doi :10.3969/j.issn.1671-1122.2020.12.008

基于 XGBoost 和 LightGBM 双层模型的恶意软件检测方法

徐国天，沈耀童
（中国刑事警察学院网络犯罪侦查系，沈阳 110854）

摘　要：目前基于网络流量的恶意软件检测方法大多依靠专家经验获取特征，此过程耗时费力且提取的流量特征较少，同时，传统特征工程在特征维度较高时复杂度大大增加。针对上述问题，文章提出一种使用极限梯度提升树（XGBoost）和轻量级梯度提升机（LightGBM）双层模型的恶意软件检测方法。在获取目标软件网络流量并提取相关特征后，使用过滤法和互信息法进行特征处理，将数据集导入首层 XGBoost 模型进行训练；然后结合网格搜索的调参方式得到最优参数组合，获取每个样本在最佳 XGBoost 模型中各棵树的叶子节点位置，以此创造新特征集；再利用 LightGBM 模型对新数据集进行训练，从而得到最终检测模型。实验结果表明，与其他检测方法相比，文章方法在恶意软件检测的准确率和实时性方面有显著提高。

关键词：恶意软件检测；流量特征；极限梯度提升树；轻量级梯度提升机；网格搜索

中图分类号：TP309　**文献标志码：**A　**文章编号：**1671-1122（2020）12-0054-10

中文引用格式：徐国天，沈耀童 . 基于 XGBoost 和 LightGBM 双层模型的恶意软件检测方法 [J]. 信息网络安全，2020，20（12）：54-63.

英文引用格式：XU Guotian, SHEN Yaotong. A Malware Detection Method Based on XGBoost and LightGBM Two-layer Model[J]. Netinfo Security, 2020, 20(12): 54-63.

A Malware Detection Method Based on XGBoost and LightGBM Two-layer Model

XU Guotian, SHEN Yaotong
(*Cyber Crime Investigation Department*, *Criminal Investigation Police University of China*, *Shenyang* 110854, *China*)

Abstract: At present, most of the malware detection methods based on network traffic rely on expert experience to acquire features. This process is time-consuming and laborious, and less traffic features are extracted. At the same time, the complexity of traditional feature engineering will greatly increase when the feature dimension is high. According to the above problem, this paper presents a use of limit gradient tree (XGBoost) and lightweight gradient

收稿日期：2020-10-09
基金项目：公安部软科学计划 [2020LLYJXJXY031]；辽宁省自然科学基金 [2019-ZD-0167,20180550841,2015020091]；中央高校基本科研业务 [3242017013]；公安部技术研究计划 [2016JSYJB06]；辽宁省社会科学规划基金 [L16BFX012]
作者简介：徐国天（1978—），男，辽宁，副教授，硕士，主要研究方向为网络空间安全、电子数据取证；沈耀童（1998—），男，河南，硕士研究生，主要研究方向为电子数据取证。
通信作者：徐国天 459536384@qq.com

hoist (LightGBM) malware detection method of double model, in the access network traffic and extract the target software related characteristics, using the characteristics of filtering method and mutual information method, and the data set into the first floor training XGBoost model, combined with the grid search of ways to get the optimal parameter combination, for obtaining the best XGBoost model in each sample of each tree in the leaf node position, to create a new collection, The LightGBM model is used to train the new data set so as to obtain the final detection model. The experimental results show that compared with other detection methods, the accuracy and real-time performance of the malware detection proposed in this paper are significantly improved.

Key words: malware detection; flow characteristics; extreme gradient boosting; LightGBM; grid search

0 引言

由于缺乏有效的监管机制，恶意软件的迅速发展和数量的不断增加给人们带来了巨大的安全隐患，因此提出一种新的恶意软件检测方法有重要意义。

当前针对恶意软件的检测方法主要分为静态检测、动态检测和基于应用软件网络流量的检测。传统的静态检测方法不需要运行程序，在对目标应用软件进行反编译后，分析代码结构和所要执行的指令，提取敏感权限、API等特征进行分析，进而确定其是否具有恶意功能。文献[1]在不依赖API版本的情况下，将API调用序列作为特征导入随机森林分类器训练，可获得93%左右的准确率，但需要分析函数关系图获取API调用序列，特征的获取较为耗时耗力。文献[2]利用敏感权限、API、系统Action等多个融合特征导入到XGBoost分类模型训练，获得了98%以上的准确率，但是由于特征维度较高导致模型训练速度过慢，同时对于使用加壳、加密技术的移动端恶意软件，其敏感权限、API等多个融合特征获取较为困难。

传统的动态检测方法是将目标程序运行在沙箱环境中，通过监视应用程序的功能，提取运行时的函数序列调用特征、网络流量特征等来判断目标软件是否具有恶意行为。文献[3]通过运行目标应用程序获取系统调用序列和控制流序列，将该应用程序与训练得出的恶意软件特征基、阈值相比对，进而判断其是否为恶意软件，使用该算法开发出的检测系统在测试样本上获得了较高的准确率，但是对于使用了加密混淆技术的恶意软件检测率有待提高。文献[4]首先运行目标程序获取其系统调用序列，再使用马尔科夫链获取相邻系统调用序列间的关联特征，然后将特征向量化，使用支持向量机进行训练和检测。该方法最终获得了98%的准确率，但将状态转移概率矩阵转化为特征向量时容易出现维度过高的问题。

