1. （必填）自己提出的问题的理解（罗列全部）：
2. 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。

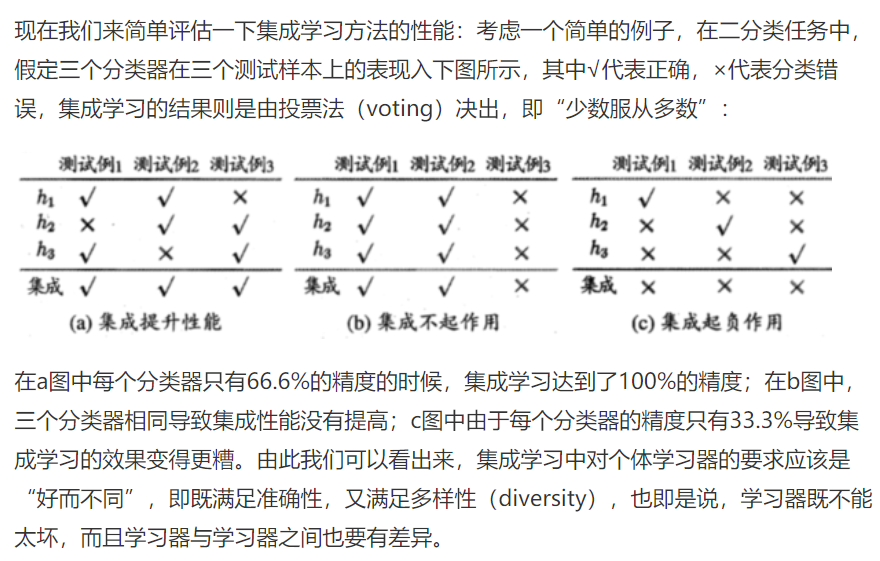
加权最简单的方法是反函数，就是返回距离的倒数，比如距离d，权重1/d。但有时候，完全一样或非常接近的商品权重会很大甚至无穷大。基于这样的原因，在求距离倒数的时候，在距离上加一个常量：weight=1/(distance+const)。这种方法会使得算法对于噪声数据变得异常敏感。

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

1. 提出的问题2：adaboost生成的一系列基本分类器的权重是相同的吗？

讨论后的理解：adaboost生成的一系列基本分类器的权重是不同的，是根据分类器的正确率ei,计算出个体学习器的系数ai.公式为

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：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

自己的理解：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. 问题4：knn算法看起来精确度似乎得不到保证，但是书上却说精确度非常高，是否有数学证明可以证明knn算法精确度存在某个确定的下限

自己的理解：可以通过数学证明，最近邻分类器的泛化错误率不超过贝叶斯最优分类器错误率的两倍



1. 问题5：knn每个样本都要遍历一遍数据集，这样算法时间复杂度很高。如何优化knn算法？

自己的理解：可以使用kd树，将时间复杂度从n将为log n 。kd树是每个节点都为k维点的[二叉树](https://baike.baidu.com/item/%E4%BA%8C%E5%8F%89%E6%A0%91)。所有非叶子节点可以视作用一个[超平面](https://baike.baidu.com/item/%E8%B6%85%E5%B9%B3%E9%9D%A2)把空间分割成两个[半空间](https://baike.baidu.com/item/%E5%8D%8A%E7%A9%BA%E9%97%B4)。节点左边的子树代表在超平面左边的点，节点右边的子树代表在超平面右边的点。

选择超平面的方法如下：每个节点都与k维中垂直于超平面的那一维有关。因此，如果选择按照x轴划分，所有x值小于指定值的节点都会出现在左子树，所有x值大于指定值的节点都会出现在右子树。这样，超平面可以用该x值来确定，其[法线](https://baike.baidu.com/item/%E6%B3%95%E7%BA%BF)为x轴的[单位向量](https://baike.baidu.com/item/%E5%8D%95%E4%BD%8D%E5%90%91%E9%87%8F)。

1. 问题6：书中的两种集成分类器方法中每个基本分类器都是使用的相同的分类算法，如果使用不同的分类算法有没有可能实验结合每种算法各自的优点的可能？

自己的理解：书上的两种分类器，bagging和boosting都属于同质弱学习器，每个基本分类器都使用相同的分类算法。但是其他的分类器方法，如stacking，使用的是异质弱学习器，并行地学习它们。Stacking通过训练一个元模型将这些弱学习器组合起来，根据不同弱模型的预测结果输出一个最终的预测结果。

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：2.1-2.5

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-近邻学习

所谓K近邻算法，即是给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例最邻近的K个实例， 这K个实例的多数属于某个类，就把该输入实例分类到这个类中。Knn算法不仅可以用于分类，还可以用于回归。通过找出一个样本的k个最近邻居，将这些邻居的属性的平均值赋给该样本，就可以得到该样本的属性。

但是，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。这个缺陷可以采用权值的方法来改进。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。

3.10 分类器的集成

Bagging是通过结合几个模型降低泛化误差的技术，可以把不稳定的学习算法的性能显著提升。主要想法是分别训练几个不同的模型，然后对每个测试样例进行分类，由k个分类投票（权重相同）决定。

AdaBoost是通过每次降低个体学习器的分类误差，加大效果好的个体学习器的重要性，得到最终的集成学习器。

集成学习器的形式为

其中系数的计算方法与错误率有关，个体学习器错误率越低，权重越高。

Boosting在大多数情况下比Bagging的效果更好，而且当基本分类器是不稳定的时候，Boosting对性能的提升更大