Computer Science and Application 计算机科学与应用, 2019, 9(5), 1029-1035
Published Online May 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.95116

Hans汉斯

Research on Prediction of the Use of Electronic Coupons Based on XGBoost

Yihua Xiong

College of Science, North China University of Technology, Beijing
Email: 359391236@qq.com

Received: May 11th, 2019; accepted: May 24th, 2019; published: May 31st, 2019

Abstract

Boosting is a very effective sequence integration algorithm, which has a wide range of applications in practice. This paper focuses on the XGBoost algorithm, which optimizes the structure of fast parallel tree and is fault-tolerant in distributed environment. This makes it possible to process billions of data accurately and quickly and give reliable results. In this paper, through simulation analysis and empirical analysis, compared with GBDT and RF algorithm, XGBoost's excellent characteristics are verified.

Keywords

XGBoost, GBDT, RF

基于XGBoost的电商优惠券使用情况预测研究

熊艺华

北方工业大学理学院，北京
Email: 359391236@qq.com

收稿日期：2019年5月11日；录用日期：2019年5月24日；发布日期：2019年5月31日

摘　要

Boosting是十分有效的序列集成算法，在实践中有着广泛的应用。本文着重研究的XGBoost算法针对快速并行树结构进行了优化，并且在分布环境下容错，这使得它可以精确快速处理亿级数据、给出可靠的结果。本文分别通过模拟分析和实证分析，对比GBDT和RF算法验证了XGBoost的优良特性。

文章引用: 熊艺华. 基于 XGBoost 的电商优惠券使用情况预测研究[J]. 计算机科学与应用, 2019, 9(5): 1029-1035.
DOI: 10.12677/csa.2019.95116

关键词

XGBoost, GBDT, RF

1. 引言

机器学习已经被广泛应用于解决各种各样的现实问题，由于现实问题往往都十分的复杂，因此仅仅用一个单一的学习器或模型得到的结果往往不尽如人意。实践证明，集成学习相较于单一学习器往往能得到更为准确的结果。所谓集成学习，就是通过组合多个学习器以期得到一个更好更全面的学习器。集成方法可以分为两类：一是并行集成方法(Bagging)，其中参与训练的基学习器并行生成，并行集成方法的原理是利用基学习器之间的独立性，通过平均可以显著降低错误。二是序列集成方法(Boosting)，其中参与训练的基学习器按照顺序生成，序列集成方法的原理是利用基学习器之间的依赖关系，通过对之前训练中错误标记的样本赋值较高的权重，可以提高整体的预测效果。从偏差方差分解角度看，Bagging 主要关注降低方差，Boosting 主要关注降低方差。本文主要研究 Boosting 相关方法及应用。

Schapire [1]于 1990 年对“弱学习是否等价于强学习”这个理论问题给出构造性证明，这也是最初的 Boosting 算法。一年后，Schapire 提出了一种更为高效的 Boosting 算法，但这两种算法都需要知道弱学习器的学习正确率的下限，难以应用实际问题。1997 年，Schapire [2]提出了 AdaBoost 算法，能够更好地利用弱学习器的优势，同时也摆脱了对弱学习器先验知识的依赖。AdaBoost 算法的很多变体都是对损失函数做了改变，比如 LogitBoost 考虑了 log 损失函数、L2Boost 考虑 L2 损失函数等。2001 年，Freund [3]将梯度下降的思想引入 Boosting 算法，提出了 Gradient Boosting 算法，其思想在函数空间以负的梯度方向优化损失函数，可以认为是 AdaBoost 的推广。

然而，现有的树模型 Boosting 算法仍局限于计算百万量级的数据，尽管已有一些关于如何并行算法的研究，但是关于同时优化系统和算法使之能用于计算亿级数据的研究很少。本文主要研究的 XGBoost 正是这样一种可以用于处理亿级数据可靠的机器学习方法。

