【重读经典/译文】Facebook广告排序模型：LR+GBDT

原创 智能应用团队 贝壳智搜 2019-11-15

来自专辑

智能应用团队文章合集

> 在技术不断迭代更新的互联网行业，需要时刻保持学习的热忱。前车之鉴，后事之师，学习前人宝贵的经验，可以少走弯路，少给自己挖“坑”。这是Facebook在14年发表的一篇论文，论文中提出了经典（现在看来略显老套）的GBDT+LR算法。所谓温故而知新在重读这篇论文上体现的淋漓尽致，学生时代的初读拾得算法框架就浅尝辄止，工作之后再读发现许多与日常工作息息相关的内容：实验设计，特征系统构建，在线学习，样本抽样等等。这篇工程性很强的论文值得精读细品，技术会过期，但是严谨的工程思想永远保质。下面分享我对论文的翻译，如有纰漏，敬请指出。
>
> 长文警告！本文包含16261个字符。

摘要

在线广告平台的主要目的是吸引广告主为用户可预见的行为（比如点击某类广告）竞价和付费。不言而喻，点击预测模块是大部分在线广告中最为重要的一环。Facebook不仅拥有亿级的日活用户，还有超过百万的活跃广告主，预测点击是一个十分具有挑战性的机器学习任务。本文中提出了一种结合决策树和LR的算法模型，该模型比单独使用这两个算法的效果均提升了3%以上，这是对整个系统的有重大影响的性能提升。与此同时，本文还探索了一些基本参数如何影响系统的预测性能。**毫无疑问，重中之重是拥有合适的特征：捕捉用户历史信息和广告优势特性的特征**。一旦拥有合适的特征和合适的模型（也就是刚刚提到的决策树加上LR），其他的元素扮演的都是陪衬的角色（即使很小的提升在规模上对总体都是巨大的，也意味着提升的难度巨大）。选择最佳的保持数据新鲜度（data freshness）处理方法，规划学习率和数据采样都是那些对模型提升不明显的陪衬角色，**相比来说远没有增加强力特征或者选择合适的模型重要**。

1.简介

数字广告是一个数十亿美元的市场，并且这一数字每年都在剧烈地增长。在大部分的在线广告平台中，广告的分发是动态变化的，通常是根据反馈信息挖掘的用户兴趣量身定制。机器学习在计算用户对候选广告的点击期望中扮演着极其中心的角色，也是通过这种方式来加速在线广告市场的效率。

Google，Yahoo和Microsoft在计算广告领域都有相应的研究，有兴趣的可找找相关资料。在线的广告的竞价拍卖的效率取决于点击预测的准确性和校验。点击预测系统需要具有鲁棒性和自适应性，同时还要具备学习海量数据的能力。本文的主要目的是分享，在时刻关注鲁棒性和自适应性的情况下，践行这些标准对真实世界的数据进行实验的深刻见解。

在各大互联网公司支持的搜索广告项目中，利用用户输入的query词检索和它具有显式（explicitly）或隐式（implicitly）匹配关系的候选广告集（检索的过程也可以称之为召回，或者作为召回的一部分）。在Facebook的场景中广告并不和query关联，而是与区分目标人群和兴趣点定位息息相关。因此，在该场景下可给用户展示的广告数量远远大于搜索广告。

每当用户访问Facebook都会触发广告请求，为了处理每次请求中大量的广告候选集，首先需要构建一系列不断增加计算成本的级联分类器（cascade classifiers）。在本文中，讨论集中在级联分类器中最后的部分，也就是为最终候选集给出预测（点击概率）的点击预测模型。接下来，以第二章节概述实验设计作为开始。第三章节中，评估了不同的基于统计概率的线性分类器，并引出多种在线学习算法。在讨论线性分类器的同时，研究了特征转化（feature transforms）和数据新鲜度对分类器的影响。根据实践中经验教训的启发，尤其是围绕数据新鲜度和在线学习，Facebook的工程师们提出了一种模型结构，它融合了在线学习层，同时设计出多个相当简洁健壮的模型。第四章节详细描述了在线学习层的关键组件，也就是在线联接（online joiner），它是实验部分的基础部件，主要被用于生产实时训练数据流。

论文的最后，探讨了多种在处理海量训练数据中平衡内存和计算效率的方法。在第五章节，分享了多种在处理大规模应用时，预防潜在内存问题的实用技巧。接着在第六章深入探讨了如何权衡训练数据量和模型准确性。

2.实验设置

为了保证实验的严谨性和可控性，利用2013年第四季度中某一周构建训练数据集。同时，为了保证在不同条件下训练数据和测试数据的一致性，准备了与在线观测相似的离线

训练数据，并从这些离线数据中划分训练集和测试集来模拟在线训练和预测的数据流。本文中所有的实验使用了相同的训练/测试数据。

评价指标（Evaluation metrics）： 由于我们最应该关心各种因素对机器模型的影响，相比于真实的利润和财政收入，直接使用模型预测的准确性更加合理。在文中，主要使用**标准熵**（Normalized Entropy, NE）和**校准率**（Calibration）作为主要的评价指标。

标准熵，更准确的说是标准交叉熵，等于每次曝光（impression）的平均log loss除以所有曝光通过模型预测的潜在CTR（background click through rate）对应的log loss。Background CTR是对整个训练数据集的经验CTR（empirical CTR）的平均，也许将它称为标准化log loss更加合适，值越小说明模型预测的效果越好。进行标准化的主要原因是当background CTR越接近0或1对应越好的log loss，除以它的熵是为了使得标准熵对它不敏感。假设一个训练样本有N个样本，$y \in \{+1, -1\}$ 表示是否点击，p_i 表示模型预测的点击概率，p 表示经验CTR的平均数，标准熵的计算过程如下所示。

公式(1):

$$NE = \frac{-\frac{1}{N}\sum_{i=1}^{n}\left(\frac{1+y_i}{2}\log(p_i) + \frac{1-y_i}{2}\log(1-p_i)\right)}{-(p*\log(p) + (1-p)*\log(1-p))}$$

