

硕士研究生学位论文

|  |  |
| --- | --- |
| 题目： | 面向微博的在线广告点击 |
|  | 行为预测系统设计与实现 |

|  |  |
| --- | --- |
| 姓 名： | 孙超 |
| 学 号： | 1301221193 |
| 院 系： | 软件与微电子学院 |
| 专 业： | 软件工程 |
| 研究方向： | 互联网信息挖掘与处理 |
| 导师姓名： | 王海峰、刘宏志 |

二〇一 年 月

版权声明

任何收存和保管本论文各种版本的单位和个人，未经本论文作者同意，不得将本论文转借他人，亦不得随意复制、抄录、拍照或以任何方式传播。否则，引起有碍作者著作权之问题，将可能承担法律责任。

# 摘要

互联网产业的迅速发展正在重塑广告业的营销模式，传统的基于平面媒体的广告投放由于难以获取目标受众的准确信息，其投放是无目标的海量投放。近年来随着互联网产业的兴起以及与之伴随的大数据、人工智能等技术的迅速发展，深刻影响了广告行业，催生了计算广告这一新兴学科，计算广告是研究如何使用互联网技术解决广告产业的实际问题，即如何通过个性化推荐做到广告的精准投放，以达到节约成本、提高收益的目标。

广告点击行为预测是计算广告的重要环节，其研究的问题是如何依据用户的特征来判断用户是否会点击某一特定广告的概率。在这方面通常的做法是先用机器学习算法在已有用户的特征数据和点击行为记录上训练出一个模型，然后用训练出的模型根据新的用户特征数据预测该用户点击行为的概率。在这个过程中，用户特征的选取是一个关键步骤，选取特征组合的优劣直接影响最终预测的准确度，已有的工作大多采用用户的结构化特征如性别、年龄、职业、教育程度等固有属性，而忽视了用户的非结构化特征如社交行为、上网浏览记录等，这些非结构化特征往往能够真实反映用户的潜在兴趣偏好，对广告推荐有重要的参考价值。

本文利用用户的微博文章（包括发表、转发、点赞过的文章集合）这一非结构化特征，通过GBDT和LR融合的方案，将非结构化特征通过特征工程转换为新的结构化特征，和原有的结构化特征结合形成新的用户特征，在此之上使用LR模型训练和预测，并设计和实现了一个可商业化运营的线上分布式广告点击行为预测系统。实验和线上运行记录表明该系统在预测准确度上较之前的方法有明显提高，同时在响应时间和吞吐量上均达到预期设计目标。

关键词：广告，点击率，预测，微博

# ABSTRACT

Click-through rate estimation plays a key role in computational advertising. The usual approach is to use machine learning algorithm to train a model based on users' featured data, and then use this model to predict a new user's click behavior. In the past, most of works focus on selecting the right arguments and features of users.

KEY WORDS: Key word 1, Key word 2, Key word 3, ……

# 目录

[摘要 I](#_Toc500719413)

[ABSTRACT II](#_Toc500719414)

[目录 III](#_Toc500719415)

[图目录 VI](#_Toc500719416)

[第一章 绪论 1](#_Toc500719417)

[1.1 项目背景 1](#_Toc500719418)

[1.2 广告点击率预测研究发展现状 1](#_Toc500719419)

[1.3 本文的研究内容和方法 2](#_Toc500719420)

[1.4 本文内容组织结构 3](#_Toc500719421)

[第二章 相关技术 4](#_Toc500719422)

[2.1 逻辑回归(LR)模型 4](#_Toc500719423)

[2.1.1 逻辑函数 5](#_Toc500719424)

[2.1.2 模型调整 7](#_Toc500719425)

[2.2 在线学习算法FTRL 8](#_Toc500719426)

[2.2.1 背景 8](#_Toc500719427)

[2.2.2 相关算法 9](#_Toc500719428)

[2.2.3 FTRL算法实现 9](#_Toc500719429)

[2.2.4 算法优化 10](#_Toc500719430)

[2.3 迭代决策树 GBDT 11](#_Toc500719431)

[2.3.1 回归决策树 Regression Decision Tree 11](#_Toc500719432)

[2.3.2 梯度迭代（Gradient Boosting） 11](#_Toc500719433)

[2.3.3 GBDT工作过程 12](#_Toc500719434)

[2.3.4 Shrinkage 13](#_Toc500719435)

[2.4 GBDT和LR结合用于CTR预测 14](#_Toc500719436)

[2.5 在线广告点击预测系统构成 15](#_Toc500719437)

[2.5.1 HTTP服务器Ngnix 15](#_Toc500719438)

[2.5.2 分布式配置和集群管理工具ZooKeeper 16](#_Toc500719439)

[2.5.3 全文检索引擎Lucene 16](#_Toc500719440)

[2.5.4 跨语言通信接口Thrift 16](#_Toc500719441)

[2.5.5 数据高速公路Flume 17](#_Toc500719442)

[2.5.6 分布式数据处理平台Hadoop 17](#_Toc500719443)

[2.5.7 特征在线缓存Redis 18](#_Toc500719444)

[2.6 本章小结 18](#_Toc500719445)

[第三章 系统需求分析 19](#_Toc500719446)

[3.1 总体架构设计 19](#_Toc500719447)

[3.2 性能需求 20](#_Toc500719448)

[3.3 功能需求 20](#_Toc500719449)

[3.4 非功能需求 21](#_Toc500719450)

[3.5 本章小结 21](#_Toc500719451)

[第四章 系统设计与实现 23](#_Toc500719452)

[4.1 系统整体架构 23](#_Toc500719453)

[4.2 Service定义 23](#_Toc500719454)

[4.3 ApiServer的设计与实现 25](#_Toc500719455)

[4.3.1 Service管理模块 25](#_Toc500719456)

[4.3.2 请求处理模块 26](#_Toc500719457)

[4.3.3 算法服务管理模块 27](#_Toc500719458)

[4.3.4 负载均衡和异常处理 29](#_Toc500719459)

[4.4 算法服务的设计与实现 29](#_Toc500719460)

[4.5 本章小结 30](#_Toc500719461)

[第五章 业务流程 31](#_Toc500719462)

[5.1 文本处理 31](#_Toc500719463)

[5.1.1 文本分词 31](#_Toc500719464)

[5.1.2 构建词向量 32](#_Toc500719465)

[5.1.3 文章向量化表示 33](#_Toc500719466)

[5.2 特征工程 35](#_Toc500719467)

[5.2.1 模型训练 35](#_Toc500719468)

[5.2.2 构建新特征 37](#_Toc500719469)

[5.3 模型上线 38](#_Toc500719470)

[5.4 本章小结 39](#_Toc500719471)

[第六章 系统测试 40](#_Toc500719472)

[6.1 系统功能测试 40](#_Toc500719473)

[6.1.1 管理交互界面 40](#_Toc500719474)

[6.1.2 服务管理 40](#_Toc500719475)

[6.1.3 线下批处理作业服务 41](#_Toc500719476)

[6.1.4 线上请求处理服务 42](#_Toc500719477)

[6.1.5 数据更新服务 43](#_Toc500719478)

[6.2 性能测试 43](#_Toc500719479)

[6.3 准确率测试 44](#_Toc500719480)

[6.4 本章小结 45](#_Toc500719481)

[第六章 总结与展望 46](#_Toc500719482)

[6.1 总结 46](#_Toc500719483)

[6.2 展望 46](#_Toc500719484)

[参考文献 48](#_Toc500719485)

[致谢 50](#_Toc500719486)

[北京大学学位论文原创性声明和使用授权说明 51](#_Toc500719487)

# 图目录

图1.1 广告点击率预测步骤……………………………………………………………………………….2

图2.1 逻辑回归(LR)曲线示例……………………………………………………………………………4

图2.2 线性回归分析示例…………………………………………………………………………………5

图2.3 逻辑函数曲线………………………………………………………………………………………6

图2.4 投影梯度下降法示意图……………………………………………………………………………8

图2.5 FTRL算法…………………………………………………………………………………………..10

图2.6 梯度迭代树算法………………………………………………………………………..………….12

图2.7 传统回归决策树预测示例………………………………………………………………………...13

图2.8 GBDT训练过程示例………………………………………………………………………………13

图2.9 GBDT自动生成特征示意图………………………………………………………………………14

图2.10 GBDT和LR结合用于广告点击预测……………………………………………………………15

图3.1系统整体架构图…………………………………………………………………………………....19

图3.2 系统业务流程………………………………………………………………………………….......20

图3.3 系统算法服务架构………………………………………………………………………………...21

图4.1 在线请求处理流程………………………………………………………………………………...22

图4.2 Service工作原理………………………………………………………………………………...…22

图4.3 系统控制台界面查看service列表………………………………………………………………..23

图4.4 ApiServer类结构图………………………………………………………………………………...23

图4.5 Service相关类图…………………………………………………………………………………...25

图4.6 系统内部模块交互图……………………………………………………………………………...26

图4.7 算法服务管理类图………………………………………………………………………………...26

图4.8 算法服务信息类图……………………………………………………..……………………….…26

图4.9 新算法实例加入流程图…………………………………………………………………………...27

图4.10 已有算法实例突出流程图…………………………………………………………………….…27

图5.1 实验流程……………………………………………………………………………………...……32

图5.2 实验准确率评估…………………………………………………………………………….......…33

# 第一章 绪论

## 1.1 项目背景

当前我国互联网产业迅速发展，根据中国互联网络信息中心（CNNIC）的最新统计，截至2016年6月，我国网民规模达7.10亿，互联网普及率达到51.7%，与2015年底相比提高1.3个百分点，超过全球平均水平3.1个百分点，超过亚洲平均水平8.1个百分点[1]。与此同时，电子商务产业也发展迅速，十二五期间，中国的网络零售交易额规模跃居世界第一，网购在网民中的普及率高达55.7%，仅在2016年双11期间，天猫交易额就突破1200亿元，物流订单量超6.5亿[2]。

在此背景下催生了互联网广告这一新兴产业，互联网广告经历了从最初铺天盖地的横幅广告到人群及兴趣精准定向的搜索广告与推荐引擎，进而催生了计算广告学这一新兴研究课题，这一课题涉及大规模搜索和文本分析、信息获取、统计模型、机器学习、分类、优化以及微观经济学等诸多领域的知识[3]。相比于基于传统媒体如电视、广播、报纸海量而无特定目标的广告投放，以互联网为媒介的广告投放可以根据用户特征（主要包括用户个人信息和上网行为）做到个性化精准投放。从而为大大提高了广告收益，为企业节省了推广成本。其中，广告点击率预测(Click Through Rate简称CTR)是计算广告学的一个重要研究内容，是互联网企业广告竞价排名和流量变现业务的重要支撑。

## 1.2 广告点击率预测研究发展现状

目前广告点击行为预测研究主要集中于特征学习、数据特点和用户行为这3个方面，本文将以这3个方面分别介绍已有的相关工作。

特征学习方面，影响CTR预估准确率的特征很多，不是选用的特征越多准确率就会越高，如何科学的选取又用的特征是不少学者和工程师的研究方向。在这方面，文献[4]研究了原始查询和重写后查询之间的相关性与广告点击率之间的关系，该文章考察了一些特征如次序、长度差异、编辑距离等对广告点击率的影响；文献[5][6]将广告显示位置和广告查询相关性作为特征，同时根据相似广告来解决非常见广告和新广告的点击率预测问题；文献[7]综合运用协同过滤、贝叶斯网络和特征工程等模型来预测点击率；文献[8]提出用协同过滤和张量分解来提取用户个性化特征用于点击率预测；

数据特点方面，广告数据多为稀疏数据，文献[4]提出利用相似广告来预估目标广告的点击率；针对新广告的历史数据缺失问题，文献[9]提出“竞拍词-广告主”矩阵，用层次聚类的方法解决历史数据不充分的问题；文献[10]设计了用于稀疏广告和新广告的点击率预测模型，提出了基于层级结构的预估模型和基于Time-Spatial的预估模型；文献[11]提出了基于经验贝叶斯的自然数据分层和基于数据一致性的两种平滑计算方法对层级模型做了改进。

用户行为建模，用假设检验和贝叶斯网络刻画用户浏览场景，进而估计出用户点击广告的概率。文献[12]基于用户的浏览行为假设提出了级联模型，如果用户点击一个文档，若该文档不能满足用户的查询需求，则该用户很可能继续查看后续搜索结果并有点击行为； 文献[13]扩展级联模型到多次点击。

