1. **自己提出的问题的理解：**

问题1：为什么Bagging算法取一个自展复制的时候每个样本平均有63.2%的原始样本？

讨论后的理解：因为自展复制的时候取样的方法使用的是自助法，即每次在D中取一个样例放入D1中，然后再将取出来的样本放回，m次取样之后某样例没有被选中取出的概率为(1-1/m)^m，对m取极限即为0.368，这样约有36.8的数据在自助法取样的过程中不会被取出，所以每个样本中平均会含有原始样本63.2%的样本。

问题2：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？

讨论后的理解：有的。Stacking算法即为使用不同的分类算法的分类器作为基本分类器，而书上所写的bagging和boosting都使用相同的分类算法。

引用：

bagging，该方法通常考虑的是同质弱学习器，相互独立地并行学习这些弱学习器，并按照某种确定性的平均过程将它们组合起来。

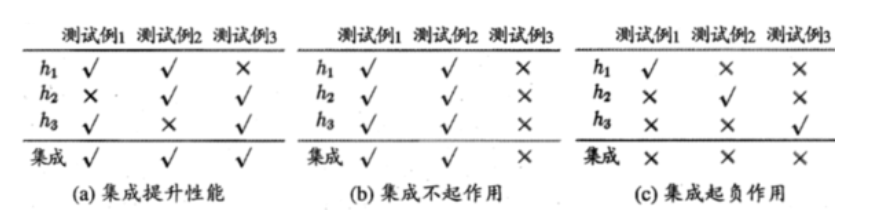
boosting，该方法通常考虑的也是同质弱学习器。它以一种高度自适应的方法顺序地学习这些弱学习器（每个基础模型都依赖于前面的模型），并按照某种确定性的策略将它们组合起来。

stacking，该方法通常考虑的是异质弱学习器，并行地学习它们，并通过训练一个「元模型」将它们组合起来，根据不同弱模型的预测结果输出一个最终的预测结果。

参考资料：<https://www.cnblogs.com/alan-blog-TsingHua/p/10903018.html>

1. **别人提出的问题的理解：**
2. 问题1：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：如图所示：



（a）中在三个基本分类器都能达到66.6的正确率的时候最终投票的结果会达到100%，在（b）中三个分类器的效果相同，最终的集成分类器效果和基本分类器相同，（c）中三个基本分类器都为33.3%的正确率，投完票之后集成分类器的正确率为0，所以最终分类结果并不是趋向于平均功效。

2、问题2：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：可以通过kd-tree算法来对knn的数据结构进行优化，在使用kd树之后将树作为数据结构，大大缩短了knn算法的求距离的时间。

参考资料：<https://blog.csdn.net/weixin_41770169/article/details/81565514>

3、问题3：Bagging为什么会对不稳定学习算法的性能显著提高，而对于稳定的的分类器反而会降低准确率？

自己的理解：稳定的分类器的分类结果相似甚至相同，不稳定的分类器作为基本分类器的时候每个分类器的分类结果不相同的可能性较大，这种情况下如果使用bagging算法可以取分类器中分类结果最多的一个结果作为最后的分类结果，减小了因为分类器的不稳定性造成的影响。稳定的分类器因为不需要使用投票的结果进行影响的消除，而且bagging又减小了分类器的训练数据量，所以可能会降低准确率。

1. 读书计划

1、本周完成的内容章节：3.9-3.10

2、下周计划：2.1-2.4

1. 读书摘要及理解或伪代码的具体实现

**(1)读书摘要：**

**3.10.1 Bagging**

Bagging是使用多个分类器投票的机制来提高分类器的稳定性，减小分类器的方差，减小过拟合。

上面几种效果概念其实差不多，之所以能够提高稳定性，是因为不稳定的分类器每次的结果可能不同，这种情况下使用Bagging可以大大减小偶然性，提高稳定性。之所以能够减小方差，和上面相同，因为分类器的不稳定性可能会出现结果偏离正确分类的情况，在分布图上表现为图像宽度大，即方差大，而Bagging因为能够提高稳定性，所以能够减小方差。对于过拟合，一个分类器在训练完成之后可能会过拟合，多个分类器因为训练效果不同，所以拟合程度不同，所以可以通过投票拟合程度比较低的分类器(也有可能是取出现频率最高的？)来减小过拟合程度。

### **3.10.2 Boosting**

步骤1：所有分布下的基础学习器对于每个观测值都应该有相同的权重

步骤2：如果第一个基础的学习算法预测错误，则该点在下一次的基础学习算法中有更高的权重

步骤3：迭代第2步，直到到达预定的学习器数量或预定的预测精度。

(1)之所以要加大错误数据项的权重，是需要后面的分类器在训练的时候能够尽量迎合这个出错的数据，尽量使得最后得到的分类器能够分类正确所有的数据项。

（2）之所以需要每个分类器的准确率大于50%，可以从上图上看出来，如果分类器的准确率小于50%，那么将分类器进行集成之后不仅不会提高分类的准确率，还会使得准确率更低（错误累加）

1. **代码实现：**

knn算法的实现：

import math

k = 3

datas = [((1,1),1),\

        ((3,1),2),\

        ((0.5,0.5),1),\

        ((1,3.5),4),\

        ((3,3),3),\

        ((3.5,2.8),3),\

        ((1.2,2.9),4)\

        ]

test = [(1,1),(2,2),(3,3),(4,4),(1.2,1.3),(2.4,2.5),(1.5,3),(3.6,3),(2.6,3.7)]

labels = {}

for pair in test:

    dis = []

    for data in datas:

        dis\_tmp =  math.sqrt(math.pow(data[0][0]-pair[0],2)+math.pow(data[0][1]-pair[1],2))

        dis.append((data[1],dis\_tmp))

    sorted\_dis = sorted(dis,key=lambda x:x[1])

    sorted\_dis = sorted\_dis[0:3]

    label = max(sorted\_dis,key = sorted\_dis[0].count)[0]

    labels[pair]=label

print("every point's class is:",labels)

Output:every point's class is: {(1, 1): 1, (2, 2): 4, (3, 3): 3, (4, 4): 3, (1.2, 1.3): 1, (2.4, 2.5): 3, (1.5, 3): 4, (3.6, 3): 3, (2.6, 3.7): 3}