计算机科学
COMPUTER SCIENCE

一种基于异质模型融合的 Android 终端恶意软件检测方法

姚烨，朱怡安，钱亮，贾耀，张黎翔，刘瑞亮

引用本文

姚烨，朱怡安，钱亮，贾耀，张黎翔，刘瑞亮. 一种基于异质模型融合的 Android 终端恶意软件检测方法[J]. 计算机科学, 2022, 49(6A): 508-515.

YAO Ye, ZHU Yi-an, QIAN Liang, JIA Yao, ZHANG Li-xiang, LIU Rui-liang. Android Malware Detection Method Based on Heterogeneous Model Fusion[J]. Computer Science, 2022, 49(6A): 508-515.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210700103

一种基于异质模型融合的 Android 终端恶意软件检测方法

姚 烨 朱怡安 钱 亮 贾 耀 张黎翔 刘瑞亮
西北工业大学计算机学院 西安 710129
(yaoye@nwpu.edu.cn)

摘 要 针对单一分类模型检测精度有限的问题,提出了一种基于异质模型融合的 Android 恶意软件检测方法。首先识别和采集恶意软件混合特征信息,采用基于 CART 决策树的随机森林算法和基于 MLP 的 Adaboost 算法分别构造集成学习模型,然后通过 Blending 算法对这两个分类器进行模型融合,最后得到一种异质模型融合分类器,在此基础上实施移动终端恶意软件检测。实验结果表明所提方法能够有效克服单一分类模型检测精度不足的问题。

关键词:Android 系统;恶意软件;模型融合;机器学习;移动终端
中图法分类号 TP391.9

Android Malware Detection Method Based on Heterogeneous Model Fusion

YAO Ye,ZHU Yi-an,QIAN Liang,JIA Yao,ZHANG Li-xiang and LIU Rui-liang
School of Computer Science,Northwestern Polytechnical University,Xi'an 710129,China

Abstract Aiming at the problem of limited detection accuracy of a single classification model,this paper proposes an Android malware detection method based on heterogeneous model fusion. Firstly,by identifying and collecting the mixed feature information of malicious software,the random forest algorithm based on CART decision tree and the Adaboost algorithm based on MLP are used to construct the integrated learning model respectively,and then the two classifiers are fused by Blending algorithm. Finally,a heterogeneous model fusion classifier is obtained. On this basis,the mobile terminal malware detection is implemented. Experimental results show that the proposed method can effectively overcome the problem of insufficient accuracy of single classification model.

Keywords Android system,Malware,Model fusion,Machine learning,Mobile terminal

1 引言

随着移动互联网的快速发展和便携、可穿戴设备的迅速普及,以 Android 系统为代表的移动端的市场占有率越来越高。根据"中国互联网信息中心"2020 年第 46 次《中国互联网络发展状况统计报告》[1]分析,2020 年上半年受疫情冲击,我国就业压力显著增大,互联网平台经济下的新业态、新模式增加了大量就业岗位,正在成为支撑就业的重要力量。上半年,我国个人互联网应用呈现平稳增长态势,截止到 6 月份,我国网民规模达 9.40 亿,互联网普及率达 67.0%;我国网民使用手机上网的比例达 99.2%。

Android 系统的高开放性和高普及性,使得它成为恶意攻击者的主要目标。近年来,针对 Android 平台的恶意软件层出不穷,给消费者的个人隐私和安全造成巨大威胁。"360 烽火实验室联合 360 安全大脑"在 2020 年发布的《2019 年 Android 恶意软件专题报告》[2]指出,2019 年全年,360 安全大脑共截获移动端新增恶意程序样本约 180.9 万个,平均每天截获新增手机恶意程序样本约 0.5 万个。新增恶意程序类型主要为资费消耗,占比 46.8%;其次为隐私窃取(41.9%)、远程控制(5.0%)、流氓行为(4.6%)、恶意扣费(1.5%)、欺诈软件(0.1%)。2019 全年从漏洞数量看,Android 系统以 414 个漏洞位居产品漏洞数量榜首。因此,针对 Android 系统恶意软件的有效检测一直是业界的一大难题。

当前,国内外研究人员针对 Android 恶意软件的检测技术研究主要分为两个方面:一方面是基于静态特征的检测技术,研究者通过解析 APK 文件,分析权限申请、函数调用和数据流等手段获取相关静态特征[3-6];另一方面是基于动态特征的检测技术,研究者通过在真机或模拟器中安装并运行应用程序,实时监控应用程序在运行过程中的行为数据,以此获取相关动态特征。

在分类器模型的选择上,国内外研究者一般采用从单一分类模型到多分类模型结合分类的方法。Zhang 等[7]提出了一种基于 Markov 链及支持向量机 SVM 的检测方法,该方法利用 Markov 链将 API 调用序列转化为特征向量,并将其

基金项目:国家重点研发计划(2020YFB1712200);陕西省重点研发(重点产业链)项目(2019ZDLGY12-07);西安市科技计划项目(GXYD192.1);太仓市大院大所创新项目(TC2019DYDS06);东莞市科技装备动员项目(KZ2018-14)
This work was supported by the National key Research and Development Program of China(2020YFB1712200),Key Research Development Plan of Shaanxi Province of China(2019ZDLGY12-07),Xi'an City Science and Technology Plan Project of China(GXYD192.1),Innovation Leading Project of Taicang City of China (TC2019DYDS06) and Dongguan City Science and Technology Equipment Mobilization Project of China(KZ2018-14).
通信作者:朱怡安(zhuya@nwpu.edu.cn)

作为分类模型 SVM 的输入进行训练和检测。Li 等[8]使用卷积神经网络 CNN 和朴素贝叶斯方法对特征进行分类，但是该方法在真实环境下的检测精度并不高。Wang 等[9]从每个应用程序中提取 11 种静态功能以表征应用程序的行为，并采用多分类器集合，即支持向量机(SVM)、K 最近邻(KNN)、朴素贝叶斯(NB)分类以及回归树(CART)和随机森林(RF)，以检测恶意行为。但这些方法都无法突破单一分类模型的精度“瓶颈”问题。