基于应用软件网络流量的检测技术属于动态检测方法，该方法首先捕获应用软件在运行过程中发送给远端控制服务器的网络流量，通过对流量日志信息进行深入分析，提取相关网络流量特征，进而判断目标软件是否具有恶意行为。文献[5]在实时采集目标软件数据流并进行清理后，利用Tshark工具提取到22个网络流量特征，分别利用多个机器学习算法对测试数据集进行训练，最终在测试集上获得了较高的准确率。但该方法只采集了应用程序运行前几分钟的全部流量，对于运行前几分钟不产生恶意通信流量的软件（如盗号软件、远程控制木马等）来说，不能完整的捕捉其恶意数据流，最终导致其对此类软件的检测率较低。文献[6]通过捕获移动端软件刚安装时、重启手机前后3个阶段的网络流量，在进行数据预处理后利用双向LSTM神经网络模型获取数据流的时序特征和空间特征，并在输出层使用Softmax激活函数进行恶意软件分类，最终获得了98%的准确率，但双向LSTM模型参数较难调整且训练时间较长。

传统的静态检测方法依赖已提取的特征库，无法检测新型恶意软件，不能满足实时检测的需要。对于使用了代码混淆、加壳等技术的移动端恶意软件，静态检测方法不能准确的对其进行分类。传统的动态检

测方法需运行目标程序，细致分析其中函数的调用情况等，耗时费力。某些代码只有在特定环境中才能被触发执行，导致恶意软件的一些功能并不能被准确记录下来[7]。此外，当前研究方法单方面的追求恶意软件分类的准确率，忽略了模型的检测时间，不适用于实时检测系统的构建。

本文提出一种基于应用软件网络流量特征，使用XGBoost和LightGBM双层模型的恶意软件检测方法。在获得原始应用软件网络流量特征后，先使用特征选择的方法去除冗余特征，然后将特征导入第一层XGBoost模型训练，利用XGBoost模型中树的叶子节点构造新特征，将新特征进行编码后导入第二层LightGBM分类模型继续训练，以此获得最终的检测结果。该方法能够检测出新型恶意软件，克服传统人工提取特征数量少、质量低、耗时费力的缺点，在保证检测准确率的同时，尽可能的缩短检测时间，提高检测的效率。

1 相关研究进展和问题分析

1.1 当前研究方法

针对移动端恶意软件的检测，传统的数据采集和检测框架模型如图1所示，主要分为3个阶段。

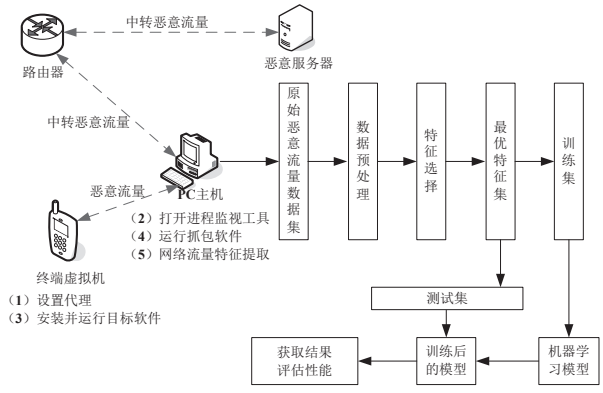


图 1 传统数据采集和检测模型框架

1）第1阶段获取目标应用程序网络数据流和特征。

（1）配置网络环境。在虚拟终端操作环境中配置代理，将代理设置为本地主机IP，确定抓包工具默认使用的端口号。

（2）使用进程监视工具仅对目标虚拟机进行监视，减少抓包过程中的冗余流量。

（3）在虚拟环境中安装并运行目标程序之前，检查后台有无其它程序和服务，若有则进行关闭，避免冗余流量的混入。

（4）本地PC主机运行抓包工具，并在移动端虚拟机中运行目标应用程序，获得网络流量。

（5）获取数据包后，利用特征提取工具提取每条数据流的发包速度、持续时间等特征。

2）第2阶段进行特征处理和模型训练。对原始网络流量数据进行预处理，使用特征选择的方法去除无关特征，以获得对模型贡献度高的特征集合，然后划分训练集和测试集，将训练集输入分类模型，通过参数优化获得最佳训练模型。

3）第3阶段对训练后的模型进行测试。将测试集导入到训练好的模型中，得到预测结果，将其与真实结果进行比对，使用多种模型评价指标对分类模型性能进行全方面评估。

常规的模型检测框架中，影响模型检测效果和运行时间的因素主要是特征的处理和分类算法的使用。特征的处理主要包括特征选择和特征创造。常见的特征选择方法有过滤法、嵌入法、包装法，以及遗传算法、基于群体智能的特征选择方法、基于随机森林的特征选择法等[8]；常见的特征创造方法有主成分分析法、核化线性降维法、基于树模型的特征创造法、基于神经网络的自动编码器特征创造法等。基于机器学习的恶意软件检测方法在模型选择方面大多使用单一的机器学习模型，常见的是随机森林算法（Random Forest，RF）、梯度提升树算法（GBDT）、支持向量机算法（Support Vector Machine，SVM）、神经网络算法等。将大量训练数据导入模型训练后，通过测试数据获取训练模型的检测效果。

1.2 目前研究中存在的问题

获取的目标应用软件网络流量特征中存在较多冗余特征，需对其进行特征处理。

1）使用过滤法进行特征选择时，需根据设置的阈值大小进行特征过滤。阈值设置太大将直接导致有效特征被过滤掉，阈值设置太小将导致冗余特征过多，故需要不断观察不同阈值过滤后的特征对模型的影响，此过程在特征维度较高时需花费大量时间。

