中山大學
SUN YAT-SEN UNIVERSITY

# 人工智能程设

题目 Title : Matlab第二次大作业

院 系
School (Department) : 智能工程学院

专业
Major : 智能科学与技术

学生姓名
Student Name : 周德峰

学 号
Student No. : 21312210

指导教师(职称)
Supervisor （Title）: 王帅

时间： 2022 年 12 月 17 日

# 实验原理

K-means算法也被称为K均值算法，是最为常见的聚类算法之一。这里的K为一个常数，代表欲聚类的数量，可由用户指定。K-means是

一个非监督的聚类过程(即在类别信息的引导下完成)，将未标注的数据进行聚类。

在聚类过程中，利用样本间的距离作为指标完成划分操作，这里，可采用基本的欧式距离完成距离测算。

该算法的执行步骤如下：

- 1. 选取*K*个点做为初始聚集的簇心（也可选择非样本点）
- 2. 分别计算每个样本点到*K*个簇核心的距离（这里可采用欧氏距离），找到离该点最近的簇核心，将它归属到对应的簇；
- 3. 所有点都归属到簇之后，*M*个点就分为了*K*个簇。之后重新计算每个簇的重心（平均距离中心），将其定为新的“簇核心”；
- 4. 反复迭代2 - 3步骤，直到达到某个中止条件(可选的条件是簇的中心变化小于某个值*e*)。

# 实验思路

- 计划采用两种方法实现本算法，一种为matlab内置函数，一种为自己设计的算法，最后可以通过结果让两者相互验证
- 关于kmeans，核心既是整个算法流程

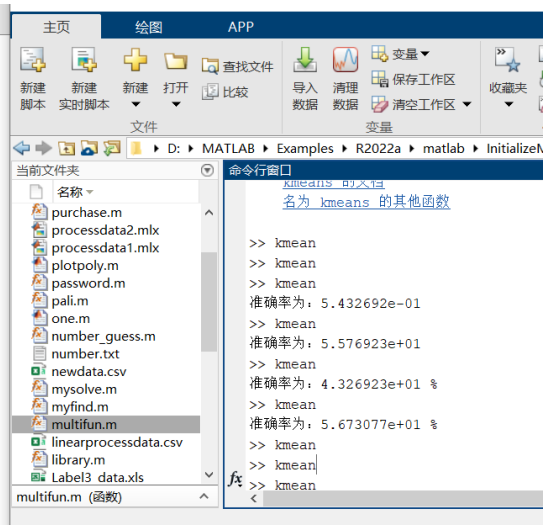
```
创建k个点作为起始质心，尽量相距一定距离
开始迭代
      对数据集中每一个点
         计算每个点到每个簇（即中心点）的距离
         对每个点进行分类
     计算每个簇的中心点
     更新每个簇的中心点
     迭代次数达到条件或者距离变化不超过阈值则终止
结束
```

# 核心代码与结果

## 内置函数

```
% 使用内置函数法
data=xlsread("sonar.xls");
global m n;
[m,n]=size(data);
global label alpha;   %声明为全局变量，方便后续使用
label=data(:, n);

X=data(:, 1:n-1);
[idx,C]=kmeans(X, 2);
prec=0;
for i =1:m
    if idx(i)==label(i)
        prec=prec+1;
    end
end
acc=prec/m;
str1=sprintf("准确率为：%d %%", acc*100);
disp(str1);
```




# 自己设计

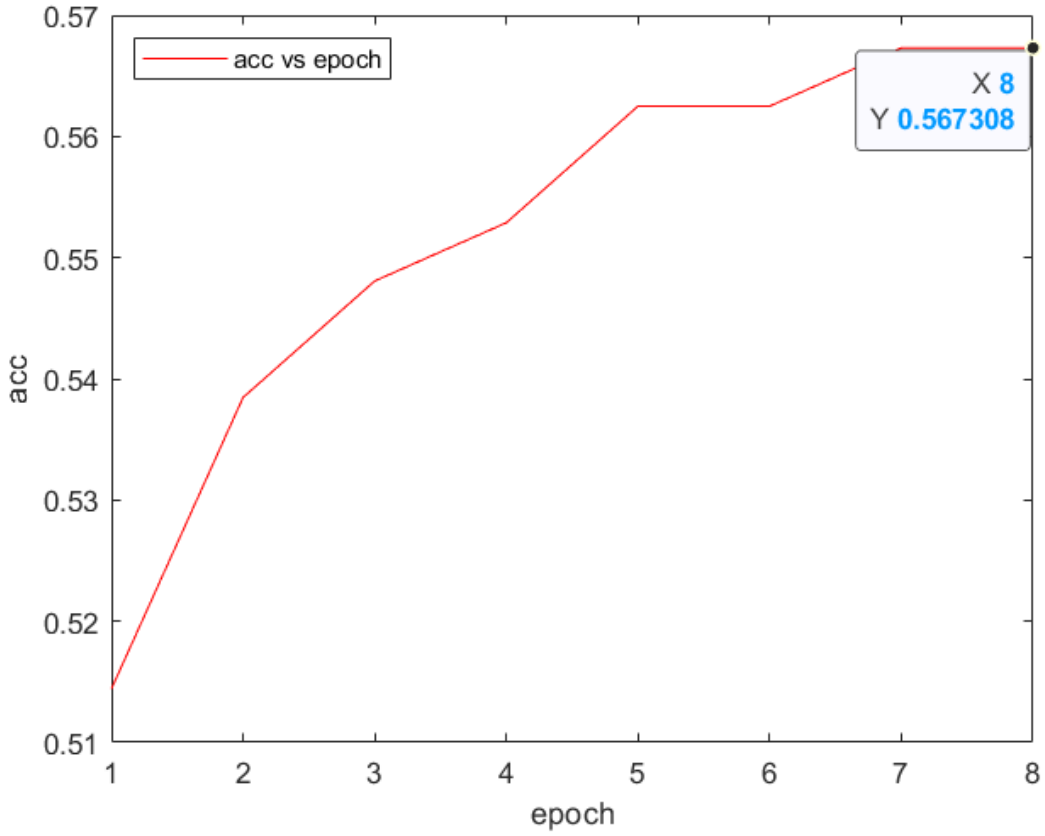
## 主要迭代过程

```
for i =1:epoch
    sumk1=zeros([1,n-1]);
    sumk2=zeros([1,n-1]);
    num1=0;
    num2=0;
    %遍历所有向量
    for j=1:m
        dis1=dist(X(j, :), k1);
        dis2=dist(X(j, :), k2);
        if(dis1>dis2)
            pred(j)=2;
            sumk2=sumk2+X(j, :);
            num2=num2+1;
        else
            pred(j)=1;
            sumk1=sumk1+X(j, :);
            num1=num1+1;
        end
    end
    %计算当前准确率
    prec(i)=acc(pred);
    %更新参数（更新k1，k2）
    k_1=sumk1/num1;
    k_2=sumk2/num2;
%     k1=k_1;
%     k2=k_2;
%     判断是否达到阈值
%     即两次中心的距离变化不超过阈值
    if norm(k_1-k1)<alpha && norm(k_2-k2)<alpha
        k1=k_1;
        k2=k_2;
        break
    else
        k1=k_1;
        k2=k_2;
    end
```

```
end
%由于前面初始化为0，所以此处负责清除0
prec(prec==0)=[];
%设置x并画图
x=1:size(prec,2);
plot(x, prec, 'r');
legend("acc vs epoch",'Location', 'NorthWest');
xlabel("epoch")
ylabel("acc");

%输出最后的中心点
str1=sprintf("x1:%f , x2: %f ", k1, k2);
disp("final center");
disp(str1);
```

