机器学习概论 实验报告

Lab3: XGBoost

2021 年 1 月 2 日

目录

1 算法数学基础 **2**

1.1 提升树 2

1.1.1 基本思想 2

1.1.2 相关公式 2

1.2 Gradient Tree Boost 3

1.2.1 目标函数与最优权重 3

1.2.2 split 的准则 3

2 算法过程简介 **3**

2.1 Split 准则算法 3

2.2 XGBoost 算法 4

3 实验结果 **4**

3.1 总体效果 4

3.2 单棵决策树 5

3.3 XGBoost 5

3.4 sklearn 的 GradientBoostingClassifier 6

4 实验小结 **6**

1 算法数学基础

XGBoost: e**X**treme **G**radient **Boost**ing, 最初是由 陈天奇 负责的研究项目, 后来其在各种数据挖掘比赛表现出了极大的性能优势.

1.1 提升树

1.1.1 基本思想

如果一个预测问题中, 训练的模型(称为 Model1)效果不好, 误差比较大, 我们可以通过基于残差的训练来拟合 Model2, 甚至 Model3, Model4,...

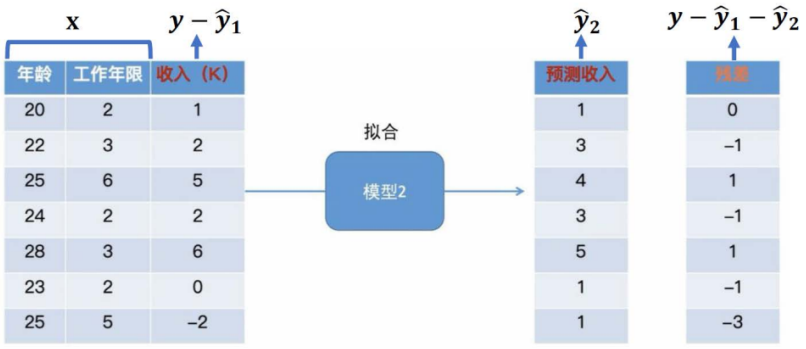

图 1: 基于 Model1 的残差训练 Model2

于是, 最后将各个模型的结果相加即可得到一个更优的模型. 这就是其基本思想.

年龄	工作年限	收入（K）		模型1		模型2		模型3		最终预测
20	2	10		9		1		0		10
22	3	13		11		3		-0.5		13.5
25	6	15		10	+	4	+	0.5	=	14.5
24	2	13		11		3		0		14
28	3	18		12		5		2		19
23	2	12		12		1		0		13
25	5	16		18		1		-2		17

图 2: 最后叠加每个模型的结果

1.1.2 相关公式

现在不妨考虑已经训练了 K 棵树, 则对第 i 个样本的预测为:

$$\hat{y}_i = \phi(x_i) = \sum_{i=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

这里目标函数为

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

其中第一项为 训练误差, 第二项为正则化项(可以是叶节点个数, 叶节点评分, 树的深度等)

1.2 Gradient Tree Boost

1.2.1 目标函数与最优权重

考虑第 i 轮的预测为:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i),$$

那么我们的目标函数就是

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

我们可以对目标函数中的训练误差项目, 进行泰勒展开, 并忽略常数项:

$$\begin{aligned}
\tilde{L}^{(t)} &= \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t(\mathbf{x}_i)] + \Omega(f_t) \\
&= \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \\
&= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) \omega_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) \omega_j^2 \right] + \gamma T
\end{aligned}$$

这里有$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ 及$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$
特别地, 如果取损失函数 $l(y_1, y_2) = \frac{1}{2}(y_1 - y_2)^2$, 那么就有

$$g_i = \hat{y}^{(t-1)} - y_i \quad \text{and} \quad h_i = 1$$

因此, 进一步我们能得到, 对于一个固定的树结构, 最优的叶子权重为:

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

而相应的最优值则为

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

1.2.2 split 的准则

根据上节给出的 $\tilde{\mathcal{L}}$, 我们很容易得到, 将叶子 split 的收益:

$$\mathcal{L}_{split} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i \right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i \right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i \right)^2}{\sum_{i \in I} h_i + \lambda} \right]$$

2 算法过程简介

2.1 Split 准则算法

根据上面的 split 准则, 我们可以设计这样的算法来计算最优 split 方法. 注意其中采用了按每个属性, 从小到大逐个分开地枚举, 从而贪心地得到当前最优 split 方法.

Algorithm 1 SPLIT

Require: I, instance set of current node

Require: m, 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 $sorted(I, by\ x_{jk})$ **do**

    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$

    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$

    $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$

return split with max score

2.2 XGBoost 算法

从上述的 split 算法, 我们就可以从一个数据集去构造一棵决策树. 随后, 我们使用 Gradient Tree Boost 的思想, 不断添加一棵树, 以此逼近决策目标.

Algorithm 2 XGBOOST

Require: dataset: 数据集

Require: : max_tree_num: 最大树数

init forest

for $t = 1$ to max_tree_num **do**

  Make a tree $tree^{(t)}$ according the objective: $\left(\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))\right) + \Omega(f_t)$ ▷ using the split mentioned above

  forest.append(this tree)

  $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

return forest

3 实验结果

3.1 总体效果

总体效果如下表所示:

算法	训练集上准确率	测试集上准确率
单棵决策树	0.74634	0.69935
XGBoost	0.83089	0.76471
sklearn 的 GradientBoostingClassifier	0.93496	0.72549

表 1: 实验结果总体效果

可以看出来, XGBoost 的对单棵决策树的改进还是比较大的, 甚至直逼 sklearn(一个由专业团队完成的库) 的结果. 而我这里的 XGBoost 经过一些参数调整等, 或许也能达到那样的效果, 但囿于时间有限, 暂时未能达到.

