Logistic Regression Report

王禹　PB18000145

I 算法原理

xgboost 是一个基于回归树的模型. 在第一棵树预测一个定值之后, 从第二棵树开始便可以利用残差以及一阶二阶导数辅助后面的回归树的建立. 每一步利用残差建树, 当建立完所有的树之后, 对任意一个样本, 便可以将所有的树的预测结果进行加和, 从而获得最终输出.

目标函数: 假设已经训练了 K 棵树, 对于第 i 个样本的预测值为: $\hat{y} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i)$. 则目标函数定义为:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{1}$$

其中第一项为训练误差, 第二项为正则化项.

叠加式训练每一轮都是由上一轮已经固定的预测值, 加上本轮新加入的函数共同预测. 即有 $\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$. 在每一轮的训练中, 都有一个预测值, 由该预测值便可以计算出目标函数的值: $Obj^{(t)} = (\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))) + \Omega(f_t) + constant$. 因此这一轮训练的任务就是找到一个 $f_t(x_i)$ 去最小化这个式子.

而对于这个函数直接优化很困难, 因此对上述目标函数进行 Taylor 展开, 同时代入 $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda||\omega||^2$(控制结点的值和结点数量的正则项), 得到:

$$\begin{aligned} Obj^{(t)} &= (\sum_{i=1}^{n}[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)]) + \Omega(f_t) + constant \\ &= \sum_{j=1}^{T}[\sum_{i\in I_j} g_i\omega_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)\omega_j^2] + \gamma T \end{aligned}$$

因此上式 Taylor 展开的结果便是新的目标函数. 可以直接求得最优值为 $\omega_j^* = -\frac{G_j}{H_j+\lambda}$.

寻找最优的树结构. 上面的方式虽然看起来很容易优化, 一步到位, 但这是在有了树的结构之后才有的最优值. 由于我们不可能将所有的树形结构全部列举一遍, 所以论文中使用的是贪心算法. 树从 0 开始增长, 每次尝试去 split, 观察目标函数值是否有下降. 而每次分裂一个结点时, 其它结点的 G_i 并不会改变. 因此可以定义在这个结点处的 $Gain$:

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

每次遍历所有的特征以及所有的分割点, 找其中最大的 Gain 对应的分割. 如果最大的 Gain 小于了一个阈值, 便不再分割.

预测结果. 得到最后的树的集合之后. 任给一个样本, 令所有的树都做一次预测, 将预测值加和, 得到的结果便作为最后的输出.

II 针对问题特殊设计

在这个问题中, 由于是二分类问题, 我们的 loss 可以有两种定义方式.

i. MSE Loss

在这种方式中, 将原问题视为一个回归问题. 第一棵树预测的值直接设为 0. 从第二棵树开始, 便可以利用前面预测结果算出来的梯度进行更新. 假设当前模型的预测结果为 $\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i)$. 那么第 i 个样本的 loss, 一阶导数, 二阶导数计算为:

$$\begin{aligned} \mathcal{L}_1 &= \frac{1}{2}(\hat{y}_i - y_i)^2 \\ g_i &= \frac{\partial \mathcal{L}_1}{\partial \hat{y}_i} = (\hat{y}_i - y_i) \\ h_i &= \frac{\partial^2 \mathcal{L}_1}{\partial \hat{y}_i^2} = 1 \end{aligned}$$

ii. Logistice Loss

在这种方法中, 将原问题视为一个二分类问题. 将多棵回归树预测的值加和之后经过一层 sigmoid, 在将 sigmoid 的结果与标签计算 CrossEntropyLoss. 因此如果当前模型的预测结果为 $\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i)$, 那么第 i 个样本的 loss, 一阶导, 二阶导计算为:

$$\begin{aligned} \mathcal{L}_2 &= y_i * \ln(\sigma(\hat{y}_i)) + (1 - y_i)\ln(1 - \sigma(\hat{y}_i)) \\ g_i &= \frac{\partial \mathcal{L}_2}{\partial \hat{y}_i} = \sigma(\hat{y}_i) - y_i \\ h_i &= \frac{\partial^2 \mathcal{L}_2}{\partial \hat{y}_i^2} = \sigma(\hat{y}_i)(1 - \sigma(\hat{y}_i)) \end{aligned}$$

其中 σ 表示 sigmoid 函数. 即 $\sigma(x) = \dfrac{1}{1 + e^{-x}}$.

III 实验

i. 数据集

实验数据集为皮马印第安人糖尿病数据，特征维度为 8 维, 具体数据统计如下:

表 1: Dataset Statistics

	正例个数	负例个数	合计
训练集 (train)	214	401	615
测试集 (test)	54	99	153

ii. 实验结果

实验结果见表2. 其中 MSE 表示使用的是 MSE Loss, 对应 2.1 中的结构. LOG 表示使用的是 Logistic Loss, 对应2.2 中的描述.

表 2: 实验结果. 其中 MSE 表示使用的是 MSE Loss, LOG 表示使用的是 Logistic Loss

	参数配置	测试集准确率	测试集准确率
Baseline			0.7
MSE	$\lambda = 0.2$, max-depth = 3, max-iter=3	0.7724	0.7386
MSE	$\lambda = 0.2$, max-depth = 5, max-iter=5	0.7951	0.7320
MSE	$\lambda = 0.2$, max-depth = 10, max-iter=10	0.7951	0.7320
LOG	$\lambda = 0.2$, max-depth = 5, max-iter=5	0.8374	0.7647
LOG	$\lambda = 0.2$, max-depth=10, max-iter=10	0.9268	0.6536
LOG	$\lambda = 1$, max-depth=5, max-iter=5	0.8455	0.7386
LOG	$\lambda = 1$, max-depth=10, max-iter=10	0.8927	0.7320

IV 实验结果分析

从这张表中可以得到如下几条结论:

1. MSE loss 比较不容易过拟合. 从生成树的过程中, 可以看出来并未生成很多很深的树. 因此不需要很大的 λ 去限制它.
2. Logistic loss 容易过拟合. 在选择 $\lambda = 0.2$, max-depth 和 max-iter 均为 10 的时候, 虽然训练集准确率很高, 但测试集的准确率却低于了 baseline. 可见已经过拟合.
3. Logistic loss 在建树过程中, 很容易产生很深, 叶子结点很多的树, 因此将 λ 调大, 增大惩罚项. 这样最后获得的结果相对来说更难过拟合一些. 即便调整 max-iter 和 max-depth, 测试集准确率相对比较稳定.

V 实验总结

本次实验中我了解了 XGboost 的实现过程, 了解到 XGBoost 是如何利用提升树来实现残差训练, 如何通过缩小残差来提升准确率. 实验过程中可见 XGboost 较为容易过拟合, 因此调参时需要根据过程中产生树的深度以及树的多少来适当调整. 在经过适当调整之后, 使用 XGboost 模型的结果成功超过了 Baseline, 达到了比较好的效果.