# 推进树(Boosted Trees)介绍

XGBoost是Extreme Gradient Boosting(极速梯度推进)的简称，术语“梯度推进”是由Friedman在论文《Greedy Function Approximation: A Gradient Boosting Machine》中提出的，XGBoost是基于此原型的。本教程主要介绍梯度推进树，大部分内容都基于XGBoost作者的这个[幻灯片](http://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf)。

GBM (boosted trees)出现有一段时间了，已经有很多讲解它的资料了。本教程将尝试以自包含和原理性的方式使用监督学习中的元素来解释boost tree。我们认为这样会更清楚、正式些，并且更容易触及xgboost一些变体的根源（ motivates the variant used in xgboost）。

## 监督学习的元素

XGBoost主要用于监督学习问题，我们使用训练数据（多个特征）来预测目标变量 。在开始介绍树之前，我们先回顾一下监督学习的基本元素。

**模型和参数**

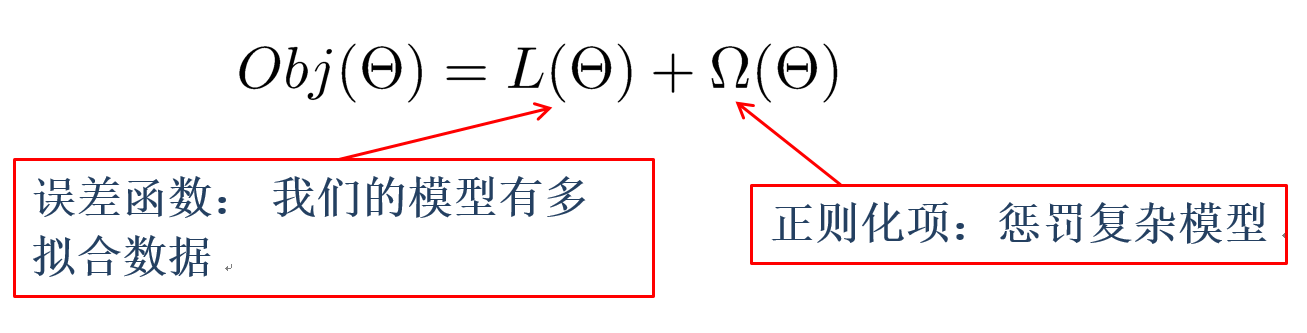
监督学习中的**模型**通常指从指定预测的一种数学结构(mathematical structure)。比如，最普通的就是*线性模型*了，预测公式为，就是把输入的特征按权重线性累加。根据具体任务，预测结果可以有很多种不同解释。比如回归或分类，可能是逻辑回归中的逻辑变换来预测概率或正类别(positive class)，也可能是打分排序如果我们用来对输出进行排序的话。

**参数**是我们需要从数据中学习的未知部分。在线性回归中，参数是系数，通常我们也用来表示系数（模型中有很多参数，我们这里定义的比较宽松）。

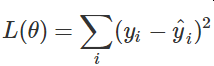
**目标函数：训练损失 + 正则项**

根据对的不同理解，我们可能需要解决各种各样问题，比如回归、分类、排序等。我们需要找到方法来寻找给定训练数据的最佳参数，为了做到这一点，我们需要定义**目标函数**，来衡量指定参数下模型的表现。

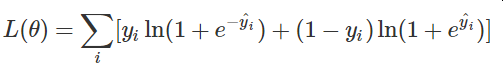
关于目标函数有一点需要特别注意，那就是它**必须总是**包含两部分：训练损失和正则项。



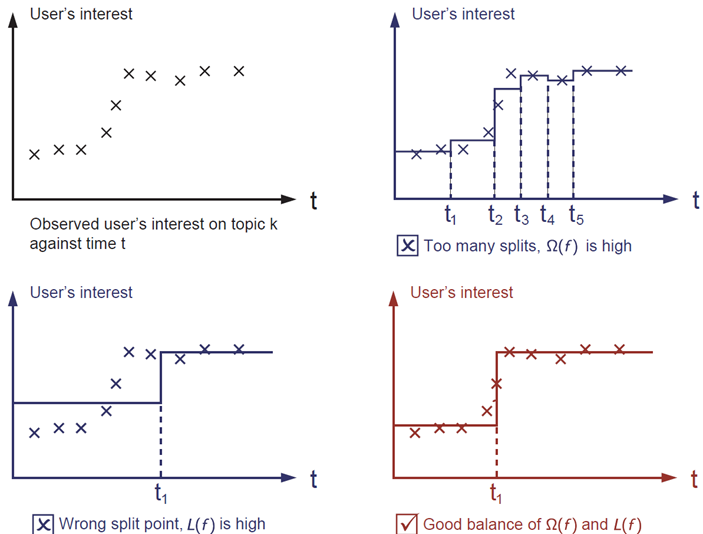
其中是训练的损失函数，是正则项。损失函数代表我们的模型对训练数据的预测，比如，最常见的损失函数是均方误差：



