XGBoost原理解析

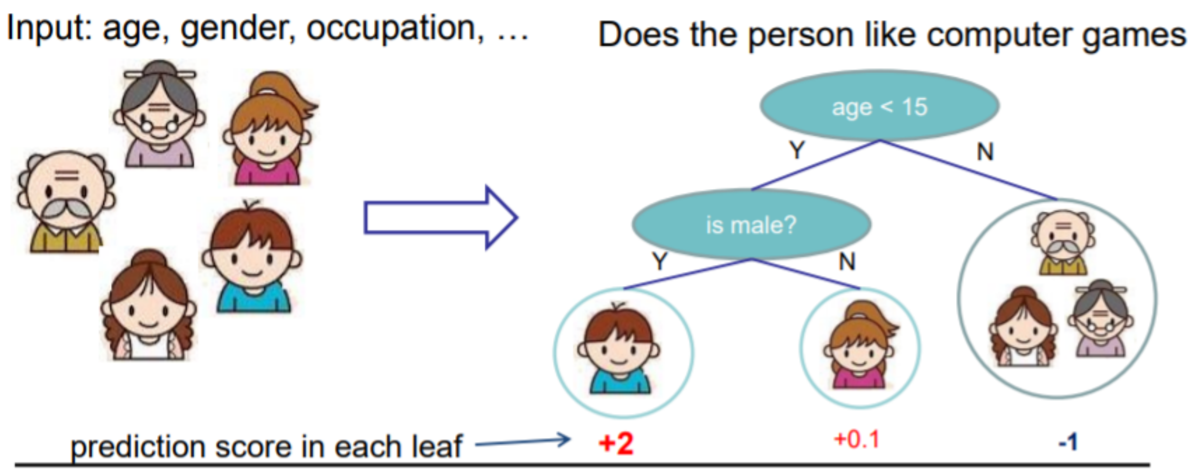
参考资料：

陈天奇博士的slides：[Introduction to Boosted Trees]()

XGBoost是“Extreme Gradient Boosting” 的缩写，是Gradient Boosting的一种改进算法，由陈天奇博士于2014年提出。这里我们直接按照他的PPT思路来对XGBoost的原理进行介绍。

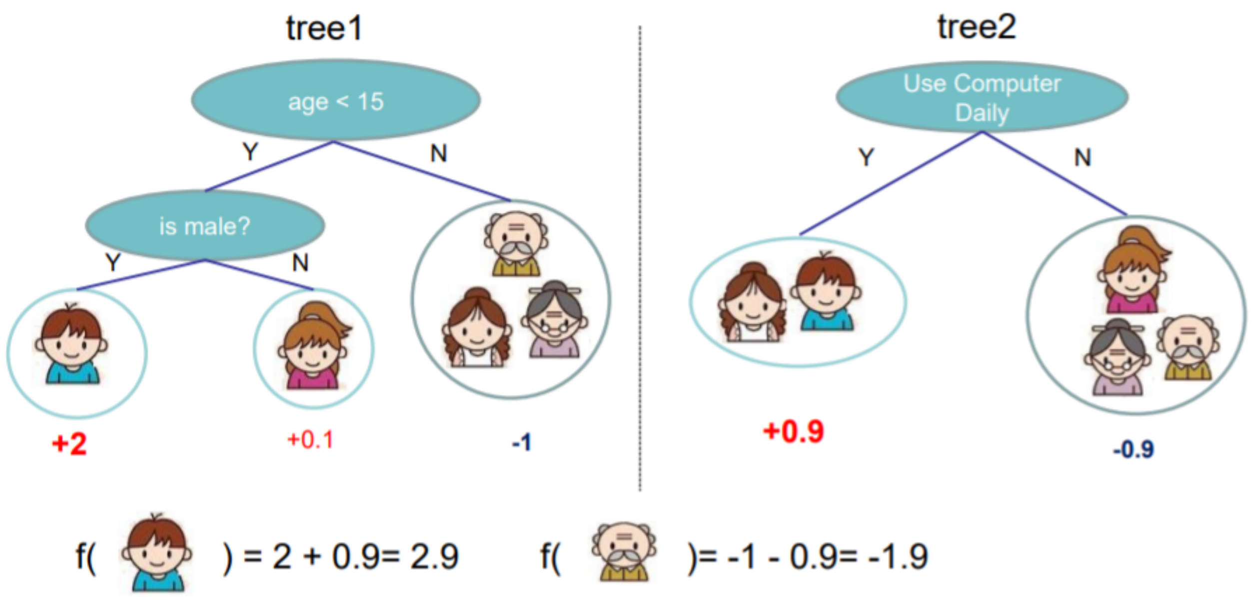
从名字就可以看出，传统的gradient boosting的基础是梯度，或者说一阶导数，而XGBoost不仅使用了一阶导数，还使用了二阶导数。

