# XGBoost 与 Boosted Tree – 我爱计算机

**作者：陈天奇，毕业于上海交通大学ACM班，现就读于华盛顿大学，从事大规模机器学习研究。**

**注解：truth4sex**

***编者按：本文是对开源xgboost库理论层面的介绍，在陈天奇原文《梯度提升法和Boosted Tree》的基础上，做了如下注解：1）章节划分；2）注解和参考链接（以蓝色和红色字体标注）。备注：图片可点击查看清晰版。***

## 1. 前言

应 @龙星镖局 兄邀请写这篇文章。作为一个非常有效的机器学习方法，Boosted Tree是数据挖掘和机器学习中最常用的算法之一。因为它效果好，对于输入要求不敏感，往往是从统计学家到数据科学家必备的工具之一，它同时也是kaggle比赛冠军选手最常用的工具。最后，因为它的效果好，计算复杂度不高，也在工业界中有大量的应用。

## 2. Boosted Tree的若干同义词

说到这里可能有人会问，为什么我没有听过这个名字。这是因为Boosted Tree有各种马甲，比如GBDT, GBRT (gradient boosted regression tree)，MART[1]，LambdaMART也是一种boosted tree的变种。网上有很多介绍Boosted tree的资料，不过大部分都是基于Friedman的最早一篇文章Greedy Function Approximation: A Gradient Boosting Machine的翻译。个人觉得这不是最好最一般地介绍boosted tree的方式。而网上除了这个角度之外的介绍并不多。这篇文章是我个人对于boosted tree和gradient boosting 类算法的总结，其中很多材料来自于我TA UW机器学习时的一份讲义[2]。

## 3. 有监督学习算法的逻辑组成

要讲boosted tree，要先从有监督学习讲起。在有监督学习里面有几个逻辑上的重要组成部件[3]，初略地分可以分为：模型，参数 和 目标函数。

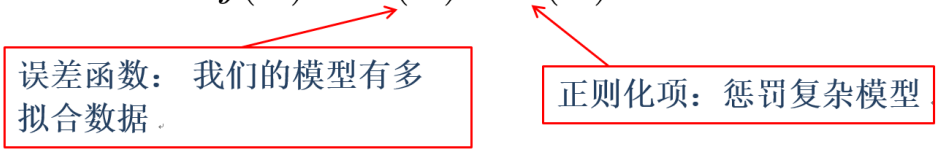
### *i. 模型和参数*

模型指给定输入$x_i$如何去预测 输出 $y_i$。我们比较常见的模型如线性模型（包括线性回归和logistic regression）采用了线性叠加的方式进行预测$\hat{y}_i=\sum_j w_j x_{ij}$ 。其实这里的预测$y$可以有不同的解释，比如我们可以用它来作为回归目标的输出，或者进行sigmoid 变换得到概率，或者作为排序的指标等。而一个线性模型根据$y$的解释不同（以及设计对应的目标函数）用到回归，分类或排序等场景。参数指我们需要学习的东西，在线性模型中，参数指我们的线性系数$w$。

### *ii. 目标函数：损失 + 正则*

模型和参数本身指定了给定输入我们如何做预测，但是没有告诉我们如何去寻找一个比较好的参数，这个时候就需要目标函数登场了。一般的目标函数包含下面两项

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$




常见的误差函数有$L=\sum_i l(y_i,\hat{y}_i)$ 比如平方误差 $l(y_i,\hat{y}_i)=(y_i-\hat{y}_i)^2$ ，logistic误差函数（$l(y_i,\hat{y}_i)=y_i\ln(1+e^{-\hat{y}_i})+(1-y_i)\ln(1+e^{\hat{y}_i})$ ）等。而对于线性模型常见的正则化项有$L2$正则和$L1$正则。这样目标函数的设计来自于统计学习里面的一个重要概念叫做Bias-variance tradeoff[4]。比较感性的理解，Bias可以理解为假设我们有无限多数据的时候，可以训练出最好的模型所拿到的误差。而Variance是因为我们只有有限数据，其中随机性带来的误差。目标中误差函数鼓励我们的模型尽量去拟合训练数据，这样相对来说最后的模型会有比较少的 bias。而正则化项则鼓励更加简单的模型。因为当模型简单之后，有限数据拟合出来结果的随机性比较小，不容易过拟合，使得最后模型的预测更加稳定。

