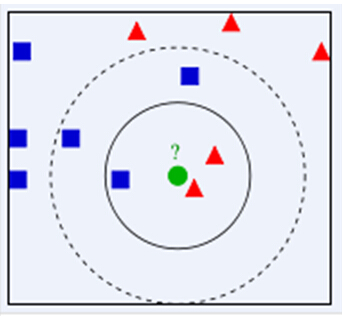
# ****K-近邻算法（KNN）****

## **一 . K-近邻算法（KNN）概述**

    最简单最初级的分类器是将全部的训练数据所对应的类别都记录下来，当测试对象的属性和某个训练对象的属性完全匹配时，便可以对其进行分类。但是怎么可能所有测试对象都会找到与之完全匹配的训练对象呢，其次就是存在一个测试对象同时与多个训练对象匹配，导致一个训练对象被分到了多个类的问题，基于这些问题呢，就产生了KNN。

     KNN是通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中K通常是不大于20的整数。KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

     下面通过一个简单的例子说明一下：如下图，绿色圆要被决定赋予哪个类，是红色三角形还是蓝色四方形？如果K=3，由于红色三角形所占比例为2/3，绿色圆将被赋予红色三角形那个类，如果K=5，由于蓝色四方形比例为3/5，因此绿色圆被赋予蓝色四方形类。



## 二、算法指导思想

kNN算法的指导思想是“近朱者赤，近墨者黑”，由你的邻居来推断出你的类别。先计算待分类样本与已知类别的训练样本之间的距离，找到距离与待分类样本数据最近的k个邻居；再根据这些邻居所属的类别来判断待分类样本数据的类别。

1、相似性度量

用空间内两个点的距离来度量。距离越大，表示两个点越不相似。距离的选择有很多，通常用比较简单的欧式距离。

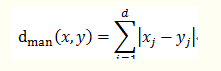
欧式距离：



马氏距离：马氏距离能够缓解由于属性的线性组合带来的距离失真，是数据的协方差矩阵。



曼哈顿距离：



切比雪夫距离：



2、 类别的判定

投票决定：少数服从多数，近邻中哪个类别的点最多就分为该类。

加权投票法：根据距离的远近，对近邻的投票进行加权，距离越近则权重越大（权重为距离平方的倒数）

## 三、优缺点

优点：

(1)简单，易于理解，易于实现，无需估计参数，无需训练；

(2)适合对稀有事件进行分类；

(3)特别适合于多分类问题(multi-modal,对象具有多个类别标签)， kNN比SVM的表现要好。

缺点：

(1)懒惰算法，对测试样本分类时的计算量大，内存开销大，评分慢；

(2)当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数；

(3)可解释性较差，无法给出决策树那样的规则。

## 四、常见问题

1、k值的设定

k值选择过小，得到的近邻数过少，会降低分类精度，同时也会放大噪声数据的干扰；而如果k值选择过大，并且待分类样本属于训练集中包含数据数较少的类，那么在选择k个近邻的时候，实际上并不相似的数据亦被包含进来，造成噪声增加而导致分类效果的降低。

如何选取恰当的K值也成为KNN的研究热点。k值通常是采用交叉检验来确定（以k=1为基准）。

经验规则：k一般低于训练样本数的平方根。

2、类别的判定方式

投票法没有考虑近邻的距离的远近，距离更近的近邻也许更应该决定最终的分类，所以加权投票法更恰当一些。

3、距离度量方式的选择

高维度对距离衡量的影响：众所周知当变量数越多，欧式距离的区分能力就越差。

变量值域对距离的影响：值域越大的变量常常会在距离计算中占据主导作用，因此应先对变量进行标准化。

4、训练样本的参考原则

学者们对于训练样本的选择进行研究，以达到减少计算的目的，这些算法大致可分为两类。第一类,减少训练集的大小。KNN算法存储的样本数据,这些样本数据包含了大量冗余数据,这些冗余的数据增了存储的开销和计算代价。缩小训练样本的方法有:在原有的样本中删掉一部分与分类相关不大的样本样本,将剩下的样本作为新的训练样本;或在原来的训练样本集中选取一些代表样本作为新的训练样本；或通过聚类,将聚类所产生的中心点作为新的训练样本。

