Tree Boosting

Regularized Learning Objective：

假设有一个包含n个样本和m个特征的数据集 $D=\{(x_i,y_i)\}$ $(|D|=n,x_i\in R^m,y_i\in R)$
为了预测这个输出，我们可以用一个使用了 K 个可加的函数的集成树来得到预测输出。

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

其中，$F=\{f(x)=w_{q(x)}\}(q:R^m\to T,w\in R^T)$ 是 CART 回归树对应的空间。以下都是对单个$f(x)$而言的，而非整体的集成树。q是一个可以把每个样本(example) x 映射到对应的叶子编号(leaf index)。T是在这个树中的叶子数量(number of leaves)。每一个f_k都对应于一个独立的树结构q和叶子权重w。每个回归树的每个叶子都包含一个连续的权重(continuous score)，第 i 个叶子的权重为w_i。

问题来了，如何得到这一系列的可加的函数f_k呢？
我们需要最小化(minimize)接下来的正则化目标函数：

$$\mathcal{L}(\phi)=\sum_i l(\hat{y}_{\mathrm{i}},y_i)+\sum_k\Omega(f_k)$$

$$\Omega(f)=\gamma T+\frac{1}{2}\lambda\|w\|^2$$

其中 γ是叶子的个数，后者是对叶子权重(leaf score)的 L2 范数。
l是可微分的损失函数(loss function)用来度量预测值$\hat{y}_{\mathrm{i}}$和真实值y_i之间的差异。

Gradient Tree Boosting

问题又来了，上述$\mathcal{L}(\phi)$中的参数是函数，故不能够像欧几里得空间一样进行优化，怎么办？
我们采用加性方法。定义$\hat{y}_{\mathrm{i}}^{(t)}$是第 i 个样本(instance)在第 t 次迭代的预测结果，故第 t 次得到的目标函数：

$$\mathcal{L}^{(t)}=\sum_{i=1}^{n}l\left(y_i,\hat{y}_{\mathrm{i}}^{(t-1)}+f_t(x_i)\right)+\Omega(f_t)$$

由二阶的泰勒展开式：

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

把$f_t(x_i)$看成Δx，得到二阶近似的$\mathcal{L}^{(t)}$，可以快速优化目标函数:

$$\mathcal{L}^{(t)}\cong\sum_{i=1}^{n}\left[l\left(y_i,\hat{y}_{\mathrm{i}}^{(t-1)}\right)+g_if_t(x_i)+\frac{1}{2}h_if_t^2(x_i)\right]+\Omega(f_t)$$

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

移除常数项$l\left(y_i, \hat{y}_{\mathrm{i}}^{(t-1)}\right)$，得到精简的目标函数：

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

问题双来了，这个损失函数是对于 sample x_i而言的，但是正则项确是对于整个树结构的，两者没有统一，不能直接计算。怎么办？

把损失函数也写成对于整个树结构而言的形式。

首先，定义集合$I_j = \{i \mid q(x_i) = j\}$，是样本$x_i$ 经过映射后的叶子节点j的集合。又因为每一个叶子节点j都有权重(score) w_j，树对样本x_i在第 t 轮的估计值就等于该样本在这棵树对应叶子节点的权重：

$$f_t(x_i) = w_{q(x_i)}$$

所得f_t即为第 t 轮的决策树。故上式经过代入和展开$\Omega(f_t)$后可以得到：

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda\|w\|^2$$

将不同样本按照叶子节点归类（因为不同样本可能映射到同一个叶子节点j）：

$$\widetilde{\mathcal{L}^{(t)}} \approx \sum_{i=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i\right) w_j^2\right] + \gamma T + \frac{1}{2}\lambda\|w\|^2$$

合并同类项得到最终的 Objective Function：

$$\widetilde{\mathcal{L}^{(t)}} \approx \sum_{i=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T$$

上式是一个二次形式，且由以下可知：

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

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

故可以知道$\mathcal{L}^{(t)}$最小的值以及此时的最佳参数w_j^*：

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} = -\frac{G_j}{H_j + \lambda}$$

$$\widetilde{\mathcal{L}^{(t)}(q)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

最佳参数可以再O(1)的时间内求出，所以这就是 XGBoost 对于 GBDT 的求解速度的提升。

$\widetilde{\mathcal{L}^{(t)}(q)}$ 也可以用来作为这颗树 q 结构的度量函数。

问题叒来了。既然可以快速的得到$\mathcal{L}^{(t)}$最小的值以及此时的最佳参数w_j^*，但我们的最终目标是最小化(minimize)这个目标函数，该如何最小化呢？

想象一颗树只有一个叶子节点，为了最小化这个树的目标函数，我们可以通过不断的增加分支来对当前数据进行拟合。

问题叕来了，该如何选择增加分支的点呢？即如何衡量分支后是否比分支前可以使得目标函数减小。

我们可以通过新得到的树的左右分支的目标函数(或者叫$loss$)与之前的目标函数做大小的比较，如：

$$\mathcal{L}_{split} = loss_{old} - \left(loss_{new-left} + loss_{new-right}\right)$$
$$split\ \ is\ \ good\ \ if\ \ \mathcal{L}_{split} > 0\ \ or\ \ \mathcal{L}_{split} > threshould$$

故可以得到：

$$\mathcal{L}_{split} = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_R + H_L + \lambda} - \gamma$$

这个式子就是用来评估候选的分裂点的。γ是因为一次 split 后，原树的一个叶子结点变为两个新的叶子节点，叶子节点数量增加了 1，其权重为γ。

这就带来了第六个问题了，如何选择最佳分裂点？待会儿讲。

Shrinkage and Column Subsampling

此节主要讲防止过拟合。除了目标函数的正则项之外，XGBoost 还用了另外两个方法。

第一个方法，是 shrinkage 对 Tree Boosting 中的每一步对新得到的权重(weights)乘上一个η。类似于随机梯度下降中的 learning rate。

第二个方法，是 column(feature) subsampling，此方法借鉴于 Random Forest。*传统的决策树再选择划分属性的时是对当前节点的属性集合(假设有 d 个属性)中选择一个最优属性划分；而在 Random Forest 中，对每个节点，先从该节点的属性集合中随机选择一个包含 k 个属性的子集，然后再从这个子集中选择一个最优的属性用于划分。一般推荐*$\mathrm{k} = \log_2 \mathrm{d}$。

Split Finding Algorithm

现在来解决第六个问题。

Basic Exact Greedy Algorithm

如果直接用暴力算法遍历每一个特征，时间复杂度是$O(mn^2)$， n个样本和m个特征。为了降低时间复杂度，先对样本根据特征的值(feature value)进行排序，然后遍历排序过后的样本。此时的时间复杂度就是$O(mn\log n)$。所以如果树深为 d，则时间复杂度是$O(dmn\log n)$。

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i\in I} g_i,\ H \leftarrow \sum_{i\in I} h_i$
for $k = 1$ *to* m **do**
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

到此，整个 XGBoost 的基本知识梳理完毕。仍有一些关于 weighted quantile sketch、稀疏、缺失值等问题用到的时候再回过头看下理论吧。

Reference：
XGBoost: A Scalable Tree Boosting System
Introduction to Boosted Trees – Washington
XGBoost all in one http://www.pengfoo.com/machine-learning/2017-03-03#toc_5
GBDT 算法原理深入解析 https://www.zybuluo.com/yxd/note/611571