2）嵌入法将特征选择与具体的模型训练过程同时进行，也存在最优阈值获取问题，且当模型算法复杂时需要耗费大量时间。

3）包装法利用特定的数据挖掘算法进行多轮迭代，剔除贡献度最低的特征来保留最佳特征集，虽能极大提高模型效果但计算开销较大。

4）群体智能算法应用于特征选择时，需通过启发式搜索策略获取全局最优特征集，虽能获得较好的特征集，但在数据集过大情况下存在迭代次数多、花费时间长等问题[9]。

5）常规的降维法可减少特征数量来加速模型的训练，但会丢失部分信息，这些信息可能是不必要的噪声，也可能是对预测结果有重要贡献度的信息，因此用传统降维法来创造特征并不一定能提高模型的最终性能。

在机器学习模型使用方面，当前研究多数使用单一的机器学习算法。传统的机器学习算法受数据集影响较大，虽能在一定程度上达到较高的检测率，但模型的泛化能力较差；神经网络算法存在参数难调整、训练耗时长等问题，模型最终的预测效果受隐藏层数目和每层神经元个数影响较大。无论是传统的机器学习还是新兴的深度学习，存在的根本问题是需要不断的衡量并尝试去解决偏差和方差问题。

2 解决方案

2.1 XGBoost 算法

极限梯度提升树(Extreme Gradient Boosting，XGBoost）算法是GBDT算法的进化形式，其基学习器通常选择决策树模型，通过不断迭代生成新树学习真实值与当前所有树预测值的残差，将所有树的结果累加作为最终结果，以此获取尽可能高的分类准确率。

XGBoost算法将模型的表现与运算速度的平衡引入目标函数，在求解目标函数时对其做二阶泰勒展开，以此加快求解速度，减少模型运行时间；同时引入正则项控制模型复杂度，避免过拟合。XGBoost算法的目标函数如公式（1）所示：

$$obj=\sum_{i=1}^{n}l\left(y_i,\hat{y}_i\right)+\sum_{k=1}^{K}\theta\left(f_k\right)=\sum_{i=1}^{n}\left[f_t\left(x_i\right)g_i+\frac{1}{2}\left(f_t\left(x_i\right)\right)^2h_i\right]+\theta\left(f_t\right) \quad (1)$$

其中，g_i和h_i分别为损失函数的一阶、二阶导数，$\theta\left(f_t\right)$为第t棵树的结构。

令T为新建树的叶子节点的总数，w为每个叶子节点的样本权重，γ和λ为权重参数，j为每个叶子节点的索引，索引为j的叶子节点上包含的样本集合为I_j，令$\theta(f)=\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}w_j^2$，$w_j=-\frac{\sum_{i\in I_j}g_i}{\sum_{i\in I_j}h_i+\lambda}$，则有

$$obj=-\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j}g_i\right)^2}{\sum_{i\in I_j}h_i+\lambda}+\gamma T \quad (2)$$

XGBoost算法在构建每棵树的过程中，通过计算所有候选特征分枝前与分枝后结构分数之差obj_{split}，从中挑选出obj_{split}最大的特征进行分枝，保证生成的树最优，以此提高预测准确率。图2展示了XGBoost算法在高维特征下单棵二叉树的构建过程，图2a）中根节点（ROOT）包含全部样本集，此处以A、B、C、D、E、F、G、H共8个样本为例。首先根据选择的f_1特征将样本集划分为右叶子节点（*Node*1）和左叶子节点（*Node*2），再挑选候选特征集中obj_{split}最大的特征继续对节点进行分裂，最终将样本F、G、H分配到*Node*1节点，样本D、E分配到*Child*1节点，样本A、B、C分配到*Child*2节点，最终生成的树结构如图2b）所示，其中，V表示当前节点分裂特征在此树中的特征值。

根据公式（2）可知，XGBoost算法中最终目标函数的大小与树结构密切相关，最终模型效果直接与叶子节点有关，在寻找最优树结构的过程中可确定叶子节点的数量。根据XGBoost算法在不断迭代构建每

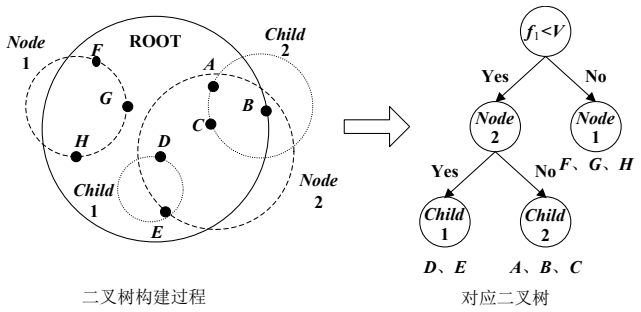


图 2 高维特征下二叉树的构建过程

棵树时选择最优特征和最优切分点的过程，可使用XGBoost算法来创造新特征组合。

2.2 LightGBM 算法

轻量级梯度提升机（Light Gradient Boosting Machine，LightGBM）算法是GBDT算法的另一种进化形式，该算法使用深度限制的叶子生长（leaf-wise）策略，从当前叶子节点中找到增益值最大的节点进行分裂，并对树的深度进行限制，防止过拟合，缩短寻找最优深度树的时间。同时保证分裂次数相同的情况下，能够降低误差，得到更高精度。在构建树的过程中，最浪费时间和计算机资源的是寻找最优分裂节点的过程，对此，LightGBM使用直方图算法、单边梯度抽样算法（GradientBased One-side Sampling，GOSS）和互斥特征捆绑算法（Exclusive Feature Building，EFB）来提升运行效率[10]。

