### 提升树模型第1次和第n次的损失函数有什么区别？(n>1)

第一次的损失函数是无法考虑残差的，所以是针对y值的损失函数，第n词（n>1）时，是对残差，即y-fm-1(xi)的损失函数。因为需要用到上一步的分类器，故而在每轮都需要计算残差，然后遍历可能的切分点，找出平方损失函数最小的切分点将输入划分为两个子集，然后依次类推，直到不能继续划分。

### 梯度提升算法改进了什么地方，有什么值得学习的应用吗？

不直接计算残差，而是通过对每个基本分类器求偏导，作为残差近似值，拟合回归树。

有一个方法GBDT，是现在各个比赛很常用的分类模型，值得学习，同时包括随机森林算法。

### 梯度提升法对比提升树，为什么更work一点？

根据算法8.4底部的叙述，对于平方损失函数，负梯度就是残差；而对于一般损失函数，就可以视为残差的估计值。那么我们就可以认为我们降低残差也就是降低方差。这其实也是提升方法神奇的地方，既能降低方差，避免欠拟合；又可以减低方差，避免过拟合。

### 梯度提升算法为什么能比提升树更容易优化一般损失函数？

提升树模型每一次的提升都是靠上次的预测结果与训练数据的label值差值作为新的训练数据进行重新训练，梯度提升算法则是将残差计算替换成了损失函数的梯度方向，将上一次的预测结果带入梯度中求出本轮的训练数据，这两种模型就是在生成新的训练数据时采用了不同的方法，提升树的残差求解比较简单，但是在使用一般的损失误差函数时，残差求解起来不是那么容易，所以就是用损失函数的负梯度在当前模型的值作为回归问题中残差的近似值。

### 梯度提升算法为什么在调参时树的深度很少就能达到很高的精度

讨论后的理解：Boosting主要关注降低偏差，因此Boosting能基于泛化性能相当弱的学习器构建出很强的集成；Bagging主要关注降低方差，因此它在不剪枝的决策树、神经网络等学习器上效用更为明显。（来自周志华《机器学习》）

### 不同损失函数分别采用的最小化办法有什么不同？

当损失函数为指数函数时，比如AdaBoost算法的损失函数是指数函数，这时通过前向分步算法来解决。前向分布算法在每轮迭代时，通过将上一轮所生成的模型所产生的损失函数最小化的方法来计算当前模型的参数。

在计算平方损失误差时，可能出现残差项（y-fm-1(x)），此时可以通过如下方法来进行优化：每一轮迭代是为了减小上一步模型的残差，为了减少残差，每次在残差的负梯度方向建立一个新的模型，这样一步一步的使得残差越来越小。

对于一般损失函数，可以通过梯度下降的方法来使得损失函数逐步减小，每次向损失函数的负梯度方向移动，直到损失函数值到达谷底。

### 算法8.4中，（2.b）中的J如何理解？（2.c）中寻找区域值怎么理解？

自己的理解：J为回归树的叶结点划分的区域数目，（2，c）是要对划分的每个区域内损失函数求和取最小，从而使（2，d）中总的损失函数最小。

### 算法8.4中，（2.b）中的J如何理解？（2.c）中寻找区域值怎么理解？

对梯度进行拟合，得到新的回归树，回归树中的叶节点区域划分的数目就是J。

对区域进行划分以后，需要为区域赋值，此时选择合适的值，使损失函数最小。

### 为什么会有Gradient boosting?

对于一般损失函数，直接优化不容易，就需要通过梯度提升计算。

负梯度是函数下降最快的方向，所以用梯度去拟合函数。只有在MSE的情况下，梯度才近似于残差的拟合。本质还是通过梯度寻求最优。

### GDBT怎么处理多分类任务？

采取一对多的方法。1 vs n-1。

对每个类别训练M个分类器。假设有K个类别，那么训练完之后总共有M\*K颗树。