我们约定$x_i \in \mathbf{R}^d$代表第i个训练样本，标签是y_i，模型的预测值为$\hat{y}_i$。使用CART回归树作为基学习器，我们首先从回归问题出发，考虑下面这个例子:



这里我们的目标是根据年龄、性别、职业等特征预测一个人对电脑游戏的喜爱程度。

传统的回归树ensemble可能得到像下面这样的模型：



假设我们有K棵树，那么ensemble的模型就是这些树的求和（即加性模型）：

$$\hat{y}_i=\sum_{k=1}^{K} f_k\left(x_i\right)$$

通常目标函数由训练loss和正则化项构成：

$$Obj=\sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right)+\sum_{k=1}^{K} \Omega\left(f_k\right)$$

这里$\Omega\left(f_k\right)$代表的是树的复杂度，它包含两部分：

- 树的结构复杂度，可以用叶结点数量、树的深度、进行划分（split）的次数等指标来衡量；
- 所有叶结点上的值（称为权重）的l2 norm。（*这一项是XGBoost独有的，它代表了对叶结点上权重的平滑*）

与gradient boosting类似，我们采用加性模型和前向分布算法来进行训练，如下：

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

在第t步，目标函数表达为：

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

考虑平方损失函数：

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

事实上，我们可以将上式改写成泰勒展开的形式。回忆一下泰勒展开：

$$f(x+\Delta x) \simeq f(x)+f'(x)\Delta x+\frac{1}{2} f''(x)\Delta x^2$$

定义

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

于是我们有：

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

对于平方损失函数，有：

$$g_i=\partial_{\hat{y}^{(t-1)}}\left(\hat{y}^{(t-1)}-y_i\right)^2=2\left(\hat{y}^{(t-1)}-y_i\right)$$
$$h_i=\partial_{\hat{y}^{(t-1)}}^2\left(y_i-\hat{y}^{(t-1)}\right)^2=2$$

代回到式（2），就是式（1）的形式。

将式（2）中与优化无关的项去掉，我们新的目标函数变为：

$$\sum_{i=1}^{n}\left[g_i f_t\left(x_i\right)+\frac{1}{2} h_i f_t^2\left(x_i\right)\right]+\Omega\left(f_t\right) \qquad (3)$$

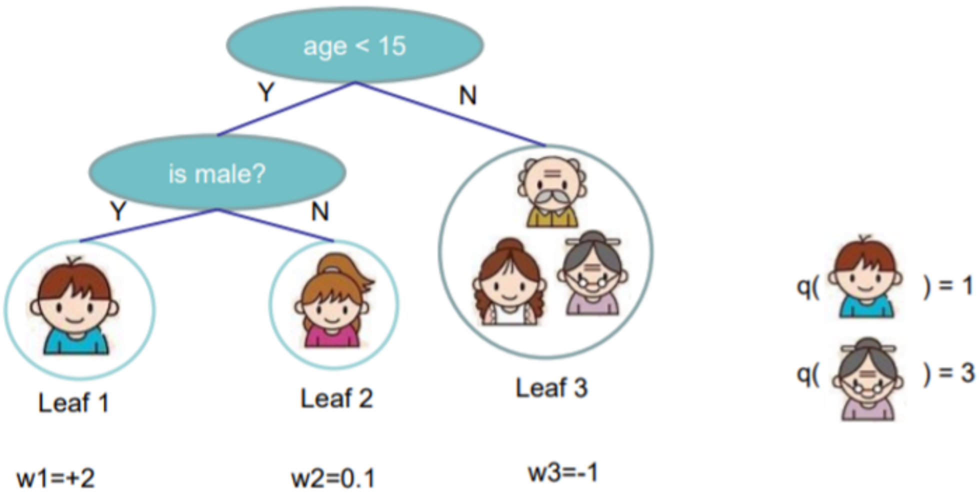
其中$g_i=\partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$，$h_i=\partial_{\hat{y}^{(t-1)}}^2 l\left(y_i, \hat{y}^{(t-1)}\right)$。

