**实验1 相似度、距离、最近邻分类器**

**一、实验目的**

（1）理解相似度、距离的度量方式。

（2）理解最近邻分类器的工作原理。

**二、实验内容**

1、编程实现任意给定两个相同维度的向量之间的欧氏距离计算函数dist\_E(x,y)。

输入：两个任意k维向量x和y，其中k的值随由数据决定。如x=[3,20,3.5], y=[-3,34,7]。

测试代码如下：

x=[3,20,3.5]; y=[-3,34,7];

dist=dist\_E(x,y);

（备注：代码粘贴在此处）

function dist = dist\_E(x,y)

dist = [x;y];

dist = pdist(dist); % ¼ÆËã¸÷ÐÐÏòÁ¿Ö®¼äµÄÅ·Ê½¾àÀë

end

2、编程实现任意给定两个相同维度的向量之间的夹角余弦相似度计算函数sim=sim\_COS(x,y)。输入：两个任意k维向量x和y，其中k的值随由数据决定。

测试代码如下：

x=[3,20,3.5]; y=[-3,34,7];

sim=sim\_COS(x,y);

（备注：代码粘贴在此处）

function sim = sim\_COS(X,Y)

sim = dot(X,Y)/(norm(X)\*norm(Y));

3、训练数据见trainingData.mat，测试数据见testingData.mat。编程实现K最近邻算法。函数为：[class]=KNN\_Classify\_E(trainingSamples, trainingLabels, testingSample,k)。

测试代码如下：

load('testingData.mat');

load('trainingData.mat');

%[trAttr]=normalize(trAttr);

%[tstAttr]=normalize(tstAttr);

k=3;

predictlabel =zeros(length(tstLabels),1);

for i=1:length(tstLabels)

predictlabel(i) =KNN\_Classify\_E(trAttr, trLabels ,tstAttr(i,:) ,k);

end

accurate =sum(predictlabel==tstLabels)/length(tstLabels);

disp(accurate);

KNN\_Classify\_E函数的处理流程与说明如下：

输入参数：k值、trainingSamples（训练数据集，M\*N矩阵，M为样本数，N为属性数）、trainingLabels（训练数据集的分类标签0、1、2...，M\*1矩阵）, testingSample（测试数据，1\*N矩阵）

输出参数：class（测试数据对应类别标签）

算法流程：

1、得到训练数据集trainingSamples的大小M，N

2、初始化Distance数组（M\*1），用来存储每个训练样本与测试样本的距离。

3、对每一个训练样本trainingSamples(i,:)【for i=1:M】，计算其与测试样本testingSample之间的距离，存储在Distance（i）中。【计算两个样本之间的欧氏距离，调用函数dist\_E(vect1,vect2)，其中dist\_E()为自定义函数，参见上机练习2】

4、对Distance数组排升序【sort函数】

5、取得排序前K个距离对应的序号，将序号对应的训练数据的分类标签得到赋给labs

6、得到labs数组的不重复元素，存储在数组All\_labs 【unique函数】

7、得到不重复元素（数组All\_labs ）的个数LabNum

8、 （for i=1: LabNum ）对每一个不重复的分类标签All\_labs(i) ，查找【find函数】最近的k个类别标签labs中，等于All\_labs(i)的有几个，将该数目作为第i类的投票数Vote(i)

9、求投票数Vote(i)的最大值所在的索引ind

10、All\_labs(ind)是最大投票数对应的类别标签，即为算法输出结果class

10、All\_labs(ind)是最大投票数对应的类别标签，即为算法输出结果class

函数[class]=KNN\_Classify\_E(trainingSamples, trainingLabels, testingSample,k)：

**（将执行正确的代码粘贴在此处，核心代码要求有注释）**

function [class] = KNN\_Classify\_E(trainingSamples, trainingLabels, testingSample,k)

% 1 µÃµ½ÑµÁ·Êý¾Ý¼¯trainingSamplesµÄ´óÐ¡M£¬N

[M,N] = size(trainingSamples);