NE本质是用来计算相对信息增益（Relative Information Gain，RIG），两者的关系是 $RIG = 1 - NE$。

校准率 是预估平均CTR和经验CTR的比值，换句话说是期望点击数和真实点击数的比值。它是一个十分重要的指标，对预估CTR的校验是一个好的在线竞价和拍卖平台必不可少的步骤。越接近1代表模型的预测越准确。由于具体的数据是不可公布的，因此仅仅说明下在实验中使用了这一指标。值得一提的是，AUC（Area-Under-ROC）也是一个不错的评价指标。但是在真实环境中，我们希望**预测更加的准确，而不仅仅为了优化排序，从而避免潜在的劣质投放或者过度投放**。标准熵评估预测的好坏，同时隐式的反应了校准率。举个例子，如果一个模型给出的预测值是真实值的两倍，那么我们需要乘以0.5来做校准，相应的标准熵也会提升，但是AUC却保持一样。

3.预测模型结构

如图1所示，在这个章节主要介绍一个混合模型：**串联梯度提升树（GBDT）和稀疏线性分类器（LR）**。在3.1中，我们证明决策树可以对输入进行有效的**特征变换（transformations），大大的提升了线性分类器的精度**。3.2主要展示越新鲜的训练数

据如何带来越准确的预测结果，这也是激发我们利用在线学习方法训练线性分类器的原因。接着，3.3中针对两种不同的概率线性模型（LR和贝叶斯），对比了一系列的在线学习变量。

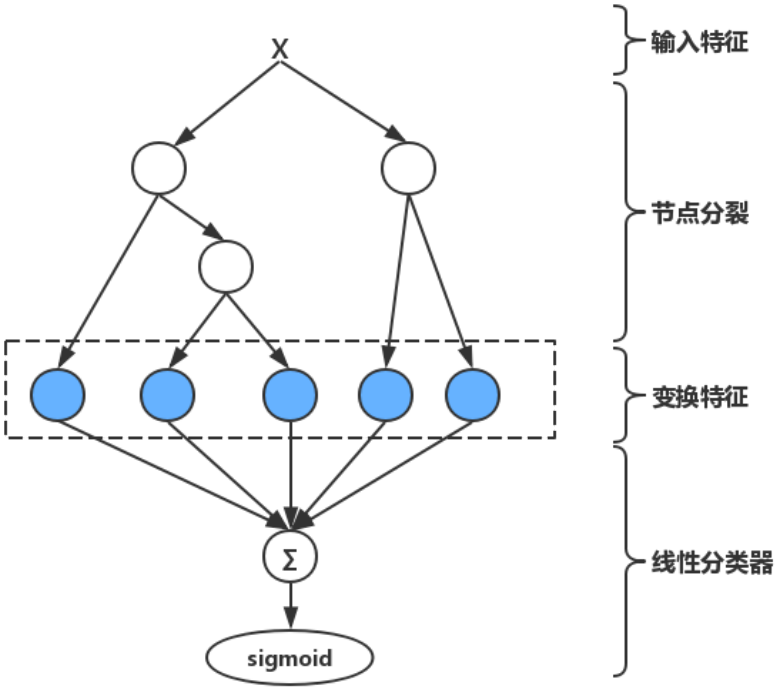

图1 GBDT+LR模型结构

研究在线学习方案时，我们主要评估了将随机梯度下降算法（Stochastic Gradient Descent，SGD ）应用于稀疏线性分类器。经过特征变换之后，每一个曝光的广告可以用向量 $\boldsymbol{x} = (e_{i_1}, \cdots, e_{i_n})$ 来表示，其中 e_i 代表第 i 个单位向量，$i_1, \cdots, i_n$ 是 n 个类目型输入特征（categorical input features）。在训练阶段，同样使用 $y \in \{+1, -1\}$ 标记广告是否被点击。对于一个拥有标签的曝光广告 $(\boldsymbol{x}, y)$，模型预测的分值可以使用

公式(2):

$$s(y, \boldsymbol{x}, \boldsymbol{w}) = y \cdot \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} = y \sum_{j=1}^{n} w_{j,i}$$

来表示，$\boldsymbol{w}$ 代表特征权重向量。

根据传统的贝叶斯概率回归在线学习（Bayesian online learning scheme for probit regression, BOPR），条件概率和先验概率可以表示为

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \Phi\left(\frac{s(y,\boldsymbol{x},\boldsymbol{w})}{\beta}\right)$$

$$p(\boldsymbol{w}) = \prod_{k=1}^{N} N\left(w_k;\mu_k,\sigma_k^2\right)$$

其中 $\Phi(t)$ 表示标准正太分布的分布函数，$N(t)$ 则代表密度函数。BOPR通过具有矩匹配（moment matching）的期望传播(expectation propagation)实现在线训练，训练结果是获取 $\boldsymbol{w}$ 的近似后验分布的均值和方差。BOPR的推导过程就是计算 $p(\boldsymbol{w}|y,\boldsymbol{x})$ ，然后通过因式分解将它映射到 $p(\boldsymbol{w})$ 的近似高斯空间（Gaussian approximation）。如此一来，更新算法只需要考虑概率不为0的 $\boldsymbol{x}$ 的均值和方差。下面的是简单的推导过程，有兴趣的可以再仔细研究一下。

公式(3):

$$\mu_{i_j} \leftarrow \mu_{i_j} + y \cdot \frac{\sigma_{i_j}^2}{\Sigma} \cdot v\left(\frac{s(y,\boldsymbol{x},\boldsymbol{\mu})}{\Sigma}\right)$$

公式(4):

$$\sigma_{i_j}^2 \leftarrow \sigma_{i_j}^2 \cdot \left[1 - \frac{\sigma_{i_j}^2}{\Sigma^2} \cdot w\left(\frac{s(y,\boldsymbol{x},\boldsymbol{\mu})}{\Sigma}\right)\right]$$

公式(5):

$$\Sigma^2 = \beta^2 + \sum_{j=1}^{n} \sigma_{i_j}^2$$

上面的出现的 v 和 w （注意和 $\boldsymbol{w}$ 区分）是惩罚函数，定义用 $v(t) := N(t)/\Phi(t)$ 和 $w(t) := v(t) \cdot [v(t) + t]$ 表示，该推导过程类似在 $\boldsymbol{\mu}$ 和 $\boldsymbol{\sigma}$ 上使用SGD。我们将BOPR类比成似然函数

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \text{sigmoid}(s(y,\boldsymbol{x},\boldsymbol{w}))$$

这里 $\text{sigmoid}(t) = \exp(t)/(1+\exp(t))$ ，也就是我们熟悉的LR。LR的推导主要在于使用对数似然求导，使用SGD算法根据渐变的步长更新参数。如公式（6）所示。公式(6):

$$w_{i_j} \leftarrow w_{i_j} + y \cdot \eta_{i_j} \cdot g(s(y,\boldsymbol{x},\boldsymbol{w}))$$