3.2 单棵决策树

可以画出它的决策树如下图: 其中叶子结点标志的值是权重

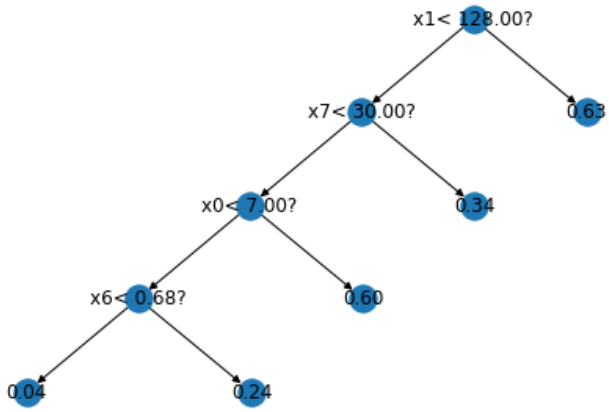

图 3: 单棵决策树的结果

可以看到, 它的结果并不如 baseline. 好在我们有 XGBoost 的加成.(见下文)

3.3 XGBoost

结合 XGBoost 技术, 多生成几棵树去弥补前面的不足, 效果有了比较显著的提升. 从原本的训练集准确率 0.74634 提升到了 **0.83089**, 测试集准确率 0.69935 提升到了 **0.76471**.

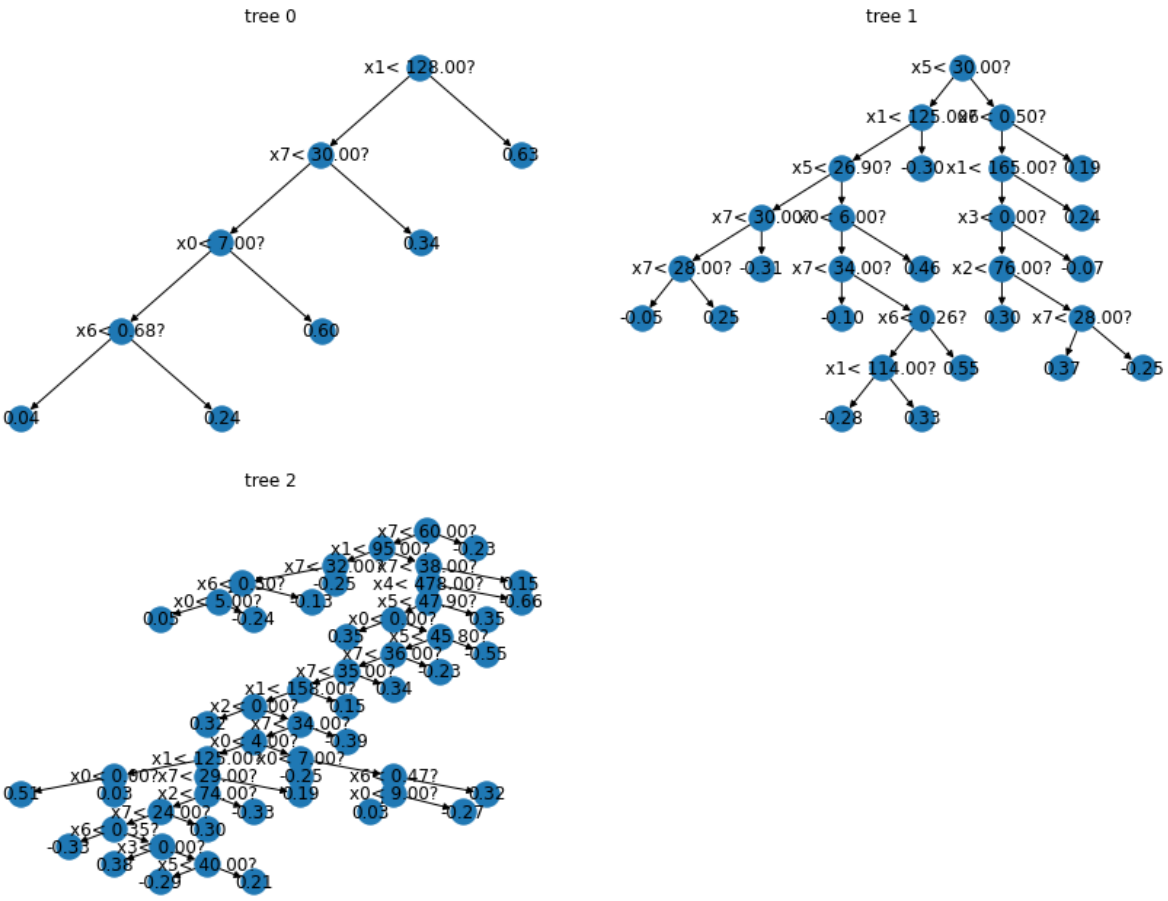

图 4: XGBoost 的结果(树结构只是给个直观感受, 因此不必在意显示的细节)

3.4 sklearn 的 GradientBoostingClassifier

sklearn 的训练集准确率能够达到 0.93496, 但测试集的准确率不如我写的, sklearn 只能达到 0.72549. 由于我这里只是调个库看看它们实现得怎么样, 就不画图了.

4 实验小结

本次实验需要先深入理解 XGBoost 的 paper 的前半部分, 对我这种没怎么读过 paper 的人来说, 收益还是很大的. 十分感谢老师和助教给了我这样的机会.