本文的主要贡献体现在以下 4 点：1)提出了一种异质模型融合方法；2)采用基于 Bagging 方法和 Boosting 方法分别构建了两个强分类器；3)将以上两个异质强分类器进行 Blending 融合，得到分类精度更高的融合模型；4)在此基础上实施恶意软件检测，并进行了对比实验，结果表明所提方法能够显著提高 Android 恶意软件的检测准确率。

2　单一分类模型分析

由于决策树算法能够很容易求得局部最优解，而神经网络算法对含噪声数据有较好的适应能力，且对未知数据具有较好的预测分类能力，因此，本文从决策树算法和神经网络算法中各选一个作为基分类器，用于构建两个异质集成强分类器，以满足模型融合对于模型的“差异性”要求。

2.1　决策树与 CART 树

决策树(Decision Tree)基于监督学习，采用一种自上而下的决策模型，其本质是通过一系列规则对数据进行分类的过程，一般分为分类树和回归树两种，分类树适用于离散变量，回归树适用于连续变量。

决策树的构造过程主要分为特征选择、决策树生成和决策树剪枝 3 步。目前主流决策树算法有 ID3，C4.5 和 CART 等，其中 C4.5 和 CART 两种算法从 ID3 算法中衍生而来。由于 CART 算法对应的决策树规模更简单，其生成效率更高，也能够处理本文所提供的数据集(离散型特征向量集)，因此，本文选择 CART 分类树来构造随机森林分类模型。

CART 分类树算法使用 GINI 指数代替信息增益比，GINI 指数代表了模型的不纯度/不确定性，GINI 指数越小，不纯度/不确定性越低，特征越好。

对于数据集 D，设有 k 种分类结果，且第 i 种结果的概率为 p_i，则概率分布的 GINI 指数计算如式(1)所示：

$$Gini(p)=\sum_{i=1}^{k}p_i(1-p_i)=1-\sum_{i=1}^{k}p_i^2 \tag{1}$$

本文面向 Android 恶意软件检测，采用二分类方法，设样本点属于第一类的概率为 p，则二分类问题下概率分布的 GINI 指数计算式如式(2)所示：

$$Gini(p)=2p(1-p) \tag{2}$$

将数据集 D 在特征 A 的条件下切分成 D_1 和 D_2，则在特征 A 的条件下，数据集 D 的 GINI 指数计算如式(3)所示：

$$Gini(D|A)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2) \tag{3}$$

其中，$|D_1|$ 是 D_1 中的样本个数，$|D_2|$ 是 D_2 中的样本个数。本文使用的 CART 分类树的生成算法描述如算法 1 所示。

算法 1　CART 分类树生成算法

输入：恶意软件特征训练数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$ 恶意软件特征特征 A 可能取值的集合 $A=\{a_1,a_2,\cdots,a_d\}$；GINI 指数阈值 ε_1；恶意软件特征样本个数阈值 ε_2

输出：从根结点 node 和初始训练数据集 D 开始，递归地建立 CART 分类树

```
Begin
1. D′←getCurrentSet();
2. if |D′|<ε2 or hasNoMoreFeature then
3.    return
4. end if
5. foreach a from A
6.    g←splitGini(D′,A);
7.    if g<ε1 then
8.       return
9.    else
10.      gmin←getCurrentMinGini(g,A);
11.   end if
12. end for
13. D1,D2←splitSet(D′,gmin);
14. goto 1
End
```

算法 1 具体描述如下：

第 1 行：获取当前特征结点递归的数据集；

第 2－4 行：若当前结点数据集的样本数小于阈值或者没有更多特征，则返回决策子树，停止当前递归；

第 5－13 行：对于当前特征 A 的每个可能取值，根据样本点对是否满足，将当前结点数据集划分成两个部分，并计算特征 A 条件下数据集的 GINI 指数，在所有可能的特征值 A 以及所有可能的切分点中，选择 GINI 指数最小的特征及其对应的切分点作为最优特征与最优切分点，依据最优特征和最优切分点，从当前结点生成两个子结点，并将当前数据集依据特征分配到两个子结点的数据集合中去；

第 14 行：返回第 1 行不断递归执行，直到跳出递归。

在随机森林中，对于每一棵树而言，都需要其在某一片的数据中有非常好的拟合性。换言之，每一棵树都不是对全数据集的拟合，它们只需要在其负责的局部数据上保证最佳拟合效果即可。因此，不需要考虑单棵树的过拟合问题，也就不需要对随机森林中的树进行剪枝操作。故本文不对 CART 分类树的后剪枝操作做过多阐述。

2.2　神经网络算法与 MLP

神经网络(Neural Network)是以人脑中的神经元结构为启发而模仿构造出来的学习模型，是深度学习中的核心算法之一。神经网络优点表现在对噪声数据有较好的适应能力，且对未知数据也具有较好的预测分类能力。

ANN(Artificial Neural Network)作为神经网络的基础，在 CNN(Convolutional Neural Network)和 RNN(Recurrent Neural Networks)以及一系列的衍生算法中的最后层基本都是 Classifier 层，即 FC(Fully Connected)层，用于对前面通过 CNN 和 RNN 处理后获得的特征参数进行最终的分类计算，以获得预测每一个样本标签的概率。

本文主要解决 Android 恶意软件检测问题，即面向分类问题构建分类器，因此选择 ANN 用于构建集成学习模型。

ANN 采用一种多层感知机(Multilayer Perceptron，MLP)技术，除了输入输出层，它中间还可以有多个隐藏层，最简单处理方式只含一个隐藏层。MLP 的层与层之间采用全连接方式，MLP 模型的训练学习过程分为两个部分：一是前向传播，即正向(输入层－＞隐藏层－＞输出层)输入训练

样本得到模型学习的输出结果；二是反向传播，对模型输出结果与样本值进行误差分析，反向(输出层－>隐藏层－>输入层)输入最小化误差以更新每个连接的权重和结点偏置，如图 1 所示。