本文分为五个部分：第一节引言部分简单介绍了 Boosting 及相关的研究进展；第二节简单概括了 Boosting 的基本思想，详细说明 XGBboost 的原理及改进优化之处；第三节详细说明 XGBoost 求解的基本算法；第四节模拟分析，把 XGBoost 用于模拟数据并于 GBDT、RF 进行比较分析；第五节实证分析，把 XGBoost 用于真实数据并评价其效果。

2. Boosting

2.1. Boosting

Boosting 分类器[4]属于集成学习模型，它基本思想是把多个“弱”分类器结合成一个“强”分类器，所谓的“弱”分类器是指错误率仅仅只比随意猜测低一点的分类器，也就是说将许多分类准确率较低的分类器组合起来，成为一个准确率很高的分类器，个体分类器间存在强依赖关系，意思是说每次迭代生成的新的分类器都是在上一个分类器的基础上生成的。通常分类器采用分类回归决策树，每次生成新的决策树时，根据已有决策树的分类效果对样本分布进行调整，给分对的样本较小的权重、给分错的样本

较大的权重，使得之前分错的样本在后续得到更多的关注，然后基于调整后的样本分布来训练生成下一个新的决策树。Boosting 最初是为解决分类问题而提出的，现在也可以用于解决回归问题。

2.2. XGBoost

与传统 Boosting 生成新决策树时对分类正确、错误的样本进行加权有着很大的区别，Friedman 提出的 GBDT 在生成每一个决策树是采用梯度下降的思想，即每一次的生成新的决策树是为了减少上一次的残差，通过在残差减少的梯度方向上建立一个新的决策树来进一步消除残差。

陈天奇[5]提出的集成学习算法 XGBoost (eXtreme Gradient Boost)在 Gradient Boost 的基础上进行了改进，在损失函数中用二次泰勒展开、增加了正则化项，使得模型更简单、减小过拟合的可能。XGBoost 能自动利用 CPU 的多线程进行并行，同时优化了算法、能分布式处理高维稀疏特征，使之更精确、计算速度可以比同类算法快十倍以上、且具有广泛适用性。

2.3. XGBoost 原理

对于有 n 个观测 m 个特征的数据集 $\mathcal{D}=\{(x_i,y_i)\}\left(|\mathcal{D}|=n,x_i\in\mathbb{R}^m,y_i\in\mathbb{R}\right)$，预测结果是由 K 个决策树加合模型得到的

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

其中 $\mathcal{F}=\left\{f(x)=\omega_{q(x)}\right\}\left(q:\mathbb{R}^m\to T,\omega\in\mathbb{R}^T\right)$ 为决策树空间，q 代表每棵树的结构，T 为叶节点数，ω 为权重。可以得到正则化损失函数

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

$$\Omega(f)=\gamma T+\frac{1}{2}\lambda\|\omega\|^2 \tag{2}$$

其中，Ω 为惩罚项控制模型的复杂度，同时使 ω 更光滑从而避免过拟合。如果是回归问题，损失函数 l 通常可以用平方误差 $(y-\hat{y})^2$ 或绝对误差 $|y-\hat{y}|$；如果是分类问题即 $y_i\in\{-1,1\}$，损失函数 l 可以是指数损失准则[6] $e^{-y\hat{y}}$，或者负二项对数似然函数[7] $\log\left(1+e^{-2y\hat{y}}\right)$。类似的这种正则化方法已经被用于正则化贪心森林(RGF) [8]。通过最小化正则化损失函数(2)得到最优参数 P^*，继而可以得到既简单又精确的模型 $f^*(x)$，即

$$P^*=\arg\min_P\mathcal{L}(\phi)$$

$$f^*(x)=f(x,P^*)$$

除了在损失函数中增加正则化项以外，还可以采用其他两种方法以避免过拟合。其一是收缩方法[9]，与随机优化中的学习率类似，收缩减少了每棵树的影响，并为将来的树留出改进模型的空间；其二是列(特征)子抽样，这一方法已被用于随机森林[10]中，在实际应用中，列子抽样避免过拟合的效果要比行子抽样的效果好，同时也能提高计算速度。