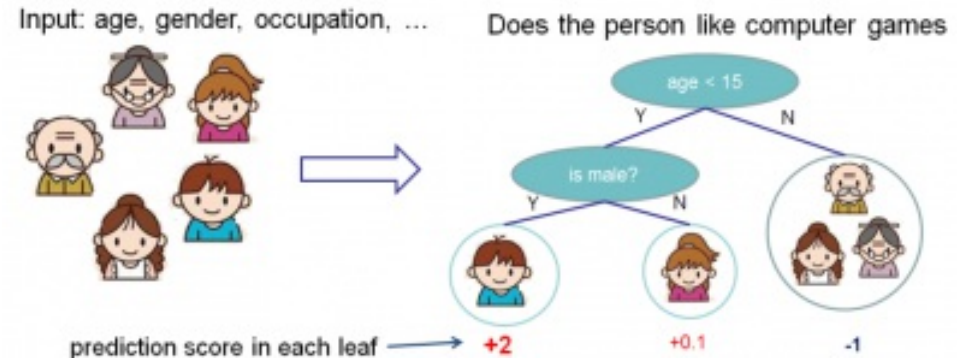
*iii. 优化算法*

讲了这么多有监督学习的基本概念，为什么要讲这些呢？ 是因为这几部分包含了机器学习的主要成分，也是机器学习工具设计中划分模块比较有效的办法。其实这几部分之外，还有一个优化算法，就是给定目标函数之后怎么学的问题。之所以我没有讲优化算法，是因为这是大家往往比较熟悉的“机器学习的部分”。而有时候我们往往只知道“优化算法”，而没有仔细考虑目标函数的设计的问题，比较常见的例子如决策树的学习，大家知道的算法是每一步去优化gini entropy，然后剪枝，但是没有考虑到后面的目标是什么。

**4. Boosted Tree**

*i. 基学习器：分类和回归树（CART）*

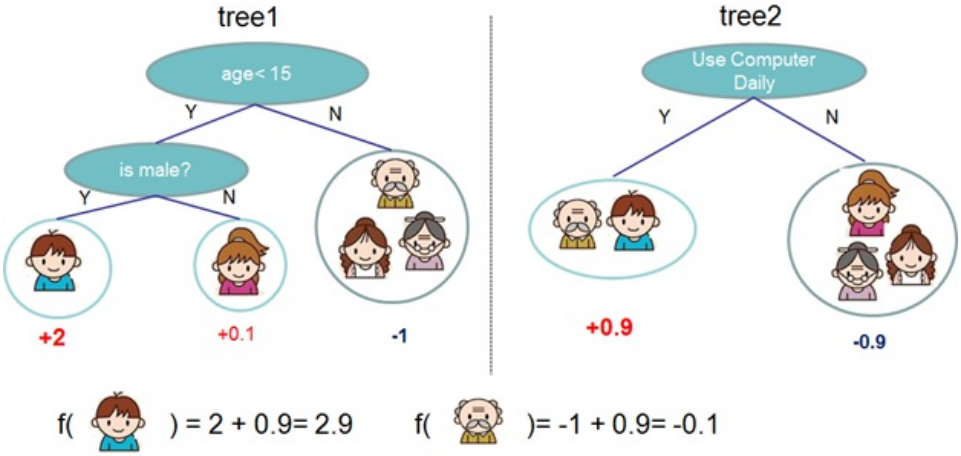
话题回到boosted tree，我们也是从这几个方面开始讲，首先讲模型。Boosted tree 最基本的组成部分叫做回归树(regression tree)，也叫做CART5。