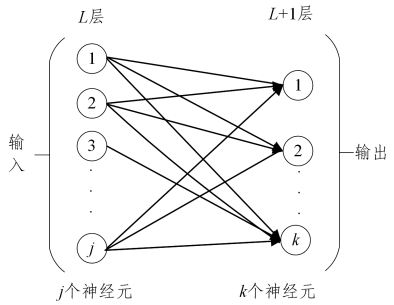

图 1 MLP 网络模型

Fig. 1 MLP network model

3 基于 CART 树构建随机森林集成学习模型

随机森林(Random Forest)算法作为多分类器的代表，在集成学习领域的分类性能表现优异，并在人工智能、知识图谱、建模分析等方面拥有广泛的应用前景[10-11]，随机森林算法的核心原理是 Bagging 思想。

3.1 Bagging 思想

Bagging 的特点在于随机采样，即从训练集里面有放回地采集固定个数的样本。对于 Bagging 算法，一般会随机采集和训练集样本一样个数的样本，使得采样集和训练集样本的个数相同，但是样本内容不同。

在 Bagging 的每轮随机采样中，训练集中大约有 36.8% 的数据没有被采集到，称之为袋外数据(Out of Bag，OOB)。这些数据没有参与训练集模型的拟合，因此可以用来检测模型的泛化能力。

一般来说，Bagging 对于弱学习器没有限制，比较常用的是决策树和神经网络。对于分类问题，其集合策略通常使用简单投票法，即将得票最多的类别作为分类模型的最终输出。Bagging 原理如图 2 所示。

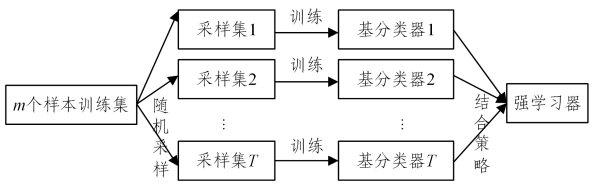

图 2 Bagging 原理图

Fig. 2 Bagging working principle diagram

3.2 基于 CART 分类树搭建随机森林集成模型

采用 Bagging 策略对样本集进行重采样可以产生不同的子数据集，并在其上建立分类器，重复多次就可以获得多个分类器，最后根据这多个分类器的投票结果，决定数据属于哪一类。随机森林算法在 Bagging 的基础上更进一步，具体步骤如下：

Step1 获取随机样本，对样本集进行自助重采样(Bootstrap)，随机选取 n 个样本；

Step2 获取随机特征，从所有特征中随机选出 K 个特征，选择最佳分割特征作为结点建立 CART 分类树；

Step3 重复步骤 1 和步骤 2 m 次，以建立 m 棵 CART 分类树；

Step4 在随机森林中，CART 树并不需要进行剪枝操作；

Step5 将这 m 个 CART 树形成随机森林，通过投票表决分类结果。

设混合特征为：

$$F'_{\text{Hybrid}}=\{F'_{\text{Static}},F'_{\text{Dynamic}}\}=\{f_1',f_2',\cdots,f'_{54},f'_{55},\cdots,f'_{94}\}$$

本文对 $n=8000$ 个 Android 软件样本 R 进行混合特征提取，得到恶意软件训练集 $\boldsymbol{S}_{RF}=\{(x_i,y_i)\mid y_i=\{0,1\},i=1,2,\cdots,n\},(X,Y)\in\boldsymbol{R}^d\times\boldsymbol{R}$。其中，$x_i=F'_{\text{Hybrid}}$ 对应第 i 个样本的混合特征向量，y_i 对应第 i 个样本的分类结果，0 表示非恶意，1 表示恶意，d 在本文中为特征向量维数 94。本文对训练集采用 Bootstrap 重采样 N_{tree} 次，获得 N_{tree} 个子训练集以生成 N_{tree} 棵 CART 分类树。则基于混合特征和 CART 分类树的随机森林集成学习模型构建的算法描述如算法 2 所示。

算法 2 基于混合特征和 CART 分类树的随机森林集成学习模型构建算法

输入：恶意软件特征训练集 $\mathbf{S}=\{(x_i,y_i),i=1,2,\cdots,n\},(X,Y)\in\mathbf{R}^d\times\mathbf{R}$；待检测恶意软件特征样本 $x_t\in\mathbf{R}^d$

输出：

1. CART 树的集合 $\{h_i,i=1,2,\cdots,N_{\text{tree}}\}$
2. 对待检测恶意软件特征样本 $x_t\in R^d$，决策树的输出 $h_i(x_t)$
3. 最终恶意软件特征样本的分类结果

$$f(x_t)=\text{majorityvote}\{h_i(x_t)\}_{i=1}^{N_{\text{tree}}}$$

Begin

For $i=1,2,\cdots,N_{\text{tree}}$

1. 对 S 采用 Bootstrap 抽样，生成训练集 S_i
2. 使用 S_i 生成不剪枝的 CART 树 h_i
   2.1. 从 d 个特征中随机选取 m 个特征
   2.2. 在每个结点上从 m 个特征中依据 GINI 指数选取最优特征
   2.3. 分裂结点直到 CART 树生长到最大

End

其中，CART 分类树构造过程参照算法 1，GINI 指数的计算参照式(3)。

本文构造随机森林集成学习模型时的部分参数设定如表 1 所列。

表 1 随机森林部分参数设定

Table 1 Artial parameter setting of random forest

参数	设定值
n_estimators	100
oob_score	False
criterion	Gini
bootstrap	True
random_state	None
splitter	Random
max_depth	None
max_depth	None
min_samples_leaf	1
min_samples_split	2
max_features	None
min_impurity_split	1×10^{-7}

由此，本文构造了基于 CART 分类树的随机森林集成学习模型 RF_{Hybrid}。

4 基于 MLP 构建 Adaboost 集成学习模型

4.1 Boosting 提升算法

Boosting 算法是将弱学习器提升为强学习器的过程。与

Bagging 并行特性不同的是，Boosting 只能串行进行。Boosting 算法主要分为两个部分：加法模型和前向分步算法。

加法模型：强学习器由一系列弱学习器线性相加而成。一般组合形式如式(4)所示。

$$F_m(x;P)=\sum_{m=1}^{n}h(x;\alpha_m)\beta_m \tag{4}$$

其中，$h(x;\alpha_m)$为弱分类器，α_m 为弱分类器学习到的最优参数，β_m 为弱分类器在强分类器中所占权重；P 是 α_m 和 β_m 的所有组合。