在特征维度较高和样本较多的情况下，传统的分类算法运算效率低下，且最终模型预测的准确率得不到保障；新型深度学习算法在处理大数据集时运算速度缓慢，且模型预测准确率受其参数影响较大。LightGBM算法具有训练速度快、消耗内存少、预测精度高的特点，能够克服两者的缺点，在训练速度和准确率之间达到平衡，适合实时检测系统的需要。

2.3 XGBoost 和 LightGBM 双层融合模型

2.3.1 双层模型的恶意软件检测框架

在样本数量一定的情况下，受原始特征维度、特征间线性关系的影响，单一的机器学习模型对原始特征集进行训练往往达不到最优预测度。使用双层融合模型可在创造大量非线性特征的前提下，最大限度提升模型整体的预测效果，故本文使用XGBoost和LightGBM双层模型的恶意软件检测方法。利用首层XGBoost模型创造新特征集后将数据导入第二层LightGBM模型中进一步训练，完成最终恶意软件的检测。新特征集融合了原始特征的多个组合关系，会对模型产生更高的贡献度，故该方法通常会产生比单模型更好的分类检测效果。同时，该检测方法大大减少了人工创造特征的开销，提高了模型检测的实时性和准确率，增强了模型的泛化能力。本文模型框架如图3所示，具体步骤如算法1所示。

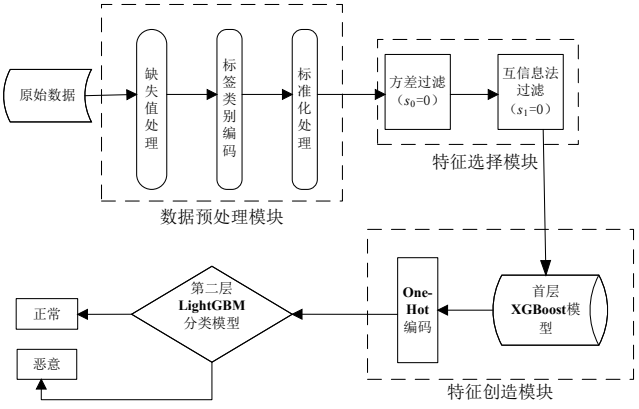


图 3 本文模型检测框架

算法1 基于XGBoost和LightGBM双层模型的恶意软件检测算法

输入：原始网络流量数据集D，原始特征集F。

输出：模型评估指标集A。

1）对F进行数据预处理。F中存在的缺失值通过填补数值0的方式处理；对F中的标签malware和normal进行编码处理，将类别转化为对应的数值0和1；对F进行标准化处理，以提升训练模型精度和加快求解速度。

2）对F进行特征选择。使用方差过滤法（令阈值s_0=0）和互信息法（令阈值s_1=0）进行两轮特征选择，在最短时间内剔除与标签完全独立的特征和对模型预测毫无贡献度的特征。由此得到新特征集F_1和新数据集D_1。

3）创造新特征集。将数据集D_1输入到首层XGBoost模型，进行参数优化后得到最终模型M_1。获取各样本在模型M_1中对应的叶子节点索引，对其进行One-Hot编码，得到新特征集F_2和新数据集D_2（详见算法2）。

4）将新数据集D_2和新特征集F_2导入第二层LightGBM分类模型训练，进行参数优化后得到该模型的评估指标集A，完成最终恶意软件分类模型构建。

2.3.2 利用首层 XGBoost 模型构造新特征集

当前研究多数使用GBDT算法来构建新特征组合，GBDT算法通过迭代构造多棵串行决策树，不断减小残差、拟合真实值。由于决策树有较强的非线性组合能力，故可通过构建多棵决策树来创造新的特征组合。原始样本数据导入到GBDT模型后，将每个样本点在该模型中每棵决策树的叶子节点位置作为高级特征集，该特征创造方法在提升模型预测能力的同时也减少了大量的人工开销[11]。但GBDT算法继承了传统Boosting算法的缺点，忽略了模型的泛化误差，导致模型在未知数据集上的表现较差。利用GBDT算法构建新特征集时同样存在该问题，在生成最优特征集过程中，充分考虑到各个特征间的非线性组合关系，使得模型中树的深度较大、数量较多。此时模型重点关注对少数样本有区分度的特征，忽略了对多数样本有区分度的特征，最终会降低新特征集整体对模型的贡献度[12]。

XGBoost算法与GBDT算法最大的不同在于求解目标函数的过程中考虑到模型的复杂度。在最小化目标函数迭代构建多棵决策树的过程中，既考虑到模型的偏差，又考虑到模型的方差，同时借鉴了随机森林的列采样方法，在目标函数中引入正则项对其复杂度进行控制，并在评估器中引入抛弃提升树（Dropouts meet Multiple Additive Regression Trees，DART）来克服树模型天生过拟合的缺点。利用XGBoost算法的上述特点来构造新的特征组合，控制树的深度和数量等影响模型复杂度的关键因素，使得新特征集包含区分少数样本和多数样本的特征，提高新特征集对模型的贡献度，同时缩短寻找高效特征组合的运行时间，达到效果与效率的双平衡。首层XGBoost模型构造新特征集的过程如算法2所示。