到此为止，我们已经得到了XGBoost中的目标函数，与gradient boosting相比，这里引入了二阶导数和正则化项。

接下来，我们重新定义一下决策树的表达式：

$$f_t(x)=w_{q(x)}, \quad w \in \mathbf{R}^T, q: \mathbf{R}^d \rightarrow\{1,2, \cdots, T\} \qquad (4)$$

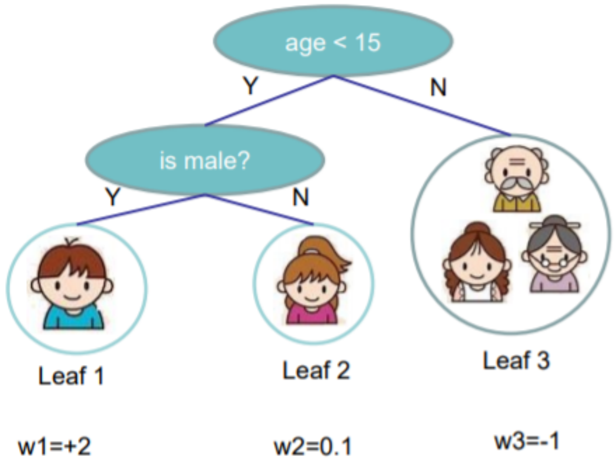
这里的w是一个向量，它的每个分量就是每个叶子结点上的值；q是一个下标映射函数，其作用是将一个样本映射到其对应的叶子结点上。式（4）的效果可以用下图来说明：



再定义树的复杂度为：

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

上式包含了叶子结点的数目以及叶子结点上值的l2 norm。比如：



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

我们定义每个叶子结点上的样本集合为:

$$I_j = \{i | q(x_i) = j\}$$

然后将目标函数重新改写为:

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t) \\ &= \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2\right] + \gamma T + \lambda\frac{1}{2}\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \end{aligned} \tag{5}$$

上式中两次求和$\sum_{j=1}^{T}\sum_{i=I_j}$实际上就是$\sum_{i=1}^{n}$。

注意到式（5）是T个独立的二次函数的和。

对于一个简单的二次函数，有如下性质:

$$\operatorname{argmin}_x Gx + \frac{1}{2}Hx^2 = -\frac{G}{H}, H > 0$$

$$\min_x Gx + \frac{1}{2}Hx^2 = -\frac{1}{2}\frac{G^2}{H}$$

我们定义$G_j = \sum_{i\in I_j} g_i$，$H_j = \sum_{i\in I_j} h_i$，于是目标函数变为:

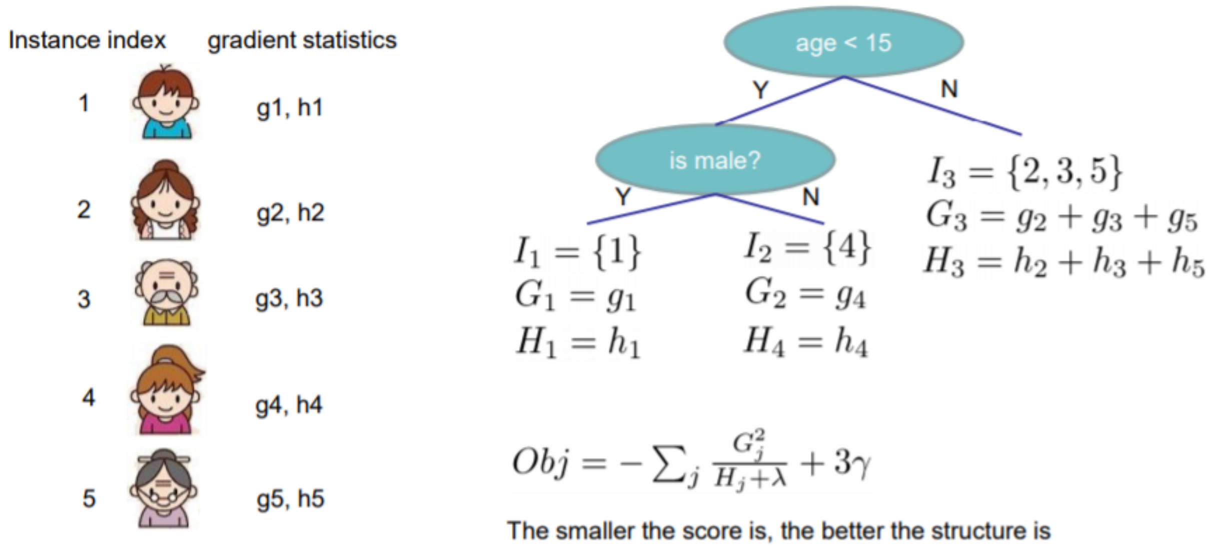
$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \\ &= \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2\right] + \gamma T \end{aligned} \tag{6}$$