另一个常用的损失函数是逻辑回归中的逻辑损失函数：



**正则项**经常会被人们忘记添加。正则项是用来控制模型的复杂度的，防止过拟合。这可能有点抽象，所以我们来看看下图中的问题。左上图中给了一堆数据点，需要你找出一个步进函数来拟合，这三个方案中你觉得哪个最合适？



正确的答案是红色那个，想想这个是不是最合理的。我们的基本原则是找到**简单**并且**足够正确(predictive)**的模型，这个折中就是机器学习中的bias-variance妥协。

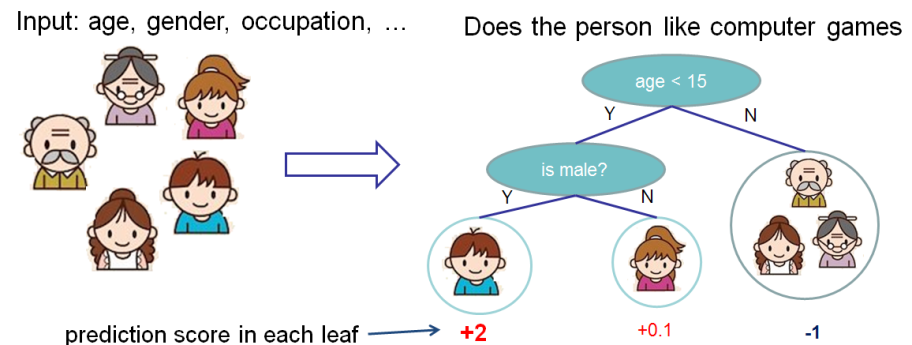
**为什么介绍这些基本原理？**

上面介绍的这些元素都是监督学习中的基本元素，他们是构建机器学习工具箱的砖石。比如，你要能描述boosted trees和随机森林的共同点和不同点，用一种正式的方式来理解这个过程也会帮助我们理解我们的学习目标以及这些启发(heuristics)背后的原理，比如修剪(pruning)和平滑(smoothing)处理。

## Tree Ensemble(树体?)

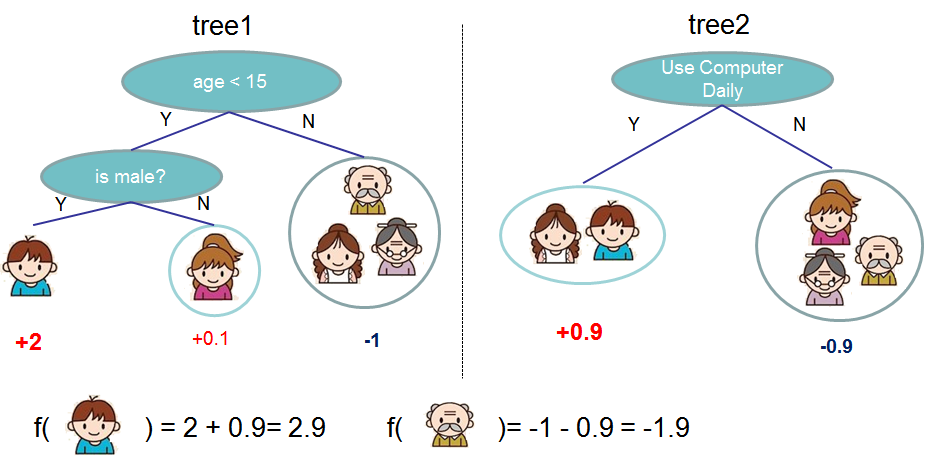
介绍完监督学习的元素后，我们开始学习树。我们以xgboost中的**模型**--树体( tree ensembles)开始。树体是一堆分类和回归树 (classification and regression trees,CART)的集合。