3. 基本算法

显然，无法直接通过运算计算出 $f^*(x)$ 的解，因此就需要用到优化的方法。上一节中已经提到 $f(x)$ 为决策树模型，给出权重 ω 和树结构 q 即可确定一颗决策树，而树结构 q 实质上就是划分分裂节点的问题，因而求解 $f^*(x)$ 可以转化成寻找最优权重 ω^* 和划分分裂节点的问题。具体过程如下：

设 $\hat{y}_i^{(t)}$ 为第 i 个实例在第 t 次迭代时的预测值，贪心的增加 f_t 使得能最大程度的改善模型，于是要最小化如下损失函数

$$\mathcal{L}^{(t)}=\sum_{i=1}^{n} l\left(y_i,\hat{y}_i^{(t-1)}+f_t\left(x_i\right)\right)+\Omega\left(f_t\right) \tag{3}$$

用泰勒二次展开来估计，得到近似损失函数

$$\mathcal{L}^{(t)}\cong\sum_{i=1}^{n}\left[l\left(y_i,\hat{y}_i^{(t-1)}\right)+g_i f_t\left(x_i\right)+\frac{1}{2}h_i f_t^2\left(x_i\right)\right]+\Omega\left(f_t\right) \tag{4}$$

其中，$g_i=\partial_{\hat{y}_i^{(t-1)}} l\left(y_i,\hat{y}_i^{(t-1)}\right)$，$h_i=\partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i,\hat{y}_i^{(t-1)}\right)$ 为梯度。将常数项去掉，得到在第 t 次迭代时简化的损失函数

$$\tilde{\mathcal{L}}^{(t)}=\sum_{i=1}^{n}\left[g_i f_t\left(x_i\right)+\frac{1}{2}h_i f_t^2\left(x_i\right)\right]+\Omega\left(f_t\right) \tag{5}$$

令 $I_j=\left\{i\mid q\left(x_i\right)=j\right\}$ 为叶节点的实例集。将(5)中 Ω 展开得到

$$\begin{aligned}\tilde{\mathcal{L}}^{(t)}&=\sum_{i=1}^{n}\left[g_i f_t\left(x_i\right)+\frac{1}{2}h_i f_t^2\left(x_i\right)\right]+\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2\\&=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j}g_i\right)\omega_j+\frac{1}{2}\left(\sum_{i\in I_j}h_i+\lambda\right)\omega_j^2\right]+\gamma T\end{aligned} \tag{6}$$

固定 $q\left(x\right)$ 可以得到叶节点 j 的最优 ω_j^*

$$\omega_j^*=-\frac{\sum_{i\in I_j}g_i}{\sum_{i\in I_j}h_i+\lambda} \tag{7}$$

将(7)代入(6)得

$$\tilde{\mathcal{L}}^{(t)}\left(q\right)=-\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j}g_i\right)^2}{\sum_{i\in I_j}h_i+\lambda}+\gamma T \tag{8}$$

(8)可用于评价一棵树结构 q 的好坏。假设分裂后分别分到左和右的实例集为 I_L 和 I_R，令 $I=I_L+I_R$，可以得到分裂减少的损失

$$\mathcal{L}_{split}=\frac{1}{2}\left[\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_L}g_i\right)^2}{\sum_{i\in I_L}h_i+\lambda}+\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_R}g_i\right)^2}{\sum_{i\in I_R}h_i+\lambda}-\sum_{j=1}^{T}\frac{\left(\sum_{i\in I}g_i\right)^2}{\sum_{i\in I}h_i+\lambda}\right]-\gamma \tag{9}$$