在训练集中，有些样本可能是更值得依赖的。可以给不同的样本施加不同的权重，加强依赖样本的权重，降低不可信赖样本的影响。

5、性能问题

kNN是一种懒惰算法，而懒惰的后果：构造模型很简单，但在对测试样本分类地的系统开销大，因为要扫描全部训练样本并计算距离。

已经有一些方法提高计算的效率，例如压缩训练样本量等。

## 五、 算法流程

就是在训练集中数据和标签已知的情况下，输入测试数据，将测试数据的特征与训练集中对应的特征进行相互比较，找到训练集中与之最为相似的前K个数据，则该测试数据对应的类别就是K个数据中出现次数最多的那个分类，其算法的描述为：

1）计算测试数据与各个训练数据之间的距离；

2）按照距离的递增关系进行排序；

3）选取距离最小的K个点；

4）确定前K个点所在类别的出现频率；

5）返回前K个点中出现频率最高的类别作为测试数据的预测分

类。

## 六、数据集描述

因为这种分类方式比较简单，就只采用简单的数据来测试

trainData = [1.0,2.0;1.2,0.1;0.1,1.4;0.3,3.5];

trainClass = [1,1,2,2];

testData = [0.5,2.3];

k = 3;

## 七、结果：

最终的分类结果为：

2

****K-均值聚类（K-means）****

## **一、 K-均值聚类（K-means）概述**

### **1. 聚类**

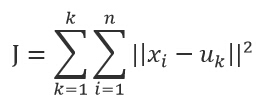
     “类”指的是具有相似性的集合。聚类是指将数据集划分为若干类，使得类内之间的数据最为相似，各类之间的数据相似度差别尽可能大。聚类分析就是以相似性为基础，对数据集进行聚类划分，属于无监督学习。

### **2. 无监督学习和监督学习**

     和KNN所不同，K-均值聚类属于无监督学习。那么监督学习和无监督学习的区别在哪儿呢？监督学习知道从对象（数据）中学习什么，而无监督学习无需知道所要搜寻的目标，它是根据算法得到数据的共同特征。比如用分类和聚类来说，分类事先就知道所要得到的类别，而聚类则不一样，只是以相似度为基础，将对象分得不同的簇。

### **3. K-means**

     k-means算法是一种简单的迭代型聚类算法，采用距离作为相似性指标，从而发现给定数据集中的K个类，且每个类的中心是根据类中所有值的均值得到，每个类用聚类中心来描述。对于给定的一个包含n个d维数据点的数据集X以及要分得的类别K,选取欧式距离作为相似度指标，聚类目标是使得各类的聚类平方和最小，即最小化：



结合最小二乘法和拉格朗日原理，聚类中心为对应类别中各数据点的平均值，同时为了使得算法收敛，在迭代过程中，应使最终的聚类中心尽可能的不变。

## **二、算法流程**

K-means是一个反复迭代的过程，算法分为四个步骤：

(1) 选取数据空间中的K个对象作为初始中心，每个对象代表一个聚类中心；

(2) 对于样本中的数据对象，根据它们与这些聚类中心的欧氏距离，按距离最近的准则将它们分到距离它们最近的聚类中心（最相似）所对应的类；

(3) 更新聚类中心：将每个类别中所有对象所对应的均值作为该类别的聚类中心，计算目标函数的值；

(4) 判断聚类中心和目标函数的值是否发生改变，若不变，则输出结果，若改变，则返回2）。

用以下例子加以说明：

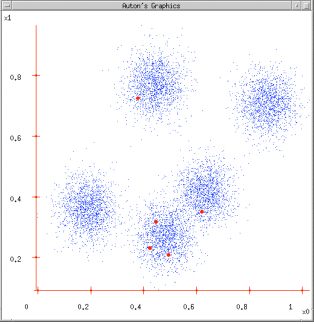
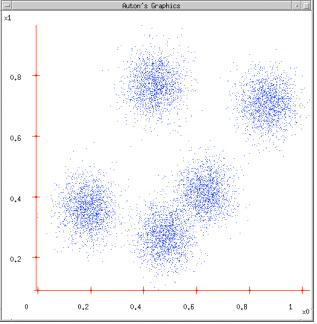
  

    　　图1　　　　　　　　　　　　　图2

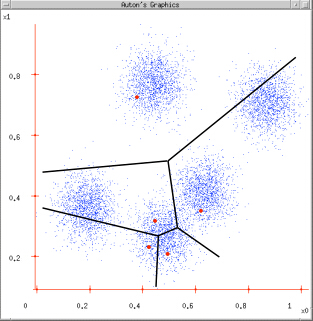
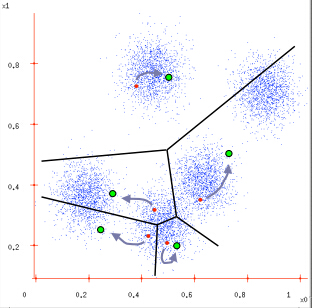
  