前向分步：在训练过程中，下一轮迭代产生的分类器是在上一轮的基础上训练得来的，如式(5)所示：

$$F_m(x)=F_{m-1}(x)+h(x;\alpha_m)\beta_m \tag{5}$$

采用的损失函数不同，Boosting 算法的类型也不同，而 Adaboost 算法就是将指数损失作为损失函数。

4.2　基于 MLP 搭建 Adaboost 分类模型

依据混合特征 $F'_{\text{Hybrid}}=\{F'_{\text{Static}},F'_{\text{Dynamic}}\}=\{f_1',f_2',\cdots,f'_{54},f'_{55},\cdots,f'_{94}\}$，本文对 $n=10000$ 个 Android 软件样本 R 进行混合特征提取，得到训练集 $S_{Ada}=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$，其中，参数 $x_i=F'_{\text{Hybrid}}\in X\subseteq R^d$ 对应第 i 个样本的混合特征向量，$y_i\in\{-1,+1\}$，$(i=1,2,\cdots,n)$对应第 i 个样本的分类结果，-1 表示非恶意，1 表示恶意，d 为特征向量维数 94。

基于 MLP 的 Adaboost 算法描述如算法 3 所示。

算法 3　基于混合特征和 MLP 的 Adaboost 集成学习模型构建算法

输入：恶意软件特征训练集

$S_{Ada}=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$，$x_i=F'_{\text{Hybrid}}\in X\subseteq R^d$

其中，$y_i\in\{-1,+1\}$，$(i=1,2,\cdots,n)$

迭代次数：M

输出：1. 面向恶意软件的强分类器

$$G(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$$

2. 最终分类预测结果

$$\text{sign}(G(x))=\begin{cases}+1, & G(x)>0\\ -1, & G(x)\leqslant 0\end{cases}$$

Begin

1. 初始化恶意软件特征训练集的权值分布：

$D_1=(w_{1,1},w_{1,2},\cdots,w_{1,n})$，$w_{1,i}=1/n$，$(i=1,2,\cdots,n)$

2. For $m=1,2,\cdots,M$

3. 使用双隐藏层 MLP 在具有权值分布 D_m 的数据集上学习，得到基分类器 $G_m(x)$

4. 计算基分类器 $G_m(x)$在训练集上的分类误差率：

$$e_m=\sum_{i=1}^{n}w_{m,i}I(G_m(x_i)\neq y_i)=\sum_{G_m(x_i)\neq y_i}w_{m,i}$$

5. 计算基分类器 $G_m(x)$在强分类器中所占权重：

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}$$

6. 更新恶意软件特征训练集权值分布：

$$w_{m+1,i}=\frac{w_{m,i}}{z_m}\exp(-\alpha_m y_i G_m(x_i)),\ i=1,2,\cdots,n$$

其中，$z_m=\sum_{i=1}^{n}w_{m,i}\exp(-\alpha_m y_i G_m(x_i))$为归一化因子，使样本的概率分布之和为 1；

End

本文构建的 Adaboost 分类模型参数设定如表 2 所列。

表 2　Adaboost 部分参数设定

Table 2　Bagging working principle diagram

参数	设定值
n_estimators	60
learning_rate	0.8
hidden_layer_sizes	(128,64)
activation	Logistic
random_state	None
solver	Sgd
alpha	0.0001
max_depth	None
learning_rate	Adaptive
learning_rate_int	0.5
max_iter	50
momentum	0.9

由此，本文构造了基于 CART 分类树的 Adaboost 分类模型 $Adaboost_{\text{Hybrid}}$。

5　面向随机森林和 Adaboost 的融合模型搭建

本文在第 3 节和第 4 节中分别构建了基于 CART 分类树的随机森林分类模型和 MLP 多层神经网络的 Adaboost 分类模型。尽管它们已经在实验中取得了很好的分类效果，但是都是属于同质模型之间的集成学习，分类精度提升有限。为了达到更好的分类效果，本文对上述两个异质强分类器进行了模型融合。

5.1　Stacking 与 Blending 算法

目前异质模型融合方法主要分为两种。

(1) Stacking 算法：Stacking[12] 是一种分层模型集成框架，其主要思想是训练模型来学习使用底层学习器的预测结果。该集成方法的思路是：使用初始训练集学习出若干个基模型之后，用这几个基模型的的预测结果作为新的训练集的特征，以训练出新的模型。

(2) Blending 算法：该集成方法的思路与 Stacking 几乎完全一样，唯一不同之处在于 Blending 过程中不进行 k 折验证，而是将原始样本训练集分为训练集和验证集，只针对验证集进行预测，生成的新训练集就只是对验证集预测的结果，而不是对全部训练集生成的预测结果。

Blending 由于不需要进行 k 折交叉验证，因此比 Stacking 简单，不会造成数据穿越，并且可以随时添加其他模型到 Blending 中。而 Stacking 由于使用 k 折交叉验证，比使用单一留出集更加稳健。大量研究表明，Stacking 和 Blending 是目前提升机器学习效果最有效的方法，并且二者的最终效果都相近。

设恶意软件数据集为 $D=\{x_i,y_i\}$，首先将恶意软件数据集 D 划分为恶意软件训练集和恶意软件测试集 D_{Test}，再将恶意软件训练集划分为恶意软件训练集 D_{Train} 和恶意软件验证集 D_{Val}。

则面向异质强分类器的 Blending 算法的基本描述如算法 4 所示。

算法 4　Blending 算法

输入：恶意软件训练集 $D_{\text{Train}}=\{(x_i,y_i)\}$，恶意软件验证集 $D_{\text{Val}}=\{(x_i,y_i)\}$，恶意软件测试集 $D_{\text{Test}}=\{(x_i,y_i)\}$