通常用贪心算法找到的使 $\mathcal{L}_{split}$ 最大的划分节点，即为要寻找寻找最优分裂节点。

4. 模拟分析

本节通过模拟数据验证 XGBoost 在大数据量下的计算快、预测结果精确的特点。同时，也引入 GBDT(梯度决策树)和 RF(随机森林)两种算法与 XGBoost 进行比较，其中 GBDT 是上文提到的将梯度下降算法与 CART 决策树相结合的一种 Boosting 算法，而 RF 是将随机选择特征引入 CART 决策树生成过程的优化的 Bagging 算法。所有的模拟计算均采用 R 语言进行。假设设计矩阵 $X_{n\times p}$ 是由 p 个独立生成的服从 $N\left(0,1\right)$ 的 $X_1,X_2,\cdots,X_p$ 组成，通过逻辑变换 $\pi=\frac{\mathrm{e}^{X\beta+\varepsilon}}{1+\mathrm{e}^{X\beta+\varepsilon}}$ 得到概率 π，其中 $\beta\sim N\left(0,1\right),\varepsilon\sim N\left(0,1\right)$。$\pi$ 大于 0.5 将对应标签记为 1，π 小于 0.5 则记为 0。

4.1. 模型评价依据

关于模型效果的评价，主要考虑两个方面：

1) 模型训练时间。模型训练时间或者运行时间越短，说明该算法收敛速度越快、高效。反之则说明收敛速度慢。

2) 预测的准确性。为了评价模型预测结果，采用错误率作为评价标准：

$$err = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i \neq \hat{y}_i\right)$$

其中 n 为观测数，y_i 为真实结果，$\hat{y}_i$ 为模型预测结果，I 为示性函数，将计算得出的 err 作为错误率。显然，err 越小意味着该模型预测结果越准确。

4.2. 结果分析

为保证结果的可比性，调节使三种算法的参数尽量保持一致，得到三组结果如下表 1：

Table 1. Simulation results
表 1. 模拟数据结果

模型	运行时长/s	错误率
XGBoost	0.251	0.1688
GBDT	0.465	0.5071
RF	17.487	0.4138

结果表明，XGBoost 在模拟数据上的表现明显优于 GBDT 和 RF。不仅模型训练时间最短，预测结果也是最精确的。

5. 实证分析

5.1. 数据来源

所用的数据是阿里云天池大赛提供的优惠券使用预测数据集。该数据提供了用户在 2016 年 1 月 1 日至 2016 年 6 月 30 日之间真实线上线下消费行为，以及用户在 2016 年 7 月领取优惠券后 15 天以内的使用情况。共有三个数据集(表 2)：

Table 2. Dataset information
表 2. 数据集信息

数据集	观测数	变量数
Offline_train	1754884	7
Offline_test	113640	6
On_line_train	11429826	7

本节用到是 Offline_train 数据集中 2016 年 4 月 14 日至 2016 年 6 月 15 日的 384,627 条数据。原始数据中包含 7 个特征，具体如下表 3。

5.2. 特征工程

由于原始数据其特征的复杂性与特殊性，无法直接用于建模。因此为了最大程度地提取有效信息以

Table 3. Data information
表 3.数据信息

Field	Description
User_id	用户 ID
Merchant_id	商户 ID
Coupon_id	优惠券 ID：null 表示无优惠券消费
Discount_rate	优惠率：折扣率或者满减
Distance	用户经常活动的地点离该商铺的最近门店距离
Date_received	领取优惠券日期
Date	消费日期

供算法和模型使用，需进行特征工程的步骤。所谓的特征工程指的是把原始数据转变为模型的训练数据的过程，它的目的就是获取更好的训练数据特征，使得机器学习模型的性能得到提升。

本节在特征构建过程中主要用了两类方法：

1) 非数值型变量处理，比如“Date”有无日期转化为标签消费券核销或消费券未核销，根据“Date”和“Date_received”计算领券日期与消费日期相差的天数等。

2) 属性分割和结合的方法，组合二个、三个不同的属性构造新的特征，从而增加特征的复杂度使之能够解决复杂的问题。比如“Discount_rate”可以拆分为该店有满减活动或者打折活动，根据“Date”和“Coupon_id”可以计算出每种优惠券的核销率等。