其中 g 代表非零元素的梯度，η_{i_j} 为步长。LR算法十分简单，梯度计算过程利用了不少求导的技巧，这里不再细说，有需要的可以查阅相关资料。值得一提的是，上面提到BOPR和SGD-based都是流式学习器（stream learners），需要一步一步拟合训练数据。

3.1 决策树特征变换

有两种简单的方法对输入特征进行特征变换，可以提高线性分类器的精度。其一，对连续型特征，很常用的非线性变换技巧是离散化（分箱，bin）变成类目特征（categorical feature）。线性分类器可以高效地学习这些分段特征。很重要的是如何让模型学习有用的分箱边界（bin boundaries），有很多信息最大化方法（information maximizing）可以做到这一点。

第二种简单但是有效的特征变换是特征组合。对于类目特征，最粗暴的组合方式是使用笛卡尔积，也就是将所有的原始特征都进行特征组合。很显然，不是所有的组合都是有效的，糟糕的是那些无效的组合不容易剔除掉。如果输入的特征是连续型特征，也可以使用特征组合，比如使用kd树进行联合分箱（joint binning）。

研究中我们发现梯度提升树可以通过刚刚提到的离散化和组合的方式，对特征进行十分有效且轻便的转化。将每棵树看做一个类目特征，用样本落在叶子节点的编号作为特征值，最终使用one-hot编码特征。举个例子，图1中的提升树有两个子树，两棵树分别包含三个和两个叶子节点。如果样本在两棵树中分别落在第二和第一个叶子节点，one-hot编码后特征可以用向量[0,1,0,1,0]表示，向量中前三个元素代表样本是否落在第一棵树上对应的叶子节点。文中使用的梯度提升树源自经典的Gradient Boosting Machine算法，并添加了L2正则（现在一般都使用XGBoost或者LightGBM）。在每一轮迭代训练中，会产生一棵新的树针对之前的树的残差（residual）进行建模。可以把**梯度提升树特征变换理解成有监督的特征编码，把原始的特征向量转化成0/1向量，从根节点到叶子节点的路径代表对非叶子节点特征的选择规则。将变换后的向量放入线性模型就是为了学习这些规则的权重**。梯度提升树通常采用批处理的方式训练模型。

我们通过实验来验证决策树转换特征作为输入对线性模型带来的影响，在实验中对比了两个LR模型，两个模型的区别为是否加入这些转换特征。同时，实验中也对比了单独使用梯度提升树预测的模型，具体的结果如表1所示。

表1: 模型标准熵对比

模型	***NE***
GBDT+LR	96.58%
LR	99.43%
GBDT	100%（对照）

从表1可以看出，使用了转化特征的LR模型对比单个GBDT模型NE指标相对下降了3.4%，这在真实环境中是十分惊人的提升。以供参考，一整套典型的特征工程也只能带来几十个百分点的NE指标下降，大部分的特征工程实验提升只有零点几个百分点。有趣的是，单独使用LR和树模型有着相识的准确率（LR稍微高一点），但是将两者结合之后有了质的飞跃。

3.2 数据新鲜度

点击预测系统通常存在于动态的环境中，需要处理的数据的分布随着时间而改变。接下来，主要研究下训练数据的实时性对预测效果的影响。为了做到这些，我们根据某一的数据训练模型，并用它来预测之后连续六天的数据。模型方面选择了GBDT和GBDT+LR两个模型，计算出NE指标如图2所示。

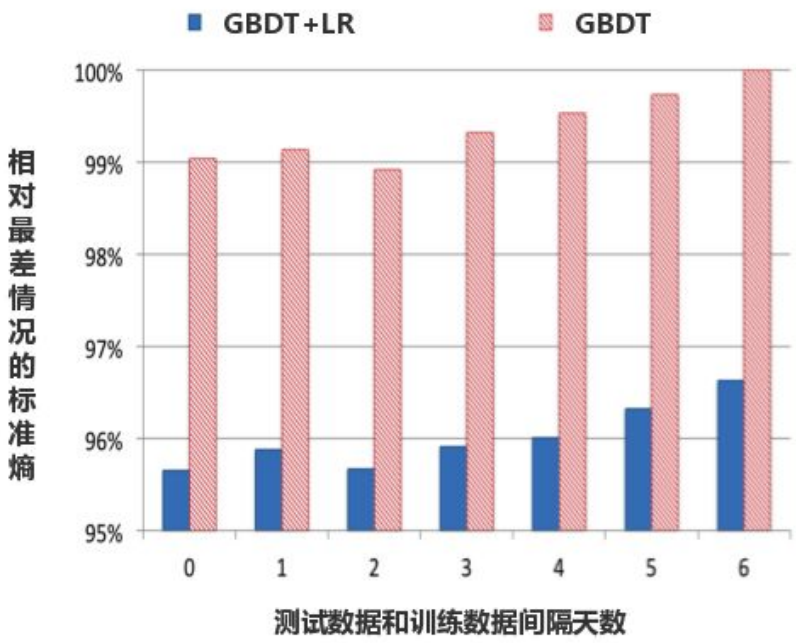

图2: 时间间隔对模型效果影响

从图中可以很明显的发现，随着测试集和训练集的时间间隔变大，模型的效果越来越差。从两个模型的表现上来看，按周更新模型相比于按天更新模型，NE指标上的损失在1%左右，这也表明天级更新模型是很有必要的。一个可行的方案是建立一个重复的每日任务去训练模型，最好是采用批处理的方式。实际工作中确定梯度提升树的更新周期受很多因素的影响，包括训练样本个数、树的数量、叶子节点的数量、CPU、内存等等。针对

数亿个样本训练一个数百棵树的boosting模型在单核CPU上可能需要超过一天的时间。在实际的生产环境中，通过多核机器并发处理等分布式可以将训练的过程压缩到几个小时。下一章中，我们将讲述另一种可行的替代方案，也就是**决策树仍然按天级或者一定的天数周期训练，而线性分类器（LR）可以利用在线学习的一些特性进行近实时训练。**

3.3 在线线性分类器

为了最大化数据的新鲜度，一种方案是直接使用曝光广告的实时点击信息在线训练线性分类器。稍后的第四章将会详细介绍如何产生实时的训练数据，这章的主要内容是评估，在LR在线学习中，常用的几种更新学习率的算法。然后，找出其中效果最好的与BOPR模型进行对比。根据公式（6），我们探索了以下学习率更新方法：

