[toc]

背景

背景介绍

在真实的机器学习问题中，我们有时不仅要预估目标的均值，还要对目标取值的置信区间进行预估。特别是样本量比较少的场景下，比如广告营销的成本管理、金融产品的定价等，行为频率低，样本获取的成本高昂，单次行为背后往往都有大额的资金成本。这类相对低频的问题往往需要较深的人工干预，预估其目标变量的分布将为商业决策行为提供更充分的参考和依据。

特别的，对于我们的广告粒度的成本预估场景，广告的成本分布受到各种因素的影响，这些因素不仅影响成本分布的均值（一阶矩 mean），还影响分布的方差（二阶矩 variance），偏度（三阶矩 skewness），峰度（四阶矩 kurtosis）。为了能够准确的预估成本的置信区间，我们先要预估成本的分布。下面我们举两个例子来说明预估完整分布的必要性。

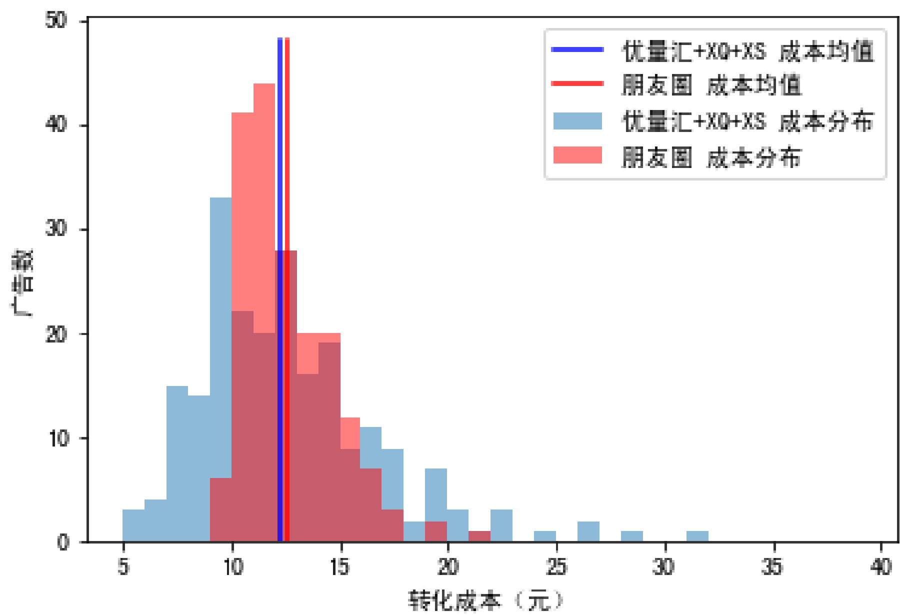


图 1 同样的广告主，在同一时间段投放相同的商品（Product id=2237556874 优化目标 = 激活 PT=12），在“朋友圈”和“优量汇”+XQ+XS 和版位的成本分布对比，可见二者均值接近，但是“优量汇”+XQ+XS 的成本分布更为分散。