## 1.3 本文的研究内容和方法

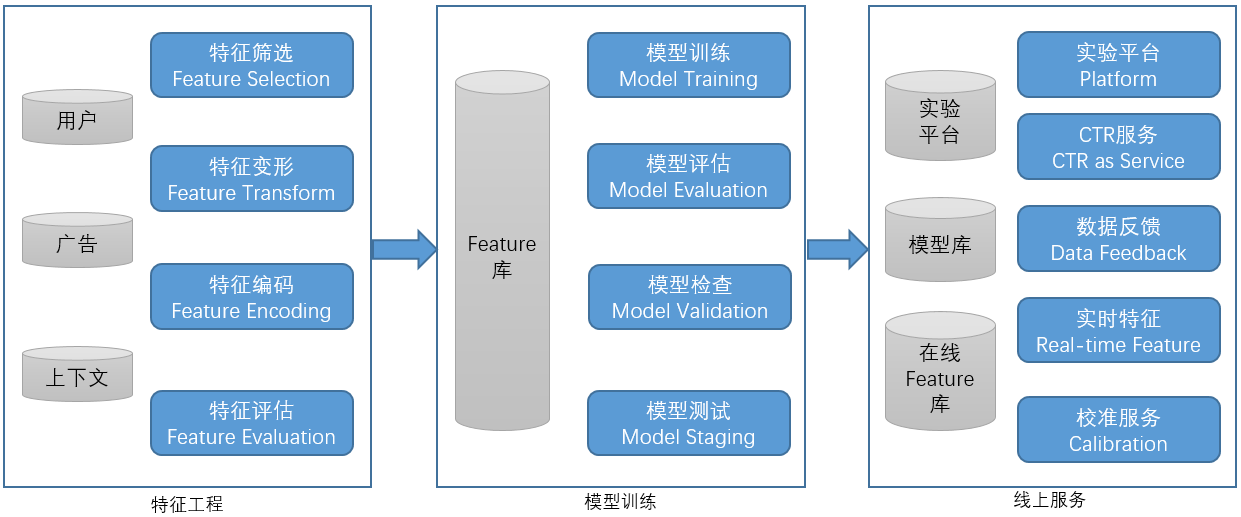
广告点击率预测过程可分为3个步骤：特征工程、魔性训练和线上服务。

图 1.1 广告点击率预测步骤

在这其中，特征工程是很重要的一个环节，特征选取的优劣对最终预测结果的准确率有很大的影响。对于用户而言，用户的固有信息如年龄、性别、地域、职业、教育程度等信息可作为用户特征，我们称之为结构化特征；而如用户的行为数据如浏览过的网页、购买过的物品、转发过的微博等非固有的，随时间和环境变化的特征，我们称之为非结构化特征。而什么样的特征适合做CTR预估，业界并没有统一的标准。往往靠工程科研人员结合具体情况和生活常识通过反复实验和以往经验来决定。以年龄特征为例：年轻人普遍喜欢运动类的广告，30岁左右的男人喜欢车，房子之类的广告，50岁以上的人喜欢保健品的广告。再以性别特征为例：化妆品的广告在女性上面的点击率就比在男性上面的点击率高很多，又如体育用品的广告在男性上面的点击率也比女性高，说明性别这个特征在化妆品和体育行业也是有预测能力的，经过多个行业的验证，就认为性别这个特征可以用了。在实际的使用中发现，性别这个特征比较有效，手机平台这个特征也比较有效，地域和年龄这两个特征有一定效果，但没有前两个那么明显，跟它们的使用方式可能有关。以上示例说明在CTR预估上特征选择往往根据具体应用场景的，主观的。

CTR预估中用的最多的是逻辑回归模型，又称LR(Logistic Regression)模型，是一种广义线性模型。LR使用了Logit变换将函数值映射到0~1区间，映射后的函数值就是CTR的预估值。LR模型的优点是效率高，容易并行化，处理上亿级别数据不是问题。但缺点是作为一种线性模型，其学习能力有限，需要大量特征工程预先分析出有效的特征、特征组合，从而去间接增强LR 的非线性学习能力[14]。上文中提到，特征选取往往依靠人工经验，耗时耗力且不一定能带来效果提升。如何自动发现有效的特征、特征组合，弥补人工经验不足，Facebook在2014年的文章介绍了通过GBDT(Gradient Boost Decision Tree)解决LR的特征组合问题[15]，随后Kaggle竞赛将此付诸实践[16]。在此之后GBDT与LR融合的方法开始引起了业界关注。

本文在结构化特征的基础之上，利用用户的微博文章(包括发表、转发、点赞的文章)，使用GBDT方法将用户的微博文章这一非结构化特征转换成结构化特征，与已有的结构化特征一起作为LR模型的训练数据训练出线上模型。实验证明，用了此方法后CTR预估准确率较之前有了明显的提升。此外，本文还详细介绍了基于此种方法的分布式在线实时广告点击率预测系统的设计和实现。

## 1.4 本文内容组织结构

第一章：绪论，介绍了广告点击率预测背景、发展现状和本文的主要工作，最后总结文章的组织结构。

第二章：相关技术，介绍了广告点击率预测的原理以及所使用的算法和工具，主要包括线上预测使用的逻辑回归(LR)模型和其算法实现FTRL，特征工程使用的GBDT算法；还有本工程所使用的各种开源工具。

第三章：系统需求分析，主要从商业应用的角度出发，分别从性能需求，功能需求和非功能需求三个方面介绍本系统设计时期望达到的目标。

第四章：系统设计与实现，详细介绍了本系统的各个组成模块以及它们之间的关系和交互过程，最后介绍了在线广告点击预测的业务流程。

第五章：系统测试，介绍了针对第三章系统需求中提出的各项指标的测试结果。

第六章：总结与展望，对本文工作进行总结以及介绍以后的改进方向。

# 第二章 相关技术

## 2.1 逻辑回归(LR)模型

逻辑回归(Logistic Regression)模型，是机器学习中的一种分类模型，由于算法的简单和高效，在社会学、生物统计学、临床、数量心理学、计量经济学、市场营销等领域得到广泛的应用。

图 2.1 逻辑回归(LR)曲线示例

逻辑回归算法最早由统计学家David Cox提出[17][18]，逻辑回归有二分类、多分类和顺序分类[19] 。二分类模型是应用最多的模型，其最终输出结果值只有两种，如“生”或“死”，“胜利”或“失败”，在本文的应用场景里就是用户“点击”或“未点击”广告，通常这两种结果值用“0”和“1”来表示[20] ，本文中“0”表示未点击，“1”表示点击。逻辑回归被用作预测基于相互独立变量基础上某一事件发生的几率，

和其他的回归分析方法类似，逻辑回归使用一个或多个预测变量，这些变量可以是连续的也可以是离散的。和一般线性回归不同的是，逻辑回归是用来预测相互依赖的二项变量而非连续值结果。而这违反了线性回归的假设前提，尤其是残差不服从正态分布。此外，线性回归可能对相互依赖的二项分布的变量做出无意义的预测。因此需要一种方法将二项分布值转换为连续实数值。要做逻辑回归首先用相互独立的变量表示不同等级事件发生的几率，然后计算出这些几率所占比率，然后再对这些计算出来的几率取对数，这一步称作对数变换，被看作是逻辑回归的连接函数。

对数函数的结果被用作线性回归分析预测，预测的结果通过自然对数的反函数（指数函数）转换为事件发生几率。因此，尽管观察到的逻辑回归预测结果是0-1值，但逻辑回归是在预测一个连续值得概率。在有些应用场合中，这个连续的概率值就是最终所需；在其他应用场景则需要一个或是或非的布尔值，通过定义一个阈值将连续值转换为离散的布尔值。

图 2.2 线性回归分析示例

逻辑回归可被看作是寻找一组最有的β参数以满足：

这里ε是逻辑斯谛分布误差。这里定义了一个隐变量 ，误差项ε未被测量，因此变量y’也是不可估的，因此被称作隐变量。与普通回归分析不同的是，β参数不可以直接用y和x的方程式表示，相反，需要用反复迭代过程寻找这样一个表达式，这正是逻辑回归训练算法要做的工作。

### 2.1.1 逻辑函数

逻辑函数是逻辑回归模型的重要组成部分，逻辑函数之所以有用是因为它的输入可以是任意实数，而输出总是在(0,1)区间上，因此比较适合用来表示概率。逻辑函数定义如下：

函数图形如图2.3所示：

图 2.3 逻辑函数曲线

假设t是x的线性函数，t可以用x表示为：

这样，逻辑函数就可以写成：

F(x)可被看作最终预测的概率值。

逻辑函数的反函数定义为：

两边取乘方得到：

其中g是对数函数，g(F(x))表明这个对数函数等价于线性回归表达式。F(x)表示因变量所代表事件发生的概率等于线性回归表达式的逻辑函数值。值得注意的是线性回归式的值域是全体实数，但是经过F(x)映射之后变为(0,1)；是线性回归方程式的截距；是回归系数。

因变量所表示的概率等同于线性回归式的指数函数。这恰好证明了对数算子是概率和线性回归式的连接函数。因为对数函数的值域是全体实数，因此它提供了足够的尺度使得在之基础之上进行线性回归。因此定义因变量的几率为：

对于连续的自变量，定义几率比(odds ratio)为：

这种指数关系解释了的作用：当x每增加1几率便乘以倍。如果因变量服从二项分布，那么几率比可定义为，这里a,b,c,d是2x2列联表的元素[21]。

### 2.1.2 模型调整

回归系数的预估一般用最大似然估计，最大似然估计也称最大概率估计，是用来估计一个概率模型参数的一种方法，其基本原理是：由已知实验结果（样本）出发，寻找一组满足这组样本分布的参数，把可能性最大的那组作为真实的参数。

设为随机变量，是多维参数向量，若随机变量相互独立且都服从，则可以得到概率函数为：

在固定时，以上函数式表示的概率；当已知时，它又变成的函数，记为：

称此函数为似然函数。似然函数值的大小意味着该样本值出现的可能性大小，由于已经得到样本值，可以估计似然函数值是比较大的，即出现上述样本可能性是高的，因此最大似然估计就是选择使达到最大值得那个作为真实模型的参数的估计。要寻找使最大的，只需在上对求导，令导函数等于0，为了求导方便，对公式2.10两边取对数得到：

对求导得到关于的k个方程组成的方程组：

解此方程组得到使函数最大的一组。

与带正态分布残差的线性回归不同的是，要寻找一个系数值的闭合表达式去最大化似然函数是不可能的，因此必须用多次迭代过程，如牛顿法[22]，该方法从一个尝试解开始在每次迭代过程中做修正看是否比之前有准确度提升直到准确度达到稳定值，称之为收敛[23]。

## 2.2 在线学习算法FTRL

FTRL(Follow The Regularized Leader)算法是由Google与2011年提出的在线学习算法[24]，在处理逻辑回归问题上有出色的性能，已在互联网行业尤其是CTR预测中得到广泛的应用。

### 2.2.1 背景

机器学习中常见的loss函数+正则化的结构风险最小化的优化问题（逻辑回归也是用的这种形式）可用两种形式描述，第一种是无约束优化形式，数学定义如下：

第二种是带约束项的凸优化：

当合理地选择g时，二者是等价的。而相关的算法都是建立在不同的描述基础之上。

首先看批量(batch)算法，批量算法中每次迭代用全体训练数据集计算全局参数，如全局梯度。优点是能够获得较好的精度，但缺点是处理大数据集时效率低，计算开销大，而且无法处理在线环境中的数据流。对于无约束优化形式，最常见的算法是全局梯度下降法：

每次迭代求一个目标函数的全局梯度，用于计算学习率。此外还有牛顿法（切线近似）、BFGS（割线拟牛顿）等方法也是基于无约束优化的；而对于带约束项的凸优化形式，有投影梯度下降法，每次迭代后迭代结果可能位于约束集合之外，取该迭代结果在约束凸集合上的投影作为新的迭代结果，如图2.4所示：

图 2.4 投影梯度下降法示意图

再来看在线算法，在线学习算法的特点是：每来一个训练样本，就用该样本产生的误差和梯度对模型进行一次修正，因此可以处理大数据量训练和在线训练。常用的算法有在线梯度下降（OGD）和随机梯度下降（SGD）等，本质思想是对单个数据的做梯度下降，因为每一步的方向并不是全局最优的，所以整体呈现出来的会是一个看似随机的下降路线。典型迭代公式如下：

这里使用混合正则化项：

其中是单点的未加和的loss函数梯度，投影集合C是约束空间。

梯度下降法的优点是精度高，但缺点主要有：简单的在线梯度下降很难产生真正稀疏的解，稀疏性在机器学习中很重要，尤其在工程应用中稀疏的特征会大大减少预测时的时间和空间复杂度；另外，梯度下降发对于不可微点的迭代会存在问题。

### 2.2.2 相关算法

上文提到稀疏性在机器学习中的重要性，解决此问题比较著名的方法有FOBOS(Forward-Backward Splitting method)[25]以及 RDA(Regularized dual averaging)[26]，FTRL综合了这两种算法的优点。

FOBOS是google和伯克利2009年的工作，方法与投影梯度法类似，不同之处在于将每一个数据的迭代过程分解成一个经验损失梯度下降迭代和一个最优化问题，其中最优化问题有两项：一是2范数项，不能离第一步loss损失迭代结果太远；二是正则化项，用来限定模型复杂度抑制过拟合和做稀疏化，从而保证了最终结果的稀疏性和理论上的完备。

RDA是微软2010年的工作，其主要特点是：用primal-dual algorithmic schema做非梯度下降；能够更好地在精度和稀疏性之间做平衡。

### 2.2.3 FTRL算法实现

Google于2013年给出了逻辑回归下的per-coordinate FTRL\_Proximal算法[27]，结合了FOBOS和RDA的优点，伪代码如图2.5所示：

|  |
| --- |
| **Algorithm:** Per-Coordinate FTRL-Proximal with L1 and L2 Regularization for Logistic Regression |
| **Input:** parameters  For t = 1 to T do  Receive feature vector and let  For compute:  Predict using the computed above  Observe label  For all do  End for  End for |