1. 自适应学习率（Per-coordinate learning rate）：对于特征 i 在第 t 轮迭代的学习率为

$$\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{j=1}^{t} \nabla_{j,i}^2}}$$

其中 α, β 是可以调节的参数（思路来着引文[8]）。

2. 单权重平方根学习率(Per-weight square root learning rate)：

$$\eta_{t,i} = \frac{\alpha}{\sqrt{n_{t,i}}}$$

其中 $n_{t,i}$ 代表直到第 t 轮为止特征 i 所有的训练样本数据。

3. 单权重学习率（Per-weight learning rate）：

$$\eta_{t,i} = \frac{\alpha}{n_{t,i}}$$

4. 全局学习率（Global learning rate）：

$$\eta_{t,i} = \frac{\alpha}{\sqrt{t}}$$

5. 常量学习率（Constant learning rate）：

$$\eta_{t,i} = \alpha$$

前三种方案针对每个特征设置不同的学习率，后两种方案所有特征共享同一学习率。利用网格搜索（grid search），各方案的最佳参数如表2所示。在训练过程中设置学习率的下限为0.00001，在相同的数据集上使用以上几种方案训练和测试LR模型，最终的实验结果如图3所示。

表2: 学习率参数

更新方法	参数
自适应学习率	
单权重平方根学习率	
单权重学习率	
全局学习率	
常量学习率	

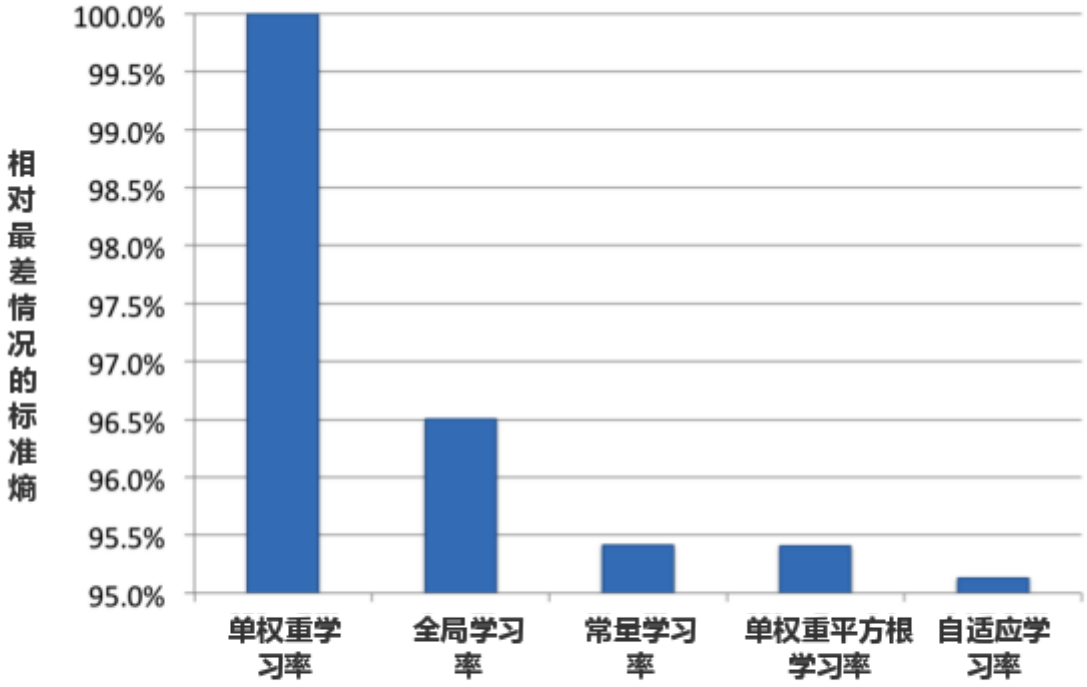

图3: 学习率更新方法实验

从上面的实验结果可以看出，使用 **自适应学习率的效果最好** ，相对最差的方案提升将近5%。结果同其他论文（引文[8]）的实验结果保持一致。使用常量学习率和单权重的方案也取得很好的效果，剩下的两种方案效果与其他方案差距太大。全局学习率落败的主要原因是没有考虑到特征之间训练样本的不平衡性。由于每个训练样本由不同的特征组成，有些特征出现的频率比其他特征高。在全局学习率的方案下，那些低频特征的学习率会很快地下降，特征权重很难收敛到最优点。虽然单权重学习率方案可以解决这个问题，但是因为所有特征的学习率都急速下降，导致它的效果很差。当模型达到局部最优点时，就会停止训练，这解释了为什么这种方案是最差的选择。

有趣的是，在BOPR模型中更新公式（3）中的平均数与基于SGD的LR模型更新自适应学习率非常相似。在BOPR中有效的学习率根据特定的维度而定，并且取决于该维度的后验方

差和之前模型预测的偏差。如表3所示，我们也对比了使用自适应学习率的LR模型和BOPR模型的预测性能。在实验中，两个模型都使用相同的数据集训练，然后评估预测下一天的准确性。和预想的一样，由于更新过程的相似性，两个模型在评价指标上的表现相差无几。

表3: 采用自适应学习率的在线模型对比

模型	***NE***
LR	100% （对照）
BOPR	99.82%

相较于BOPR，**LR的优点在于模型大小只有前者的一半**，主要是由于LR只需要计算特征的权重，而不是均值和方差。考虑到线上部署，越小的模型可以优化缓存局部性（cache locality），从而加速缓存命中（cache lookup）。对预测的计算成本而言，LR模型只需要对计算特征向量和权重向量取内积，而BOPR模型需要将特征向量分别与方差向量和均值向量做共内积运算。

BOPR比相比LR的优势在于，作为贝叶斯公式的应用，它**提供了点击概率的完整预测分布**。这可以用来计算预测分布的百分位数，而百分位数对探索和开发学习方案有帮助。

4.在线数据联接

之前的章节中提到新鲜的数据可用提升预测精度，还提出了在线训练的简单模型框架。这章的重点是介绍一个实验性的系统，它主要用于产生在线学习所需的实时训练数据。因为它的关键作用是在线上环境**联接标签（是否点击）和训练输入（广告曝光）**，我们称这个系统为“在线联接器”（**Online Joiner**）。相似的结构也被运用于流学习（stream learning），比如谷歌的广告系统。在线联接器输出实时训练数据流到Scribe（Facebook的开源的日志收集系统）。