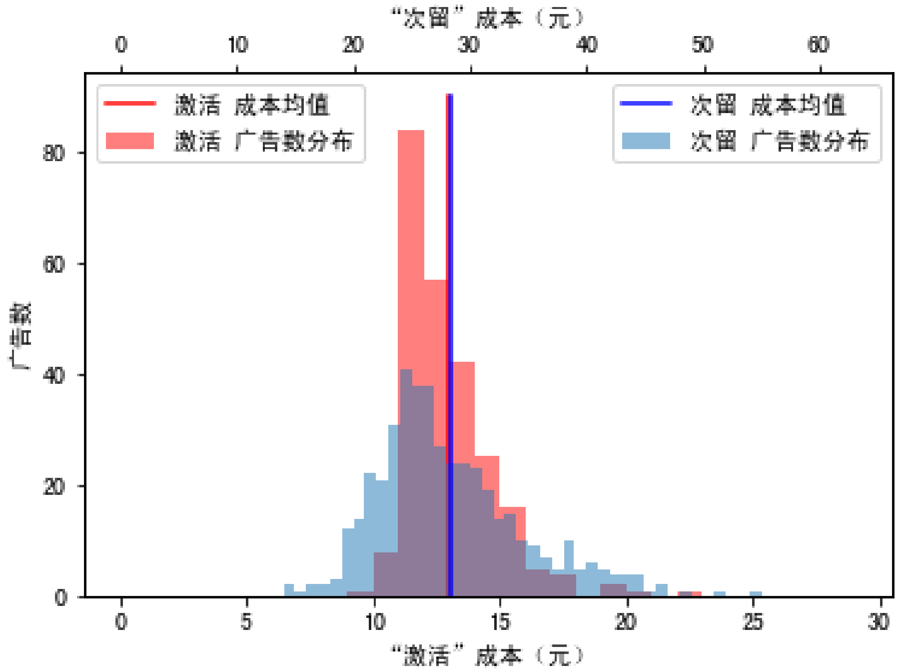


图 2 同样的广告主，在同一时间段投放相同的商品（Product id=2237556874 PT=12 版位 =“朋友圈”），在浅层优化目标“激活”阶段和深层优化目标“次留”阶段的成本分布对比，在将二者变换到一个尺度之后，能够看到深层优化目标的分布更为分散。

现有方案

在我们遇到的广告粒度的成本预估场景中，我们能够满足以下需求的一款开箱即用概率预估工具：

- 预估精度：预估精度高，支持连续特征、离散特征、特征交叉，点估计精度至少不低于 Xgboost。
- 求解复杂度：在有限时间内能完成广告场景下工业级数据规模的模型求解。
- 线上服务：能够集成到线上的 C++ 编写的线上服务，且能够满足 10ms 的返回。

目前我们调研到的一些比较有代表性的分布预估方案中，其优缺点列举如下：

模型名称	简介	现有实现方式	精度满足业务需求	求解复杂度	支持线上服务
Gaussian Process Regression [NIPS 1995]	用于时间序列等序列数据场景的概率预估，求解计算量大。	Python/R	x	x	x
GAMLSS [JSS 2007]	使用广义加法模型分别预估均值、方差、偏度、尖度，本质上仍然是线性模型，精度较差	Python/R	x	✓	x
BART[AOAS 2010]	Bayesian Additive Regression Tree，属于非参数模型，使用采样技术进行概率预估，计算量大	Python/R	✓	x	x
XGBOOST [2016 KDD] + Quantile Regression	Quantile Regression 思路，每个分位训练一个模型，灵活性较差	C++	x	✓	✓

模型名称	简介	现有实现方式	精度满足业务需求	求解复杂度	支持线上服务
NGBOOST [ICML 2020][2]	用分布的自然梯度（Natural Gradient）代替参数梯度，在实现上需要频繁的对参数的信息矩阵 (FIM) 求逆。	基于 Sk-learn	✓	x	x
PGBM[SIGKDD 2021][3]	用 gradient boosting 的方式预估分布的一阶矩和二阶矩，（一阶矩和二阶矩不能完全确定一个分布）	基于 PyTorch	x	✓	x