算法2 首层XGBoost模型构建新特征集

输入：原始网络流量数据集D，原始特征集F。

输出：新网络流量数据集D_1，新特征集F_1。

1）将原始网络流量数据集D和原始特征集F导入XGBoost模型，通过图2的方式构造多棵决策树，提升模型预测效果。

2）使用网格搜索调参方式获取最优参数组合，得到最佳XGBoost模型（详见算法3）。

3）根据当前最佳XGBoost模型的结构，获取数据集D中每个样本点在各棵树中叶子节点的位置，并进行One-Hot编码，即包含该样本点的叶子节点用1表示，否则用0表示。将所有叶子节点对应的One-Hot编码横向拼接得到各个样本对应的新特征集F_1，新网络流量数据集D_1。

XGBoost算法构造新特征集时，每棵树的叶子节点可看作是这棵树在寻找最优切分点时所选择特征的组合，每次迭代生成的树是在上棵树学习到的组合特征基础上再去学习新特征组合，以此不断拟合真实值，最终生成树的叶子节点就是对原始特征的多维组合。将叶子节点看作新创造的特征，这些新特征既包含了特征与标签之间的线性与非线性关系，也包含了标签中各类别间的关系。此处以包含4个样本（A、B、C、D），5个原始特征（m_1、m_2、m_3、m_4、m_5）的数据集D为例，结合图2中XGBoost算法在高维特征下单棵二叉树的构建过程，以3棵二叉树（T_1、T_2、T_3）的构建为例展示数据集D在XGBoost算法中构造新特征集的过程，如图4所示。其中V_1、V_2、V_3、V_4、V_5、V_6、V_7、V_8为当前节点分裂特征在此树中的特征值。

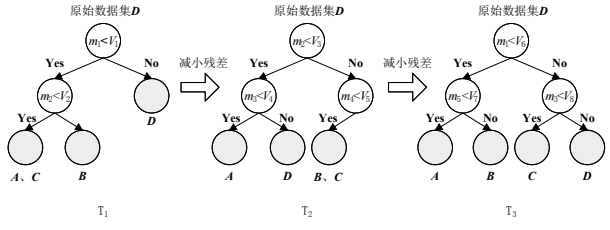


图 4 XGBoost 算法构造新特征过程

在XGBoost算法构造的每棵决策树中，从根节点到叶子节点的路径可看作是一个决策过程，每棵树的叶子节点可看作是交叉特征。如图4决策树T_1中，（$m_1<V_1$）&（$m_2<V_2$）的决策过程使得A、C样本落在同一叶子节点，该叶子节点可看作是A、C样本的共同高阶特征。获取样本点落在各棵树中所在的叶子节点位置，将其进行One-Hot编码后横向拼接的向量组合作为该样本的新特征集。如图4中，T_1有3个叶子节点，则A样本在T_1中对应的新特征向量为[1,0,0]；T_2有3个叶子节点，则A样本在T_2中对应的新特征向量为[1,0,0]；T_3有4个叶子节点，则A样本在T_3中对应的新特征向量为[1,0,0,0]。将3棵树的新特征向量横向拼接得到A样本在XGBoost算法中产生的新特征集为[1,0,0,1,0,0,1,0,0,0]，由此得到包含4个样本和新特征组合的数据集D_1，如表1所示。

表 1 包含新特征组合的数据集 D_1

样本	新特征集（One-Hot 编码）									
A	1	0	0	1	0	0	1	0	0	0
B	0	1	0	0	0	1	0	1	0	0
C	1	0	0	0	0	1	0	0	1	0
D	0	0	1	0	1	0	0	0	0	1

2.3.3 获取首层 XGBoost 模型最优参数组合

上述利用XGBoost算法叶子节点创造新特征集的过程中，影响最终新特征集质量的关键因素是决策树个数（*n_estimators*）。决策树个数少，会导致新创造的特征组合不充分，降低新创造特征集对模型的贡献度；决策树个数多，会导致新特征集维度较高，模型训练速度较慢，且可能会出现多个冗余的特征组合。故选取合适的决策树个数对新特征集的创造至关重要，本文采用网格搜索的参数优化方式获取首层XGBoost模型中最优决策树个数。网格搜索是一种穷举式调参方式，需事先指定好参数范围，然后循环遍历给定范围内的参数，尝试所有可能性，最终选择预测结果最好的参数，利用此方法可获取到所有参数最优组合。首层XGBoost模型的参数优化流程如算法3所示。

算法3 获取首层XGBoost模型最优参数组合

输入：原始网络流量数据集D，原始特征集F。

输出：最优参数组合P。

1）将原始数据集D和特征集F导入XGBoost模型，并将较高的学习率（*eta*）初始化，选择其默认值为0.3。

2）确定决策树的数量（*n_estimators*）。设定*n_estimators*参数范围为（0,200），通过分析不同决策树个数对首层XGBoost模型评估指标的影响，确定最优*n_estimators*的大致范围，进而缩小候选参数范围，细化学习曲线，获得最终的参数优化结果。

3）设定树的深度（*max_depth*）参数范围为（3,10），最小化叶子节点权重（*min_child_weight*）参数范围为（1,6），决策树节点分裂时的最小损失减少量（*gamma*）参数范围为（0,1），数据集样本上子样本比例（*subsample*）和列采样比例（*colsample_bytree*）参数范围均为（0.5, 0.9）。