主要思路是从五个角度分别提取特征，即从“商家”、“用户”、“优惠券”、“商家和用户”，“其他”这五个角度分别进行属性处理、组合与分割，最后再经过筛选得出 50 个用于建模的特征。

5.3. 模型构建

为了预测用户领取优惠券以后 15 以内的消费情况，将这 384,627 条数据、50 个特征用 XGBoost 进行训练。首先，数据划分为训练集与测试集两部分以供训练模型和调参，其中训练集占 70%，测试集占 30%。然后，根据第 2、3 两节中的原理和算法进行模型训练。

5.4. 结果分析

为保证结果的可比性，使三种算法的参数尽量保持一致，得到三组结果如下表 4：

Table 4. Coupon data results
表 4. 优惠券数据结果

模型	运行时长/s	错误率
XGBoost	6.703	0.0784
GBDT	15.867	0.0894
RF	1322.855	0.0801

结果表明，无论是从运行时长还是预测结果的准确性来看，XGBoost 的表现均明显优于其他两种模型。

6. 结语

本文首先概述了 XGBoost 的基本原理和算法，以及相关优化改进之处，使之成为更为高效、精确的机器学习方法。然后通过随机模拟将 XGBoost、GBDT、RF 在模拟数据上的表现进行对比，验证了上述

优点。最后用阿里天池提供的真实数据，对数据进行处理和特征工程后，分别用 XGBoost、GBDT、RF 建模并比较三者在运行时间、预测精确性上的表现，结果表明 XGBoost 可以在最短的时间内给出最精确的预测结果。综合来看，XGBoost 在大样本且数据稀疏的情况下有十分优秀的表现。但是，本文没有继续对模型进行优化，未来可以进一步改进模型或者通过模型平均的方法来改进预测精度。

致　谢

首先，感谢我的导师徐礼文老师在我的写作过程中多加提点。然后，要感谢我的师姐以及同学在我的学习过程中给予了我莫大的帮助。最后，感谢我的父母给予我生活和精神上的支持。

基金项目

北京市教委基础科研计划项目(大数据的统计学基础理论与分析方法)。

参考文献

[1] Schapire, R.E. (1990) The Strength of Weak Learnability. *Machine Learning*, **5**, 197-227. https://doi.org/10.1007/BF00116037

[2] Freund, Y. and Schapire, R.E. (1997) A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, **55**, 119-139. https://doi.org/10.1006/jcss.1997.1504

[3] Friedman, J. (2001) Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, **29**, 1189-1232. https://doi.org/10.1214/aos/1013203451

[4] Friedman, J., Hastie, T. and Tibshirani, R. (2001) The Elements of Statistical Learning. Springer Series in Statistics, New York. https://doi.org/10.1007/978-0-387-21606-5

[5] Chen, T. and Guestrin, C. (2016) Xgboost: A Scalable Tree Boosting System. *Proceedings of the* 22*nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 785-794. https://doi.org/10.1145/2939672.2939785

[6] Freund, Y. and Schapire, R.E. (1996) Experiments with a New Boosting Algorithm. *ICML*, **96**, 148-156.

[7] Friedman, J., Hastie, T. and Tibshirani, R. (2000) Additive Logistic Regression: A Statistical View of Boosting (with Discussion and a Rejoinder by the Authors). *The Annals of Statistics*, **28**, 337-407. https://doi.org/10.1214/aos/1016218223

[8] Zhang, T. and Johnson, R. (2014) Learning Nonlinear Functions Using Regularized Greedy Forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**. https://doi.org/10.1109/TPAMI.2013.159

[9] Friedman, J. (2002) Stochastic Gradient Boosting. *Computational Statistics & Data Analysis*, **38**, 367-378. https://doi.org/10.1016/S0167-9473(01)00065-2

[10] Breiman, L. (2001) Random Forests. *Maching Learning*, **45**, 5-32. https://doi.org/10.1023/A:1010933404324