输出：恶意软件测试集的预测结果 $D_{\text{Test}}^{\text{Predict}}=\{y_1^{\text{Predict}},y_2^{\text{Predict}},\cdots,y_e^{\text{Predict}}\}$

Begin

1. 创建第一层训练模型 $H_1=\{h_1,h_2,\cdots,h_{t_1}\}$；

2. 用恶意软件训练集 D_{Train} 训练第一层模型 H_1，并用训练好的模型预测恶意软件验证集 D_{Val} 和恶意软件测试集 D_{Test}，生成新的恶意

软件训练集 $D_{Train}^{new}=\{(x_i^{new},y_i^{new})\}$ 和恶意软件测试集 $D_{Test}^{new}=\{(x_i^{new},y_i^{new})\}$；

3. 创建第二层训练模型 $H_2=\{h_1,h_2,\cdots,h_{t_2}\}$；

4. 用新恶意软件训练集 D_{Train}^{new} 训练第二层模型 H_2，并用训练好的模型对新恶意软件测试集 D_{Test}^{new} 进行最终预测；

5. 输出最终预测结果

$D_{Test}^{Predict}=\{y_1^{Predict},y_2^{Predict},\cdots,y_e^{Predict}\}$

End

5.2 面向随机森林和 Adaboost 的 Blending 模型融合

将所有数据集划分为训练集 D_{Train} 和测试集 D_{Test}。传统 Blending 方法将训练集 D_{Train} 直接划分出部分(10%)数据作为验证集 D_{Val}，而用不相交的数据训练不同的基模型，并将它们的输出取加权平均。该方法实现简单，但是对训练数据的利用率不高。

为了提高训练数据的利用率，本文进行了改进：使用 Bootstrap 重采样方法，对每一个基模型的训练集都进行重采样。使用该方法重采样后，约有 36.8%的数据(袋外数据)未被采集到，再将每个模型的袋外数据作为该模型的验证集使用。

(1)第一层训练模型——基模型

本文分别构造 3 个不同参数的 CART 随机森林模型和 3 个不同参数的 MLP 多层神经网络 Adaboost 模型作为第一层的训练模型 $H_1=\{h_1,h_2,\cdots,h_6\}$。

针对基模型 H_1，本文首先对大训练集 D_{Train} 进行 Bootstrap 重采样得到 6 组训练集 $D_{Train\text{-}1}=\{D_{T1},D_{T2},\cdots,D_{T6}\}$，其次将训练集 $D_{Train\text{-}1}$ 对应的袋外数据作为验证集 $D_{Val}=\{D_{V1},D_{V2},\cdots,D_{V6}\}$。由于验证集 D_{Val} 中包含重复数据，因此本文需要先对其中数据进行去重，只保留不重复的数据，最后记为验证集 $D_{Val\text{-}1}=\{(x_i,y_i)\}$。相比传统 Blending 使用固定比例的训练数据作为验证集而言，使用袋外数据作为验证集极大地提高了数据的利用率。

本文对基模型 H_1 的训练和预测分为如下两步：

第 1 步　先用训练集 $D_{Train\text{-}1}$ 训练基模型 H_1，再用训练好的基模型预测验证集 $D_{Val\text{-}1}$，生成新的训练集 $D_{Train\text{-}2}$。由于训练基模型使用的 6 组训练集是从大训练集 D_{Train} 中随机重采样得到的，保证了各组数据之间的差异性；同时预构造的 6 组模型参数设定都不一样，保证了训练得到的各组基模型之间的差异性；6 组训练集之间的部分数据有重复，这些重复数据经过多组不同的模型进行训练，降低了某些数据被单一模型错误学习的概率。使用同一验证集分别在 6 组训练后的基模型上进行预测，得到 6 组独立的预测结果，即生成了包含 6 列特征向量的特征集，作为第二层训练模型的训练集。则在第一层模型上的训练和对验证集的预测过程如图 3 所示。

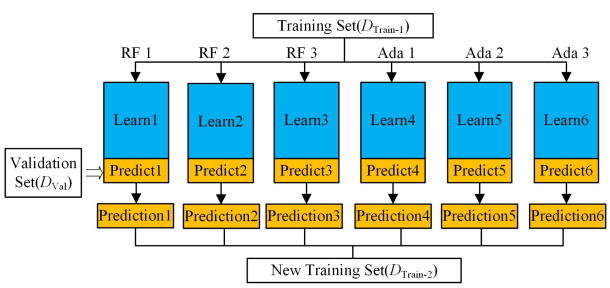

图 3　基模型训练和预测验证集过程

Fig. 3　Base model training and prediction verification set process

第 2 步　用训练好的基模型对测试集 $D_{Test\text{-}1}$ 进行预测，生成新的测试集 $D_{Test\text{-}2}$。即将同一测试集分别在 6 组训练后的基模型上进行预测，得到 6 组独立的预测结果，同时也生成了包含 6 列特征向量的特征集，作为第二层训练模型的测试集。则在基模型上对测试集进行预测的过程如图 4 所示。

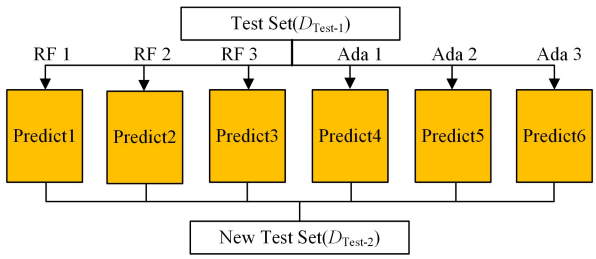

图 4　基模型对测试集的预测过程

Fig. 4　Prediction process of base model on test set

(2)第二层训练模型——次模型

在第二层训练模型中，本文分别构造一个随机森林分类器和一个 Adaboost 分类器，以生成第二层的训练模型 $H_2=\{h_1,h_2\}$。对训练集 $D_{Train\text{-}2}$ 进行 Bootstrap 重采样获得 2 组训练集 $D_{Train\text{-}2}^{new}=\{D_{T1},D_{T2}\}$，同时将两次袋外数据进行合并去重，得到第二层训练模型的验证集 $D_{Val\text{-}2}=\{(x_i,y_i)\}$。在第二层上对训练模型 H_2 进行训练，并使用训练好的次模型对验证集进行预测，生成第三层的训练集 $D_{Train\text{-}3}$。这一层的训练和预测过程同第一层基本一致，如图 5 所示。