下面是关于CART 的一个简单例子，对用户进行分类看他们是不是喜欢玩电脑游戏。

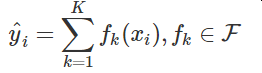


我们将一个家庭中的成员分到不同的叶节点上，然后打上相应的分数。在CART中，每个叶节点都关联了一个实数分数，给了除分类之外的更多信息。这也让统一优化( unified optimization)变得更容易，我们在本教程的后面部分可以看到。

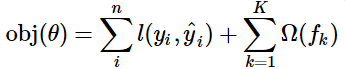
通常单个树没法应用到实际中，实际中我们使用的是被称为树体的模型，它将多个树的预测结果相加。



这是由两棵树构成的树体的例子，对每颗树的预测分数相加就得到了最终的预测分数。如果你仔细看这个例子就会发现，这两颗树都尝试去*补充*（complement）对方。从数学上我们可以将写成如下形式：



其中，是树的个数，是函数空间(functional space)中的一个函数，是所有可能的CARTs。于是我们的优化目标可以写成如下形式：

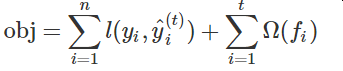


那么问题来了，随机森林的*模型*是啥呢？就是树体！所以随机森林和boosted trees在模型方面并没有不同，区别只是如何训练他们。这意味着如果你写一个树体的预测服务，你只需要写其中的一个就行了，他们应该都能直接应用于boosted trees和随机森林。

## 树推进（Tree Boosting）

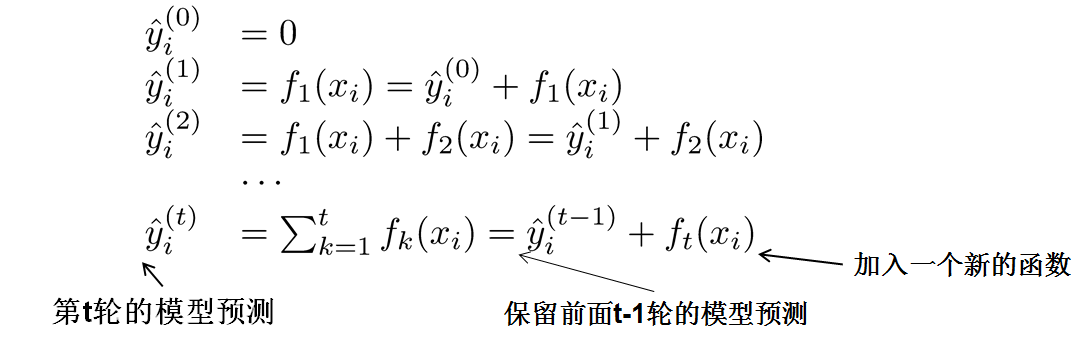
介绍完模型后，我们开始将真正的训练部分。我们该怎么训练树呢？针对所有监督学习模型，答案永远都是：定义目标函数，然后优化它*！*

假设我们有如下目标函数（记住它永远包含训练损失和正则项）：

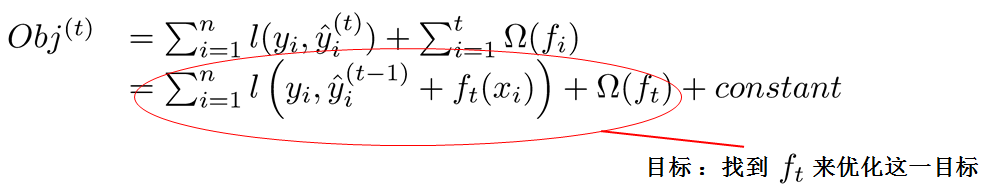


### 渐进训练（Additive Training）

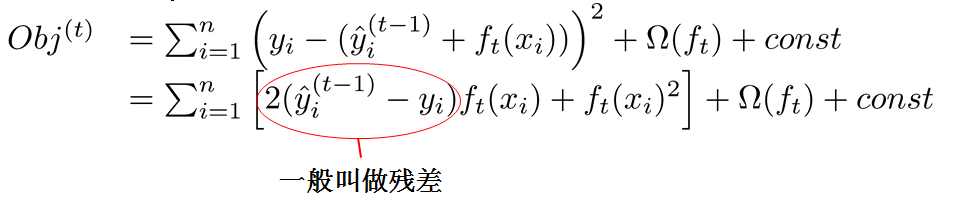
我们要问的第一问题是：树的***参数***是啥？你会发现我们需要学习一堆函数，每个都包含树结构和叶节点分数。这比传统的优化问题要难很多，你可以使用梯度(gradient)来推进。一次训练所有的树不太好做，我们可以使用渐进策略：将我们已经学习到的固定下来，然后每次增加一颗新的树。我们将第步的预测结果记为，于是得到：