图 2.5 FTRL算法

其中per-coordinate表示 FTRL是对w每一维分开训练更新的，每一维使用的是不同的学习速率。与所有维度使用统一的学习速率相比，这种方法考虑了训练样本本身在不同特征上分布的不均匀性，如果包含w某一个维度特征的训练样本很少，那么该特征维度对应的训练速率可以独自保持比较大的值，没来一个包含该特征的样本，就可以在该样本的梯度上前进一大步，而不需要与其他特征维度的前进步调强行保持一致。

### 2.2.4 算法优化

在工程应用中会从不同的方面对算法做出一些优化来提高效率，主要优化措施有以下几种：

1). 内存节省，主要方法有：在线丢弃训练数据中很少出现的特征；对某一维度特征所来的训练样本，以一定的概率接受并更新模型；用bloom filter从概率上做某一特征出现k次才更新。

2). 浮点数重新编码，不用计算机中常用的32bit(C/C++里的float)或64bit(C/C++里的double)来存储浮点数，改用16bit编码，但是要注意处理rounding技术对regret带来的影响。

3). 训练若干相似model，对同一份训练数据序列，同时训练多个相似的model，这些model有各自独享的一些feature，也有一些共享的feature，因为有的特征维度可以是各个模型独享的，而有的各个模型共享的特征，可以用同样的数据训练。

4). Single Value Structure，多个model公用一个feature存储（例如放到cbase等），各个model都更新这个共有的feature结构，对于某一个model和它所训练的特征向量的某一维，直接计算迭代结果并与旧值做平均。

5). 训练数据重采样，在实际应用中CTR远小于50%，因此正样本更加有价值。通过对训练数据集进行重采样，可以大大减小训练数据集的大小。一般采样方法是正样本（至少有一个广告被点击）全部采样，负样本（完全没有广告被点击）按一定的比例r采样。但是直接在这种方法采样的数据集上进行训练会导致较大的预测偏差，解决方法是先采样减少负样本数目，在训练的时候再用权重弥补负样本，该权重直接乘到loss上面，从而梯度也会乘以这个权重：

## 2.3 迭代决策树 GBDT

GBDT（Gradient Boost Decision Tree）是一种迭代的决策树算法，该算法由多棵决策树组成，每次迭代都在减少残差的梯度方向新建立一颗决策树，迭代多少次就会生成多少颗决策树，所有树的结论累加起来做最终答案。它在被提出之初就和SVM一起被认为是泛化能力较强的算法。近些年更因为被用于搜索排序的机器学习模型而引起大家关注。GBDT主要由三个概念组成：Regression Decistion Tree（即DT)，Gradient Boosting（即GB)，Shrinkage (算法的一个重要演进分枝，目前大部分实现都是依据该算法）。

### 2.3.1 回归决策树 Regression Decision Tree

在机器学习和人工智能中，决策树可分为两大类，回归树和分类树。前者用于预测实数值，如明天的温度、用户的年龄、网页的相关程度；后者用于分类标签值，如晴天/阴天/雾/雨、用户性别、网页是否是垃圾页面。前者的结果加减是有意义的，后者则无意义。GBDT的核心在于累加所有树的结果作为最终结果，而分类树的结果是没办法累加的，所以GBDT中的树都是回归树，不是分类树。

### 2.3.2 梯度迭代（Gradient Boosting）

Boosting迭代，即通过迭代多棵树来共同决策。GBDT的核心就在于，每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量。比如A的真实年龄是18岁，但第一棵树的预测年龄是12岁，差了6岁，即残差为6岁。那么在第二棵树里我们把A的年龄设为6岁去学习，如果第二棵树真的能把A分到6岁的叶子节点，那累加两棵树的结论就是A的真实年龄；如果第二棵树的结论是5岁，则A仍然存在1岁的残差，第三棵树里A的年龄就变成1岁，继续学习。

Gradient Boosting算法如图2.6所示：

|  |
| --- |
| Algorithm: Gradient Tree Boosting |
| Initialize  For m = 1 to M  For i = 1,2,…,N compute:  Fit the regression tree to the targets giving terminal regions ,  For compute:  Update  Output |

图 2.6 梯度迭代树算法

### 2.3.3 GBDT工作过程

以年龄预测为例，假设训练集只有4个人A,B,C,D，他们的年龄分别是14,16,24,26。其中A,B分别是高一和高三学生；C,D分别是应届毕业生和工作两年的员工。若用一棵传统的回归决策树来训练，过程如图2.5所示：

图 2.7 传统回归决策树预测示例



如果使用GBDT来来训练，过程如下：（限定叶子节点做最多有两个，并且限定只学两棵树）

A: 14岁高一学生，购物较少，经常问学长问题；预测年龄A = 15 – 1 = 14

B: 16岁高三学生；购物较少，经常被学弟问问题；预测年龄B = 15 + 1 = 16

C: 24岁应届毕业生；购物较多，经常问师兄问题；预测年龄C = 25 – 1 = 24

D: 26岁工作两年员工；购物较多，经常被师弟问问题；预测年龄D = 25 + 1 = 26

购物金额<=1k

购物金额>1k

残差：A=-1, B=1

残差：C=-1, D=1

经常到百度知道提问

经常到百度知道回答

残差：A=0, C=0

残差：B=0, D=0

图 2.8 GBDT训练过程示例

### 2.3.4 Shrinkage

Shrinkage(缩减)的思想认为：每次走一小步逐渐逼近结果，要比每次迈一大步很快逼近结果的方式更容易避免过拟合。即它不完全信任每一个棵残差树，它认为每棵树只学到了结果的一小部分，累加的时候只累加一小部分，通过多学几棵树弥补不足。数学公式描述如下：

这里v是学习速率，一般都比较小，导致各个树的残差是渐变的而不是陡变的。

## 2.4 GBDT和LR结合用于CTR预测

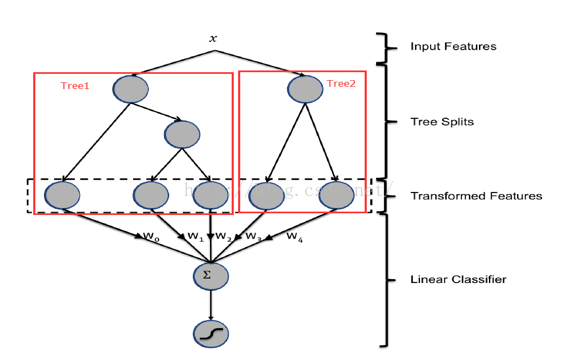
 如前文所述，逻辑回归（LR）模型是一种很好的分类器，很适合广告点击预估的应用场景，然而缺点是不易选择有效的特征。GBDT的思想使其具有天然优势可以发现多种有区分性的特征以及特征组合，决策树的路径可以直接作为LR输入特征使用，省去了人工寻找特征、特征组合的步骤。这种通过GBDT生成LR特征的方式（GBDT+LR）最早由Facebook提出[15]，应用效果不亚于人工特征选取。具体过程如图2.7所示：

图 2.9 GBDT自动生成特征示意图

上图中x为一条输入样本，遍历两棵树后，x样本分别落到两颗树的叶子节点上，每个叶子节点对应LR一维特征，通过遍历树就能得到该样本对应的所有LR特征。由于树的每条路径是通过最小化均方差等方法最终分割出来的有区分性路径，因此根据该路径得到的特征和特征组合都具有相对区分性。

具体应用到广告点击预测中，常用的做法是：对于曝光充分训练样本充足的广告，单独建树，称为ID类树，用于发掘对单个广告有区分度的特征，但对于曝光不充分样本不充足的长尾广告，不以细粒度的属性建树，此类树作为非ID类树，即便曝光少的广告、广告主仍可以通过此类树得到有区分性的特征和特征组合。当一条样本进来时遍历两类树到叶子节点，得到的特征作为LR的输入。当广告曝光不充分不足以训练树时，其它树恰好作为补充。过程如图2.8所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input x | | | | X是一条输入样本 |
|  | | | |
| ID类树 | | | 非ID类树 | 根据GBDT树转换原始特征 |
| Tree of ad id | Tree of advertiser id | Tree of ad cat id |
|  | | | | 特征输入LR进行训练或预测 |
| LR | | | |

图 2.10 GBDT和LR结合用于广告点击预测

通过GBDT映射得到的特征空间维度等于GBDT树的叶子节点数。一个叶子节点对应一种有区分性的特征或特征组合。针对不同的应用场景可以选择不同的GBDT建树方案，以达到最优效果。

## 2.5 在线广告点击预测系统构成

一个完整的可商业化运行的在线广告点击预测系统主要包括：1)请求接口：一般多用基于HTTP的web服务器；2)计算服务，为了处理大规模高并发高响应请求，一般使用分布式计算集群；3)数据存储服务，用于存储访问用户的特征信息以便更新模型用。一些大型企业一般会根据自身需求专门开发相关的系统，这样可以针对具体业务做专门的优化，而大多数公司，尤其是一些初创公司，通常选用开源的工具搭建自己的平台，以达到节省成本，快速开发部署，将主要精力投入到具体业务上来。本文采用后一种方法。下面分别介绍搭建在线广告点击预测系统所用到的工具。

### 2.5.1 HTTP服务器Ngnix

由于广告点击率预测系统有高并发、低延迟的性能要求，因此在选择前端处理HTTP请求的web服务器上，ngnix(<http://www.nginx.org>)成为这类系统的首选。Nginx是一款开源服务器软件，兼有HTTP服务器和反向代理服务器的功能。其主要特点在于高性能、高并发和低内存消耗，并且具有负载均衡、高速缓存、访问控制、带宽控制以及高效整合各种应用的能力，这些特性使得Nginx 非常适合计算广告这种并发很高的互联网服务。此外，Nginx 还提供了fastCGI这一与各种编程语言之间的通信接口，开发者可以很方便地将服务器的功能逻辑用fastCGI插件的形式实现，而无需关注响应HTTP 请求的细节。在广告点击预测系统中，用Nginx作为前端Web服务器，而将广告投放机的功能用C/C++语言实现成fastCGI插件，是一个开发成本较低、性能又很不错的方案。实际上，这一方案已经实现了一个基本的广告投放机，从事最简单的广告投放业务，而其他模块和功能则可以根据需求逐步开发[28]。

### 2.5.2 分布式配置和集群管理工具ZooKeeper

实际商业运营中的广告点击预测系统的流量惊人，单台服务器往往不能满足需要。在使用多台服务器的时候，会遇到很多诸如配置文件更新、集群上下线管理等分布式环境下的同步问题。ZooKeeper(http://zookeeper.apache.org)是解决这些问题非常有用的开源工具。

ZooKeeper 是为分布式应用建立更高层次的同步、配置管理、群组以及名称服务的通用工具。它的基础原理是参考文献[29]中的Paxos算法，在编程上ZooKeeper的设计很简单。所使用的数据模型非常类似于文件系统的目录树结构和Windows中注册表的结构，有名称、树节点、键/值对等，可以看作一个树形结构的数据库，可以分布在不同的机器上做名称管理。由于ZooKeeper并不传递计算数据而是传递节点的运行状态，所以运行负载很低。ZooKeeper另一个典型应用是：当某台服务器宕机或者新机器上线，Nginx的负载均衡方案需要及时作出调整。显然，如果用人工地维护，响应时间会很长，不可避免地会带来一些流量上的损失。利用ZooKeeper的Ephemeral类型节点可以很方便地实现此功能[30]。

### 2.5.3 全文检索引擎Lucene

大多数广告业务在初始运营阶段并不见得需要一个真正的倒排检索引擎，不过当广告业务开始面向长尾广告主，广告库规模较大时，采用“倒排检索”加“排序”这样的两段式决策过程是必要的。然而，实现一个功能全面、效率较高的倒排索引并不是一件简单的事，并且由于其与核心业务逻辑关系并不大，也可以用开源方案来实现。在开源工具中，Lucene(http://lucene.apache.org)是比较常用的基于Java的全文检索工具包。Lucene并不是一个完整的搜索引擎，但是针对计算广告系统的需要，它可以方便地实现全文索引和检索功能。Lucene能够为文本类型的数据建立索引，其主要功能是替文档中的每个关键词建立索引[31]。另外，Lucene还提供一组解读、过滤、分析文档，编排和使用索引的API。之所以选用Lucene除了它的高效和简单外，还因为它允许用户对其中的关键环节自定义功能逻辑。不过一些特殊的检索算法，如相关性检索，在Lucene中并不能直接支持，还需要在深入理解源代码的基础上改动或者另行开发。

