西瓜书复习笔记08

- 个体与集成：
  - 集成学习的结构：
    先产生一组个体学习器，再使用某种策略将它们结合起来。
  - 个体学习器：
    - 同质：
      同质集成中，个体学习器算法都是统一的，也叫做基学习器、弱学习器。
    - 异质：
      异质集成中，个体学习器可能由不同算法生成，例如同时包括决策树和神经网络。
  - 同质学习器的两类：
    Boosting：弱学习器之间有强依赖关系，必须通过串行生成的序列化方法。
    Bagging：弱学习器之间没有依赖关系，用同时生成的并行化方法。
    — Boosting:
  - 什么是Boosting：
    Boosting是一种将弱学习器提升为强学习器的算法。先训练一个弱学习器；再根据弱学习器的表现对训练样本进行调整，使得先前做错的训练样本在后续收到更多关注；然后基于调整后的样本继续训练下一个弱学习器。最后弱学习器数目到达定值，将这几个弱学习器进行加权结合。
  - 弱学习器没达到特定数目怎么办：
    一旦不满足条件（例如弱学习器的误差率>0.5）,那么将这个学习器抛弃。这时还没到预定的学习轮数。我们需要对数据进行重采样，根据当前的样本分布重新对训练样本进行采样，再基于新采样的结果训练弱学习器。这样可以防止早停。
  - Boosting方法主要关注降低偏差。
  - AdaBoost:
    - 用于分类的算法，仅限于二分类[-1,1]
    - 思想：
      前一个弱分类器分错的样本会得到加强，加权后的全体样本再次用来训练下一个弱分类器。
    - 步骤：
      1. 初始样本的权值分布$\frac{1}{N}$：

$$D_1 = (W_{11}, w_{12} \cdots W_{1N}) \quad W_{1i} = \frac{1}{N}$$

      2. 多轮迭代：
         如果样本被正确分类，样本权值降低；

如果被错误分类，样本权值变高
计算本轮学习器权重：

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

其中：
ϵ_t为误差率
计算下一轮样本权重：

$$w_{m+1,i} = \frac{w_{mi}}{z_m}\exp\left(-a_m y_i \mathrm{G_m}\left(x_i\right)\right)$$