经过调研，有几点结论：

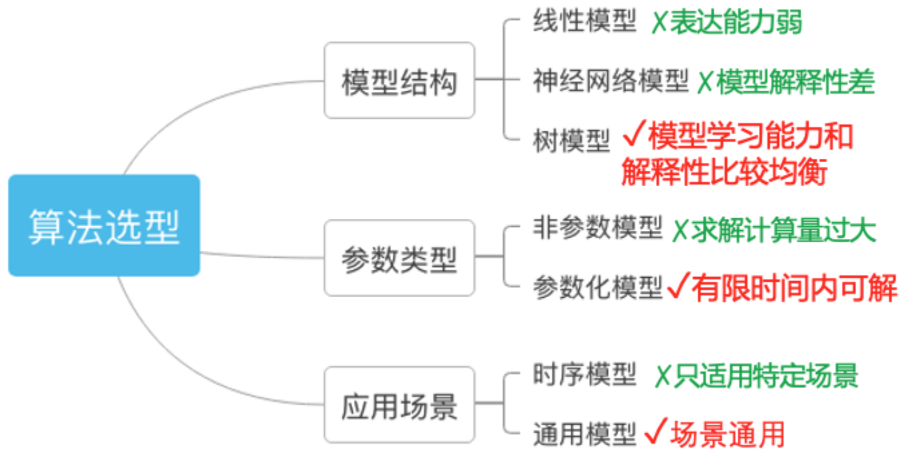
- 概率预估问题，近年来越来越受到机器学习和数据挖掘领域的关注，十年前的一些经典工具和论文，主要发表在统计学刊物上，而近两年比较有代表性的 NGBOOST 和 PGBM 开始出现在 ICML 和 KDD 这样的机器学习顶会上。
- 目前对于概率预估，仍然是学术界的探讨较多，目前暂时没有适合工业界是用的开箱即用的工具包。

对于我们要解决的生产场景下的成本、曝光、转化分布问题，目前暂无案例可供借鉴，也无成熟的框架可供使用，因此我们提出了 Probabilistic Boosting Tree 的框架。

模型设计

方案选型

综合考虑和模型的精确性、可解释行、实现难度、业界现有的思路之后，我们决定开发一种基于提升树(Boosting Tree)的分布预估算法，也就是 Probabilistic Boosting Tree 名字的由来。



单颗概率树

我们在构建单颗树模型时参考了论文 [1]，每一个节点代表一个概率分布，不同之处时我们并没有把父节点的输出作为子节点的先验概率。

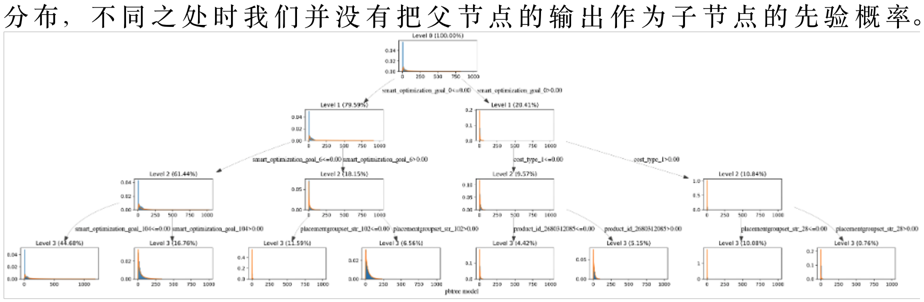


图 3 决策树的每个叶子结点都代表一个概率分布。

多棵概率树的集成学习

在构建多棵概率树进行集成学习时，我们把当前的样本的预估概率分布（第 1 轮时为初始分布）作为先验分布，样本落到的叶子节点的分布作为似然分布，然后根据贝叶斯公式求出后验分布。具体的，设 y 为要估计的变量，x 是我们观测到的特征，我们需要计算目标变量 y 的分布，也即：$p(y|x)$。

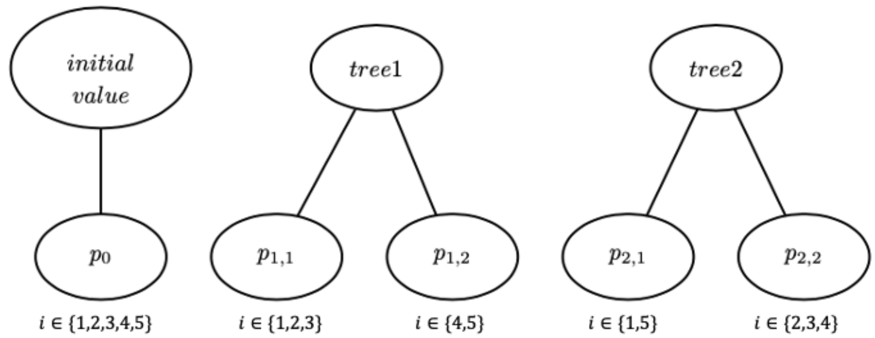


图 4 多棵概率树集成学习的建模思路示意

设树模型 T_1 的每个叶子节点 l_i 都是一个分布函数 $p_{l_i}(y)$。那么单颗树模型可以表示为:

$$p_{T_1}(y|x)$$

包含 d 颗树的概率树 Boosting 模型可以表示为:

$$q(y|x) = \frac{\prod_{i=0}^{d-1} p_{T_i}(y|x)}{C}$$

这里 C 相当于归一化项。从贝叶斯的角度来理解,

$$q_{T_m}(y|x) = \frac{p_{T_m}(y|x) \prod_{i=0}^{m-1} p_{T_i}(y|x)}{C}$$

$q_{T_{m-1}}(y|x) = \prod_{i=0}^{m-1} p_{T_i}(y|x)$ 相当于先验概率,$p_{T_m}(y|x)$ 相当于似然,q_{T_m} 相当于后验概率。

这个思路简单直接,但是由于是非启发式的算法,我们将之实现后发现其效果有天花板,我们考虑设计启发式算法进行求解。

模型求解

对于启发式算法,我们需要确定一个启发函数(损失函数),不断的去搜索这个启发函数的全局最优解。点估计场景下,损失函数一般由由 MLE(最大似然估计),在概率分布预估场景下,除了 MLE,还可以用 CRPS[4] 推导。

参数模型

以 Gamma 分布为例(Gamma 分布的定义域为 0 到正无穷,可能比正太分布更适合刻画某些真实场景下的问题)。

$$q_{T_m}(y|x) = \frac{1}{\Gamma(k_m){\theta_m}^{k_m}} y^{k_m - 1} e^{-\frac{y}{\theta_m}}$$

这时上式子中的似然函数的表达式会比较复杂，但如果我们仅仅对比先验分布与后验分布的差异，可以发现，先验分布乘以似然函数并且归一化以后，后验分布相对于先验分布的变化为：$k_{m-1} \rightarrow k_m$，$\theta_{m-1} \rightarrow \theta_m$。

设 $k_m = k_{m-1} + \Delta k$，$\theta_m = \theta_{m-1} + \Delta\theta$，因此我们只需要求解 Δk 和 $\Delta\theta$。

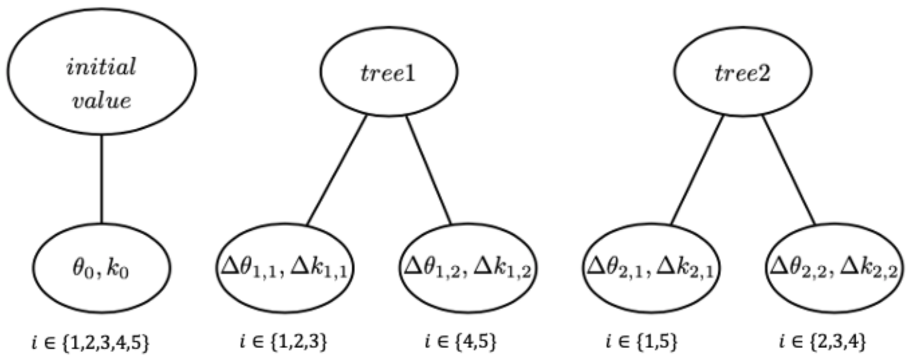


图 5 构建参数分布构建多棵概率树集成学习的示意图

- $\theta_{2,1} = \theta_0 + \Delta\theta_{1,1} + \Delta\theta_{2,1}$
- $\theta_{2,2} = \theta_0 + \Delta\theta_{1,1} + \Delta\theta_{2,2}$
- $\theta_{2,5} = \theta_0 + \Delta\theta_{1,2} + \Delta\theta_{2,1}$