% 2 ³õÊ¼»¯DistanceÊý×é£¨M\*1£©£¬ÓÃÀ´´æ´¢Ã¿¸öÑµÁ·Ñù±¾Óë²âÊÔÑù±¾µÄ¾àÀë

Distance = zeros(M,1);

% 3 ¶ÔÃ¿Ò»¸öÑµÁ·Ñù±¾trainingSamples(i,:),¼ÆËãÆäÓë²âÊÔÑù±¾testingSampleÖ®¼äµÄ¾àÀë£¬´æ´¢ÔÚDistance£¨i£©ÖÐ

for i=1:M

training = trainingSamples(i,:);

Distance(i) = dist\_E(training,testingSample); % dist\_EÎªÅ·ÊÏ¾àÀë¼ÆËãº¯Êý

end

% 4 ¶ÔDistanceÊý×éÅÅÉýÐò

[val ind] = sort(Distance); % sortº¯ÊýÄ¬ÈÏÉýÐò£¬´Ë´¦ÓÐÁ½¸öÊä³ö²ÎÊý£¬val£º¾àÀëÅÅÐòºóµÄÖµ£¬ind£ºvalÖÐÃ¿¸öÔªËØÔÚÔ­¾ØÕóDistanceÖÐµÄÎ»ÖÃ

% 5 È¡µÃÅÅÐòÇ°K¸ö¾àÀë¶ÔÓ¦µÄÐòºÅ-->ind(1:k)£¬½«ÐòºÅ¶ÔÓ¦µÄÑµÁ·Êý¾ÝµÄ·ÖÀà±êÇ©trainingLabelsµÃµ½¸³¸ølabs

labs = trainingLabels(ind(1:k));

% 6 µÃµ½labsÊý×éµÄ²»ÖØ¸´ÔªËØ£¬´æ´¢ÔÚÊý×éAll\_labs

All\_labs = unique(labs); % uniqueº¯Êý£º·µ»ØÓëlabsÖÐÏàÍ¬µÄÊý¾Ý£¬µ«ÊÇ²»°üº¬ÖØ¸´Ïî

% 7 µÃµ½²»ÖØ¸´ÔªËØ£¨Êý×éAll\_labs £©µÄ¸öÊýLabNum

LabNum = length(All\_labs);

% 8 ¶ÔÃ¿Ò»¸ö²»ÖØ¸´µÄ·ÖÀà±êÇ©All\_labs(i) £¬²éÕÒ×î½üµÄk¸öÀà±ð±êÇ©labsÖÐ£¬µÈÓÚAll\_labs(i)µÄÓÐ¼¸¸ö£¬½«¸ÃÊýÄ¿×÷ÎªµÚiÀàµÄÍ¶Æ±ÊýVote(i)

for i=1:LabNum

Vote(i) = length(find(labs==All\_labs(i)));

end

% 9 ÇóÍ¶Æ±ÊýVote(i)µÄ×î´óÖµËùÔÚµÄË÷Òýind

[val ind] = max(Vote); % maxº¯Êý ·µ»ØVoteµÄ×î´óÔªËØ

% 10 All\_labs(ind)ÊÇ×î´óÍ¶Æ±Êý¶ÔÓ¦µÄÀà±ð±êÇ©£¬¼´ÎªËã·¨Êä³ö½á¹ûclass

class = All\_labs(ind);

end

选做题：函数[Samples2]=normalize(Samples)：

**（将执行正确的代码粘贴在此处，核心代码要求有注释）**

function [Samples2]=normalize(Samples)

[M,N] = size(Samples);

Samples2 = zeros(M,N);

for i=1:N

sam = Samples(:,i);

STD = std(sam); % Çó±ê×¼²î

MEAN = mean(sam); % Çó¾ùÖµ

x = (sam-MEAN)/STD;

Samples2(:,i)=x;

end

end