我们假设树的结构已经确定（即$q(x)$固定），那么每棵树的最优权重和最优目标值为:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{7}$$

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{8}$$

我们仍然通过电脑游戏的例子来对式（8）进行说明：



于是我们可以得到搜寻单棵树的简单算法：

- 穷举所有可能的树的结构q
- 根据式（8）计算每个特定结构q的得分，确定最佳结构：

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

- 找到最佳结构后，使用最优的叶子结点权重值：

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

这里的问题是，可能的树结构有无穷多个，再实际操作中我们应该怎么列举呢？这里XGBoost使用了类似于信息增益的划分准则。

考虑划分某个结点前的目标函数值：

$$\begin{aligned} Obj &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \\ &= -\frac{1}{2}\left(\sum_{j=1}^{T-1}\frac{G_j^2}{H_j + \lambda} + \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right) + \gamma T \end{aligned}$$

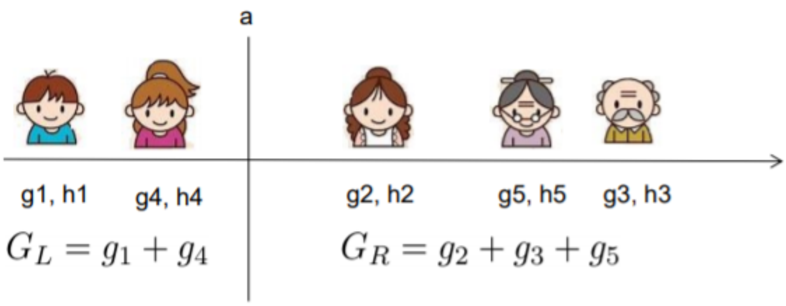
划分后的目标函数值为：

$$Obj = -\frac{1}{2}(\sum_{j=1}^{T-1}\frac{G_j^2}{H_j+\lambda}+\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda})+\gamma(T+1)$$

将划分前的目标值减去划分后的目标值，即可得到增益：

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma \quad (9)$$

其中$\frac{G_L^2}{H_L+\lambda}$代表了划分后左子结点的目标得分，$\frac{G_R^2}{H_R+\lambda}$代表了划分后右子结点的目标得分，$\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$是父结点（即不进行划分）的目标得分，$\gamma$代表了划分后引入新的叶子结点所增加的复杂度。例如，对于划分准则$x_j < a$，左右结点G的值计算如下：



剩下的问题就是如何找到最佳划分点了，这里我们采用传统决策树算法中的做法即可：

- 对于每个结点，对所有特征穷举：
  - 对每一个特征，根据该特征的取值对所有样本进行排列；
  - 利用线性扫描的方式找到该特征的最佳划分点；
  - 取所有特征最佳划分点中的最佳值。

至此，我们可以将XGBoost下的提升树算法总结如下：

- 每次迭代增加一棵新的树
- 在每次迭代开始时，计算：

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

- 从深度为0开始，利用贪心法来生成一棵树$f_t(x)$，目标函数为：

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

- 将$f_t(x)$增加到模型中：$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$。
  - 通常作为替代，新的模型为：$y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$
  - ϵ被称作步长或者shrinkage，通常设置在0.1左右
  - 这意外着在每一步，我们不执行最优的模型，并保留继续提升的空间，这中做法可以防止过拟合。

以上就是XGBoost的基本原理，当然在实际操作中，XGBoost还在这之上添加了许多的技巧和算法上的优化，比如在训练决策树时采用了类似随机森林的采样方式，这里就不一一介绍了。