## Acc




## 最终中心点

（结果太长放不下，助教/老师有兴趣可以直接跑下代码）

```
final center
x1:0.029810 , x2: 0.039137 x1:0.046586 , x2: 0.059633 x1:0.087072 , x2: 0.112131 x1:0.130231 , x2: 0.142735 x1:0.
```

# 心得体会

- **在设计算法的过程中，要灵活使用debug功能，可以使调试代码更加便捷**
- **重视数据的维度！**

- 注意kmeans为启发式算法，所以每次得到的结果都会是局部最优解，而不是全局最优解，所以每次的结果可能都不一样，取决于初始化时选择的向量
- 两种方法得到的最终acc并不高，所以可能是原数据的label并不准确，或者原数据的可区分度并不大
- 有没有可能分更多类效果越好?

开始证明，分别对三类，四类，五类进行4-5次实验，结果发现acc并不高，所以推测acc不高的原因在于数据间本身可分性并不强

```
>> kmean
三类准确率为：3.509615e+01 %
>> kmean
三类准确率为：2.644231e+01 %
>> kmean
三类准确率为：3.365385e+01 %
>> kmean
三类准确率为：2.692308e+01 %
>> kmean
三类准确率为：3.317308e+01 %
>> kmean
四类准确率为：1.971154e+01 %
>> kmean
四类准确率为：2.067308e+01 %
>> kmean
四类准确率为：1.778846e+01 %
>> kmean
四类准确率为：3.125000e+01 %
>> kmean
五类准确率为：3.653846e+01 %
>> kmean
五类准确率为：1.923077e+01 %
>> kmean
五类准确率为：1.490385e+01 %
>> kmean
五类准确率为：1.923077e+01 %
>>
```

# 加分项

- 关于终止条件，设定了两个，一个为迭代次数，当迭代次数超过50时便会停止，一个是阈值，当两次中心距离变化小于阈值时，同样会停止

```
epoch=50;
%      判断是否达到阈值
%      即两次中心的距离变化不超过阈值
    if norm(k_1-k1)<alpha && norm(k_2-k2)<alpha
        k1=k_1;
        k2=k_2;
        break
    else
        k1=k_1;
        k2=k_2;
    end
```

- **使用内置子函数计算两个向量间的距离和每次迭代过程的acc**

```
function y=dist(x1, x2, n)
d=0;
global n;
for i =1:n-1        %n表示向量维度！总共n-1列！
    d=d+(x1(i)-x2(i))^2;
end
d=sqrt(d);
y=d;        %返回两个向量间的距离
end
%计算单次迭代的acc
function y=acc(pred, m)
y=0;
global label m;     %使用全局变量
for i =1:m
    if pred(i)==label(i)
        y=y+1;
    end
end
y=y/m;  %返回单次迭代acc
end
```

- **关于初始选择向量，同样设置了一个阈值，此阈值的目的在于使最开始随机选择的两个向量尽可能远，有利于后续算法的计算与更新！**

```
%初始化选择向量,尽量选择相距较大的，方便后续聚类
beta=3;
while 1
    a=randperm(m,2);
    k1=X(a(1), :);
    k2=X(a(2), :);
    if dist(k1,k2)>beta
        break
    end
end
```

- **使用两种算法，相互验证！自己设计的算法结果与内置函数相比极相似，甚至acc略高于内置函数，更进一步说明自己算法设计的正确性**
- **对三类，四类，五类聚类进行进一步实验，来推测实验本身acc不高的原因**