上面就是一个CART的例子。CART会把输入根据输入的属性分配到各个叶子节点，而每个叶子节点上面都会对应一个实数分数。上面的例子是一个预测一个人是否会喜欢电脑游戏的 CART，你可以把叶子的分数理解为有多可能这个人喜欢电脑游戏。有人可能会问它和decision tree的关系，其实我们可以简单地把它理解为decision tree的一个扩展。从简单的类标到分数之后，我们可以做很多事情，如概率预测，排序。

*ii. Tree Ensemble*

一个CART往往过于简单无法有效地预测，因此一个更加强力的模型叫做tree ensemble。




在上面的例子中，我们用两棵树来进行预测。我们对于每个样本的预测结果就是每棵树预测分数的和。到这里，我们的模型就介绍完毕了。现在问题来了，我们常见的随机森林和boosted tree和tree ensemble有什么关系呢？如果你仔细的思考，你会发现RF和boosted tree的模型都是tree ensemble，只是构造（学习）模型参数的方法不同。第二个问题：在这个模型中的“参数”是什么。在tree ensemble中，参数对应了树的结构，以及每个叶子节点上面的预测分数。

最后一个问题当然是如何学习这些参数。在这一部分，答案可能千奇百怪，但是最标准的答案始终是一个：定义合理的目标函数，然后去尝试优化这个目标函数。在这里我要多说一句，因为决策树学习往往充满了heuristic。 如先优化吉尼系数，然后再剪枝啦，限制最大深度，等等。其实这些heuristic的背后往往隐含了一个目标函数，而理解目标函数本身也有利于我们设计学习算法，这个会在后面具体展开。

对于tree ensemble，我们可以比较严格的把我们的模型写成是：

$$\hat{y}_i=\sum_{k=1}^{K} f_k(x_i), f_k \in \mathrm{F}$$

其中每个$f$是一个在函数空间6(F)里面的函数，而F对应了所有regression tree的集合。我们设计的目标函数也需要遵循前面的主要原则，包含两部分

$$Obj(\Theta)=\sum_i l(y_i,\hat{y}_i)+\sum_{k=1}^{K}\Omega(f_k)$$

*iii. 模型学习：additive training*

其中第一部分是训练误差，也就是大家相对比较熟悉的如平方误差，logistic loss等。而第二部分是每棵树的复杂度的和。这个在后面会继续讲到。因为现在我们的参数可以认为是在一个函数空间里面，我们不能采用传统的如SGD之类的算法来学习我们的模型，因此我们会采用一种叫做additive training的方式（另外，在我个人的理解里面7，boosting就是指additive training的意思）。每一次保留原来的模型不变，加入一个新的函数$f$到我们的模型中。

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

加入一个新的函数

第t轮的模型预测　　保留前面t-1轮的模型预测

现在还剩下一个问题，我们如何选择每一轮加入什么$f$呢？答案是非常直接的，选取一个$f$来使得我们的目标函数尽量最大地降低8。

$$\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}$$

目标：找到 $f_t$ 来优化这一目标

这个公式可能有些过于抽象，我们可以考虑当$l$是平方误差的情况。这个时候我们的目标可以被写成下面这样的二次函数9：

$$\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const \\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}$$

一般叫做残差

更加一般的，对于不是平方误差的情况，我们会采用如下的泰勒展开近似来定义一个近似的目标函数，方便我们进行这一步的计算。

- 目标 $Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$
- 用泰勒展开来近似我们原来的目标
  - 泰勒展开：$f(x+\Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$
  - 定义：$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$，$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$$Obj^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) + constant$$

