Xgboost

xgboost 思想：

1. xgboost 的套路看陈天奇的论文和 gbdt.pdf 就好了，网络博客也很多

自己的思考：

1. xgboost 怎样做回归？与分类有何区别？

xgboost 做回归只需要更改损失函数即可，如交叉熵损失函数。只要损失函数可以二阶导

问题是：有些样本的预测值会在 0-1 之外，这个怎么处理？

1）粗暴方法，小于 0 为 0，大于 1 为 1

2）做归一化，（x-最小值）除以最大值减最小值

2. xgboost 的每棵树的叶子节点的值怎么确定的？

叶子节点的值其实代表的是某一类样本的值，就是这一批样本在这个叶子结点上都等于这个值。然而这个值是在上一棵树训练好之后就确定下来的，因为 gi,hi 都是上一棵树训练完成之后就知道了。问题是怎么得到树的结构了。树的结构就是本文下面总结的，用了一个特殊的信息增益公式。w 值在树的结构确定了之后自然就得知了。

3. 第一棵树的叶子节点怎么确定？

因为上一棵树是没有的，所以可以计上一棵树的叶子节点全是 0，跟计算第二棵树都是一样的套路。

4. 加入了 shrinkge 的时候怎么确定叶子节点值？

shrinkge 是学习率这个东西，因为每一棵树训练的目标是残差，而论文中或者资料中一般不会讲怎么加学习率。学习的目标残差等于 y-学习率*yi^即可。

1. 首先是每一棵树都学习上一颗树的残差

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

2.损失函数使用二阶泰勒展开式展开

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

3. 因为每一颗树学习完成之后 y（t-1）预测值是已知了。因此下一棵树的损失函数为如下所示

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

4. 因为要求的参数是叶子节点，做变换

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

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

5. 因为是二元一次方程，故可求最大值及其参数的值

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

6. 将 w 代入损失函数即可得出。但是这是最优目标损失，但是问题是，做分类的时候基本无法做到这样，只能用这个函数来指导分裂，因为树分类是使用特征进行分裂。

7. 其中每次学习的分裂函数中包括上一颗树的叶子节点的值

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

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

把 $f_t, \Omega(f_t)$ 写成树结构的形式，即把下式代入目标函数中

$$f(\mathbf{x}) = w_{q(\mathbf{x})} \qquad \Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$$

8.

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

- 将公式中的常数项去掉，得到：

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

- 把 $f_t, \Omega(f_t)$ 写成树结构的形式，即把下式代入目标函数中

$$f(\mathbf{x}) = w_{q(\mathbf{x})} \qquad \Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$$

- 得到：

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

9.

- 怎么统一起来？

  定义每个叶节点j上的样本集合为 $I_j = \{i|q(x_i) = j\}$

  则目标函数可以写成按叶节点累加的形式：

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

如果确定了树的结构（即q(x)确定），为了使目标函数最小，可以令其导数为0，解得每个叶节点的最优预测分数为：

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

代入目标函数，得到最小损失为：

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

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

标红部分衡量了每个叶子节点对总体损失的的贡献，我们希望损失越小越好，则标红部分的值越大越好。

因此，对一个叶子节点进行分裂，分裂前后的增益定义为：

$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

Gain的值越大，分裂后 L 减小越多。所以当对一个叶节点分割时，计算所有候选(feature,value)对应的gain，选取gain最大的进行分割

精确算法

遍历所有特征的所有可能的分割点，计算gain值，选取值最大的（feature，value）去分割

**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

精确算法时间复杂度非常高。所以经常使用近似算法作为替代

- 近似算法

  对于每个特征，只考察分位点，减少计算复杂度

  - Global：学习每棵树前，提出候选切分点
  - Local：每次分裂前，重新提出候选切分点

**Algorithm 2:** Approximate Algorithm for Split Finding

**for** $k = 1$ ***to*** $m$ **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature $k$.
  Proposal can be done per tree (global), or per split(local).
**end**
**for** $k = 1$ ***to*** $m$ **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
**end**
Follow same step as in previous section to find max score only among proposed splits.

- 行抽样（row sample）
- 列抽样（column sample）
  借鉴随机森林
- Shrinkage（缩减），即学习速率
  将学习速率调小，迭代次数增多，有正则化作用

缩减这个怎样用到 xgboost 中去？
拿回归树举例子，当第一棵树学习好的时候，往往会通过目标值与第一棵树叶子结点做差，得到的残差拿来学习。然而第一棵树往往会学到的信息非常多，导致第二棵树开始的时候需要学习的信息已经非常少。这样就容易导致学习的不太好，往往会通过第一棵树的结果乘以一个缩减作为下一棵树学习的目标。

xgboost 是回归树，如何应用到分类中去？