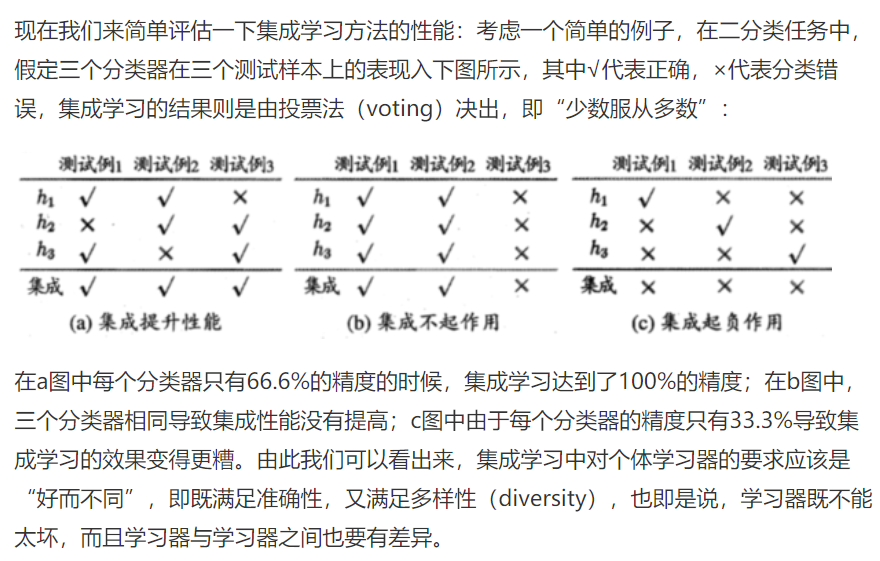
1. （必填）自己提出的问题的理解（罗列全部）：

提出的问题1：knn在样本不平衡时会有缺陷，例如有一类样本容量很大，其他类样本容量小，导致大容量的样本在k个邻居中占多数。有无改进的办法？

讨论后的理解：可以采用设置权值的方法，和该样本距离小的邻居权值大，来改进Knn算法。加权最简单的方法是反函数，就是返回距离的倒数，比如距离d，权重1/d。但有时候，完全一样或非常接近的商品权重会很大甚至无穷大。基于这样的原因，在求距离倒数的时候，在距离上加一个常量：weight=1/(distance+const)。这种方法会使得算法对于噪声数据变得异常敏感。还可以使用高斯函数，高斯函数在图形上的形状像一个倒悬着的钟，在距离为零的时候权重为1，随着距离的增大，权重减小，但不会变为0

提出的问题2：集成学习选取多个学习器然后综合评价为什么可以获得更好的分类结果，换句话说，分类器中必定有好的有坏的，最终结果从生活经验来看不是应该趋向于最终的平均值嘛？

讨论后的理解：个体分类器如果不能做到好而不同，就有可能会导致下图c 的情况，即集成反而起了副作用

1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：
2. 问题3：什么是余弦相似度？

自己的理解：余弦相似性通过测量两个[向量](https://baike.baidu.com/item/%E5%90%91%E9%87%8F)的夹角的[余弦](https://baike.baidu.com/item/%E4%BD%99%E5%BC%A6)值来度量它们之间的相似性。0度角的余弦值是1，而其他任何角度的余弦值都不大于1；并且其最小值是-1。从而两个向量之间的角度的余弦值确定两个向量是否大致指向相同的方向。两个向量有相同的指向时，余弦相似度的值为1；两个向量夹角为90°时，余弦相似度的值为0；两个向量指向完全相反的方向时，余弦相似度的值为-1。这结果是与向量的长度无关的，仅仅与向量的指向方向相关。余弦相似度通常用于正空间，因此给出的值为-1到1之间。

1. 问题4：knn算法K的取值是怎么确定的？

自己的理解：常用的方法是从k=1开始，使用检验集估计分类器的误差率。重复该过程，每次K增值1，允许增加一个近邻。选取产生最小误差率的K。一般k的取值不超过20，上限是n的开方，随着数据集的增大，K的值也要增大。

1. 问题5：bagging什么时候会降低稳定分类器的准确率？

自己的理解：在稳定分类器本身效果不好，且相互之间差异较大，即“差而不同”的时候也。此时分类器的准确率可能会比最差的分类器还差。

1. （必填）读书计划

1、本周完成的内容章节：如3.9-3.10

2、下周计划：5.1

四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1、读书摘要及理解（选做）

3.9 k-近邻算法

from numpy import \*

import operator as op

# 原始训练集

def createDataSet():

group = array([[1.0, 1.1], [1.0, 1.0], [0, 0], [0, 0.1]])

labels = ['A', 'A', 'B', 'B']

return group, labels

def classify0(intX, dataSet, labels, k):

dataSetSize = dataSet.shape[0]

diffMat = tile(intX, (dataSetSize,1)) - dataSet

sqdiffMat = diffMat\*\*2

sqDistances = sqdiffMat.sum(axis=1)

distances = sqDistances\*\*0.5

sortedDistIndicies = distances.argsort()

classCount = {}

for i in range(k):

voteIlabel = labels[sortedDistIndicies[i]]

classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

sortedClassCount = sorted(classCount.items(), key=op.itemgetter(1), reverse=True)

return sortedClassCount[0][0]