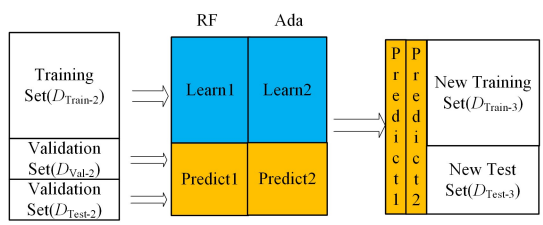

图 5　次模型的训练和预测过程

Fig. 5　Training and prediction of sub-model

(3)第三层训练模型——元模型

与 Stacking 算法一样，Blending 算法在最后一层训练模型中也可以采用元模型进行结果的最后预测。本文将随机森林模型作为第三层训练的元模型，将在第二层预测的验证集和测试集分别作为元模型的训练集和测试集，将最后在训练好的元模型上测试的测试集作为模型融合后的最终预测结果。

(4)模型融合过程

本文基于异质强分类器的模型融合过程如图 6 所示。

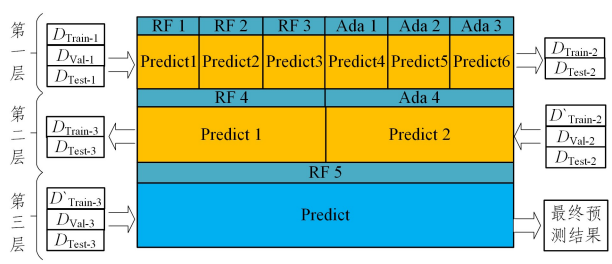

图 6　基于异质强分类器的模型融合过程示意图

Fig. 6　Schematic diagram of model fusion process based on heterogeneous strong classifier

综上,本文基于 Blending 算法构建了面向随机森林和 Adaboost 的融合模型。

6　对比实验

6.1　实验样本与配置环境

本文实验所用数据均为 APK 文件,根据 APK 文件是否恶意将其分为良性 APK 数据集和恶意 APK 数据集。其中,良性 APK 数据集是从国内外 Android 应用市场(豌豆荚和 Google Play)上下载的,而恶意 APK 数据集是从著名恶意软件发布网站 VirusShare 和 Drebin 上下载的。

由于从网站上下载的 APK 文件参差不齐,有部分文件经过了高度加固防护,部分文件缺少关键配置,还有一些文件无法在模拟器或移动终端上运行。因此,本文需对上述 APK 文件进行筛选去重。首先,使用 ApkTool 工具反编译各个文件,获取文件基本信息,去除无法反编译和基本信息不全的文件;其次,根据文件版本信息和校验码对比信息,去除重复的 APK 文件;最后,在模拟器中安装运行各个 APK 文件,去除无法安装或安装失败以及无法运行的文件。经过筛选去重步骤后,本文最终用于实验部分的数据共计 8000 个。其中,恶意 APK 文件数量和良性 APK 文件数量各为 4000 个。通过静态特征提取、动态特征提取和特征数据预处理等步骤,共计获得 8000 个样本数据用于训练分类模型的训练集。

本文的实验环境分为两部分:一部分为 Hadoop 私有云平台,由 3 台集群服务器组成,配置如表 3 所列;另一部分为移动端测试设备,配置如表 4 所列。

表 3　Hadoop 云平台设备配置

Table 3　Hadoop cloud platform device configuration

操作系统	内存/GB	显卡	JDK 版本	Hadoop 版本	Spark 版本
CentOS 7	32	RTX 3070	1.8.0_181	2.9.1	1.5.1

表 4　移动端测试设备配置

Table 4　Mobile terminal test equipment configuration

操作系统	移动端系统	内存/GB	外存/GB
MIUI 10.1	Android 7.0	4	32

6.2　分类模型评价指标

设 TP(True Positive)为恶意软件被预测为恶意的样本数,TN(True Negative)为良性软件被预测为良性的样本数,FP(False Positive)为良性软件被预测为恶意的样本数,FN(False Negative)为恶意软件被预测为良性的样本数。则本文针对移动端恶意软件检测模型采用的分类评价指标如下所示。

(1)TPR(True Positive Rate):真阳性率,即当前被预测为恶意软件的样本占所有恶意软件样本的比例,取值范围为[0,1],取值越大,则模型的预测能力越好。

$$TPR=\frac{TP}{TP+FN} \tag{6}$$

(2)FPR(False Positive Rate):假阳性率,即当前被错误预测为恶意的良心软件样本占所有良性样本的比例,取值范围为[0,1],取值越小,则模型的预测能力越好。

$$FPR=\frac{FP}{TN+FP} \tag{7}$$

(3)精度 ACC(Accuracy):软件被正确预测的概率,取值范围为[0,1],取值越大,则模型的预测能力越好。

$$ACC=\frac{TP+TN}{TP+FP+TN+FN} \tag{8}$$

(4)准确率(Precision):被正确预测为恶意软件的样本占所有所有被预测为恶意软件样本的比例,取值范围为[0,1],取值越大,则模型的预测能力越好。

$$P=\frac{TP}{TP+FP} \tag{9}$$

(5)召回率(Recall):被正确预测为恶意的软件样本占所有恶意软件样本的比例,取值范围为[0,1],取值越大,则模型的预测能力越好。

$$R=\frac{TP}{TP+FN} \tag{10}$$

(6)F_1-Score:准确率和召回率的调和平均。

$$F_1=\frac{2P*R}{P+R} \tag{11}$$

其中,TPR=Recall,都表示当前被预测为恶意的样本中,真正为恶意的样本占所有恶意样本的比例。召回率又叫查全率,反映检测模型在某一类别上的分类效果,也是检索出的相关文档数和文档库中所有的相关文档数的比率,衡量的是检索系统的查全率,在此处是被准确预测为恶意软件的软件样本占实际所有恶意软件样本的比例,当召回率的值越接近 1,证明模型的预测准确度越高,以此反映模型的好坏。另外精确率和召回率是相互影响的,理想情况为两者都高,但是一般情况下,准确率高时召回率就低,反之亦然。同时,F 值结合了精确率和召回率,公式中也需要用到召回率的值。