　　　图3　　　　　　　　　　　 　图4

图1：给定一个数据集；

图2：根据K = 5初始化聚类中心，保证　聚类中心处于数据空间内；

图3：根据计算类内对象和聚类中心之间的相似度指标，将数据进行划分；

图4：将类内之间数据的均值作为聚类中心，更新聚类中心。

最后判断算法结束与否即可，目的是为了保证算法的收敛。

## 三、 Kmeans算法优缺点

优点：

(1). 算法原理简单。需要调节的超参数就是一个k。

(2). 由具有出色的速度和良好的可扩展性。

缺点：

(1). 在 Kmeans 算法中 kk 需要事先确定，这个 kk 值的选定有时候是比较难确定。

(2). 在 Kmeans 算法中，首先需要初始k个聚类中心，然后以此来确定一个初始划分，然后对初始划分进行优化。这个初始聚类中心的选择对聚类结果有较大的影响，一旦初始值选择的不好，可能无法得到有效的聚类结果。多设置一些不同的初值，对比最后的运算结果，一直到结果趋于稳定结束。

(3). 该算法需要不断地进行样本分类调整，不断地计算调整后的新的聚类中心，因此当数据量非常大时，算法的时间开销是非常大的。

(4). 对离群点很敏感。

(5). 从数据表示角度来说，在 Kmeans 中,我们用单个点来对 cluster 进行建模，这实际上是一种最简化的数据建模形式。这种用点来对 cluster 进行建模实际上就已经假设了各 cluster的数据是呈圆形(或者高维球形)或者方形等分布的。不能发现非凸形状的簇。但在实际生活中，很少能有这种情况。所以在 GMM 中，使用了一种更加一般的数据表示，也就是高斯分布。

(6). 从数据先验的角度来说，在 Kmeans 中,我们假设各个 cluster 的先验概率是一样的,但是各个 cluster 的数据量可能是不均匀的。举个例子,cluster A 中包含了10000个样本,cluster B 中只包含了100个。那么对于一个新的样本,在不考虑其与A cluster、 B cluster 相似度的情况,其属于 cluster A 的概率肯定是要大于 cluster B的。

针对Kmeans算法的缺点，很多前辈提出了一些改进的算法。例如 K-modes 算法，实现对离散数据的快速聚类，保留了Kmeans算法的效率同时将Kmeans的应用范围扩大到离散数据。还有K-Prototype算法，可以对离散与数值属性两种混合的数据进行聚类，在K-prototype中定义了一个对数值与离散属性都计算的相异性度量标准。

## 四、数据集描述

%% 构造产生多维正态随机数

mu1=[0 0 0];%期望向量

S1=[0.23 0 0;0 0.87 0;0 0 0.56];%协方差矩阵

n=100;%规模

data1=mvnrnd(mu1,S1,n); %产生高斯分布数据

%%第二类数据

mu2=[1.25 1.25 1.25];

S2=[0.23 0 0;0 0.87 0;0 0 0.56];

data2=mvnrnd(mu2,S2,100);

%第三个类数据

mu3=[-1.25 1.25 -1.25];

S3=[0.23 0 0;0 0.87 0;0 0 0.56];

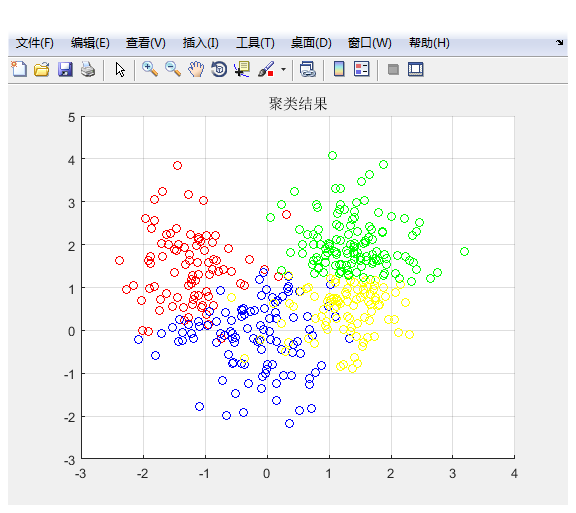
data3=mvnrnd(mu3,S3,100);

mu4=[1.5 1.5 1.5];