4）通过绘制并细化各参数与模型评估指标间的学习曲线，获取步骤3）中所涉及参数的最优组合。

5）重新调整*eta*值，提高模型预测效果，最终得到首层XGBoost模型所有参数的最优组合P。

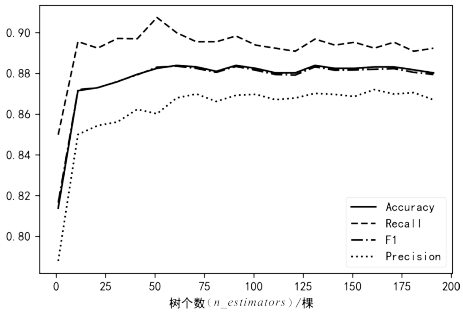
本文使用加拿大网络安全研究所的CICInvesAndMal2019数据集，挑选其中的1000条样本数据用于举例分析XGBoost算法的优化调参过程。为最大限度的缩短模型的运行时间，保证新创造特征集的质量，提高新创造特征集对第二层模型的贡献度，本文重点对首层XGBoost算法的*n_estimators*参数进行优化。通过网格搜索的调参方式获取到最终首层XGBoost模型的最优参数组合P如表2所示。

表 2 首层 XGBoost 模型最优参数组合 P

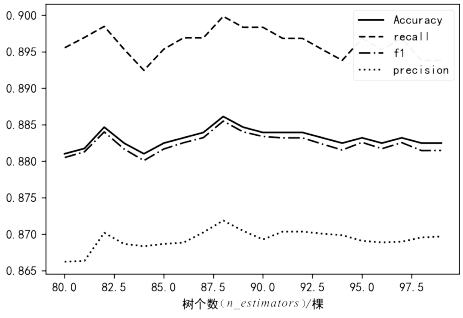
参数	最优值
n_estimators	88
eta	0.2
max_depth	6
min_child_weight	1
gamma	0
subsample	0.9
colsample_bytree	0.5

首层XGBoost模型参数优化举例如图5所示。图5a）绘制了XGBoost算法中决策树个数与模型评估指标

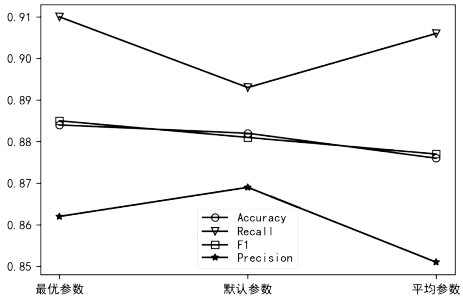
之间的关系图，当决策树个数在（50,75）间时，其*Recall*值大大增加，但*Precision*值较低。综合考虑检测效率和运行时间，可确定其最优*n_estimators*参数范围在（80,100）之间。进一步分析*n_estimators*参数取值在（80,100）时对首层模型整体的影响，由图5 b）最终确定首层XGBoost模型中决策树个数为88。为充分证明表2最优参数组合对首层XGBoost模型的优化效果，在此使用同一数据集分别对包含默认参数和平均参数组合的XGBoost模型进行训练，如图5 c）所示，对比发现，在最优参数组合下，首层XGBoost模型的效果在整体上能达到最优。



a）获取最优 *n_estimators* 参数值范围



b）获取最优 *n_estimators* 参数值



c）不同参数组对首层 XGBoost 模型影响

图 5 首层 XGBoost 模型参数优化举例

表 3 本文所用网络流量数据集的主要特征

序号	特征名称	特征描述
1	fl_dur	数据流持续时间
2	tot_fw_pk	转发总包数
3	tot_bw_pk	反向转发总包数
4	tot_l_fw_pkt	报文总大小
5	fw_pkt_l_max	正向报文最大数据包大小
6	fw_pkt_l_min	正向报文最小数据包大小
7	fw_pkt_l_avg	正向报文平均大小
8	fw_pkt_l_std	数据包正向的标准偏差大小
9	Bw_pkt_l_max	反向最大包数
10	Bw_pkt_l_min	反向最小包数
⋮	⋮	⋮

3 实验分析

3.1 实验数据集准备和预处理

本文使用加拿大网络安全研究所的CICInvesAndMal2019数据集进行基于网络流量的移动端恶意软件检测实验，从应用软件网络流量部分选取并整合了1300条数据集，其中正常流量和恶意流量各占50%以解决分类学习中的类别不平衡问题。该数据集中含有42个恶意软件家族以及大量的良性软件，并捕获了安装运行软件中、重启手机前后3个阶段的双向网络流量，使用CICFlowMeter工具获取到80个网络流量特征[13]，主要特征如表3所示。与其他数据集相比，该数据集能够较为全面准确的反映恶意软件的隐藏行为。

3.2 实验环境和模型评价指标

本文实验环境为一台内存为12GB、搭载2.90GHz的Intel i5-4210H CPU、1T硬盘空间、安装Windows10系统的电脑。编程语言和平台版本为Python 3.8.0，主要的机器学习库为scikit-learn 0.22.1、XGBoost 1.0.2、LightGBM 2.3.1。

为充分衡量模型的检测精度、泛化能力和运行效率，本文使用准确率（*Accuracy*）、查准率（*Precision*）、查全率（*Recall*）、*F*1值、模型检测时间来多方面评估模型的优劣。对于二分类问题，通过如表4所示混淆矩阵可清楚地反映真实值与预测值划分情况。

3.3 实验结果与分析

为充分证明本文提出的XGBoost和LightGBM双层模型在恶意软件检测领域的实际效果，实验中使用上

表 4 二分类问题混淆矩阵

真实类别	正例	反例
正例	*TP*（真正例）	*FP*（假正例）
反例	*FN*（假反例）	*TN*（真反例）

述预处理后的数据集，采用10折交叉验证方法，分别将数据集导入到SVM、RF、自适应梯度提升树（Adaptive Boosting，AdaBoost）、GBDT、XGBoost、LightGBM、多层感知机（MLP）以及本文提出的模型中，将本文提出的双层模型与多个单独的机器学习模型分类效果进行比较分析，实验结果如表5和图6所示。