### 2.5.4 跨语言通信接口Thrift

如前文所述，一个完整的在线广告点击预测系统由若干不同的模块组成，这些模块可能是用不同的编程语言开发的。为了方便在不同语言的模块之间实现调用接口，避免应用开发者过多地将精力放在底层通信上，跨语言通信协议就是用来解决这个问题。其中比较著名的有Apache Thrift，gRPC等。以本系统所使用的Thrift (http://thrift.apache.org)为例，它有自己的跨机器的通信框架，提供了一套代码生成工具，可以生成多种编程语言的通信过程代码，目前支持开发用的大多数编程语言如C++、Java、Python、PHP、Ruby、Erlang、Perl、Haskell、C#、Cocoa、JavaScript等。Thrift有一种描述对象和服务的界面定义语言IDL(Interface Definition Language)，它提供了一种网络协议，使用这些对象和服务定义的进程之间基于这种网络协议彼此进行通信，因此服务器端实现语言不会影响到客户端，这给系统各模块之间的通信提供了很大的便利[32]。此外，Thrift 还提供了实践中非常有用的版本兼容性功能，即服务器端能在不影响现有的客户端的情况下增加数据结构、字段、服务方法和函数参数。这一特性使得大型工程中模块间的依赖性大为减弱，也能够显著降低开发成本。

### 2.5.5 数据高速公路Flume

像广告点击实时预测这样的高并发系统，其在日常运行中产生的日志量也非常大。为了不影响效率应该避免对数据做单点的集中式读写，而是尽量应该让数据的处理形成环形的流动，即由数据高速公路将线上日志准实时地送至离线或在线处理平台，再将处理结果存放在缓存中供线上决策使用。在这样的架构中，一个分布式、高吞吐率的数据传送通道至关重要。

在这类数据传输工具中，Flume(http://flume.apache.org)是比较常用的开源解决方案之一。Flume是Cloudera提供的一个高可用的、高可靠的、分布式的海量日志采集、聚合和传输的系统，它支持在日志系统中定制各类数据发送方，用于分布式地收集和汇总日志数据。Flume提供了从控制台、RPC（thrift）、文本、Tail操作（UNIX tail）、日志系统（syslog，支持TCP和UDP两种模式）以及命令执行（exec）等数据源上收集数据的能力。同时，Flume还提供了对数据进行简单处理并输出到各种数据接收方的能力。如果广告投放机采用syslog方式记录投放、点击等日志，可以很方便地通过配置Flume将日志传送到Hadoop上。

### 2.5.6 分布式数据处理平台Hadoop

在线广告点击预测系统在日常运行中会生成大量的离线数据，离线数据处理部分需要一个能够存储和加工海量数据的基础设施，实际上这也是大多数大数据系统都需要的平台。在开源的解决方案中Hadoop（http://hadoop.apache.org）几乎是工业界的标准选择。Hadoop的核心架构主要包括分布式文件系统HDFS、MapReduce和HBase，其中HDFS是GFS(Google File System)[33]的开源实现，MapReduce是Google MapReduce[34]的开源实现，而HBase则是Google BigTable[35]的开源实现。

Hadoop MapReduce是一种分布式计算框架，它由两个部分组成：Map和Reduce。Map是将一个作业分解成多个任务，而Reduce是将分解后多任务处理的结果汇总起来。在程序设计中，一项工作往往可以被拆分成为多个任务，任务之间的关系可以分为两种：一种是不相关的任务，可以并行执行；另一种是任务之间有相互依赖，先后顺序不能够颠倒，这种任务是无法并行处理的。MapReduce适用于第一种类型，庞大的集群可以看作是硬件资源池，将任务并行拆分，然后交由每一个空闲硬件资源去处理，能够极大地提高计算效率，同时这种资源无关性对于计算集群的横向扩展提供了最好的设计保证。为了降低MapReduce编程的复杂性，人们还开发了Hive、Pig等开源工具产品，使用类似于SQL的脚本语言发起各种数据计算任务。

在广告系统中，Hadoop主要承担着离线数据的存储和计算需求，是系统进行大规模数据处理不可或缺的基础平台。无论是受众定向、点击率预测还是基础的报表生成，都需要在Hadoop上进行大规模的数据处理。

### 2.5.7 特征在线缓存Redis

在广告点击预测系统中，需要缓存点击率模型的参数和特征数据，以便用作模型定期更新。由于数据规模比较大，一般来说都无法直接存放在在线广告投放机的内存中，而是要用独立的缓存服务。在线用到的特征缓存有两个显著的特点，首先是往往只需要存储简单的键/值对，其次是大多数情形下需要支持高并发的随机读和不太频繁的批量写。在这样的需求下， Redis（http://www.redis.io）是比较合适的开源工具之一。

Redis是一种NoSQL数据库，它主要提供的是高性能的键值存储，采用的是内存数据集的方式。Redis的键值可以包括字符串、哈希、列表、集合和有序集合等数据类型，因此也被称作是一款数据结构服务器。Redis会周期性地把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了主从同步，具有非常快速的非阻塞首次同步、网络断开自动重连等功能。同时，Redis还具有其他一些特性，其中包括简单的check-and-set机制、pub/sub和配置设置等，使得它能够表现得更像高速缓存。Redis 还提供了丰富的客户端，支持现阶段流行的大多数编程语言，使用起来比较方便。在实际应用中当以批处理方式更新其中内容时，为了避免对线上高并发的读请求产生影响，一般采用多次写入的方案。

## 2.6 本章小结

本章主要介绍了广告点击行为预测的主要的算法逻辑回归模型与实现Ftrl；特征工程方法GBDT决策树；以及搭建一个可商业化运行平台用到的主要开源工具，这些算法和工具是构建本文的在线广告点击预测系统的基础设施。

# 第三章 系统需求分析

本项目的需求分析阶段提出了要设计并实现一个通用性高的分布式大数据处理平台能够运行各种主流的大数据和机器学习相关的算法，包括本文所针对的主要业务广告点击预测所涉及的文本处理，特征工程，和线性回归预测等任务。利用分布式系统的高并发多任务优势实现业务的高效处理。要求系统能够处理线上和线下两种类型的业务，线上业务主要是针对线上实时请求如实时预测在线用户的广告点击行为，线下业务主要是指定输入输出文件的批处理作业如用户浏览微博文章集的特征提取。

## 3.1 总体架构设计

系统总体架构采用分布式的总-分架构设计，由一台入口服务器接收任务请求，按照负载均衡的原则分发到计算服务器上处理，再由入口服务器返回处理结果，整体架构如图3.1所示：

图 3.1系统整体架构图

其中ApiServer即是入口服务器，其在后台运行一个http服务器用于接收线上webservice请求，前台运行一个类似Linux Shell的命令行界面用于提交批处理作业和查看系统运行状态，相当于用户交互界面。计算服务集群上运行不同的算法服务，ApiServer收到任务请求后根据请求内容分配到相应的计算服务器上执行请求，结果通过ApiServer返回给请求方。特征缓存和模型在线训练更新服务主要是针对线上线性回归预测的，本系统选用的线性回归预测Online-FTRL算法支持模型增量更新，因此需要将线上的用户请求行为数据缓存起来供数据模型的增量训练与更新。

## 3.2 性能需求

本项目的设计目标是实现一个可线上商用的广告点击预测系统，因此对性能的要求主要体现在线上请求处理的响应时间和吞吐量上，在需求调研阶段确定的线上业务(主要是线性回归预测)处理的平均请求响应时间控制在10ms以内，吞吐量指标为每秒至少可以处理4000个线上请求。

此外对线下批处理作业如文本处理(主要是分词、文章向量化等工作)，特征工程等业务流程要比之前的单机版耗时明显减少。

## 3.3 功能需求

本系统的主要功能是利用用户的微博文章数据做在线广告点击率预测。其中特征工程，即将用户的微博文章这一非结构化特征转换成结构化特征，是关键环节。业务流程如图3.2所示：

通过文本处理工具转换为数值向量

通过GBDT特征工程转换为结构化特征

和已有的结构化特征结合成新的特征

用逻辑回归(LR)模型进行训练或预测

图 3.2 系统业务流程

依据此业务流程，系统需要实现的功能主要有：a)文本处理，将用户的微博文章转换成数值向量，主要通过jieba工具实现，将在下一章中详细介绍；b)特征工程，用GBDT方法从用数值向量表示的文章集合中发现新特征；c)逻辑回归(LR)模型算法实现，在第二章已详细介绍。

## 3.4 非功能需求

除了支持业务的功能需求和性能需求之外，非功能需求主要包括易用性，可扩展性和准确度。

易用性方面：要求分布式集群上的所有算法通过APIServer对外提供两种服务接口，一是基于WebService的实时线上请求，主要用于提供在线服务的线上算法；二是命令行请求，用于批处理数据文件，主要用于线下算法如文本处理、分词、特征工程。

可扩展性方面：要求能在系统上做二次开发，要在系统上添加一个新的算法服务需要做的工作有：a)将算法实现按照系统规范进行改造封装，部署于算法集群上；b)在ApiServer上实现该算法的请求处理和任务分发功能，称其为算法的proxy。其架构如图3.3所示：



ApiServer

计算集群

任务分发

任务分发

任务分发



外部请求

图 3.3 系统算法服务架构

一个算法服务对应一个Proxy，它主要负责对收到的请求进行预处理如格式转换、判断合法，并将请求分发到算法集群上，要求proxy相对于ApiServer是独立的，添加新的Proxy无需修改或重新编译ApiServer的代码。

准确度方面：采用均方根误差(RMSE)测量预测结果的准确度，要求RMSE值较之前只使用用户已有结构化特征的方法降低至少一个数量级。

## 3.5 本章小结

本章从实际应用的角度介绍了系统需求，主要分三个方面，性能需求主要用分布式计算框架解决；功能需求要求能够实现整个业务流程所需的所有功能，非功能需求主要从准确度、高响应、高并发、易用性和可扩展性几个方面阐述。后续系统设计实现部分将以本章所介绍的需求为目标来实现。

# 第四章 系统设计与实现

## 4.1 系统整体架构

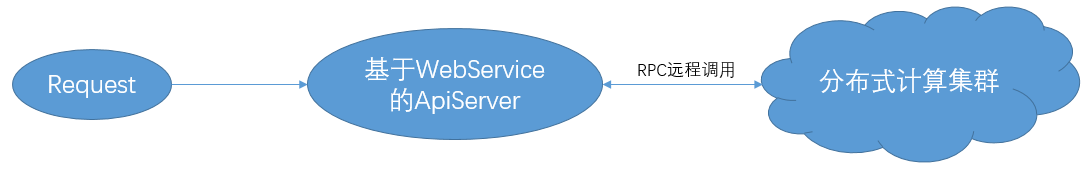
 为了满足性能上高并发、快响应的需求，系统整体采用一个请求处理/分发单元+多个计算单元组成的分布式计算集群的总-分架构。如图4.1所示：

图 4.1 在线请求处理流程

ApiServer是一个基于WebService的在线请求处理服务器，用于接收外部请求，并将收到的请求转换为RPC调用并按照负载均衡的原则分发到计算集群上的主机。计算集群完成请求的计算任务并将结果返回给ApiServer，ApiServer将收到的结果以WebService的形式（http+json）返回给请求者。

分布式计算集群上除了算法运行任务的计算服务器之外，还包括用于特征缓存的数据服务器和用于模型更新的服务器。特征缓存服务用作当收到广告点击预测请求时缓存请求数据中的用户特征和用户实际行为（点击或未点击），当缓存到一定数目时通过增量训练更新线上模型，从而能够实时反映用户兴趣行为的变化。

## 4.2 Service定义

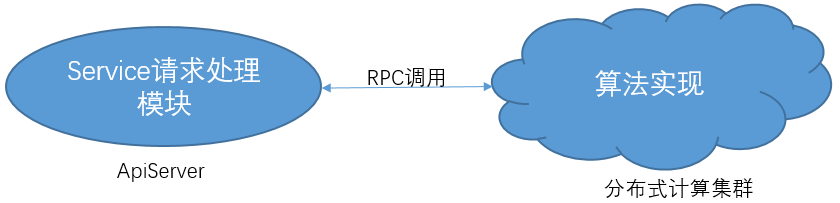
 在本系统中定义一个service为能够对外提供线上服务的算法运行实例，主要包括请求处理模块（ApiServer端运行）和算法实现模块（在分布式集群上运行），如图4.2所示：

图 4.2 Service工作原理示意图

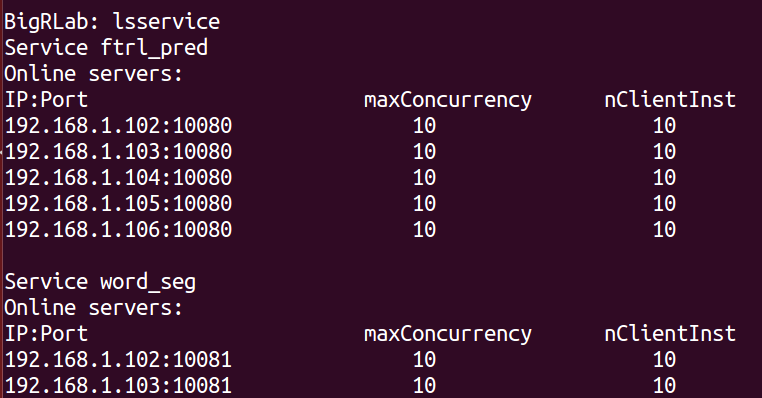
算法实现模块主要由运行算法的计算服务主机构成，根据业务需求可能也会有其他的支持服务如特征缓存数据库等。在系统控制台端可以通过lsservice命令查看当前线上运行的service列表，如图4.3所示：