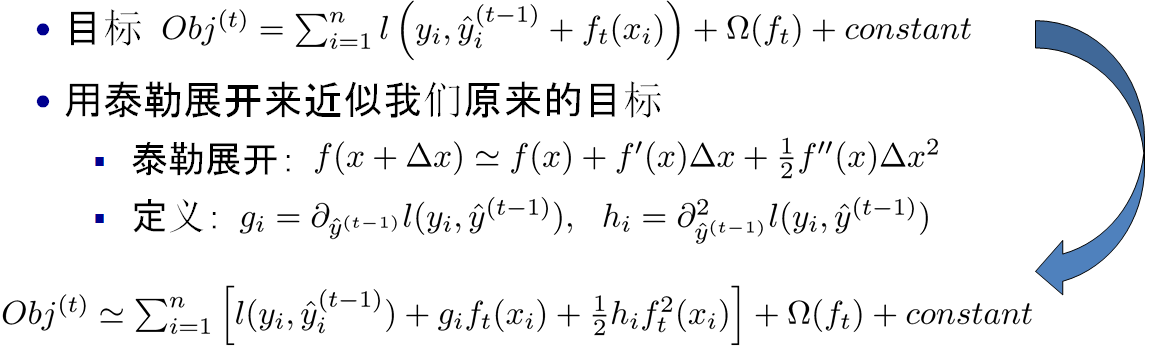
还有个问题，我们如何选择每一轮加入什么呢？答案很直接，给目标函数增加一个优化：



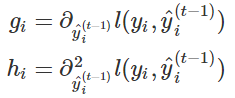
如果我们使用MSE作为损失函数，变成如下形式：



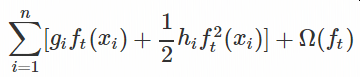
MSE的形式比较好懂，前面一个排序项(order term,通常被叫做剩余项residual)，然后是二次项。对于其他的损失函数（比如逻辑损失），不太好定义一个这么完美的形式。所以通常情况下，我们使用损失函数的二阶泰勒展开：



其中和定义如下：



移除所有常量后，第步的目标函数变为：



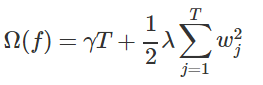
这就是我们对新的树的优化目标。这种定义的一个重要优势是它只依赖于和，xgboost就是这么实现对自定义损失函数的支持的。我们能优化每个损失函数，包括逻辑回归和加权逻辑回归( weighted logistic regression)，就使用和作为输入！

### 模型复杂度

介绍完训练步骤后，还有一件重要的事，**正则项**！我们需要定义树的复杂度。为了这一点，我们将树重新定义为：



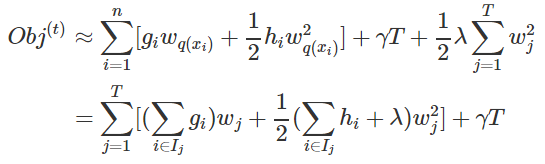
其中是叶节点的分数向量，是将每个数据映射到相应叶节点的函数，是叶节点个数。在XGBoost中我们将复杂度定义为：



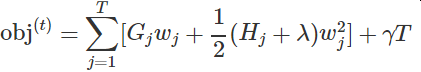
当然有很多种方法来定义复杂度，这种定义在实践中效果不错。大多数框架都不太重视正则项或者直接忽略，这是因为传统的树训练只强调改进训练效果，而复杂度控制则留给了启发式(heuristics)。通过正式定义，我们能更清楚的知道我们正在学习啥，并且它在实践中确实工作的不错。