设有 n 条样本，则第 m 轮的损失函数为：

$$L_m = -log(\prod_{i=1}^{n} p(y_i | k_{m-1,i} + \Delta k_{m,l}, \theta_{m-1,i} + \Delta\theta_{m,l})), \quad i \in S_{m,l}$$

$$k_{m-1,i} = k_0 + \sum_{j=1}^{m-1} \Delta k_{j,l}$$

$$\theta_{m-1,i} = \theta_0 + \sum_{j=1}^{m-1} \Delta\theta_{j,l}$$

$i \in S_{m,l}$ 表示样本 i 在第 m 轮被划分到第 l 个叶子节点。

寻找分裂点和求解 $\Delta\theta$、Δk 的目标是使得上式最小化。

$$\arg\min_{S,\Theta,K} L_m$$

$$L_m = -log(\prod_{i=1}^{n} p(y_i|k_{m-1,i} + \Delta k_{m,l}, \theta_{m-1,i} + \Delta\theta_{m,l})) = -\sum_{i=1}^{n} log(p(y_i|k_{m-1,i}+\Delta k_{m,l}, \theta_{m-1,i}+\Delta\theta_{m,l}))$$

设第 m 轮的节点 l 的样本为 $(x_i, y_i), i = \{1...s\}$。采用类似梯度下降的方法:

$$\frac{\partial L_m}{\partial k_{m-1,l}} = -\sum_{i=1}^{s} (log(\frac{y_i}{\theta_{m-1,l}}) + \psi(k_{m-1,l}))$$

$$\frac{\partial L_m}{\partial \theta_{m-1,l}} = -\sum_{i=1}^{s} (\frac{y_i}{\theta_{m-1,l}^2} - \frac{k_{m-1,l}}{\theta_{m-1,l}})$$

设学习率为 η_1, η_2，则

$$\Delta k_{m,l} = \eta_1 \frac{\partial L_m}{\partial k_{m-1,l}}$$

$$\Delta \theta_{m,l} = \eta_2 \frac{\partial L_m}{\partial \theta_{m-1,l}}$$

Gamma 分布拟合某个具体数据分布的收敛的动态过程如下:

图 6 Gamma 分布模式下 PBTree 对某一组数据的收敛过程

非参数模型

同样的，我们可以拓展到非参数分布的场景。根据某种策略（等频、等宽、指数间隔等方式）把目标 y 划分为划分为 B 个区间 $\{(-\infty, b_1), (b_1, b_2), (b_2, b_3), (b_B, +\infty)\}$。设 $p(b_j \leq y_i \leq b_{j+1}) = \frac{e^{k_{i,j}}}{\sum_{b=0}^{B} e^{k_{i,b}}}$ ，对于第　棵树的结点 l，设划分到这个节点的样本为 $(x_i, y_i), i = 1, 2, ..., s$。

我们可以基于 MLE 推导其损失函数和参数更新过程。

$$L_m = -\frac{1}{s} \sum_{i=1}^{s} log(p_i)$$

$$\frac{\partial L_{m,l}}{\partial k_{m-1,l,b}} = -\frac{1}{s} \sum_{i=1}^{s} \left(\delta(y_i, b) \frac{e^{k_{i,j}}}{\sum_{b=0}^{B} e^{k_{i,b}}} + (1 - \delta(y_i, b))(1 - \frac{e^{k_{i,j}}}{\sum_{b=0}^{B} e^{k_{i,b}}}) \right)$$

$$\delta(y_i, j) = \begin{cases} 1 & b_j \leq y_i \leq b_{j+1}, \\ 0 & otherwise \end{cases}$$

$$\Delta k_{m,l} = -\eta_1 \frac{\partial L_m}{\partial k_{m-1,l,b}}$$

还可以由 CRPS 推导损失函数，比较繁琐，在此不再展开。

效果分析

收敛性对比

在广告转化成本数据集上，采用相同的参数（树深度、迭代次数等），分别用 PBTree 的不同模式训练模型，并与 Xgboost 对比（xgboost 损失函数采用 tweedie loss，我们在之前的工作中验证过，此损失函数在成本预估场景下效果最好），按照点估计准确度衡量：