图 4.3 系统控制台界面查看service列表

可以看到service的服务器列表，每个服务器属性主要有IP地址和端口号、支持的最大并发数（Thrift RPC Server参数）、以及当前的连接数（从ApiServer发起的连接）。

由于不同的业务处理请求的方式不同，当向系统中添加一个新的service时，除了要实现分布式集群上运行的算法模块之外，还要实现ApiServer上运行的请求处理模块，各个service请求处理模块由service的二次开发人员所编写，因此和ApiServer是松耦合的，支持运行时动态添加而不必重新编译ApiServer。



图 4.4 ApiServer类结构图

在Service基类中提供两个纯虚函数，分别是handleRequest和handleCommand，前者用来处理http在线请求，对外提供WebService服务，后者提供基于命令行的用户接口。每一个具体的service派生类都要根据自己特定的业务需求和处理方法提供这两个函数的功能实现。

此外，ApiServer端运行的Service请求处理模块还负责计算集群上服务器的管理，包括新服务器加入，已有服务器下线，以及服务器异常处理；还负责将服务请求按照一定的规则分发给计算服务集群上的服务器处理。系统默认的是基于负载均衡的分发方案，随机挑选一个服务器处理当前请求，各具体的Service派生类可根据自身业务需求改写这一方案。对于异常处理：如集群中的某一台服务器崩溃，则RPC调用会抛出异常或请求超时，此时系统会自动重新选择另一台服务器处理请求，并将出现问题的服务器从候选列表中删除。

## 4.3 ApiServer的设计与实现

ApiServer在本系统中主要负责接收外部业务处理请求并且请求分发到计算集群上处理，其主要包括3个模块，Service管理，请求处理，和算法服务集群管理，本节将分别对这3个模块展开介绍。

### 4.3.1 Service管理模块

在本系统中ApiServer主要任务是接收外部请求，并交给请求所指定的service处理。为了能够支持运行时动态加载，所有在ApiServer上运行的service请求处理模块都被要求编译成动态链接库。在ApiServer端通过loadlib命令加载。为了省去每次添加新的service都要手动加载的麻烦，可以将service动态链接库路径记录在autoload.conf文件中，每次ApiServer启动时会先依据此文件记载的内容加载相应的service请求处理动态链接库。由加载的service动态链接库创建service实例。

Service动态链接库和service实例都由ServiceManager类管理，该类采用单例模式，只有一个全局唯一实例。ServiceLib类是对service动态链接库的类抽象，每一个该类的对象拥有一个service动态链接库的handle，此外提供两个函数用于创建该service实例和获取库的名称；每一个service动态链接库都有一个全局唯一的名称，此外还必须提供用于创建实例和获取名称的函数接口以供ServiceLib调用。它们之间的关系如图4.5所示：

图 4.5 Service相关类图



### 4.3.2 请求处理模块

ApiServer接受的外部请求分为两种，一种是WebService请求，通过http协议传输json封装的字符信息，用于处理实时在线请求，每次请求只处理一条数据并立刻返回结果；另一种是终端输入的命令行，用于线下批量处理存放于文件中的请求数据，结果一般输出到指定的文件中。

对于WebService请求，请求发送者将处理该请求的service名称封装在URL中，一般格式为：

<http://apiserver_addr:port/service_name>

当ApiServer的http服务器模块收到该URL请求后，会通过service\_name字段得到请求服务的service名称，然后通过ServiceManager查找当前系统中有没有该service，若没有返回错误信息，若找到则将请求正文内容作为参数传给该service的handleRequest函数处理。处理结果同样以json格式的文本返回给请求者。

对于命令行格式的请求，其一般格式为：

service $service\_name $args

ApiServer的命令行处理模块从终端读入一行用户输入，若第一个单词为service则表示该命令请求要交给具体的service处理，否则都是系统命令；紧接着读入指定处理该请求的service名称service\_name，然后通过ServiceManager查找这个service，若找到就将剩余的args参数列表传给该service的handleCommand处理，若找不到指定的service就在终端上输出错误信息。

图 4.6 系统内部模块交互图



### 4.3.3 算法服务管理模块

在计算服务集群上，每一个算法实例启动时都要连接ApiServer（由启动参数指定ApiServer的地址和端口）以完成注册，这样集群上的该算法实例会成为对应service的计算集群的一员，成为一个作业分配的候选者。这个过程在ApiServer端是由算法服务管理模块AlgManager实现的。AlgManager本身也是一个Thrift RPC Server，计算集群上的算法实例通过Thrift RPC远程过程调用和ApiServer进行通信。AlgManager提供如下接口：



图 4.7 算法服务管理类图

当计算集群上算法实例启动时会将自己的信息填入到AlgSvrInfo结构中，然后以此为参数远程调用ApiServer上AlgManager模块的addServer完成注册；在该算法实例退出的时候相应地会调用rmServer从ApiServer上删除自己的注册信息。AlgSvrInfo记录了一个算法实例的关键信息，定义如下：



图 4.8 算法服务信息类图

其中addr和port分别定义了该算法实例所在集群主机的ip地址和所用的服务端口号；maxConcurrency指定了该算法实例所支的最大并发数，即同时能处理的最大请求数；serviceName指定了该算法实例属于哪一个service；libName指定了ApiServer所加载的service请求处理动态链接库的名称标识。

计算服务集群上一个新的算法实例加入系统过程如图4.9所示：



图 4.9 新算法实例加入流程图

当计算集群上一个算法实例退出时会远程调用AlgManager的rmServer，进而从对应service服务器列表中删除自己和已经建立的连接，如果删除后该service服务器列表为空，那么ServiceManager会删除该service的信息，后续到来的指定该service处理的请求都会返回错误。过程如图4.10所示：



图 4.10 已有算法实例突出流程图

### 4.3.4 负载均衡和异常处理

为了最大限度提高并发处理能力，对于多台主机构成的计算集群，负载均衡是必须要考虑的。ApiServer在service基类提供了一套默认的负载均衡方案，该方案为每个service创建一个空闲RPC连接池，当新的算法服务器加入的时候，根据其声明的maxConcurrency创建相应数目的空闲连接，然后放入空闲连接池中并执行random permutation的随机乱序操作，确保连续到来的请求尽可能地平均分配到不同的主机上。当一个请求到来的时候，该默认调度方案会从空闲连接池中取出第一个空闲连接，用它完成指定的RPC调用请求后再放回到空闲连接池中以供后续请求使用。每一个继承自service的具体派生类（动态链接库实现的）都可以定义自己的调度方案以覆盖默认方案，只需重写addServer函数并提供相关的数据结构支持即可。

异常处理方面，ApiServer端将异常分为用户异常和系统异常。用户异常是由算法定义的逻辑错误如非法输入，此类异常定义为InvalidRequest异常类，是Thrift Exception的子类。当ApiServer执行远程调用请求时收到这类异常会直接返回给调用者错误信息（终端输出或json文本，取决于请求类型是命令行还是WebService），并按正常流程将用过的RPC连接放回到空闲连接池中；系统异常主要是不可恢复的错误，如远程服务器意外停机，此时若在已经意外断开的连接上执行RPC调用会抛出Thrift Exception异常，对于这类异常系统不会将当前使用的RPC连接放回连接池。为了及时更新各线上算法计算服务器的状态，要求集群上的算法服务器每隔15秒钟向ApiServer发送一次信号，由AlgManager的informAlive远程函数实现，若连续1分钟未收到该主机的信号，系统会从相应service的服务器列表中将其删除。

## 4.4 算法服务的设计与实现

算法服务是对现有算法的改造封装以便和本系统兼容，算法服务实例运行于分布式计算集群上，通过Apache Thrift RPC远程调用和ApiServer端的service lib通信。算法服务本身是一个Thrift Server，按照自身业务逻辑提供远程调用接口供ApiServer端的service lib使用，计算集群上的算法服务实例和ApiServer端该算法对应的service lib实现共同构成了一个完整的service服务。

当算法实例启动时，要填写一个AlgSvrInfo对象，包括自己所在主机的ip地址，和本实例的服务端口号；ApiServer端service lib的名字；service名称；以及所支持的最大并发数，使用这个AlgSvrInfo对象去ApiServer注册，ApiServer会根据service名称找对应的service，若找到就把这个新到的实例加入到其服务器列表中，若找不到就用service lib名称找对应的动态链接库句柄，并用这个链接库创建一个新的service。

为了防止ApiServer意外崩溃，算法实例每隔一段时间（15秒）会启动一次注册流程，ApiServer收到注册请求若发现该实例已经注册，会直接返回实例已注册的错误，算法实例这边会忽略这个错误。这样当ApiServer崩溃重启后，之前所有在线的算法实例都会自动重新上线而不需要人工手动重启。

此外，为了实现模型自动更新，算法实例会自动保存启动时指定的模型数据文件路径，之后若要更新模型文件，只需把要更新的文件放在源文件的同一目录下，并在源文件名之后加上.update后缀，如源文件是model.dat，更新文件就是model.dat.update，算法实例会每隔一段时间检查是否有新的模型更新，若发现更新，就自动下线服务，然后将模型更新文件替换掉源文件，再初始化算法，到ApiServer注册，完成重新上线服务。

## 4.5 本章小结

本章详细介绍了面向微博的在线广告点击预测系统的设计和实现，首先介绍了系统整体架构和系统中的核心概念service，然后详细介绍了ApiServer各个组件和它们之间的关系以及与系统其他模块间交互流程，最后介绍了分布式计算集群上算法服务模块的设计和实现。

# 第五章 业务流程

广告点击预测业务流程可大致分为文本处理，特征工程，线上预估和模型更新几个步骤，本章将分别详细介绍。

## 5.1 文本处理

文本处理的目标是将用户发生行为的微博文章作为用户的新特征，即非结构化特征作为机器学习算法训练和预测的依据。文本处理主要包括文本分词，生成词向量，文章向量化表示等若干步骤。

### 5.1.1 文本分词

文本分词的目的是以中文词语为单位将原来的整篇文章分解，并去掉停用词，未后续的文章向量化作准备。这一步是用Jieba实现的[36]。Jieba是一款非常优秀的中文自然语言处理工具，其分词功能准确而快速，对未登录词有很好的处理方法，Jieba分词的基本步骤为：

1. 加载词典；
2. 从内存的词典中构建要分词句子的DAG（有向无环图）；
3. 对于词典中未收录词，使用HMM模型的Viterbi算法尝试分词处理；
4. 已收录词和未收录词全部分词完毕后，使用动态规划方法寻找DAG图的最大概率路径；
5. 输出分词结果为字符串数组。

Jieba使用的词典包括自带默认词典和用户自定义词典，还有用户自定义停用词集合，系统自带词典主要是依据做过词性标注的1998年全年人民日报预料库训练得到，在大多数应用场景结果是令人满意的，本项目中文本分词就使用其默认词典。语料库主要有两个功能，其一是对词语的频率进行统计，作为登录词使用，其二是对单字在词语中的出现位置进行统计，使用BMES模型进行统计，供后续的HMM模型Viterbi算法使用。为了快速地索引词典以加快分词性能，Jieba使用前缀数组来构造并存储词典，之前曾使用字典树trie，相比于trie树，前缀数组节省内存存储空间并且有更高的查询效率。

Jieba有三种分词模式供选择，分别是：1）全切分模式，即输出一个句子所有可能成词的字组合；2）精确模式，在所有的分词组合中，找出一个概率最佳的；3）搜索引擎模式，基于精确模式的再分词，将长词再次拆分为短词。举例说明对于输入语句:

“我来到北京清华大学，他去了网易杭研大厦”

全切分模式的分词结果为：

“我/来到/北京/清华/清华大学/华大/大学/他/去/了/网易/杭/研/大厦”

精确模式的分词结果为：

“我/来到/北京/清华大学/他/去/了/网易/杭研/大厦”

搜索引擎模式的分词结果为：

“我/来到/北京/清华/华大/大学/清华大学/他/去/了/网易/杭研/大厦”

可见对于文本分析任务，精确模式比较适合，此外精确模式和搜索引擎模式都能识别未登录词，如本例中的“杭研”，除了全切分模式未能识别出来之外，其他两种模式都使用了HMM模型和Viterbi算法都识别出来了。本项目中文本分词工作都使用Jieba的精确分词模式。

### 5.1.2 构建词向量

构建词向量的目的是将每一个词语用一个固定维度的浮点数组表示，为文章的向量化表示做基础。将文章中的词语转换成数值向量是由word2vec工具完成的。Word2vec 是 Google 在 2013 年推出的一款将词表征为实数值向量的开源高效工具[37]，word2vec通过人工神经网络算法训练出一个语言上下文模型，它能在一个输入文本语料的基础上生成一个向量空间，语料中的每一个单词都被映射到这个空间上的一个向量。在语料库中有相似上下文的词语，在向量空间中距离越近。

Word2vec用到CBOW(continuous bags-of-words)和skip-grams NLP模型，CBOW模型通过一个上下文窗口预测当前的单词，且上下文单词的顺序对预测结果没有影响；skip-grams模型会赋予邻近的上下文词汇更高的权重，相比于CBOW，skip-grams速度要慢一些，但对非频繁词有较好的效果。Word2vec使用命令行参数-cbow (1:使用cbow，0:使用skip-grams)来指定训练使用的模型，默认为1，本项目也遵循其默认选择cbow。