6.3　对比实验和分析

本文针对异质分类融合模型的对比实验内容主要包括 3 个部分:首先,使用 CART 决策树构造随机森林集成学习模型;其次,使用 MLP 构造 Adaboost 集成学习模型;最后,使用 Blending 算法对上述两个异质强分类器进行模型融合,生成最终的强分类融合模型。具体对比实验如下。

(1)使用 CART 决策树构造随机森林集成学习模型

传统的 CART 决策树分类算法无论如何进行剪枝优化,始终无法突破单分类模型在预测精度上的瓶颈问题。因此,本文基于"Bagging"思想使用 CART 决策树为基学习器构造随机森林集成学习模型。分类效果的对比如表 5 所列。

表 5　使用单 CART 决策树和基于 CART 的随机森林对于分类效果的影响

Table 5　Impact of using single cart decision tree and random forest based on cart tree on classification effect

(单位:%)

分类器	*TPR*	*FPR*	*Precision*	*Recall*	*ACC*	F_1
单 CART 决策树	84.7	13.0	86.7	84.7	85.9	85.7
CART 随机森林	93.5	6.0	94.0	93.5	93.8	93.7

由此可见,本文在将 CART 决策树作为基学习器构建随机森林集成学习模型后,最终分类效果有了显著提升。

(2)使用 MLP 构造 Adaboost 集成学习模型

为了构造异质分类融合模型,本文以 MLP 作为基学习器构建 Adaboost 集成学习模型。使用双隐藏层的 MLP 和 Adaboost 集成学习模型的分类效果对比结果如表 6 所列。

表 6 使用单 MLP 和基于 MLP 的 Adaboost 对于分类效果的影响

Table 6 Impact of MLP and Adaboost based on MLP on classification effect

（单位：%）

分类器	*TPR*	*FPR*	*Precision*	*Recall*	*ACC*	F_1
单 MLP	78.5	19.5	80.1	78.5	79.5	79.3
MLP 的 Adaboost	94.1	7.3	92.8	94.1	93.4	93.4

可以看出，本文在将 MLP 作为基学习器构建 Adaboost 集成学习模型后，最终分类效果有了显著提升。

(3)使用 Blending 算法进行模型融合

为了进一步提高分类模型的最终分类效果，本文使用 Blending 方法，对上述 CART 构建的随机森林集成学习模型和 MLP 构建的 Adaboost 集成学习模型进行模型融合。经过模型融合后的最终分类模型和单一集成学习模型的分类效果对比如表 7 所列。

表 7 单一集成学习模型和融合模型对于分类效果的影响

Table 7 Effect of single ensemble learning model and fusion model on classification

（单位：%）

分类器	*TPR*	*FPR*	*Precision*	*Recall*	*ACC*	F_1
CART 随机森林	93.5	6.0	94.0	93.5	93.8	93.7
MLP 的 Adaboost	94.1	7.3	92.8	94.1	93.4	93.4
本文融合模型	96.0	3.5	96.4	96.0	96.3	96.2

(4)使用 Stacking 算法进行多模型融合

Stacking 是一种集成学习算法，该算法通过分类算法或者回归算法集成多个分类模型或者回归模型。为了验证本文提出的基于异质模型融合方法对 Android 恶意软件的检测效果，将其与文献[25]中提出的基于多模型融合方法[25]以及 Milosevic 等[26]提出的基于权限和基于源代码的分类器融合方法进行对比实验，实验结果如表 8 所列。由于对实践中的结果而言，Stacking 和 Blending 的效果差不多，并且相比之下，Blending 方法不用进行 k 次的交叉验证来获得新特征，因此 Blending 比 Stacking 方法相对简单一些，因此本文模型融合采用 Blending 方法。

表 8 基于源代码方法、权限方法和融合模型的对比结果

Table 8 Comparison results based on source code method, permission method and fusion model

（单位：%）

	Recall	*Precision*	*FPR*
本文方法	96.0	96.4	3.5
文献[26]基于源代码方法	95.7	95.8	4.2
文献[26]基于权限方法	89.4	89.5	10.4
文献[25]方法	95.9	96.3	3.5

可以看出，本文在对两个集成学习模型进行模型融合后，最终分类效果有了明显提升。

最后，本文对上述对比实验采用 5 次、10 次、15 次和 20 次十折交叉验证法，对检测精度 ACC 分别取平均值，最终分类效果的柱状对比如图 7 所示，折线对比如图 8 所示。其中，Model_1 代表单 CART 决策树模型；Model_2 代表以 CART 为基学习器的随机森林模型；Model_3 代表 MLP 模型；Model_4 代表以 MLP 为基学习器的 Adaboost 模型。

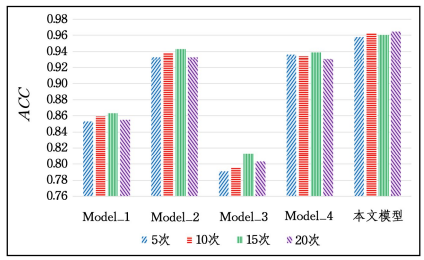

图 7 十折交叉验证柱状对比图

Fig. 7 Ten fold cross-validation Bar Chart

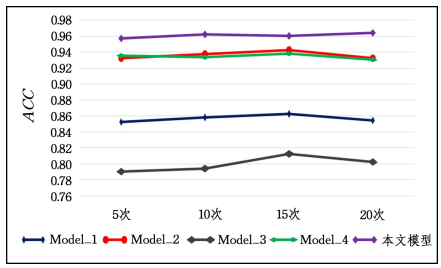

图 8 十折交叉验证折线对比图

Fig. 8 Ten fold cross-validation line comparison chart

综上，通过对对比实验的分析可知，所提异质强分类器模型融合方法相较于传统单分类模型而言，突破了传统单分类器模型分类效果的瓶颈问题，在面向移动端恶意软件检测的分类精度上有了很明显的改善和提升。