- bayesian 模式下初始效果较好，但是收敛曲线很快进入平台期，效果有天花板。
- nonparametric 模式下收敛速度和效果比 Xgboost 差，可能与这种模式下模型要学习的权重值较多有关。
- gauss 分布模式下收敛速度和最终效果也不理想，原因是广告成本分布与正太分布差异过大（由于大部分广告的优化目标为浅层目标，广告成本整体分布更接近 tweedie 分布）。
- Gamma 分布模式下的 PBTree 在收敛性上要好于 xgboost。

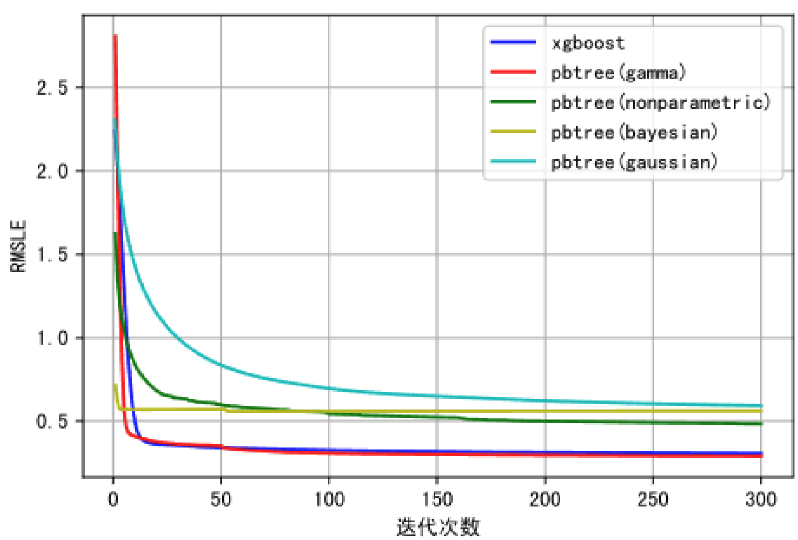
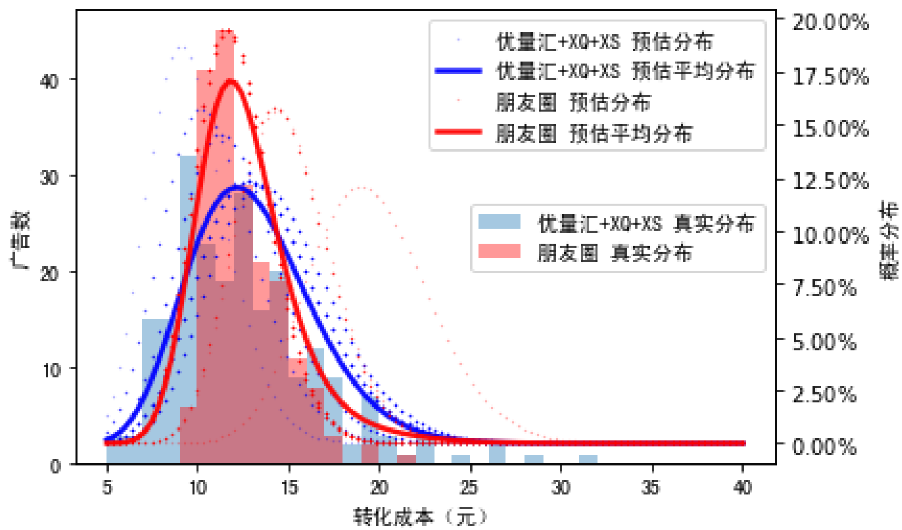