Word2vec训练采用层次化softmax函数法(hierarchical softmax)和负采样法(negative sampling)以使所求的条件似然函数值最大化。层次化softmax方法使用哈夫曼树以减少计算；而负采样法通过最小化负样本的似然函数值求解。层次化softmax能更好地处理非频繁词，而负采样法对频繁词和低维度向量效果更好。Wordvec使用命令行参数-hs(1:使用hierarchical softmax，0:使用negative sample)来指定，默认为0，即使用负采样法，本项目该选项也采纳其默认选择。

关于维度的选取，维度越高准确度也越高，但训练时间会变慢，当维度达到达到特定值之后，再继续增加维度收益不明显。Wordvec使用命令行参数-size指定，默认100，为提高准确度本项目中设定该参数为200。

上下文窗口大小设置：上下文窗口大小决定了给定词语前后有多少词语参与训练过程，依据作者建议，若使用CBOW设置为5，若使用skip-grams设置为10比较合适。Wordvec使用命令行参数-window指定，默认为5。本项目为了提高准确度，设定该参数为10。

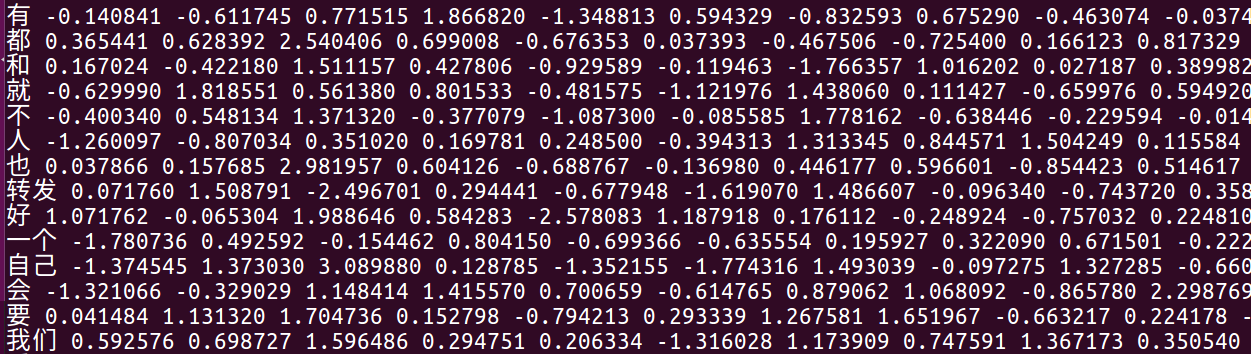
 Word2vec的输出结果又两种，一种是每个词语的向量表示，如图5.1所示：

图 5.1 word2vec输出的词向量结果

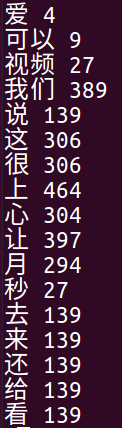
还有一种是对词向量空间进行聚类，通过参数-class指定类的个数，输出每个词所在的类id，如图5.2所示：

图 5.2 word2vec输出的词向量聚类结果

Class id相同的词语具有较高的相似度。这一步的目的是为接下来的文章向量化表示做数据准备，针对上述两种不同的输出结果，都有对应的文章向量化算法，这是下一小节要介绍的内容。

### 5.1.3 文章向量化表示

文章向量化的目的是将供人类阅读的文本转换成机器学习算法所能识别的数值向量。上一步中借助word2vec工具已经能够将词语用浮点数值向量表示，并且能在一定程度上反映现实世界，如意义相似的词语在向量空间中距离也是相近的，本节在其基础之上将整篇微博文章映射到一个向量空间，并且争取能做到真实反映现实世界，即内容相近的文章在向量空间中的距离也相近。

上一节中提到word2vec有两种输出形式，第一种是词向量，第二种是词的class ID，相对应的基于这两种数据基础的文章向量化方法也有两种，第一种是基于词向量的算法，将文章中的每个词向量相加再求平均值，算法如下：

|  |
| --- |
| **Algorithm**: 基于词向量的文章向量化 |
| **Input**: 由word2vec生成的“词语-数值向量对照表”wordVecTable;  要转换成数值向量的分过词的文章article，一般用string数组表示  **Output**: article的数值向量vector  Initialize vector[N] = {0,0,…0}, N是word2vec中指定的词向量的长度,  count = 0 用于统计article中在wordTable中词的个数  For word in article do  if wordVecTable.contains(word) do  vector += wordVecTable[word] 将word的向量累加到所求的vector上  count = count + 1  End if  End for  再对vector上每一个元素求平均值  If count > 0 do  vector = vector / count  End if  Return vector |

图 5.3 基于词向量的文章向量化算法

第二种文章向量化算法是基于文章中词汇的class id分布，算法描述如下：

|  |
| --- |
| **Algorithm**: 基于词class id的文章向量化 |
| **Input**: 由word2vec生成的“词语-class id对照表”wordIdTable;  要转换成数值向量的分过词的文章article，一般用string数组表示  **Output**: article的数值向量vector  Initialize vector[N] = {0,…0} N是word2vec中参数-class指定的class数量,  max = 0 存储最大的id出现次数  wordSet = unique(article) wordSet存储文章article中词的集合，不重复统计  For word in wordSet do  If wordIdTable.contains(word) do  index = wordIdTable[word] 将当前词语word的class id存入index  vector[index] = vector[index] + 1  if vector[index] > max, then  max = vector[index]  End if  End for  再将vector归一化  If max > 0 do  vector = vector / max  End if  Return vector |

图 5.4 基于词class id的文章向量化算法

通过实验对比，第二种方法效果更好一些，其采用的特征向量能更好地表示不同文章间相似程度。本项目文章向量化部分主要使用第二种方法。

## 5.2 特征工程

特征工程的目的是在原有用户结构化特征基础上增加新的特征，以更准确地反映用户的兴趣偏好，从而提高预测准确率。前文中提到，如何选取用户特征，以往通常是依靠人工工作，通过大量反复的实验和经验获得。本项目是以用户的微博兴趣文章为基础通过迭代决策树GBDT自动构建新特征。

本项目选用的GBDT实现工具为xgboost[39]，xgboost全称是eXtreme Gradient Boosting，是Gradient Boosting Machine的一个C++实现，其目标是提供一个可扩展、可移植、分布式的Gradient Boosting算法库；它支持常见的分布式框架如Hadoop、Spark；并且提供多种语言的编程接口如C++、Java、Python；具有很高的运行效率和准确度。

用xgboost实现特征工程包括模型训练和特征提取两个步骤，下文将分别予以介绍。

### 5.2.1 模型训练

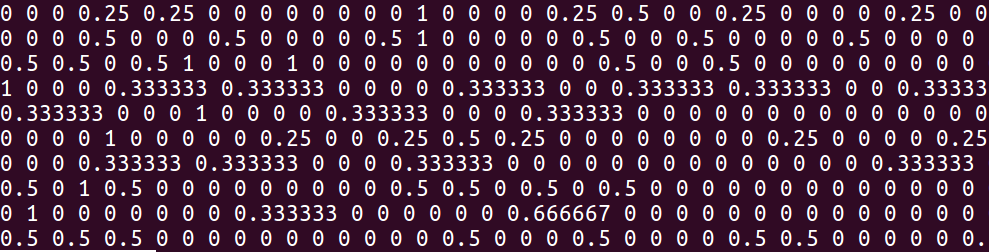
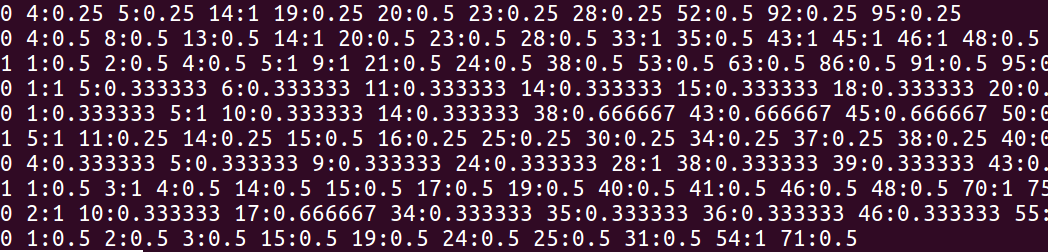
 在上一节的文本处理中，我们已经可以将一篇文章用一个数值向量表示，现在我们要用这些文章向量结合期望输出作为xgboost的训练样本来训练出一个迭代决策树模型。首先要做格式处理，文章向量化后的结果如下图所示：

图 5.5 文章的向量化表示(使用class id方法)

其中的每一行代表了一个用户感兴趣的微博文章集合，加上期望输出（即用户是否点击了广告）形成xgboost的训练数据集如下图所示：

图 5.6 用于特征工程的xgboost训练数据集

其中最左侧一列是期望输出，表示用户是否点击了广告，随后的是文章向量，格式为index:value，index从1开始。

Xgboost训练过程参数设置如下：

booster = gbtree

objective = binary:logistic

eta = 1.0

gamma = 1.0

min\_child\_weight = 1

max\_depth = 10

num\_round = 5

图 5.7 特征工程中xgboost训练参数设置

其中max\_depth是生成决策树的最大深度，该值越大，决策树上的叶子结点就越多，生成新特征的维度也就越大。用以上参数设置和训练数据集训练出模型FeatureModel，模型内容是二进制存储的一组迭代决策树，可以用xgboost的dump功能将其转换为可读文本，内容如下图：

booster[0]:

0:[f29<-9.53674e-07] yes=1,no=2,missing=1

1:[f56<-9.53674e-07] yes=3,no=4,missing=3

3:[f60<-9.53674e-07] yes=7,no=8,missing=7

7:[f23<-9.53674e-07] yes=13,no=14,missing=13

13:[f24<-9.53674e-07] yes=19,no=20,missing=19

19:leaf=1.99735

20:leaf=-1.8

14:leaf=-1.80952

8:leaf=-1.95062

4:[f21<-9.53674e-07] yes=9,no=10,missing=9

9:leaf=1.77778

10:leaf=-1.98104

2:[f109<-9.53674e-07] yes=5,no=6,missing=5

5:[f67<-9.53674e-07] yes=11,no=12,missing=11

11:[f8<-9.53674e-07] yes=15,no=16,missing=15

15:leaf=-1.99117

16:leaf=1

12:[f39<-9.53674e-07] yes=17,no=18,missing=17

17:leaf=1.77143

18:leaf=-1.5

6:leaf=1.85965

booster[1]:

… …

图 5.8 用于特征工程的xgboost训练模型

booster[4]:

0:[f29<-9.53674e-07] yes=1,no=2,missing=1

1:[f21<-9.53674e-07] yes=3,no=4,missing=3

3:leaf=0.949695

4:[f27<-9.53674e-07] yes=7,no=8,missing=7

7:[f30<-9.53674e-07] yes=13,no=14,missing=13

13:leaf=-0.846263

14:leaf=0.571051

8:leaf=0.604424

2:[f39<-9.53674e-07] yes=5,no=6,missing=5

5:[f115<-9.53674e-07] yes=9,no=10,missing=9

9:leaf=-0.1213

10:leaf=0.945553

6:leaf=-0.863671

上述训练模型中共有5棵决策树，分别是从booster[0]到booster[4]，booster之下每一行是对一个数结点的描述，冒号之前是结点编号，标记有leaf字样的是树叶子结点，相同缩进的结点位于同一层中，而预测结果就是看每棵决策树上最终预测流程是落在哪个叶子结点上，xgboost提供了predict\_leaf功能，使用刚刚用过的训练数据运行一遍predict\_leaf，得到输出样例如下：

[[10 8 16 10 6]

[20 18 18 6 8]

[11 13 15 13 3]

... ...

[10 8 16 4 14]]

图 5.9 Predict\_leaf输出样例

每一行对应一个输入预测样本在模型中每棵决策树上所触发的叶子结点序号，我们将以此为基础来构建新的特征。

### 5.2.2 构建新特征

在得到了训练样本数据对应模型决策树种的叶子结点信息之后，下面就要用这些叶子结点编号构建新特征。采用的方法是对每一颗决策树先求出它的index偏移量offset为前一棵决策树的offset加上前一棵决策树的最大叶子结点序号再加1，第一棵决策树(编号为0)的offset是已有结构化属性的最大index加1，见公式5.1(i是决策树编号)：

(公式5.1)

相应地，新特征的编号定义为：

(公式5.2)

得到了新特征的编号，就可以将新特征添加到训练样本中，以图5.9为例，假设数据原有结构化特征为100维，对应index编号为[0-99]，训练模型中5棵决策树的最大叶子结点编号为[20,18,16,16,14]，这5棵决策树对应的offset为[100,121,140,157,174]，新特征的内容为：

[[110:1 129:1 156:1 167:1 180:1]

[120:1 139:1 158:1 163:1 182:1]

[111:1 132:1 155:1 170:1 177:1]

... ...