当我们把常数项移除之后，我们会发现如下一个比较统一的目标函数。这一个目标函数有一个非常明显的特点，它只依赖于每个数据点的在误差函数上的一阶导数和二阶导数10。有人可能会问，这个材料似乎比我们之前学过的决策树学习难懂。为什么要花这么多力气来做推导呢？

$$\sum_{i=1}^{n}\left[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\Omega(f_t)$$

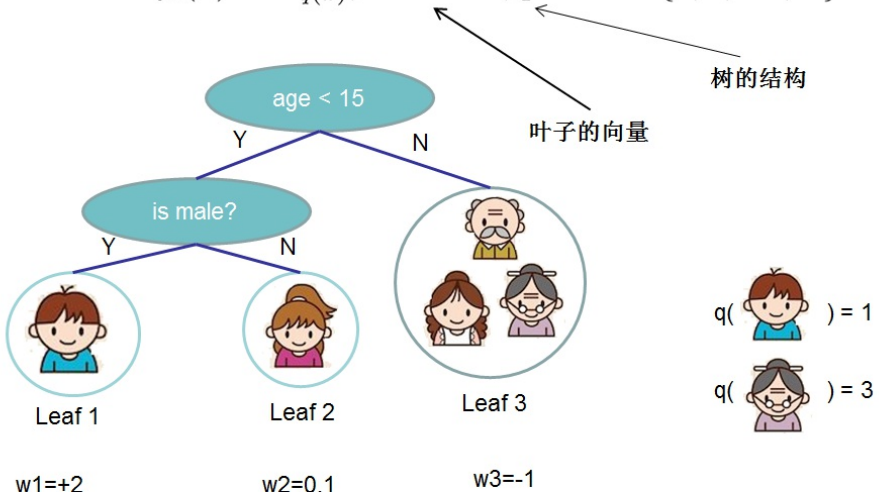
- where $g_i=\partial_{\hat{y}^{(t-1)}}l(y_i,\hat{y}^{(t-1)}),\quad h_i=\partial^2_{\hat{y}^{(t-1)}}l(y_i,\hat{y}^{(t-1)})$

因为这样做使得我们可以很清楚地理解整个目标是什么，并且一步一步推导出如何进行树的学习。这一个抽象的形式对于实现机器学习工具也是非常有帮助的。传统的GBDT可能大家可以理解如优化平法$a$残差，但是这样一个形式包含可所有可以求导的目标函数。也就是说有了这个形式，我们写出来的代码可以用来求解包括回归，分类和排序的各种问题，**正式的推导可以使得机器学习的工具更加一般**。

*iv. 树的复杂度*

到目前为止我们讨论了目标函数中训练误差的部分。接下来我们讨论如何定义树的复杂度。我们先对于f的定义做一下细化，把树拆分成结构部分$q$和叶子权重部分$w$。下图是一个具体的例子。结构函数$q$把输入映射到叶子的索引号上面去，而$w$给定了每个索引号对应的叶子分数是什么。

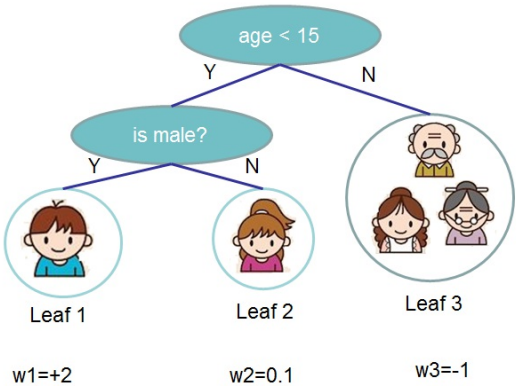
$$f_t(x)=w_{q(x)},\quad w\in\mathbf{R}^T, q:\mathbf{R}^d\rightarrow\{1,2,\cdots,T\}$$