表 5 不同单模型分类效果对比

模型方法	*Accuracy*	*Recall*	*F*1	*Precision*
SVM	0.583	0.897	0.677	0.545
RF	0.814	0.859	0.821	0.802
AdaBoost	0.842	0.858	0.841	0.826
GBDT	0.880	0.896	0.879	0.864
XGBoost	0.886	0.900	0.885	0.871
LightGBM	0.875	0.904	0.876	0.851
MLP	0.816	0.837	0.816	0.800
本文模型	0.945	0.950	0.946	0.942

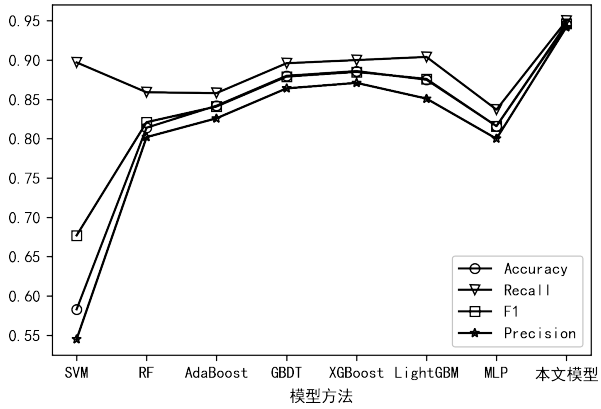


图 6 单模型分类实验结果对比

对实验结果进行分析，发现本文模型无论是准确率、查准率、查全率还是*F*1值都显著高于其它单一模型。相比于RF、AdaBoost、XGBoost、LightGBM这些基于树的单一集成学习模型，本文提出的双层模型在*F*1值上分别高出12.5%、10.5%、6.1%和7%，表明其在恶意软件检测上取得了良好的分类效果。

在考虑模型的分类性能后，对上述模型的运行时间进行比较分析。本文模型的运行时间是指对目标数据进行10折交叉验证划分后导入模型训练预测的时间，如表6所示。

表 6 不同单模型运行时间对比

模型	SVM	RF	AdaBoost	GBDT	XGBoost	LightGBM	MLP	本文模型
运行时间 /s	5.25	13.26	21.57	56	11.59	9.27	191	9.87

实验对比发现SVM模型运行时间最短，但预测的准确率较低；MLP模型由于具有多个隐藏层和神经元其运行时间太长不适用于实时检测；其它几个基于树的单一学习模型中，只有LightGBM的运行时间比本文模型稍短一些。通过对比其分类效果，综合分析发现，本文模型在保证分类效果最好的情况下拥有较短的运行时间。

另外，在使用相同数据集的前提条件下，本文在实验中分别使用RF+逻辑回归（LR）双层模型、RF+XGBoost双层模型、RF+LightGBM双层模型、XGBoost双层模型来验证分类的效果，对比结果如表7和图7所示。

表 7 不同双层融合模型分类效果对比

模型方法	*Accuracy*	*Recall*	*F*1	*Precision*
RF+LR	0.942	0.954	0.943	0.933
RF+XGBoost	0.904	0.925	0.906	0.893
RF+LightGBM	0.903	0.916	0.904	0.899
XGBoost+XGBoost	0.944	0.932	0.945	0.944
本文模型	0.945	0.950	0.946	0.942

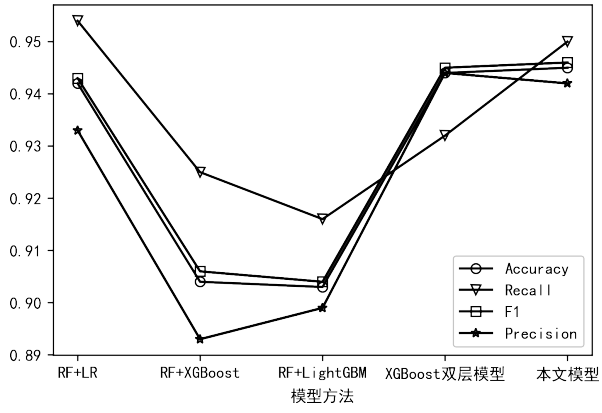


图 7 融合模型分类实验结果对比

通过上述实验结果对比发现，本文提出的双层融合模型在某些评价指标上略低于其它双层融合模型。这主要是由第一层模型创造的特征决定的，极少数新特征之间不可避免的存在共性甚至负相关，会影响新

特征整体对模型的贡献度。从整体来看，本文双层融合模型的*Accuracy*、*Recall*、*F*1、*Precision*值较为稳定，整体性能上优于其他模型。同时，双层融合模型的效果一般要优于单层模型的效果。

多个双层融合模型的运行时间进行对比，如表8所示，RF+LR模型的运行时间最短，但其*Precision*值有待进一步提高；其他几个模型的运行时间比本文模型运行时间长，不能满足实时检测的时间要求。因此，本文提出双层融合模型在运行时间、检测准确率和稳定性方面表现良好。

表 8 不同双层模型运行时间对比

模型	RF+LR	RF+XGBost	RF+LightGBM	XGBoost+XGBoost	本文模型
运行时间 /s	6.58	112	12.38	87	9.87