通过点击可以很好的定义正样本，但是没有“不点击”按钮提供给用户。正因如此，**一般将没有产生点击的广告曝光作为负样本，但是要控制广告的停留时长（曝光时长）。必须注意的是，调节负样本停留时长的窗口大小需要非常谨慎**。使用太长的窗口会导致实时训练数据的延迟推送，还会增加因已曝光广告等待点击信号带来的内存开销。设置太短的窗口的问题是点击信息的丢失，其原因是相应的广告曝光被迅速刷新导致误标记为负样本。这一负面影响干扰了点击覆盖（click coverage），也就是点击在所有曝光的比重。因此，在线联接器系统必须权衡好**时效性和点击覆盖**。

不完整的点击覆盖意味着**实时训练集存在偏置**：预测的经验CTR比真实CTR要低。那是因为在时间窗口足够长的情况下，有部分被标记为无点击的曝光样本应为点击样本。然而我们发现，在真实环境中，通过设置合理的窗口大小很容易将这种偏置减少到零点几个百分点，此外这些很小的偏置是可以计算和校正的。设计在线数据联接是为了完成广告曝光和点击中**流对流（stream-to-stream）的数据整合**，其中最重要的组件是请求ID（request ID）。每当用户在Facebook上触发内容刷新动作，都会生成一个请求ID。图4简单介绍了从在线联接到在线学习的逻辑。

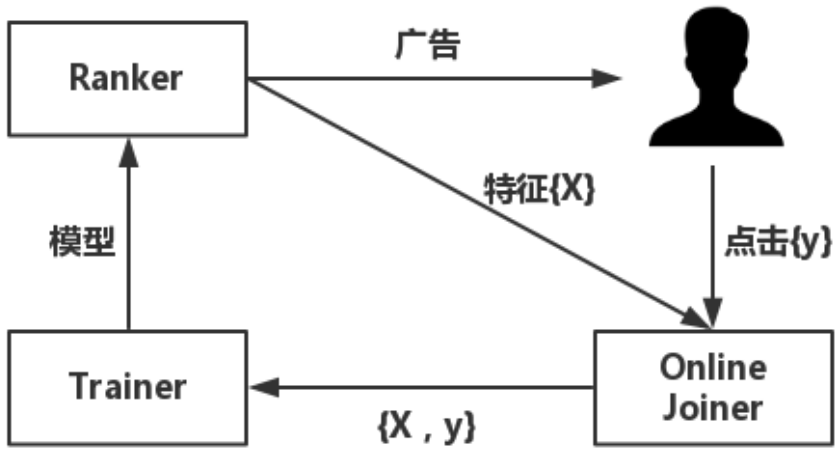

图4：在线学习数据/模型流

当用户访问Facebook时产生最开始的数据流，接着触发Ranker对广告候选集的排序请求。然后广告被回传推送给用户，同时每个曝光广告和关联的特征被添加到曝光流。如果用户点击了广告，点击信息还会被添加到点击流。为了实现流对流的数据联接，系统中使用了由先进先出（First-In- First-Out）策略的**哈希队列**（HashQueue）作为缓存窗口，还使用了**哈希表**以便对曝光标签进行快速随机访问。典型的哈希队列对key-value组成的键值对有三种操作：入队，出队和查找。举个例子，对一个元素（广告）进行入队操作，我们需要把它添加到队首，同时在哈希表中增加一个key-value对标记它和它在哈希队列的位置。只有当所有的联接窗口都过期的情况下，才将曝光的广告推到训练流。如果关联不上点击，曝光的广告将被标记为负样本。

在这个实现性的系统中，Trainer通过对训练数据流不断地学习，周期性的发布新的模型推送给Ranker。最终为机器学习模型构造了紧密的闭环，可以在短时间内捕捉和学习特征分布，以及对模型进行性能监控和实时调整。当实验实时训练数据生成系统的时候，我

们意识到建立保护机制的重要性，保护机制的作用是应对潜在腐蚀整个在线学习系统的异常现象。举个例子，当点击流因为基础数据设施的错误导致无法刷新，在线联接器会产生一些只能学习出预测值很小（甚至为零）的训练数据，紧接着在线分类器会错误地预估出很低的点击率。由于广告的曝光依赖模型的预测结果，很低的预测值会引导系统减少广告的曝光数量。异常检测在这种情况下就大有用处。比如当在线实时训练数据的分布急剧变化的时候，系统可以自动地切断在线Trainer和Online joiner的联接。

5.优化内存和延迟

5.1 提升树数量

在梯度提升树模型中，树的数量越多预测所需的时间越长。这一章节，我们研究树的数量对模型精度的影响。

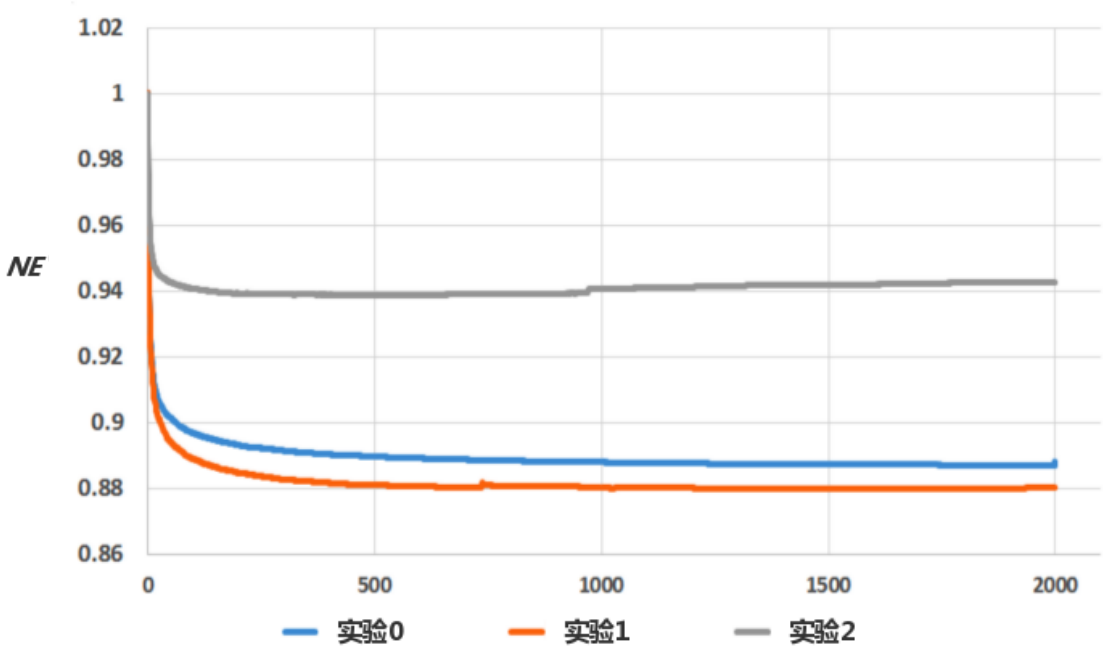

图5：提升树数量对精度的影响

如图5所示，我们使用同一天的数据分别训练了1-2000棵树组成的GDBT模型来预测下一天的数据，对每一棵树控制其叶子节点数小于12。同之前的一系列实验一样，使用标准熵作为评价函数。随着树的数量增加，NE不断下降。然而增加树的收益在不断下降，最后