[110:1 129:1 156:1 161:1 188:1]]

图 5.10 新特征样例

构建新特征算法描述如下：

|  |
| --- |
| **Algorithm**: 以用户微博文章向量为基础构建新非结构化特征 |
| **Input**: 基于文章向量训练得到的xgboost决策树模型model；用户结构化特征的维度baseOffset；  **Output**: 用户非结构化特征non\_structured\_featureset  **Step1**：用xgboost的predict\_leaf功能得到训练数据(微博文章向量)在model中每棵决策树上对应的叶子结点编号集合leaf\_dataset  **Step2**：用xgboost的dump功能将model中的决策树输出为可读文本，设有N棵决策树  **Step3**：找出每棵决策树的叶子结点的最大编号存为max\_leaf\_idx[0,…,N-1]  **Step4**：求新特征index的offset，定义idx\_offset[0,N-1]，令idx\_offset[0] = baseOffset  For i = 1 to N-1 do  idx\_offset[i] = idx\_offset[i-1] + max\_leaf\_idx[i-1] + 1  End for  **Step5**：得到非结构化特征  For leaf\_data in leaf\_dataset do  newFeatureSet = “”  For i = 0 to N-1 do  newIdx = leaf\_data[i] + idx\_offset[i]  newFeatureSet.add(“$newIdx:1”)  End for  non\_structured\_featureset.add(newFeatureSet)  End for  **Step6**：return non\_structured\_featureset |

图 5.11 构建非结构化特征算法

将得到的用户非结构化特征与原有的用户结构化特征一起构成了用于CTR预估的全部用户特征。接下来要做的是用全部用户特征作为新的训练数据集训练线上模型。

## 5.3 模型上线

在得到了用户所有特征(包括原有的结构化特征和由用户的微博文章转化而来的非结构化特征)之后，接下来就是用这些特征训练出一个可用作线上实时预测的模型。在第二章中提到，逻辑回归模型是一种很高效的分类器，很适合做CTR预估。本系统中使用ftrl\_proximal\_lr实现这一功能[38]。ftrl\_proximal\_lr实现了ftrl逻辑回归算法，并且最大限度地利用并行多线程以提高效率；而且支持模型在线增量更新，即在已有模型的基础上用线上新获取的数据（用户的特征和具体点击行为）来增量训练，矫正原有模型的各个参数，从而实现更高的预测准确度。通过本系统的线上运行实验证明ftrl\_proximal\_lr有很高的运行效率和准确度，能够应对实际生产环境中的高并发请求。

## 5.4 本章小结

本章详细介绍了基于用户微博行为的在线广告点击行为预测的业务流程，主要分为文本处理、特征工程、模型上线3个主要步骤，对每一个步骤分别详细介绍了其工作原理和处理方法。

# 第六章 系统测试

## 6.1 系统功能测试

功能测试的主要目的是检查系统所实现的功能是否达到了当初的设计目标，并且能够提供稳定可靠的服务。本系统的功能测试主要包括：1)管理交互界面；2)服务管理，包括分布式集群管理，服务上/下线，异常处理如线上服务进程实例非正常退出等；3)线下批处理作业服务，主要包括分词、文本转向量等；4)线上请求处理服务，主要是线上预测；5)数据更新服务，主要是线上预测模型的更新。

### 6.1.1 管理交互界面

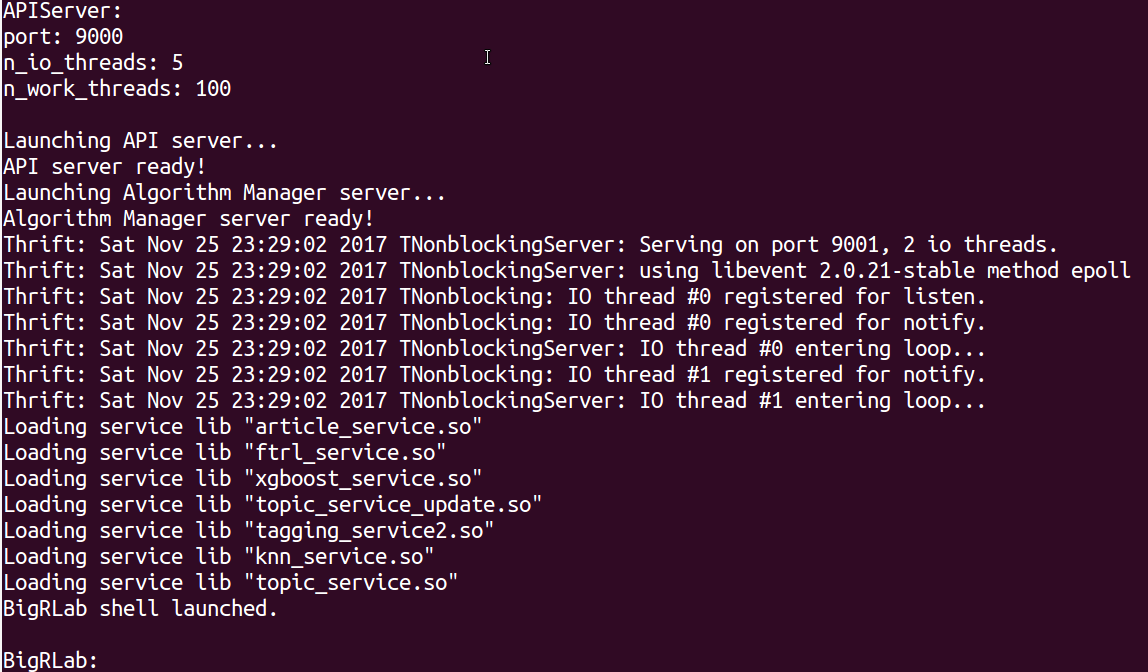
 管理员用户可以通过交互界面(UI)来实现线上服务管理和作业管理，服务管理主要功能包括：1)查看当前服务运行状态；2)强制结束某个服务(可以是整个服务，也可以是服务下的某个进程实例)。作业管理的主要功能包括：1)建立并提交批处理作业；2)查看作业运行状态；3)强制结束某个作业。目前采用类似shell的命令行接口实现交互界面，如图6.1所示：

图 6.1 系统交互命令行接口

### 6.1.2 服务管理

服务(Service)是一组运行同一算法、对外提供统一服务接口的进程实例，在本文第四章介绍过其概念和实现，对这一部分的功能测试主要是测试各算法服务是否能正常地创建，新实例是否正常加入和退出。

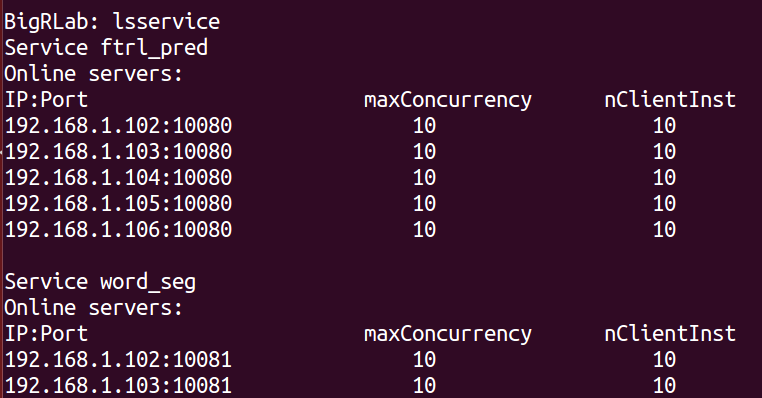
启动一个服务实例进程必须提供这几个参数：1)服务名称，系统会将同一服务名称的进程实例归为一组，对外提供统一服务接口，内部实现负载均衡，若该服务不存在则自动创建，若已存在会将这个新的进程实例加入到服务实例列表中；2)ApiServer地址，用于在ApiServer上注册；3)本服务的网络端口号，需手工指定，其注册IP地址是自动获取。注册成功后，ApiServer的控制台会显示新实例加入的通知信息。此外还可以通过控制台命令lsservice显示所有的服务和实例详细信息，如图6.2所示：

图 6.2 查看服务和实例信息

可以看到当前线上有2个服务ftrl\_pred和word\_seg，服务ftrl\_pred有5个实例(servers)，word\_seg有2个，列表中显示了这些实例所在主机的IP和服务端口号，所支持的最大并发数maxConcurrency和当前所建立的工作连接数nClientInst。

当某个实例正常退出时会给ApiServer发退出消息，ApiServer会从相应的服务中删除该实例及为其所建立的工作连接，并在控制台上输出提示消息。当某个服务实例异常退出，没来得及向ApiServer发送退出通知时，系统会自动检测到发往该服务实例的请求异常，并自动选取下一个服务实例重新发送请求，当发往这个实例的连续请求失败次数达到一阈值（当前设定为5）时，系统会从相应的service中删除该服务实例并通过控制台消息通知。

### 6.1.3 线下批处理作业服务

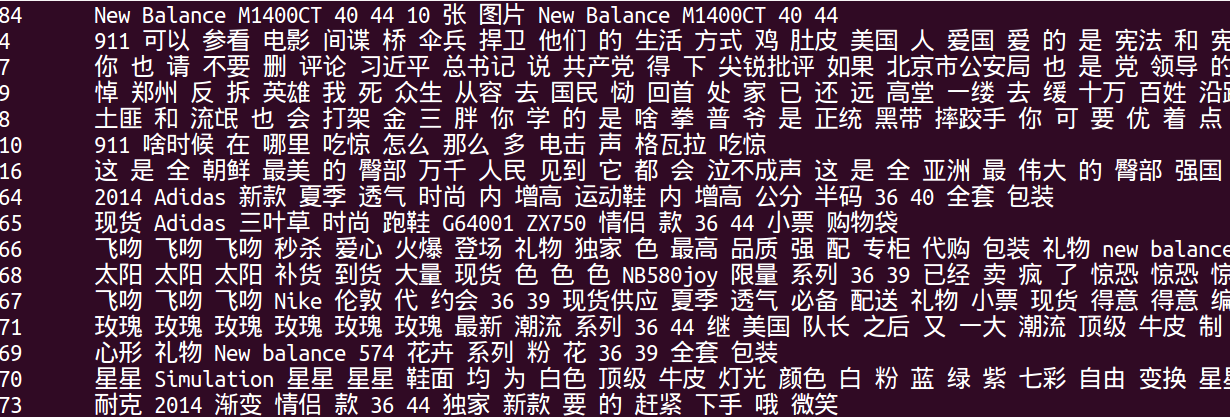
批处理作业服务是借助分布式集群高并发的性能优势实现对海量数据(千万级别)的高效快速处理。广告点击预测业务所涉及的线下批处理作业主要有分词和文本的向量化表示。一般建立批处理作业需要指定输入输出文件的路径(支持分布式文件系统如HDFS)、作业名称、所用线上服务的名称以及该服务所要求的其他参数。批处理作业的输入输出均为可读文本格式，如要分词或转为向量的微博文章，输入文件的每一行代表一个请求，会按照负载均衡的原则分发到服务所属的进程实例来处理。由于是并行处理，结果返回的顺序与请求提交的顺序很可能是不一致的，这需要对每个请求进行标识，可以通过参数指定输入文件的第一列为请求标识，如不指定用默认的行号做请求标识，输出文件的每一行是一个请求的返回结果，格式为请求标识+返回内容。如图6.3所示：

图 6.3 分词结果示例，最左侧栏是请求标识，对应输入文件的行号

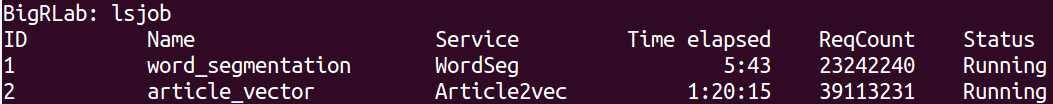
 在系统控制台可以用lsjob命令查看当前系统中的作业运行状态，如图6.4所示：

图 6.4 查看作业状态

显示内容依次是作业ID(按提交顺序依次编号)、作业名(由命令行指定)、所用服务名、已经运行时间、已处理的请求数、当前状态(四种状态运行中Running，已完成Finish，人为终止Abort和出错Error)

### 6.1.4 线上请求处理服务

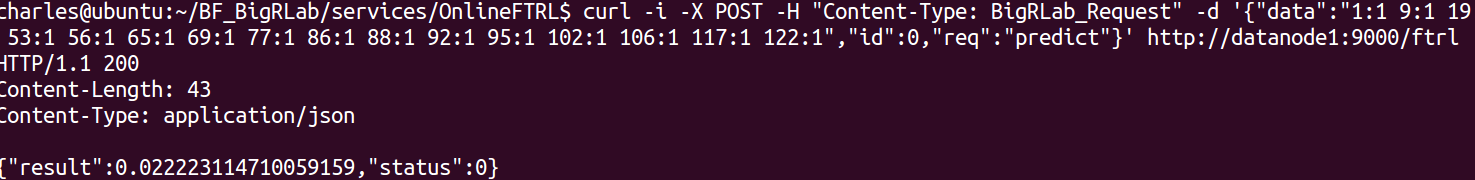
 线上服务请求处理使用Restful API的形式，用Http POST方式发送请求，URL是ApiServer的地址端口和service名称，正文内容是请求参数和内容，用Json格式封装。在广告点击预测业务中线上服务主要是ftrl预测，此部分功能测试主要使用Http请求工具curl来做，如图6.5所示：