当我们给定了如上定义之后，我们可以定义一棵树的复杂度如下。这个复杂度包含了一棵树里面节点的个数，以及每个树叶子节点上面输出分数的$L2$模平方。当然这不是唯一的一种定义方式，不过这一定义方式学习出的树效果一般都比较不错。下图还给出了复杂度计算的一个例子。

$$\Omega(f_t)=\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}w_j^2$$

叶子的个数　　w的L2模平方




$$\Omega=\gamma 3+\frac{1}{2}\lambda(4+0.01+1)$$

*v. 关键步骤*

接下来是最关键的一步[11]，在这种新的定义下，我们可以把目标函数进行如下改写，其中I被定义为每个叶子上面样本集合 $I_j=\{\ i|q(x_i)=j\}$

$$
\begin{aligned}
Obj^{(t)} &\simeq \sum_{i=1}^{n}\left[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\Omega(f_t)\\
&=\sum_{i=1}^{n}\left[g_i w_{q(x_i)}+\frac{1}{2}h_i w_{q(x_i)}^2\right]+\gamma T+\lambda\frac{1}{2}\sum_{j=1}^{T}w_j^2\\
&=\sum_{j=1}^{T}\left[(\sum_{i\in I_j}g_i)w_j+\frac{1}{2}(\sum_{i\in I_j}h_i+\lambda)w_j^2\right]+\gamma T
\end{aligned}
$$

这一个目标包含了$T$个相互独立的单变量二次函数。我们可以定义

$$G_j=\sum_{i\in I_j}g_i H_j=\sum_{i\in I_j}h_i$$

那么这个目标函数可以进一步改写成如下的形式，假设我们已经知道树的结构$q$，我们可以通过这个目标函数来求解出最好的$w$，以及最好的$w$对应的目标函数最大的增益

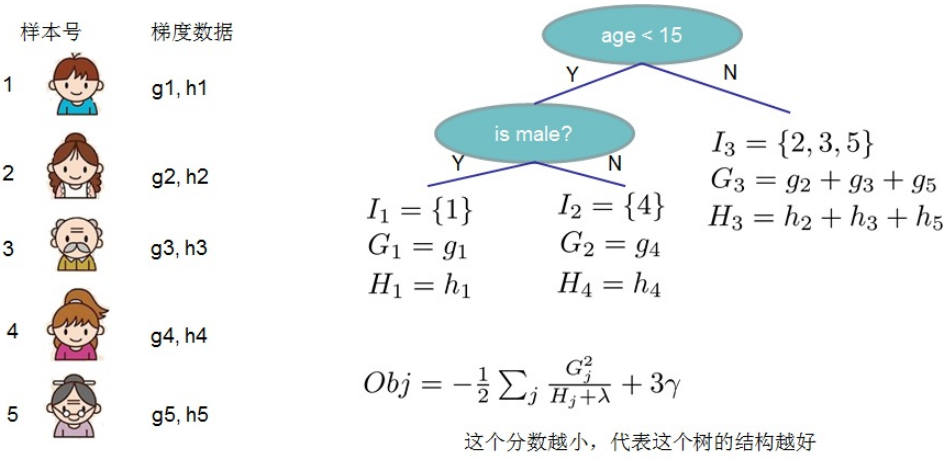
$$Obj^{(t)}=\sum_{j=1}^{T}[(\sum_{i\in I_j}g_i)w_j+\frac{1}{2}(\sum_{i\in I_j}h_i+\lambda)w_j^2]+\gamma T=\sum_{j=1}^{T}[G_j w_j+\frac{1}{2}(H_j+\lambda)w_j^2]+\gamma T$$

这两个的结果对应如下，左边是最好的$w$，右边是这个$w$对应的目标函数的值。到这里大家可能会觉得这个推导略复杂。其实这里只涉及到了如何求一个一维二次函数的最小值的问题[12]。如果觉得没有理解不妨再仔细琢磨一下

$$w^*_j=-\frac{G_j}{H_j+\lambda}Obj=-\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda}+\gamma T$$