S4=[0.23 0 0;0 0.87 0;0 0 0.56];

data4 =mvnrnd(mu4,S4,100);

## 五、结果:



两者比较



KNN代码

%% KNN

clear all

clc

%% data

trainData = [1.0,2.0;1.2,0.1;0.1,1.4;0.3,3.5];

trainClass = [1,1,2,2];

testData = [0.5,2.3];

k = 3;

%% distance

row = size(trainData,1);

col = size(trainData,2);

test = repmat(testData,row,1);

dis = zeros(1,row);

for i = 1:row

diff = 0;

for j = 1:col

diff = diff + (test(i,j) - trainData(i,j)).^2;

end

dis(1,i) = diff.^0.5;

end

%% sort

jointDis = [dis;trainClass];

sortDis= sortrows(jointDis');

sortDisClass = sortDis';

%% find

class = sort(2:1:k);

member = unique(class);

num = size(member);

max = 0;

for i = 1:num

count = find(class == member(i));

if count > max

max = count;

label = member(i);

end

end

disp('最终的分类结果为：');

fprintf('%d\n',label)

K-means代码

%%%K-means

clear all

clc

%% 构造随机数据

mu1=[0 0 0];

S1=[0.23 0 0;0 0.87 0;0 0 0.56];

data1=mvnrnd(mu1,S1,100); %产生高斯分布数据

%%第二类数据

mu2=[1.25 1.25 1.25];

S2=[0.23 0 0;0 0.87 0;0 0 0.56];

data2=mvnrnd(mu2,S2,100);

%第三个类数据

mu3=[-1.25 1.25 -1.25];

S3=[0.23 0 0;0 0.87 0;0 0 0.56];

data3=mvnrnd(mu3,S3,100);

mu4=[1.5 1.5 1.5];

S4=[0.23 0 0;0 0.87 0;0 0 0.56];

data4 =mvnrnd(mu4,S4,100);

%显示数据

figure;

plot3(data1(:,1),data1(:,2),data1(:,3),'+');

title('原始数据');

hold on

plot3(data2(:,1),data2(:,2),data2(:,3),'r+');

plot3(data3(:,1),data3(:,2),data3(:,3),'g+');

plot3(data4(:,1),data4(:,2),data3(:,3),'y+');

grid on;

data=[data1;data2;data3;data4];

[row,col] = size(data);

K = 4;

max\_iter = 300;%%迭代次数

min\_impro = 0.1;%%%%最小步长

display = 1;%%%判定条件

center = zeros(K,col);

U = zeros(K,col);

%% 初始化聚类中心

mi = zeros(col,1);

ma = zeros(col,1);

for i = 1:col

mi(i,1) = min(data(:,i));

ma(i,1) = max(data(:,i));

center(:,i) = ma(i,1) - (ma(i,1) - mi(i,1)) \* rand(K,1);

end

%% 开始迭代

for o = 1:max\_iter

%% 计算欧氏距离,用norm函数

for i = 1:K

dist{i} = [];

for j = 1:row

dist{i} = [dist{i};data(j,:) - center(i,:)];

end

end

minDis = zeros(row,K);

for i = 1:row

tem = [];

for j = 1:K

tem = [tem norm(dist{j}(i,:))];

end

[nmin,index] = min(tem);

minDis(i,index) = norm(dist{index}(i,:));

end

%% 更新聚类中心

for i = 1:K

for j = 1:col

U(i,j) = sum(minDis(:,i).\*data(:,j)) / sum(minDis(:,i));

end

end

%% 判定

if display

end

if o >1,

if max(abs(U - center)) < min\_impro;

break;

else

center = U;

end

end

end

%% 返回所属的类别

class = [];

for i = 1:row

dist = [];

for j = 1:K

dist = [dist norm(data(i,:) - U(j,:))];

end

[nmin,index] = min(dist);

class = [class;data(i,:) index];

end

%% 显示最后结果

[m,n] = size(class);

figure;

title('聚类结果');

hold on;

for i=1:row

if class(i,4)==1

plot3(class(i,1),class(i,2),class(i,3),'ro');

elseif class(i,4)==2

plot3(class(i,1),class(i,2),class(i,3),'go');

elseif class(i,4) == 3

plot3(class(i,1),class(i,2),class(i,3),'bo');

else

plot3(class(i,1),class(i,2),class(i,3),'yo');

end

end

grid on;