1000树带来指标的变化不到0.1%，甚至编号为2的模型在1000棵树后出现精度回退。其原因是相比其他模型，模型2使用了少4倍的数据训练，导致过拟合现象。

5.2 提升树特征重要性

在模型构建过程中，特征数量是权衡预测准确性和计算性能的重要因素。为了更好地理解特征数量的影响，我们首先需要定义每个特征的重要性。我们使用统计量 *Boosting Feature Importance* 计算特征的重要性，它的思路是计算每个特征贡献的累计损失下降（the cumulative loss reduction）。在树中每个节点都会挑选一个最佳的特征作为分裂依据，从而最大化平方误差的下降。由于同一特征可用在多棵树，该统计量等于特定特征在所有树中误差下降之和。通常来说，少量的特征蕴藏着大部分可解释性信息，剩下的大部分特征贡献微乎其微（边际效益）。通过绘制特征数量及其重要性的示例图（图6），我们印证了相关的规律。

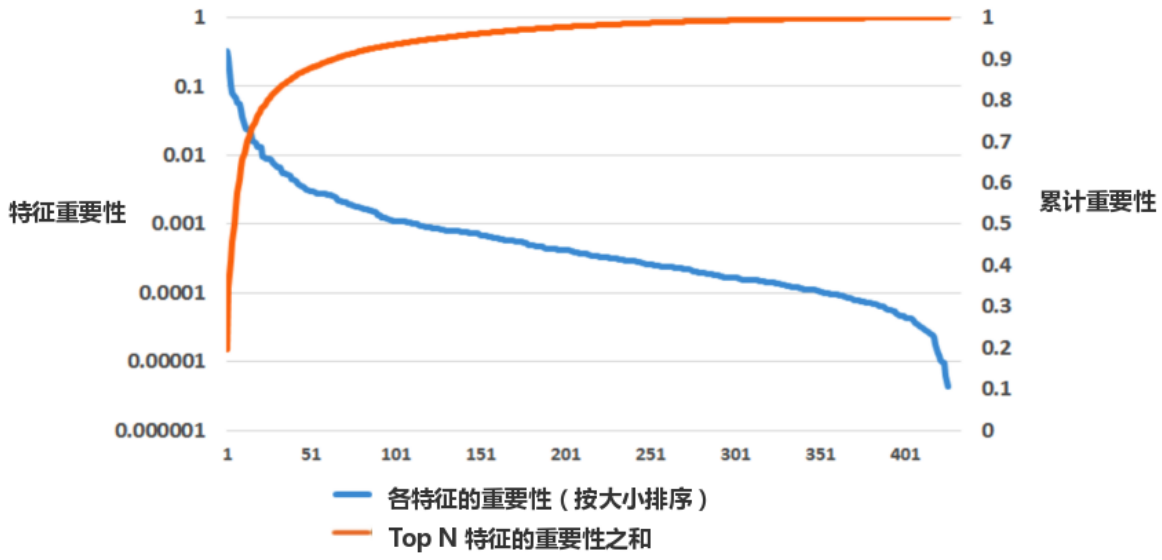

图6：提升树特征重要性

从图中的结果发现，排在前十的特征贡献了接近一半的“重要性”，相比最后300个特征贡献不到1%。根据这一发现，我们做了进一步实验，即只保留top 10，20，50等数量的特征用于模型训练，来探究特征数量如何影响性能。实验的结果如图7所示，特征数量的增加带来评价指标的收益不断减少。

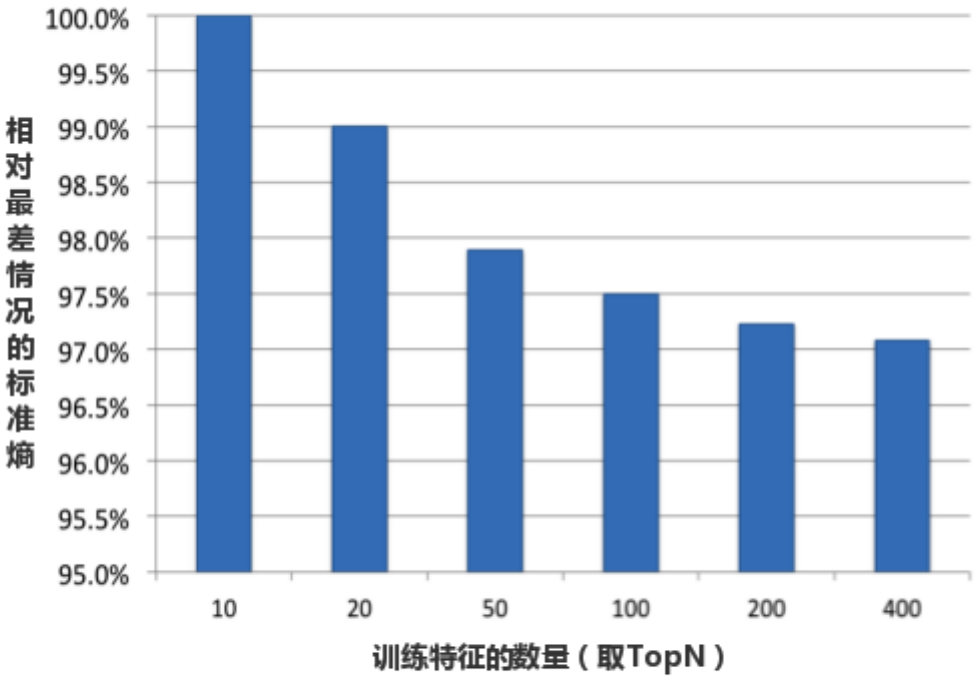

图7: 特征数量对模型性能的影响

接下来的章节，本文将探究历史特征和上下文特征的实用性。由于数据敏感性和公司政策，无法透露这些特征的细节。上下文特征主要包括时间，日期等等，历史特征可以是对广告的历史点击统计等等。