其中：
G_m为弱分类器
Z_m是归一化因子

3. 组合弱学习器：
   加大分类错误率小的学习器的权重；
   减小分类错误率大的学习器的权重。
   sign为阶跃函数：

$$sign\begin{cases} 1, x>0 \\ 0, x=0 \\ -1, x<0 \end{cases}$$

$$G(x) = sign\left(\sum_{m=1}^{m} a_m G_m(x)\right)$$

- 损失函数：

$$loss(x) = \exp\left(\hat{y}_i f\left(x_i\right)\right)$$

求导：

$$-e^{f(x)}p(f(x)=1|x) + e^{f(x)}p(f(x)=-1|x) = 0$$

求解：

$$f(x)=\frac{1}{2}\ln\left(\frac{p(f(x)=1|x)}{p(f(x)=-1|x)}\right)$$

- 何时停止：
  到达预定的错误率
  到达预定的迭代数
- 在SkLearn里，AdaBoost默认是树模型。

- GBDT（梯度提升树）：
  - 用于回归的算法，虽然经过调整后可以进行分类，但是GBDT是回归树。
  - 思想：
    GBDT核心在于每一棵树拟合前面树的残差；在残差减小的方向（负梯度）建立一个新模型；最后累加所有树的结果作为最终的结果。显然分类树结果是无法累加的。
    (它是一种基于boosting增强策略的加法模型，训练的时候采用前向分布算法进行贪婪的学习，每次迭代都学习一棵CART树来拟合之前 t-1 棵树的预测结果与训练样本真实值的残差。)
  - 分类树与回归树在处理连续值时的区别：
    分类树是穷举每个特征的每个阈值，按最大信息增益或者最小基尼指数来分裂节点。
    回归树是也是穷举每个特征的阈值，按组小均方误差来确定分裂点。
  - 如果训练集不变，那么训练三次得到的树是一样的。
  - GBDT的结合策略不是投票法也不是平均法，而是累加所有树的结果作为最终结果。
  - 过程：
    GBDT使用多轮迭代,每轮迭代产生一个弱学习器，每个学习器器是在上一轮学习结果的残差基础上进行训练，最后将所有学习器的结果累加。
  - 哪里体现了梯度：
    如果损失函数为均方误差的情况下，要保证每一轮保证损失函数最小。对损失函数求导得到(y^*-y)刚好是残差，也就是负梯度就是残差。
    但是如果损失函数不是均方误差，比如最大熵损失，那么负梯度就不是残差了。所以说提升树都是基于梯度的，只不过均方误差求导后刚好是残差。
  - 停止条件：
    - 直到每个叶子上的年龄都唯一（很难），一般是用叶子上的平均值作为该节点的预测值。
    - 到达规定的叶子上限。
  - sklearn中主要参数：
    - 每回合树的深度
    - 学习器个数
- XgBoost：
  作者陈天奇，华盛顿大学计算机系毕业，华人之光！
  - 基本思想：
    不断的添加树，不断的进行特征分裂生成树，每次添加树其实就是学习一个新函数，去拟

合上次预测的残差（train_y-y1）。训练好K颗树，要预测一个样本，就根据这个样本的特征选择每棵树的叶子结点。最后将这些叶子结点的对应分数加起来就行了（

$$\hat{y}_i = \sum_{k=1}^{k} f_k(x)$$

）。

(XGBoost对GBDT进行了一系列优化，比如损失函数进行了二阶泰勒展开、目标函数加入正则项、支持并行和默认缺失值处理等，在可扩展性和训练速度上有了巨大的提升，但其核心思想没有大的变化。)

- 下一棵树的输入是什么：
  train_x还是train_x，y变成了残差(train_y-y1)。
- 目标函数：
  当前第t颗树：

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

其中：
y_i是定值；
$\hat{y}_i^{(t-1)} + f_t\left(x_i\right)$是当前树t的预测值，由前t-1颗树的预测值和当前树t的输出值相加而成；
l就是损失函数（
可以为平方损失（用于回归）：

$$l\left(y_i, y^i\right) = \left(y_i - y^i\right)^2$$

也可以为logistic损失（用于分类）：

$$l\left(y_i, \hat{y}_i\right) = y_i \ln\left(1 + e^{-i\hat{y}_i}\right) + \left(1 - y_i\right) \ln\left(1 + e^{\hat{y}_i}\right)$$

）；
$\Omega\left(f_t\right)$为当前t树的正则项；
constant为前t-1颗树的复杂度之和；

- 什么是泰勒展开：
  用原式的导数构建一个多项式来近似函数在某一领域的值。
- 用泰勒展开来近似目标函数：
  三项泰勒展开：

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

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

其中：
一阶导：

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

二阶导：

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

为了对目标函数进行优化，将所有常数项删除：

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

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

也就是不同的结点分裂组合方式，对应不同的obj。

- 树的定义：

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

其中：
q(x)是每个样本x落在某叶子结点上，1.....T;
wq是该节点的分数。

- 树的复杂度：

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

其中：
γ(gamma)为结点个数的惩罚力度；
T为结点的个数；
λ为正则项的惩罚力度；
w为L2膜的平方；
L2为每个结点加L2平滑防止过拟合。
值越小复杂度越低，泛化能力越强。

- 叶子结点归组：
  将Obj在样本上遍历转为Obj在叶子结点上遍历。
  将上述展开带入得到：

$$Obj^{(t)}=\sum_{j=1}^{T}\left[G_jw_j+\frac{1}{2}\left(H_j+\lambda\right)w_j^2\right]+\gamma T$$

其中：

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

Gj表示某叶子结点内所有一阶导的累加值；
Hj表示某叶子结点内所有二阶导的累加值。
对wj求导等于0，并带入：

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

这个数越低代表树的结构越好。

- 节点分裂：
  贪婪算法：
  对每个特征的特征值进行排序，然后保存为block结构（后面的迭代中重复地使用这个结构，大大减小计算量）；
  对每个block选择使目标函数下降最大的点作为最优切割点（大量计算）；