*vi. 打分函数计算举例*

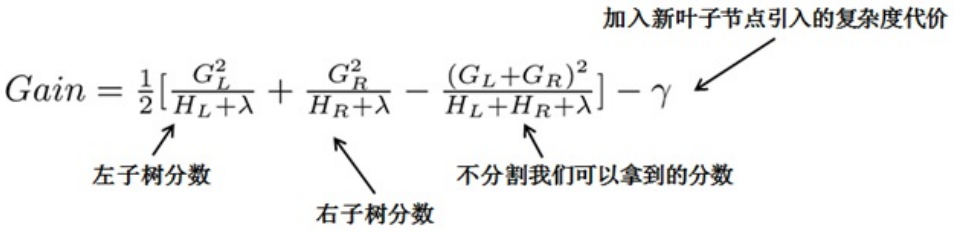
Obj代表了当我们指定一个树的结构的时候，我们在目标上面最多减少多少。我们可以把它叫做结构分数(structure score)。你可以认为这个就是类似吉尼系数一样更加一般的对于树结构进行打分的函数。下面是一个具体的打分函数计算的例子



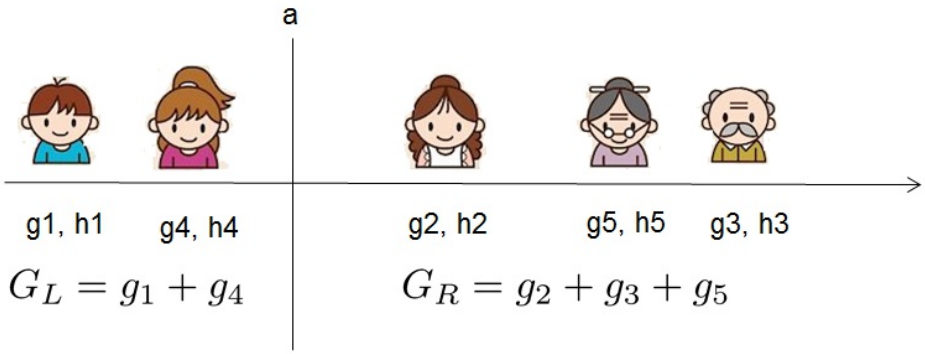

这个分数越小，代表这个树的结构越好

*vii. 枚举所有不同树结构的贪心法*

所以我们的算法也很简单，我们不断地枚举不同树的结构，利用这个打分函数来寻找出一个最优结构的树，加入到我们的模型中，再重复这样的操作。不过枚举所有树结构这个操作不太可行，所以常用的方法是贪心法，每一次尝试去对已有的叶子加入一个分割。对于一个具体的分割方案，我们可以获得的增益可以由如下公式计算




对于每次扩展，我们还是要枚举所有可能的分割方案，如何高效地枚举所有的分割呢？我假设我们要枚举所有 $x<a$ 这样的条件，对于某个特定的分割$a$我们要计算$a$左边和右边的导数和。




我们可以发现对于所有的$a$，我们只要做一遍从左到右的扫描就可以枚举出所有分割的梯度和$G_L$和$G_R$。然后用上面的公式计算每个分割方案的分数就可以了。

观察这个目标函数，大家会发现第二个值得注意的事情就是引入分割不一定会使得情况变好，因为我们有一个引入新叶子的惩罚项。优化这个目标对应了树的剪枝， 当引入的分割带来的增益小于一个阀值的时候，我们可以剪掉这个分割。大家可以发现，当我们正式地推导目标的时候，像计算分数和剪枝这样的策略都会自然地出现，而不再是一种因为heuristic而进行的操作了。

讲到这里文章进入了尾声，虽然有些长，希望对大家有所帮助，这篇文章介绍了如何通过目标函数优化的方法比较严格地推导出boosted tree的学习。因为有这样一般的推导，得到的算法可以直接应用到回归，分类排序等各个应用场景中去。