5.3 历史特征

在提升树模型中使用的特征可以归为两类：**上下文特征（contextual features）和历史特征（historical features）**。上下文特征的价值体现在描述了当前曝光广告的独有环境，比如用户使用的设备或者当前浏览的页面。相反，历史特征刻画了用户或者广告在历史的（行为）信息，例如某个广告在过去一周的点击率或者用户历史上的平均点击率。这一章中，我们主要研究这两类特征在系统中的表现。首先，我们检查了与这两类特征相关的重要性。为了达到这一目的，我们将所有特征按重要排序，然后计算出历史特征在前 k 个特征中的占比，计算的结果如图8所示。

图8: 历史特征重要性分析

从结果可以看出，相比上下文特征，历史特征提供了更多有用的信息，重要性前10都为历史特征。虽然在整个数据集中历史特征占全部特征大约75%左右，前20的特征中也只有两个上下文特征。为了更好的理解两类特征的差别，我们分类只利用某一类训练出两个模型，同时还训练了包含所有特征的模型。对比的结果如表4所示。

表4: 使用不同类型特征的模型实验

特征类型	***NE***
所有特征	95.65%
历史特征	96.32%
上下文特征	100%（对照）

根据表中的评价指标，我们可以再次确认，总体上历史特征扮演着更加重要的角色。在缺少历史特征的情况下，模型损失的精度达到4.5%。相反，去掉上下文特征只带来不到1%的损失。应该注意的是，上下文特征在应对冷启动（cold start）问题中起到了十分重要的作用。对于新用户和新广告，想要得到合理的预估点击率上下文特征是不可或缺的。

接下来，通过评估单独利用历史特征或上下文特征训练的模型在连续时间段的表现，来验证这两类特征对数据新鲜度的敏感程度。

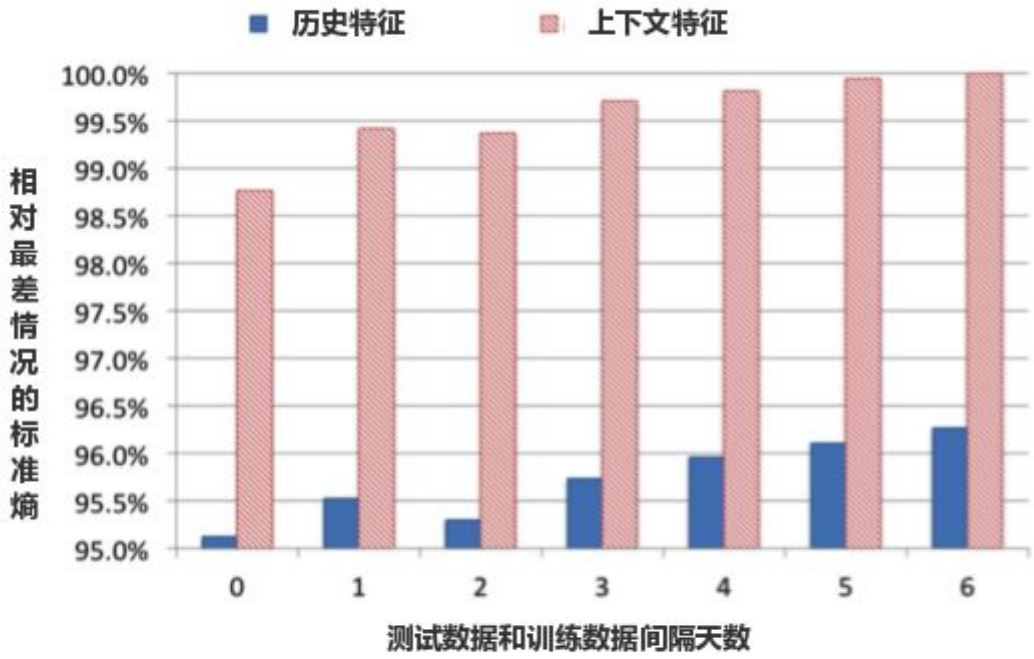

图9: 各类型特征对数据新鲜度的依赖

如图9所示，我们发现利用上下文特征的模型对数据新鲜的依赖比历史特征的重。这符合我们的直觉，因为历史特征描述的是用户的长期行为，相比于上下文特征更加稳定。

6. 处理大规模训练数据

Facebook一天广告曝光数据包含数量巨大的样本，具体的数值因为保密不方便透露，即使是很小的一部分有效数据也可能达到数亿规模。常用加速模型训练的方法之一是减少训练样本量。这章主要介绍两种欠采样方法：**均匀欠采样（uniform subsampling）和负降采样（negative down sampling）**。对不同的采样方法，分别训练包含600棵树的梯度提升树模型，并对比相应的评价指标。

6.1 均匀欠采样

由于**均匀欠采样容易实现**，所以它是处理训练集最常见的方法，同时这种方法得出的模型可以不加修改地作用在抽出的训练数据和未抽出的测试数据。在这一小节中，我们评估了大致指数增长的抽样率，选取的抽样率为 {0.0001, 0.01, 0.1, 0.5, 1}。基于相同的基本数据集，使用每一个抽样率得到样本来训练提升树。在评价指标上的表现如图10所示。