最后对每个Block的最优切割点求增益，选择增益最大的那个特征做分裂。
（整个过程用并行）

- Gain增益：

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

第一项为左子树分数；
第二项为右子树分数；
第三项为不分割我们可以拿到的分数；
最后一项使加入新节点引入的复杂度。

- 如何加快寻找最佳分裂结点：
并行查找： XGBoost利用多个线程并行计算每个特征的最佳分裂点。
- 分裂停止条件：
-Gain<0,也就是分后小于阈值(gamma);
树到达最大深度;
样本权重和小于设定值（min_child_weight）；
- 串行和并行：
XGBoost是串行生成CART树；
XGBoost并行原理体现在最优切分点的选择。
- 调参心得：
（未整理笔记）
- 强烈推荐：
https://mp.weixin.qq.com/s?__biz=MzI1MzY0MzE4Mg==&mid=2247485159&idx=1&sn=d429aac8370ca5127e1e786995d4e8ec&chksm=e9d01626dea79f30043ab80652c4a859760c1ebc0d602e58e13490bf525ad7608a9610495b3d&scene=21#wechat_redirect

- LightGBM：

- **Bagging：**
  - BootStrap：
  有放回的随机采样。为了使样本空间既有联系又有差异，如果不放回的话，仅仅是每个基学习器用到了一小部分训练数据，不能进行有效训练。所以要使用互有交叠的采样子集。
  - 包外估计：
  36.8%的样本没有被采样，可作为验证集来对泛化性能进行包外估计。还可以辅助剪枝。
  - Bagging思想：
  可采样出T个m个样本的数据集，然后基于每个采样集训练出一个基学习器，再通过策略将这些基学习器结合。

- 方差偏差分解：
  Bagging主要关注于降低方差。
- 随机森林RF：
  - 什么是随机森林：
    在Bagging基础上，进一步在决策树的训练中引入随机属性选择。一般k=log2d。
  - 为什么要引入属性随机：
    样本随机给样本带来多样性，属性随机可以使进一步影响基学习器之间的差异，使得最终集成的泛化性能提升。
  - ET树是什么（Extra-Tree）：
    样本不随机，属性随机。
  - 选择划分：
    Bagging要对结点的所有属性进行考察；
    随机森林只对结点的一个属性子集进行考察。

- **结合策略：**
  - 多学习器的好处：
    解决泛化性能不佳的问题；
    降低局部最小的风险。
  - 平均法：
    - 简单平均：

$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x})$$

    - 加权平均：

$$H(x) = \sum_{i=1}^{T} w_i h_i(x)$$

      其中：
      权值大于0，权值之和为1。
      权值从训练数据中学习得到。
      怎么选用：
      基学习器结果相差较大选择加权平均，否则简单平均。
      加权平均不一定比简单平均好，权重可能会造成过拟合。
  - 投票法：
    - 绝对多数投票：
      某标记得票过半，则预测为该标记；否则拒绝预测。

- 相对多数投票：
  得票最多的标记为最终的预测标记，如果相同那么随机选取一个。
  一般使用相对多数投票。
- 加权投票
  个体学习器的输出类型：
  - 硬投票：
    基于类标记，预测标记是{0,1}，选择输出最多的标签。
  - 软投票：
    基于类概率，预测概率在[0,1]之间，选择概率最大的输出。
    （概率模型，LR，决策树，KNN（k临近最多的类的个数除以k），SVM）
  - *怎么看决策树的概率：
    把树模型建立好后，把训练数据在模型里跑一下，看最终各类别所在比例，这个比例就成了最后所需的概率了。

- 学习法：
  - Stacking：
    先从初始数据集中训练出N个初级学习器，将每个学习器的输出作为新数据集的输入，样本标记不变。在用次学习器去学习这个新数据集，得出结果。