**Introduction**

决策树这种[**算法**](http://lib.csdn.net/base/datastructure)有着很多良好的特性，比如说训练时间复杂度较低，预测的过程比较快速，模型容易展示（容易将得到的决策树做成图片展示出来）等。但是同时，单决策树又有一些不好的地方，比如说容易over-fitting，虽然有一些方法，如剪枝可以减少这种情况，但是还是不太理想。

模型组合（比如说有Boosting，Bagging等）与决策树相关的算法比较多，如randomForest、Adaboost、GBRT等，这些算法最终的结果是生成N(可能会有几百棵以上）棵树，这样可以大大的减少单决策树带来的毛病，有点类似于三个臭皮匠赛过一个诸葛亮的做法，虽然这几百棵决策树中的每一棵都很简单（相对于C4.5这种单决策树而言），但是他们组合起来确是很强大。虽然这些算法都是通过决策树演变过来的，但在处理的过程上有着一些差异，我会在后面对此做一个本质上的比较。下面先来介绍下本文的梯度提升算法。

**Gradient Tree Boosting**

梯度树提升（Gradient Tree Boosting）是一种组合算法，也叫做梯度提升回归树（gradient boosting regression tree），它的基分类器是决策树，既可以用来回归，也可以用作分类。在分类性能上，能够和随机森林媲美，甚至在有的数据集上表现的有过之而无不及。如今，Gradient Tree Boosting模型已经广泛的运用在Web搜索排行榜以及生态学上。在阿里内部也用的比较多，所以值得我们去花点时间认真学习。

根据scikit-learn官网的介绍，GBRT的优势有：

* 自然而然地处理混合类型的数据
* 预测能力强
* 在输出空间对于异常值的鲁棒性强（通过强大的损失函数）

然而，GBRT也有劣势：

* 可扩展性方面，由于提升的时序性，不能进行并行处理

尽管如此，由于GTB的表现性能很好，所以它仍然受广大业界人士的青睐。下面来介绍下梯度提升树的算法原理。

**GTB算法**

梯度提升（gradient boosting）算法最初是FreidMan在2000年提出来的，其核心就在于，每棵树是从先前所有树的残差中来学习。利用的是当前模型中损失函数的负梯度值

rmi=−[∂L(yi,f(xi))∂f(xi)]f(x)=fm−1(x)

作为提升树算法中的残差的近似值，进而拟合一棵回归（分类）树。

梯度提升属于Boost算法的一种，也可以说是Boost算法的一种改进，原始的Boost算法是在算法开始时，为每一个样本赋上一个相等的权重值，也就是说，最开始的时候，大家都是一样重要的。在每一次训练中得到的模型，会使得数据点的估计有所差异，所以在每一步结束后，我们需要对权重值进行处理，而处理的方式就是通过增加错分类点的权重，同时减少正确分类点的权重，这样使得某些点如果老是被分错，那么就会被“严重关注”，也就被赋上一个很高的权重。然后等进行了N次迭代（由用户指定），将会得到N个简单的基分类器（basic learner），最后将它们组合起来，可以对它们进行加权（错误率越大的基分类器权重值越小，错误率越小的基分类器权重值越大）、或者让它们进行投票等得到一个最终的模型。

Gradient Boost与传统的Boost有着很大的区别，它的每一次计算都是为了减少上一次的残差(residual)，而为了减少这些残差，可以在残差减少的梯度(Gradient)方向上建立一个新模型。所以说，在Gradient Boost中，每个新模型的建立是为了使得先前模型残差往梯度方向减少，与传统的Boost算法对正确、错误的样本进行加权有着极大的区别。

**梯度提升算法（以回归为例）**

对于给定的输入：训练数据集T=(x1,y1),(x2,y2),...,(xn,yn),损失函数L(y,f(x));   
输出结果：一棵回归树f~(x)

（1）首先初始化

f0(x)=argminc∑i=1NL(yi,c)

估计一个使损失函数极小化的常数值，此时它只有一个节点的树；

（2）迭代的建立M棵提升树   
for m=1 to M:（第一层循环）   
for i=1 to N：（第二层循环） 计算损失函数的负梯度在当前模型的值，并将它作为残差的估计值。

rmi=−[∂L(yi,f(xi))∂f(xi)]f(x)=fm−1(x)

对于rmi拟合一棵回归树，得到第m棵树的叶节点区域Rmj,j=1,2,…,J

for j=1 to J：（第二层循环）,计算：

cmj=argminc∑xiϵRmjL(yi,fm−1(xi)+c)