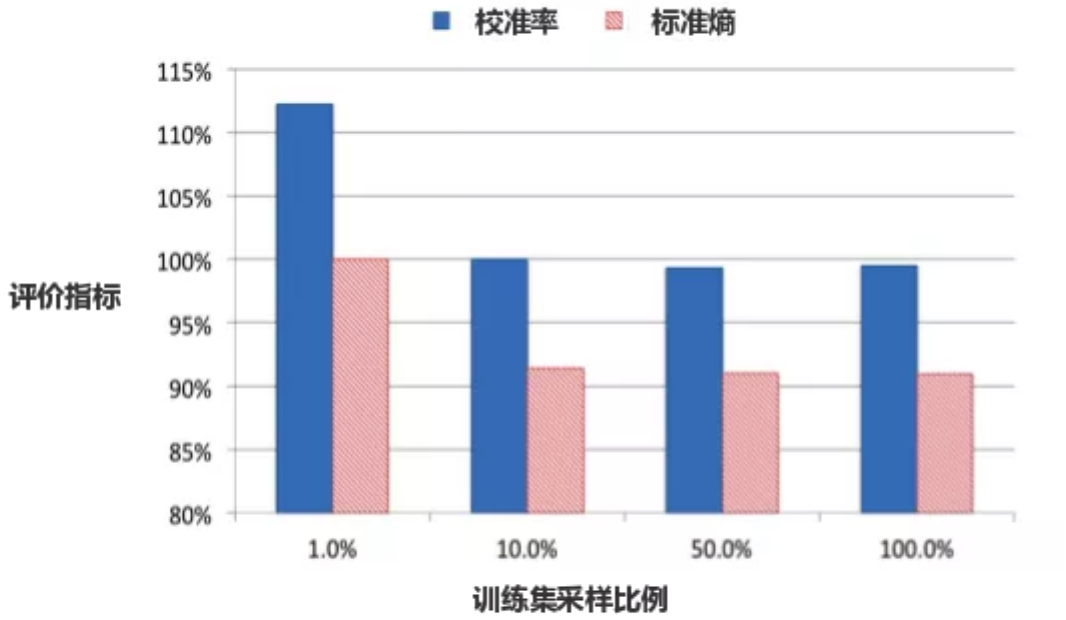

图10: 样本量对模型性能的影响

与我们的直觉相符，数据越多，性能越好。不仅如此，因增加抽样率而带来收益的增量在不断减小。相比于全部数据，使用10%的数据在评价指标上的损耗只有1%左右。

6.2 负降采样

很多学者都对标签不平衡问题进行了深入的研究，而且也被证实这类问题对模型性能有着十分重要的影响。这一小节，我们将调研使用负降采样方法去处理标签不平衡问题的效果（负降采样为保持正样本数量，只对负样本进行欠采样）。根据经验，我们尝试了不同的负降采样率来抽取样本进行实验，采样率的变化过程为{0.1, 0.01, 0.001, 0.0001}。最终的实验结果如图11所示。

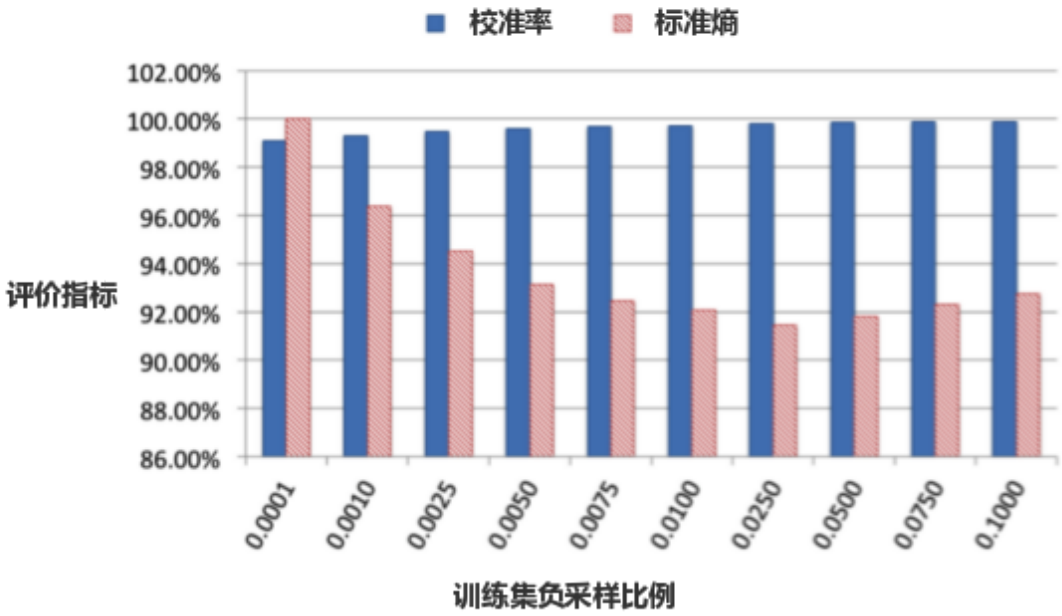

图11: 负降采样实验结果

从结果来看，负降采样率的取值大小对模型的性能影响很大，其表现最好的值为0.025。

6.3 模型校准

上面提到负降采样可以加速训练过程和提升模型性能，请注意，如果模型使用降采样，需要对降采样空间的预测值进行校准。举个例子，假设在采样之前的平均CTR为0.1%，如果我们使用了0.01的负降采样，那么模型预测CTR大致为10%（100倍）。这种情况下，我们需要使用公式

$$q = \frac{p}{p + (1-p)/w}$$

将对实时流预测值拉回到0.1%的范围，公式中 p 和 w 分别代表降采样空间的预测值和负降采样率。

7. 总结

上面的内容中，已经给出了在处理Facebook真实广告数据中实际经验。这些宝贵的经验，支撑着我们设计出在点击预测方向上效果可观的混合模型结构。

- 数据新鲜度举足轻重。至少天级的再训练是有必要的。论文中更近一步，讨论了多种在线学习方案，同时也分享了生成实时训练数据的系统。
- 利用梯度提升树进行特征变换对线性概率分类器提升明显。由此出发，设计出结合提升树和稀疏线性分类器的混合模型结构。
- 最佳在线学习方法：使用自适应学习率的**LR**。它的性能和BOPR基本持平，模型大小只有后者的一半，并且它的表现比使用其他学习率方案的LR模型更加优秀。

另外，我们描述了开发大规模机器学习应用时处理内存和时延问题的技巧。

- 文中介绍了如何权衡模型精度与提升树的数量。选择尽可能少的树对控制内存和优化计算十分有帮助。
- 梯度提升树通过计算特征重要性可以便捷地进行特征选择。帮助我们减少特征的数量，而不过多影响预测性能。
- 我们分析了使用历史特征及上下文特征对模型的影响。对于有历史记录的广告和用户，历史特征的表现远远好于上下文特征。

最后，我们讨论了各种降采样方案，其中负降采样是一种简单有效的均匀方法。

本文译者

> 赵群，2019 年 1 月毕业于哈尔滨工业大学智能计算研究中心，毕业后加入贝壳语言智能与搜索部，主要从事搜索排序优化工作。

你可能还想读

一镜到底：FM们的原理及在贝壳搜索的实践