**5. 尾声：xgboost**

这篇文章讲的所有推导和技术都指导了xgboost https://github.com/dmlc/xgboost 的设计。xgboost是大规模并行boosted tree的工具，它是目前最快最好的开源boosted tree工具包，比常见的工具包快10倍以上。在数据科学方面，有大量kaggle选手选用它进行数据挖掘比赛，其中包括两个以上kaggle比赛的夺冠方案。在工业界规模方面，xgboost的分布式版本有广泛的可移植性，支持在YARN, MPI, Sungrid Engine等各个平台上面运行，并且保留了单机并行版本的各种优化，使得它可以很好地解决于工业界规模的问题。有兴趣的同学可以尝试使用一下，也欢迎贡献代码。

***注解和链接：***

- 1: Multiple Additive Regression Trees, Jerry Friedman, KDD 2002 Innovation Award 创新奖 http://www.sigkdd.org/node/362
- 2: Introduction to Boosted Trees, Tianqi Chen

, 2014 http://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

- 3: *Principles of Data Mining*, David Hand et al,2001. Chapter 1.5 *Components of Data Mining Algorithms*, 将数据挖掘算法解构为四个组件：1）模型结构（函数形式，如线性模型），2）评分函数（评估模型拟合数据的质量，如似然函数，误差平方和，误分类率），3）优化和搜索方法（评分函数的优化和模型参数的求解），4）数据管理策略（优化和搜索时对数据的高效访问）。
- 4: 在概率统计中，参数估计量的评选标准有三：1）无偏性（参数估计量的均值等于参数的真实值），2）有效性（参数估计量的方差越小越好），3）一致性（当样本数从有限个逐渐增加到无穷多个时，参数估计量依概率收敛于参数真值）。《概率论与数理统计》，高祖新等，1999年，7.3节估计量的评选标准
- 5: Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press. Breiman获KDD 2005 Innovation Award 创新奖
- 6: F是泛函空间，或者叫假设空间；f是函数，或者叫假设。这里的ensemble一共有K个组件/基学习器，他们的组合方式是不带权重的累加。*NOTE*：【一般boosting里面（如AdaBoost）$f$是有权重叠加 原因是那些权重自然地被吸收到叶子的weight里面去了。】 by Chen Tianqi
- 7: Friedman等（ Friedman, J., Hastie, T., & Tibshirani, R. (2000). *Additive logistic regression: a statistical view of boosting* (with discussion and a rejoinder by the

authors). The annals of statistics, 28(2), 337-407. ）在2000年为boosting这种集成学习方法范式提供了一种统计解释，即加性逻辑斯蒂回归。但是他们不能完全解释AdaBoost算法的抗过拟合性。南大周志华等人（http://www.slideshare.net/hustwj/ccl2014-keynote）从统计学习理论的间隔分布出发，较完满的解决了这个问题。

- 8：这个思想即是AdaBoost等boosting算法的思想：串行生成一系列基学习器，使得当前基学习器都重点关注以前所有学习器犯错误的那些数据样本，如此达到提升的效果。
- 9：注意，这里的优化搜索变量是$f$（即搜索当前的基学习器f），所以其他不关于$f$的项都被吸纳到const中，如当前的误差损失
- 10：1). 分别作为要优化的变量$f$的一次系数$g$和二次系数$1/2*h$

2). 误差函数对 $\hat{}y_{(t-1)i}$求导时，即涉及到前$t-1$个基学习器，因为他们决定了当前的预测结果

- 11：因为每个数据点落入且仅落入一个叶子节点，所以可以把$n$个数据点按叶子成组，类似于合并同类项，两个数据点同类指的是落入相同的叶子，即这里的指示变量集$I_j$
- 12：变量是$w_j$，关于它的二次函数是：$G_j*w_j+1/2(H_j+\lambda)*w2_j$，该函数对变量$w_j$求导并令其等于0，即得到$w*_j$的表达式。