### 评分结构

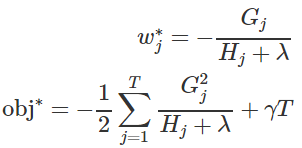
下面是求导的魔术部分了。对树模型重新整理后，我们将第颗树的目标函数记为：



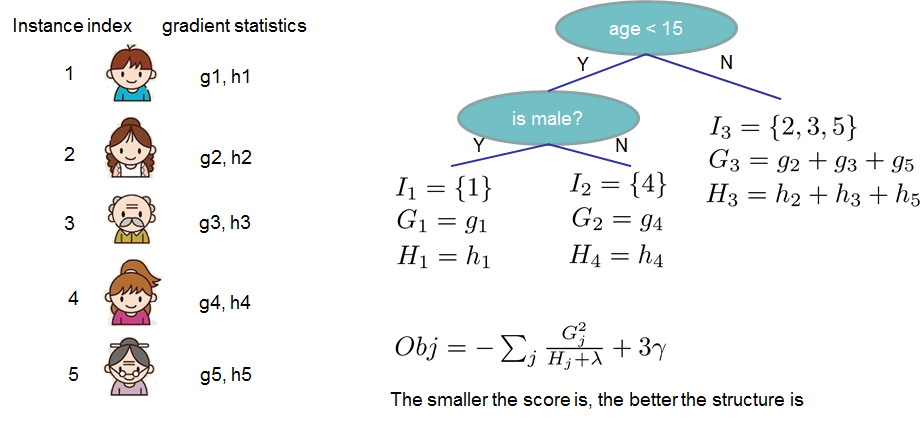
其中是分配给第个叶节点的数据点的索引集合，注意在第二行中我们改变了和的索引，因为用一个叶节点上的所有数据点的分数都一样。我们可以通过定义和来进一步压缩表达式：



在这个方程中是彼此独立的，是二次项，并且可以得到对给定结构的最佳和最佳目标reduction（损失?） 是：



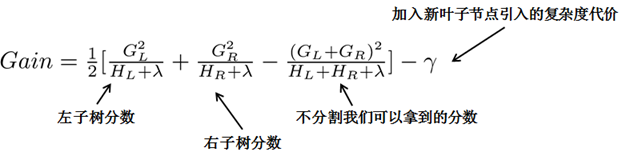
最后一个求导用来衡量我们的树结构到底**有多好**。



如果这些看起来有点复杂，那我们看看上图中分数是怎么计算出来的。基本上，对一个给定的树结构，我们将和分配到属于他们的叶节点上，然后把他们的分数(statistics)相加，最后使用上面的公式计算这棵树有多好。分数就像决策树中的杂质度（impurity measure ），除了它也考虑到模型复杂度。

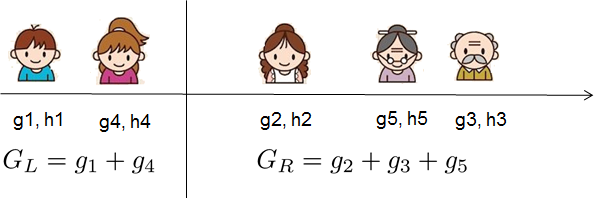
### 学习树结构

现在我们知道怎么衡量一棵树的好坏，理论上我们枚举所有可能的树然后选择最好的一个。实际上这比较麻烦，所以我们将尝试每次优化一个level的树。我们尝试将一个叶节点分裂成两个，获得的分数是：



这个公式可以被分解成1）新的左叶节点的分数 2)新的右叶节点分数 3）原叶节点分数 4）叶节点上的正则项。我们可以看到：如果获得的分数小于，我们最好不要添加树的分支。这就是基于树的模型中的修剪技术！通过使用监督学习中的原理，我们很自然的弄清楚了为什么这些技术有用。

对一个实数数据，我们通常会找到一个最优分裂。为了提高效率，我们将所有的实例排序，如下图：



从左到右扫描就可以计算所有可能的分裂的分数，然后就可以很快的找到最佳分裂。

### 写在最后

现在你搞清楚boosted trees 是啥了，可能会问XGBoost的[介绍](https://github.com/dmlc/xgboost)呢？XGBoost就是本教程正式介绍的这个工具，重要的一点是它是对***系统优化***和***机器学习原理***进行过深思熟虑后开发的。XGBoost的目标是通过提供***可扩展***、***可移植***和***足够精确***的库来推动机器计算的极限，一定要[试试](https://github.com/dmlc/xgboost)哦！最重要的是记得将自己的智慧(代码、示例、教程)回馈给社区吧。