可调整为止，计算分类后的簇内变差，每个k进行10次实验，结果如图

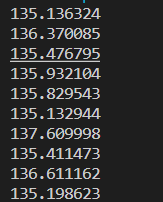
K=20



K=10



K=5



从上述结果看，即使随机选择了初始的向量，只要是在数据集中的，其最终的分类效果都很近似，且分的簇越多，簇内变差倾向于下降，意味着分类更准确。

正面地讲，k-均值通过一个简单的方式实现了多维数据的分类，且其准确度相对较好，效率尚可接受；负面地讲，k-均值对初始k个簇的选择要求较高，且其效率与初始簇选择有很大关联，若选择的簇离开群体较远，效率将会很低，且k-均值对偶现的离群点处理性能较差，当出现少量极端奇点时，k-均值将会给出较差的最终簇内变差结果

而k-中心点算法使用样本点本身作为簇中心，通过找取最优的样本点来作为中心，这主公算法计算量相对大，准确度较好，且比k-均值胜在可以有效处理离群点，但是同样对初始点的选择有一定依赖，潜在地可能导致算法效率下降，但是其最终结果很有可能优于k-均值

本次实验由于python缺乏对应的算法库，因此使用效率更高的c语言进行实验，最终发现在初始簇选择更好的情况下，算法整体的效率很高，大约迭代10到20次即可得出答案，但是如果随机初始簇的值，将会需要较长的运行时间

以下为代码

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<time.h>

typedef int ll;

typedef double d;

#define K 5

d data[320][690];

d core[20][690];

d dist(ll x,ll y){

    d otc=0;

    for(ll i=0;i<690;i++){

        d gap=data[x][i]-core[y][i];

        otc+=gap\*gap;

    }

    return sqrt(otc);

}

int main(){

    FILE\*xx=fopen("data.txt","r");

    for(ll i=0;i<320;i++)for(ll j=0;j<690;j++)fscanf(xx,"%lf",&data[i][j]);

    fclose(xx);

    for(ll loop=10;loop;loop--){

    srand(time(NULL));

    for(ll i=0;i<K;i++){

        ll row=rand()%320;

        for(ll j=0;j<690;j++)core[i][j]=data[row][j];

    }

    for(ll l=0;;l++){

        //printf("%d--\n",l);

        d temp[K][690]={.0},var=.0;ll ct[K]={0},flag=0;

        for(ll i=0;i<320;i++){

            d min=10000;ll pos=0;

            for(ll j=0;j<K;j++){

                d tem=dist(i,j);

                if(min>tem)min=tem,pos=j;

            }

            for(ll j=0;j<690;j++)temp[pos][j]+=data[i][j];

            var+=min\*min;

            ct[pos]++;

        }

        for(ll i=0;i<K;i++)

            for(ll j=0;j<690;j++){

                temp[i][j]/=ct[i];

                if(temp[i][j]!=core[i][j])core[i][j]=temp[i][j],flag=1;

            }

        if(!flag||l>=600/K){printf("%lf\n",var);break;}

    }

    sleep(2);

    }

}

1. 实验中遇到的问题及解决办法。

首先是由于k-均值算法的距离计算要消耗较多时间，因此尝试了预计算向量内积方阵的

方法，结果并不好，思考后发现实际运算量没有减少，因此回归原当场计算距离的方法

然后是初始k簇选择问题，尝试了随机选择k个样本点或是完全随机0-1向量值，发现后者会消耗大量运算时间且结果不如前者，最终选择前者作为初始k簇的选择方法

另外，当运行速度过快时，srand引入的time函数将会返回相同结果，导致连续两次运行所得的结果完全一致，因此在每次尝试后加入2秒的暂停时间，以此充分发挥随机的作用

1. 实验还存在哪些问题。

k-均值算法作为初级算法仍然存在很大优化空间，如对任意的初始簇都有较好的优化

方法的可能性，或者对初始簇的自动估计法

1. 本实验有哪些收获和心得体会。

了解了k-均值算法的核心思想，加深了对其思想的理解，同时引发了对优化其的方法的

一系列思考，并尝试了实践，虽然最终以失败告终，但是在此过程中也是加深了对c语言运行方式的了解

1. 其它需要补充的问题。

暂无

**实验成绩：\_\_\_\_\_\_\_\_\_\_\_\_ 教师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**