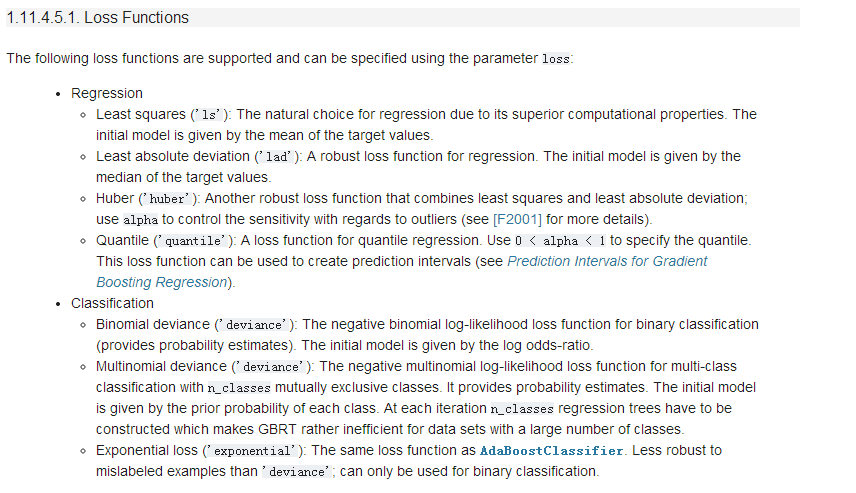
利用线性搜索估计叶节点区域的值，使损失函数极小化；

然后，更新fm(x)=fm−1(x)+∑Jj=1cmjI(xϵRmj)

（3）最后得到的fm(x)就是我们最终的模型

f~(x)=fM(x)=∑m=1M∑j=1JcmjI(xϵRmj)

**使用scikit-learn中的GTB**

在scikit-learn中对GTB算法有了很好的封装，对于分类可以选择的损失函数有逻辑回归和指数函数，对于回归的损失函数相对比较多，有最小二乘法、最小绝对偏差函数、huber以及分位数等。具体描述参考下面的图片：   


下面是sklearn中的一个分类原例：

>>> from sklearn.datasets import make\_hastie\_10\_2

>>> from sklearn.ensemble import GradientBoostingClassifier

>>> X, y = make\_hastie\_10\_2(random\_state=0)

>>> X\_train, X\_test = X[:2000], X[2000:]

>>> y\_train, y\_test = y[:2000], y[2000:]

>>> clf = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0,

... max\_depth=1, random\_state=0).fit(X\_train, y\_train)

>>> clf.score(X\_test, y\_test)

0.913...

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

其中n\_estimators表示弱分类器的个数，learning\_rate表示学习率，max\_depth表示最大的深度等。GTB的参数比较多，在实际应用中需要自己去调整合适的参数。

**基于决策树的组合算法比较**

基于决策树的组合算法常用的有三个，分别是Adaboost、RandomFrest以及本文的GBRT。

Adaboost是通过迭代的学习每一个基分类器，每次迭代中，把上一次错分类的数据权值增大，正确分类的数据权值减小，然后将基分类器的线性组合作为一个强分类器，同时给分类误差率较小的基本分类器以大的权值，给分类误差率较大的基分类器以小的权重值。Adaboost使用的是自适应的方法，其中概率分布式变化的，关注的是难分类的样本。详细内容请参考我之前的文章：[机器学习算法-Adaboost](http://blog.csdn.net/dream_angel_z/article/details/46764845)。

随机森林RandomForest算法，与adaboost有错区别，可以说一种改进的装袋组合算法。随机森林则(randomForest)，不仅对样本进行抽样，还对变量进行抽样。它通过随机的方式建立一个森林，森林里面有许多棵决策树，并且每一棵树之间是没有联系的。在得到森林之后，当有一个新的输入样本进来的时候，就让森林中的每一棵决策树分别对其进行判断，看这个样本应该属于哪一类（就分类算法而言），然后看看哪一类选择最多，就预测这个样本为该类。在建立每一棵决策树的过程中，有两点需要注意，即**采样**与**完全分裂**。首先是两个随机采样的过程，RF对输入的数据要进行行采样和列采样。对于行采样，是采用有放回的方式，也就是在采样得到的样本集合中，可能有重复的样本。假设输入样本为N个，那么采样的样本也为N个。这样使得在训练的时候，每一棵树的输入样本都不是全部的样本，使得相对不容易出现over-fitting过拟合。然后进行列采样，从M个feature特征中，选择m个(m << M)。之后就是对采样之后的数据使用**完全分裂**的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个类别。一般很多的决策树算法都一个重要的步骤-**剪枝**，但是这里不这样干，由于之前的两个随机采样的过程保证了随机性，所以就算不剪枝，也不会出现over-fitting。按照这种算法得到的随机森林中的每一棵决策树都是非常弱的，但当它们组合在一起的时候，就相当厉害了。随机森林就好比是：每一棵决策树就是一个精通于某一领域的专家（因为我们从M个feature中选择m个让每一棵决策树进行学习），这样在随机森林中就有了很多个精通不同领域的专家，对一个新的问题（新的输入数据），可以用不同的角度去看待它，最终由各个专家，投票得到结果。随机森林的分类准确率可以与adaboost媲美。它对噪声数据更加鲁棒，运行速度比adaboost也快得多。

对于梯度提升树，它的每一次计算都是为了减少上一次的残差(residual)，而为了减少这些残差，可以在残差减少的梯度(Gradient)方向上建立一个新模型。这与adaboost和随机森林有很大的区别。

**References**

[1] Introduction to Data Mining 数据挖掘概论. Pang-Ning Tan Michael Steinbach Vipin Kumar著   
[2] 统计学习方法 李航 著   
[3] scikit-learn官网组合算法 [点击这里](http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting)

**参考文章**： [随机森林与GBDT](http://www.cnblogs.com/LeftNotEasy/archive/2011/03/07/1976562.html)