图 6.5 线上服务功能测试示例

一般返回结果中status为0表示成功，非0表示失败并附有errmsg错误信息字段。

### 6.1.5 数据更新服务

数据更新服务主要是针对线上预测服务的模型数据增量更新，Online-Ftrl算法支持模型增量训练，这样可以能够实时反映用户兴趣行为的变化，提高预测精确度。线上请求数据存储是借助基于内存的数据库Redis实现，设定缓存近一周的请求数据。数据更新服务部署在一台独立主机上，当新请求的数量累计到一定阈值时会触发增量训练，随后数据更新服务会逐个通知线上预测服务实例重新加载新模型，为了防止所有服务同时重启引起的线上服务抖动，只有当前一个服务实例重启(重新加载模型)成功并通知数据更新服务后，数据更新服务才会继续通知下一个服务实例重启。整个框架结构如图6.6所示：

图 6.6 模型数据更新流程图

## 6.2 性能测试

性能测试的主要目的是测试系统处理请求效率是否达到了设计要求，按业务种类不同分线下批处理业务性能测试和线上实时请求处理能力测试。批处理业务性能评价主要是处理时间，即相比于普通单机环境处理时间的优化；线上请求处理性能评价主要是请求平均响应时间和单位时间内处理线上请求的数量qps。

线上运行环境：目前系统处于小流量试运行阶段，线上环境为4台同样配置的服务器，CPU：Intel Xeon ES-2609 v2 2.5GHz 4核，32GB内存。在每台主机服务实例数目方面，据测试当实例数目略多于CPU核数时性能达到最优，过多的实例会导致并发调度的开销过高，实际运行中每台服务器跑6-8个服务实例。

线下批处理作业性能方面：这部分主要是针对文本分词和向量化表示的，使用数据集含有4600余万篇微博文章。在同样配置的服务器上，使用单机版的Jieba分词平均耗时3小时，在本计算服务平台上平均耗时11分钟，仅为单机版的1/15，文本向量化表示服务单机版平均耗时5.5小时，在本平台上运行平均耗时1.5小时，时间缩短为单机版的1/4，效率提升非常明显，完全达到设计要求。

线上实时请求处理性能方面：这主要是针对Online-Ftrl做线上实时预测的性能测试，请求平均响应时间在4-7毫秒之间，很少有出现10毫秒以上的请求响应。每秒钟处理的平均请求数(qps)稳定在9000到11000之间，也达到了设计要求。

## 6.3 准确率测试

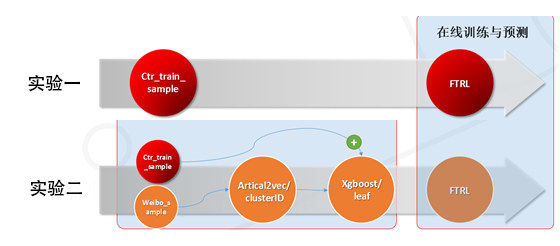
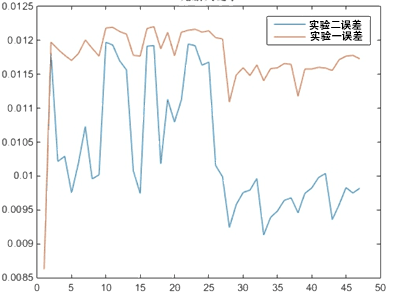
准确率测试的目的是针对广告点击预测业务比较本文所用方法，即使用用户固有结构化特征结合用户行为数据(微博行为记录)的非结构化特征一起来预测用户的广告点击行为，较之传统方法即只使用用户的结构化特征做预测在准确率上是否有所改进。为了测试本文所有方法方法的准确率，分别设计了两组组实验加以比较。实验过程如图6.7所示：

图 6.7 对比实验流程

其中实验一是直接用ftrl算法在用户的结构化特征数据上做训练和预测；实验二是用本文的方法将用户的微博行为数据通过特征工程转化为数值特征，再和用户原有的结构化特征结合生成新数据，在之基础上用ftrl做训练和预测。实验结果如图6.8所示：

图 6.8 对比实验准确率评估

图中橙色线条表示用传统方法实验一的均方根误差，蓝色线条表示用本文的方法实验二的均方根误差。可以看出本文所用方法在准确率上有明显优势。

## 6.4 本章小结

本章分别从功能、性能和准确度3个方面展示了系统上线后的实际运行效果。运行结果表明其各项指标均达到了设计目标，某些方面还有所超越。

# 第六章 总结与展望

## 6.1 总结

本文旨在介绍一种结合用户微博行为记录的广告点击率预测方法，并实现了一个可实用的基于此种方法的在线广告点击预测系统。本文所用方法的重点是特征工程，即如何自动选取有用特征。为此首先将用户感兴趣的微博文章转换为数值向量，用这些向量和用户最终点击行为数据训练出一个迭代决策树GBDT模型，再用此GBDT模型将连续的微博文章向量特征转换成和结构化特征形式一致的离散特征，实际上是使用了GBDT模型中每棵树不同的叶子序号作为特征。

系统实现方面，本文考察了多种搭建互联网大数据处理平台的开源框架，综合利用多种开源工具，设计并实现了一个分布式大数据处理平台，其主要优势在于将大量计算任务分散到多台服务器上执行，既能够处理线上实时请求，也能够处理线下作业，同时有很好的扩展性和兼容性。引入服务管理和作业管理的设计概念，经过二次开发也可用于处理其他类型的业务，具有很好的通用性。

## 6.2 展望

本文提出了一种基于用户微博的广告点击预测，其目的是通过用户微博浏览记录发现用户兴趣，从而更准确地投放广告。整个业务流程中，微博文章数据的特征转换是关键环节。目前所用的方法是将用户感兴趣的所有文章拼接成一篇长文章，再将拼接后的文章转换为数值向量，转换方法是用每个词的向量叠加并归一化。这个过程没能够考虑到不同的文章所代表的不同的主题，而拼接过程事实上是消除了不同文章间的个性差异。势必会对最终预测准确度造成负面影响。在这方面今后的改进方向是使用NLP技术如话题模型找出每篇用户感兴趣文章的话题关键词，给这些关键词更高的权重，并以此为基础分析用户兴趣以达到更高的投放准确度。

在系统设计方面，为了支撑业务需求，本文实现了一个基于分布式的大数据处理平台，但目前这个平台功能上还有很多待完善的地方，首先从架构上看，目前采用的是一个入口节点(ApiServer)带多台计算服务器，在高访问量环境下，ApiServer将成为影响系统吞吐量的瓶颈。后续还要对此进行结构优化，使之能支持多个对外服务入口服务器。在用户交互界面上，目前的仿Linux Shell命令行交互界面还不够直观易用，后续会在其上层加一个图形界面接口，能够实时观察各个服务、作业运行状态。随着业务规模的扩大，后续会加入线上日志报表功能，用于统计用户访问量，系统响应时间，预测准确率变化等信息，同时还要加入用户管理和访问控制等功能。在系统功能方面，目前线上只支持预测（支持在线更新模型的算法如Online Ftrl除外），训练模型的过程需要线下完成，在后续的改进版本中可以通过图形管理界面选择算法和参数，模型训练和部署一步完成。最终目标是将其打造成一个功能完善的、界面友好的、多任务高并发可扩展的综合大数据处理平台。

# 参考文献

1. 第38次中国互联网络发展状况统计报告 <http://www.cnnic.net.cn/>
2. 新浪财经频道 <http://finance.sina.com.cn/roll/2016-11-12/doc-ifxxsfip4572557.shtml>
3. 刘鹏, 王超, 计算广告,互联网商业变现的市场与技术, 人民邮电出版社, 2015
4. Zhang, Wei Vivian, and Rosie Jones. "Comparing click logs and editorial labels for training query rewriting." WWW 2007 Workshop on Query Log Analysis: Social And Technological Challenges. 2007.
5. Richardson M, Dominowska E, Ragno R. Predicting clicks: Estimating the click-through rate for new ads, Proceedings of the 16th International conference on World Wide Web, Banff, Canada, 2007: 521-530
6. Guo F, Liu C, Kannan A, et al. Click chain model in web search, Proceedings of the 18th International conference on World Wide Web. Madrid, Spain, 2009: 11-20
7. Jahrer M, Toscher A, Lee J Y, et al. Ensemble of collaborative filtering and feature engineered models for click through rate prediction, Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDDCup Workshop, Beijing, China, 2012
8. Shen S, Hu B, Chen W, et al. Personalized click model through collaborative filtering, Proceedings of the 5th ACM International Conference on Web Search and Data Mining, Seattle, USA. 2012: 323-332
9. Regelson M, Fain D. Predicating click-through rate using keyword clusters, Proceedings of the second workshop on Sponsored Search Auctions, Ann Arbor, USA, 2006: 9623-9628
10. Agarwal, Deepak, et al. "Estimating rates of rare events at multiple resolutions." Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.
11. Wang X, Li W, Cui Y, et al. Click-through rate estimation for rare event in online advertising, Hua Xian-Sheng, Mei Tao, Hanjialic A eds. Online Multimedia Advertising: Techniques and Technologies. Hershey Pennsylvania, USA: IGI Global, 2010: 1-12
12. Kempe D, Mahdian M, A cascade model for externalities in sponsored search, Proceedings of the 4th International Workshop on Internet and Network Economics, Chicago, USA, 2008: 585-596
13. Guo F, Liu C, Wang Y M, Efficient multiple-click models in web search, Proceedings of the 2nd ACM International Conference on Web Search and Data Mining, Barcelona, Spain, 2009: 124-131
14. 腾讯大数据：CTR预估中GBDT与LR融合方案 <http://www.cbdio.com/BigData/2015-08/27/content_3750170.htm>
15. He, Xinran, et al. "Practical lessons from predicting clicks on ads at facebook." Proceedings of the Eighth International Workshop on Data Mining for Online Advertising. ACM, 2014.

1. <https://github.com/guestwalk/Kaggle-2014-criteo>
2. Cox, DR. "The regression analysis of binary sequences (with discussion)". J Roy Stat Soc B. 20, 1958: 215–242.
3. Walker SH, Duncan DB. "Estimation of the probability of an event as a function of several independent variables". Biometrika. 54, 1967: 167–178.
4. <https://en.wikipedia.org/wiki/Logistic_regression#Logistic_function.2C_odds.2C_odds_ratio.2C_and_logit>
5. Hosmer, David W.; Lemeshow, Stanley. Applied Logistic Regression (2nd ed.). Wiley. ISBN 0-471-35632-8, 2000
6. Everitt, Brian. The Cambridge Dictionary of Statistics. Cambridge, UK New York: Cambridge University Press, 1998
7. Tjalling J. Ypma, Historical development of the Newton-Raphson method, SIAM Review 37 (4), 531–551, 1995.
8. Menard, Scott W, Applied Logistic Regression (2nd ed.). SAGE, 2002
9. H. Brendan McMahan, Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization, Google, Inc. 2011
10. J. Duchi and Y. Singer. Efficient learning using forward-backward splitting. In Advances in Neural Information Processing Systems 22, pages 495-503. 2009
11. L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. In NIPS, 2009
12. McMahan, H. Brendan, et al. "Ad click prediction: a view from the trenches." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013.
13. 陶辉, 深入理解Nginx: 模块开发与架构解析, 机械工业出版社, 2013
14. Lamport, Leslie. "Paxos made simple." ACM Sigact News 32.4 (2001): 18-25.
15. Junqueira, Flavio, and Benjamin Reed. ZooKeeper: distributed process coordination. " O'Reilly Media, Inc.", 2013.
16. McCandless, Michael, Erik Hatcher, and Otis Gospodnetic. "Lucene 实战." (2011): 6-7.
17. Rakowski, Krzysztof. Learning Apache Thrift. Packt Publishing Ltd, 2015.
18. Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.
19. Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
20. Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." ACM Transactions on Computer Systems (TOCS) 26.2 (2008): 4.

1. <https://github.com/fxsjy/jieba/>
2. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

1. <https://github.com/bingzhengwei/ftrl_proximal_lr>
2. <https://github.com/dmlc/xgboost>

# 致谢

本论文是在xx老师的悉心指导下完成的。xx老师作为一名优秀的、经验丰富的教师，具有丰富的xx知识和xx经验，在整个论文实验和论文写作过程中，对我进行了耐心的指导和帮助，提出严格要求，引导我不断开阔思路，为我答疑解惑，鼓励我大胆创新，使我在这一段宝贵的时光中，既增长了知识、开阔了视野、锻炼了心态，又培养了良好的实验习惯和科研精神。在此，我向我的指导老师表示最诚挚的谢意！

……

(仅为网络示例，可根据论文实际进行撰写，使用时把模板示例内容尽皆删除即可)

# 北京大学学位论文原创性声明和使用授权说明

**原创性声明**

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名： 日期： 年 月 日

**学位论文使用授权说明**

（必须装订在提交学校图书馆的印刷本）

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

* 按照学校要求提交学位论文的印刷本和电子版本；
* 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
* 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
* 因某种特殊原因需要延迟发布学位论文电子版，授权学校□一年/□两年/□三年以后，在校园网上全文发布。

（保密论文在解密后遵守此规定）

论文作者签名： 导师签名：

日期： 年 月 日