结束语 本文提出一种基于模型融合的 Android 恶意软件检测方法。首先通过分析当前单模型分类器在 Android 恶意软件检测领域分类精度无法提高的现状，提出了基于异质强分类器的融合分类模型；其次对比、介绍了本文使用到的基分类器——CART 分类树和 MLP 多层神经网络；然后基于 CART 决策树的随机森林算法和基于 MLP 的 Adaboost 算法分别构造集成学习模型；最后通过 Blending 算法对这两个分类器进行模型融合得到异质模型融合分类器，在此基础上实施恶意软件检测。实验结果表明，所提方法能够有效克服单一分类模型检测精度不足的问题，各项评价指标均达到了预期要求。

参 考 文 献

[1] China Internet Network Information Center. The 46th《Statistical Reports on Internet Development in China》[EB/OL]. http://www.gov.cn/xinwen/2020-09/29/content_5548176.htm.

[2] 360 Beacon Lab, 360 Security Brain. 2019 Android Malware Special Report [EB/OL]. https://blogs.360.cn/post/review android_malware_of_2019.html.

[3] China Academy of Information and Communications Technology. White Paper on Mobile Application (App) Data Security and Personal Information Protection (2019) [EB/OL]. http://www.caict.ac.cn/kxyj/qwfb/bps/201912/t20191229_272847.htm.

[4] Network and Information Technology Center. Information Security Technology Personal Information Security Specification (2020 Edition) [EB/OL]. http://www.ahstu.edu.cn/wlzx/info/ 1011/1478.htm.

[5] National Engineering Laboratory, China Academy of Information and Communications Technology, iJiami. National Mobile

App Risk Monitoring and Evaluation Report (2020 3rd Quarter Edition) [EB/OL]. https://www.anquanke.com/post/id/219502.

[6] SHEN F, VECCHIO J D, MOHAISEN A, et al. Android Malware Detection Using Complex-Flows[C]// IEEE Transactions on Mobile Computing. 2017.

[7] ZHANG C, HU G, WANG Z, et al. A NOVEL SVM-BASED DETECTION METHOD FOR ANDROID MALWARE[J]. Computer Applications and Software, 2018, 35(10): 298-304.

[8] LI C F, LEE W L, SUN W. Android Malware Detection Algorithm Based on CNN and Naive Bayesian Method[J]. Journal of Information Security Research, 2019, 5(6): 470-476.

[9] WANG W, LI Y, WANG X, et al. Detecting android malicious apps and categorizing benign apps with ensemble of classifiers [J]. Future Generation Computer Systems, 2018, 78: 987-994.

[10] Android Developers. Motion Event [EB/OL]. https://developer.android.com/reference/android/view/MotionEvent#getAction%28%29.

[11] GREGORUTTI B, MICHEL B, SAINT-PIERRE P. Correlation and variable importance in random forests[J]. Stats & Computing, 2017, 27(3): 659-678.

[12] SIKORA R, AL-LAYMOUN O H. A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms [J/OL]. https://www.igi-global.com/Files/Ancillary/7a51f757-7e8d-4feb-8afd-2d16a8257b18_TOC.pdf.

[13] DONG K Y. Research and implementation of Android malware detection method[D]. Nanjing: Nanjing University of Science and Technology, 2018.

[14] DU W, LI J. Android malware detection and malicious behavior analysis based on semi-supervised learning[J]. Journal of Information Security Research, 2018, 4(3): 242-250.

[15] QIU H J, LIAN G X, LIU Z J. Android malware detection based on combined machine learning algorithm[J]. Journal of Information Technology, 2019(7): 59-64.

[16] WANG T, LI J. Design and implementation of Android malware detection based on deep learning[J]. Journal of Information Security Research, 2018, 4(2): 140-144.

[17] JIANG C. Research on Android malware detection technology based on deep learning [D]. Changsha: Hunan University.

[18] HOU L Y, LUO L L, PAN L M, et al. Android Malware Detection Method Fusion Multi-feature[J]. Chinese Journal of Network and Information Security, 2020(1): 67-74.

[19] WANG G Y. Research on Android malware detection method based on multi-features [D]. Xi'an: Xidian University, 2020.

[20] SONG L. Research on Android Local Layer Code Obfuscation Analysis Based on Machine Learning [D]. Xi'an: Northwest University, 2019.

[21] WANG X. Research and implementation of Android mobile terminal data security protection technology [D]. Beijing: Beijing University of Posts and Telecommunications, 2019.

[22] XU H. Research on Malware Detection Technology Based on Recurrent Neural Network [D]. Beijing: Beijing University of Posts and Telecommunications, 2016.

[23] ALZAYLAEE M K, YERIMA S Y, SEZER S. DL-Droid: Deep learning based android malware detection using real devices[J]. Computers & Security, 2020, 89(2): 101663.1-101663.11.

[24] JIANG F S. Research and implementation of malware identification based on deep learning [D]. Beijing: Beijing University of Posts and Telecommunications, 2019.

[25] YAN B. Research on Android malware detection technology based on multi-model fusion [D]. Xi'an: Xidian University, 2019.

[26] MILOSEVIC N, DEHGHANTANHA A, CHOO K K R. Machine learning aided Android malware classification[J]. Computers & Electrical Engineering, 2017, 61: 266-227.

YAO Ye, born in 1972, associate professor, master supervisor. His main research interests include software security testing, network system security evaluation, industrial Internet and security technology.

ZHU Yi-an, born in 1961, professor, doctoral supervisor. His main research interests include parallel computing, network and information security, complex system modeling and analysis, big data intelligent processing technology, security critical operating system.