图 7 PBTree 的不同分布模式与 xgboost 的收敛速度对比

案例分析

回到我们在背景介绍部分提到的两个问题，pbtree 能比较好的拟合不同场景下的分



布形状。

图 8 PBTree 能够区分不同版位上的成本分布的分散程度

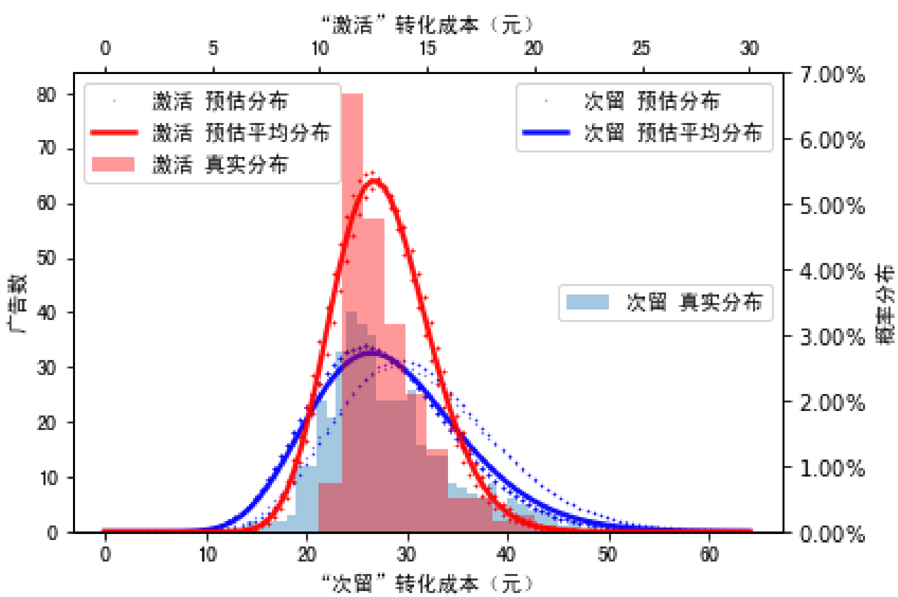


图 9 PBTree 能够区分转化链路不同位置的成本分布的分散程度

准确率指标

在测试集上，不同优化目标的预估准确率如下：| 优化目标 | 广告数 | XGB 准确率 |PBTree 准确率 | 提升幅度 | :— | :—: | —: | :—: | —: | | 非 OCPA | 16,922 |62.31%|68.80%|6.49%| | 关注 | 18,103 | 79.05% |79.37% |0.34% | | 点击 | 3,824 | 59.26% |66.89% |7.64% | | 激活 | 12,235 | 73.74% |75.42% |1.68% | | 注册 | 3,380 | 68.43% |74.56% |6.12% | | 下单 | 6,441 | 80.89% |81.11% |0.22% | | 商品详情页浏览 | 1,214 | 66.47% |73.15% |6.67%|

工程实现

- 使用纯 C++ 开发，只依赖 protobuf/glog/gperftools/c++boost 等少数第三方库，方便跨环境迁移部署和集成到线上服务。
- 支持 Gamma/Gauss/非参数分布等多种场景。
- 项目地址见：https://git.woa.com/paleylv/pbtree.git

附录

寻找分裂点的伪代码如下：

```
## 假设有F个特征，特征i的候选分裂点数量为C，当前结点有S条样本，
## S_L为左子节点的样本集，S_R为右子节点的样本集。
min_loss = inf
For i in (1 to F):
    For j in (1 to C):
        S_L = empty
        S_R = empty
        For k in (1 to S):
            if (x_ik < C_j):
                S_L <- k
            else:
                S_R <- k
        ## 计算梯度
        g_L = calculate_gradient(S_L)
        g_R = calculate_gradient(S_R)
        ## 计算损失函数
        loss_L = calculate_loss(S_L, g_L)
        loss_R = calculate_loss(S_R, g_R)
        loss = loss_L + loss_R
        if (loss < min_loss):
            best_split = i, j
```

参考文献

1. Zhuowen Tu, "Probabilistic boosting-tree: learning discriminative models for classification, recognition, and clustering," Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, 2005, pp. 1589-1596 Vol. 2, doi: 10.1109/ICCV.2005.194.
2. Duan, Tony, et al. "Ngboost: Natural gradient boosting for probabilistic prediction." *International Conference on Machine Learning* . PMLR, 2020.
3. Sprangers, Olivier, Sebastian Schelter, and Maarten de Rijke. "Probabilistic Gradient Boosting Machines for Large-Scale Probabilistic Regression." *arXiv preprint arXiv:2106.01682* (2021).
4. https://www.lokad.com/continuous-ranked-probability-score