4 结束语

本文将XGBoost与LightGBM双层融合模型应用于恶意软件检测领域，通过将原始网络流量特征导入到第一层XGBoost模型中进行训练来创造新特征集，将新特征集导入到第二层LightGBM模型中去预测最终的分类结果。使用移动端应用软件网络通信流量进行实验，结果表明，本文提出的双层融合模型在分类效果和运行时间上达到了整体平衡，优于单一的机器学习模型和绝大多数双层融合模型，能够满足实时检测系统的需要。但实验中使用的网络流量样本数据有限，未能够充分体现出本文双层融合模型在处理大数据集时的时间优势，下一步将收集更多的网络流量进行模型的训练预测，并通过实验探究本文模型在处理大数据集中的优势所在。另外，未来将会考虑使用其它融合模型对不同家族的恶意软件进行分类，以便对目标恶意软件的功能进行细化，为后续的软件防护提供更多的信息。（责编 姜丽辉）

参考文献：

[1] CUI Yanpeng, YAN Bo, HU Jianwei. Android Malware Detection Method Based on Abstract API Call Sequence[J]. Computer Applications and Software, 2019, 36(9): 321–326.
崔艳鹏，颜波，胡建伟 . 基于抽象 API 调用序列的 Android 恶意软件检测方法 [J]. 计算机应用与软件，2019，36（9）：321–326.
[2] HOU Liuyang, LUO Senlin, PAN Limin, et al. Multi–feature Android Malware Detection Method[J]. Netinfo Security, 2020, 20(1): 67–74.
侯留洋，罗森林，潘丽敏，等 . 融合多特征的 Android 恶意软件检测方法 [J]. 信息网络安全，2020，20（1）：67–74.
[3] ZHANG Zhen, CAO Tianjie. Detecting Android Malware Based on Matching Sequence of Behavioral Characteristic Value[J]. Computer Engineering andApplications, 2018, 54（24）：97–102.
张震，曹天杰 . 行为特征值序列匹配检测 Android 恶意应用 [J]. 计算机工程与应用，2018，54（24）：97–102.
[4] ZHANG Chaoqin, HU Guangwu, WANG Zhenlong, et al. A Novel SVM–Based Detection Method For Android Malware[J]. Computer Applications and Software, 2018, 35(10): 292–298.
张超钦，胡光武，王振龙，等 . 一种基于支持向量机的安卓恶意软件新型检测方法 [J]. 计算机应用与软件，2018，35（10）：292–298.
[5] LI Hao, MA Kun, CHEN Zhenxiang, et al. Unknown Malware Detection Based on Network Traffic Analysis[J]. Journal of Jinan University, 2019, 33(6): 500–505.
李浩，马坤，陈贞翔，等 . 基于网络流量分析的未知恶意软件检测 [J]. 济南大学学报（自然科学版），2019，33（6）：500–505.
[6] WANG Shuwei, ZHANG Linjie, JIA Zhe, et al. Android Malware Recognition Based on Network Traffic[J]. Radio Engineering, 2020, 50(7): 612–618.
王澍玮，张林杰，贾哲，等 . 基于网络流量的安卓恶意软件识别 [J]. 无线电工程，2020，50（7）：612–618.
[7] PENG Guojun, LI Jingwen, SUN Runkang, et al. Android Malware Detection Research and Development[J]. Journal of Wuhan University (Science Edition), 2015, 61(1): 21–33.
彭国军，李晶雯，孙润康，等 .Android 恶意软件检测研究与进展 [J]. 武汉大学学报（理学版），2015，61（1）：21–33.
[8] LI Zhanshan, LIU Zhaogeng. Feature Selection Algorithm Based on XGBoost[J]. Journal on Communications, 2019, 40(10): 101–108.
李占山，刘兆赓 . 基于 XGBoost 的特征选择算法 [J]. 通信学报，2019，40（10）：101–108.
[9] WANG Shengwu, CHEN Hongmei. Feature Selection Method Based on Rough Sets and Improved Whale Optimization Algorithm[J]. Computer Science，2020, 47(2): 44–50.
王生武，陈红梅 . 基于粗糙集和改进鲸鱼优化算法的特征选择方法 [J]. 计算机科学，2020，47（2）：44–50.
[10] BIAN Lingyu, ZHANG Linlin, ZHAO Kai, et al. Ethereum Malicious Account Detection Method Based on LightGBM[J]. Netinfo Security, 2020, 20(4): 73–80.
边玲玉，张琳琳，赵楷，等 . 基于 LightGBM 的以太坊恶意账户检测方法 [J]. 信息网络安全，2020，20（4）：73–80.
[11] HE X R, PAN J F, JIN O, et al. Practical Lessons from Predicting Clicks on Ads at Facebook[C]// ACM SIGKDD. The 8th International Workshop on Data Mining for Online Advertising, August 24, 2014, New York，USA. New York：ACM SIGKDD, 2014: 1–9.
[12] WANG Yao, LI Wei, WU Kehe, et al. Application of Fusion Model of GBDT and LR in Encrypted Traffic Identification[J]. Computer and Modernization, 2020(3): 93–98.
王垚，李为，吴克河，等 .GBDT 与 LR 融合模型在加密流量识别中的应用 [J]. 计算机与现代化，2020（3）：93–98.
[13] LAYA T H, ANDI F A, ARASH H L. Extensible Android Malware Detection and Family Classification Using Network–flows and API–calls[J]. The IEEE(53rd) International Carnahan Conference on Security Technology, 2019, 4(1): 26–30.