**随机森林**

//参考http://backnode.github.io/pages/2015/04/23/random-forest.html 深度学习7.11

//参考https://zhuanlan.zhihu.com/p/22097796

Bagging是通过结合几个模型降低泛化误差的技术。

bagging + CART decision trees=随机森林。解决了决策树容易过拟合的劣势。

g[t]的out-of-bag**(OOB)样本**：随机抽取数据集时没有被抽到的数据。用OOB error代替验证集错误，在实践中效果非常好，节省了验证集数据开销。

**随机森林的构造过程：**

　　1. 假如有N个样本，则有放回的随机选择N个样本(每次随机选择一个样本，然后返回继续选择)。这选择好了的N个样本用来训练一个决策树，作为决策树根节点处的样本。

　　2. 当每个样本有M个属性时，在决策树的每个节点需要分裂时，随机从这M个属性中选取出m个属性，满足条件m << M。然后从这m个属性中采用某种策略（比如说信息增益）来选择1个属性作为该节点的分裂属性。

　　3. 决策树形成过程中每个节点都要按照步骤2来分裂（很容易理解，如果下一次该节点选出来的那一个属性是刚刚其父节点分裂时用过的属性，则该节点已经达到了叶子节点，无须继续分裂了）。一直到不能够再分裂为止。注意整个决策树形成过程中没有进行剪枝。

　　4. 按照步骤1~3建立大量的决策树，这样就构成了随机森林了。

**随机森林的优点：**

a. 在数据集上表现良好，两个随机性的引入，使得随机森林不容易陷入过拟合

b. 在当前的很多数据集上，相对其他算法有着很大的优势，两个随机性的引入，使得随机森林具有很好的抗噪声能力

c. 它能够处理很高维度（feature很多）的数据，并且不用做特征选择，对数据集的适应能力强：既能处理离散型数据，也能处理连续型数据，数据集无需规范化

d. 可生成一个Proximities=（pij）矩阵，用于度量样本之间的相似性： pij=aij/N, aij表示样本i和j出现在随机森林中同一个叶子结点的次数，N随机森林中树的颗数

e. 在创建随机森林的时候，对generlization error使用的是无偏估计

f. 训练速度快，可以得到变量重要性排序（两种：基于OOB误分率的增加量和基于分裂时的GINI下降量

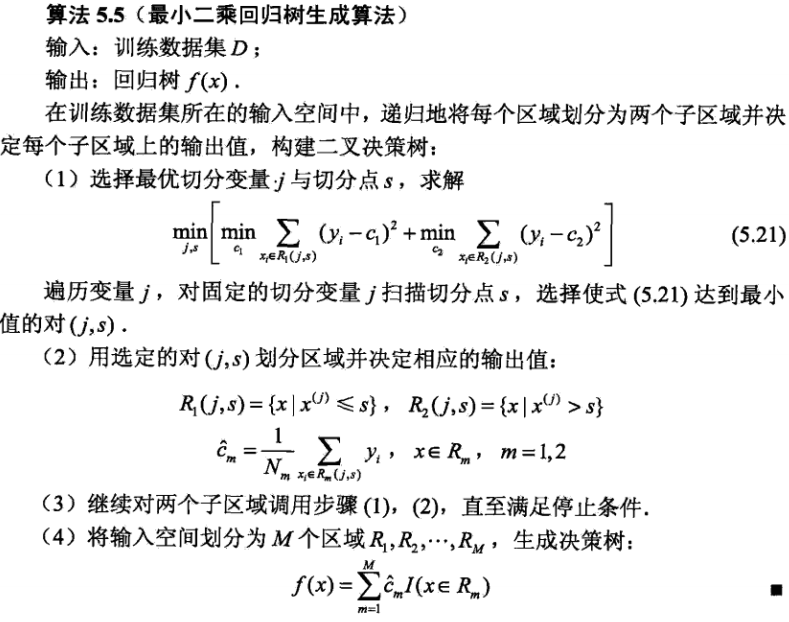
g. 在训练过程中，能够检测到feature间的互相影响

h. 容易做成并行化方法

**GBDT**

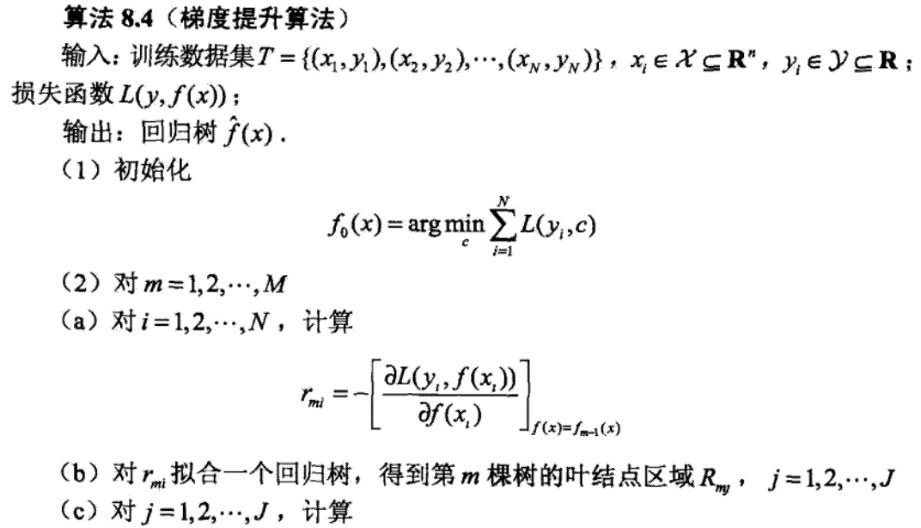
//李航P69 P151

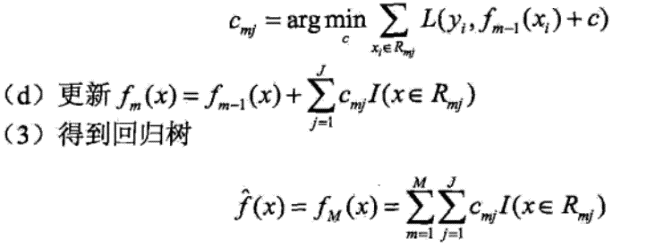
决策树+boosting=GBDT



在GBDT的迭代中，假设我们前一轮迭代得到的强学习器是, 损失函数是, 我们本轮迭代的目标是找到一个CART回归树模型的弱学习器 ，让本轮的损失

最小。





**Gradient boosting算法（GBM）和随机森林都是基于树的算法，它们有什么区别？**

//参考http://www.jianshu.com/p/18a400ad47cc

相同点：

(1)都是由多棵树组成的，都是集成学习算法

(2)最终的结果都是由多颗树一起决定

不同点：

(1)组成随机森林的树可以是分类树，也可以是回归树，但是GBDT只能由回归树组成。

(2)组成随机森林的树可以并行生成，但是组成GBDT的树只能串行生成。

(3)对于最终的输出结果，随机森林采用多数投票；而GBDT是将所有的结果累加起来，或者加权起来

(4)随机森林对异常值不敏感，而GBDT对异常值非常敏感

(5)随机森林通过减小方差来提高性能，GBDT通过减小偏差来提高性能

**gbdt在调参时为什么树的深度很少就能达到很高的精度？**

对于Bagging算法来说，由于我们会并行地训练很多不同的分类器的目的就是降低这个方差(variance) ,因为采用了相互独立的基分类器多了以后，h的值自然就会靠近.所以对于每个基分类器来说，目标就是如何降低这个偏差（bias),所以我们会采用深度很深甚至不剪枝的决策树。

  对于Boosting来说，每一步我们都会在上一轮的基础上更加拟合原数据，所以可以保证偏差（bias）,所以对于每个基分类器来说，问题就在于如何选择variance更小的分类器，即更简单的分类器，所以我们选择了深度很浅的决策树。