doi:10.11959/j.issn.1000-436x.2017073

基于改进随机森林算法的 Android 恶意软件检测

杨宏宇，徐晋
（中国民航大学计算机科学与技术学院，天津 300300）

摘 要：针对随机森林（RF, random forest）算法的投票原则无法区分强分类器与弱分类器差异的缺陷，提出一种加权投票改进方法，在此基础上，提出一种检测 Android 恶意软件的改进随机森林分类模型（IRFCM, improved random forest classification model）。IRFCM 选取 AndroidManifest.xml 文件中的 Permission 信息和 Intent 信息作为特征属性并进行优化选择，然后应用该模型对最终生成的特征向量进行检测分类。Weka 环境下的实验结果表明 IRFCM 具有较好的分类精度和分类效率。
关键词：随机森林；加权投票；恶意软件；分类检测
中图分类号：TP309 **文献标识码**：A

Android malware detection based on improved random forest

YANG Hong-yu, XU Jin
(School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China)

Abstract: Aiming at the defect of vote principle in random forest algorithm which is incapable of distinguishing the differences between strong classifier and weak classifier, a weighted voting improved method was proposed, and an improved random forest classification (IRFCM) was proposed to detect Android malware on the basis of this method. The IRFCM chose Permission information and Intent information as attribute features from AndroidManifest.xml files and optimized them, then applied the model to classify the final feature vectors. The experimental results in Weka environment show that IRFCM has better classification accuracy and classification efficiency.
Key words: random forest, weighted vote, malware, classification detection

1 引言

Android 系统是一款基于 Linux 内核的开源操作系统，已经成为目前市场占有量最大的移动设备终端平台。Android 系统的开放性使它成为恶意软件最大的发展平台，2015 年，国家互联网应急中心捕获和厂商获得的针对安卓平台的移动互联网恶意程序数量位居第一。随着 Android 恶意软件的占有率逐年上升，如何将其快速高效地分析并检测出来已经成为目前的研究热点。

目前，机器学习中的很多分类算法被应用于 Android 恶意软件检测中。文献[1]提出一种基于朴素贝叶斯（NB, naive Bayes）的 Android 应用恶意行为识别方法，抽取软件是否申请过多权限、是否存在敏感权限组合等作为分类属性，通过对 Android 安全框架的扩展，实现了对恶意行为的实时分析和处理。文献[2]利用 Android 权限间、Android 权限和软件恶意倾向间的相关性，通过改进贝叶斯算法实现了恶意软件的检测。文献[3]使用 Android 权限信息作为特征并采用信息增益（IG, information gain）算法对其进行优化选择，再利用拉普拉斯校准和乘数取自然对数对 NB 算法进行改进，从而对 Android 恶意应用进行分析检测。然而，上述研究只针对 Android 应用的权限信息进行检测

收稿日期：2016-12-03；**修回日期**：2017-02-21
基金项目：国家科技重大专项基金资助项目（No.2012ZX03002002）；中国民航科技基金资助项目（No.MHRD201009, No.MHRD201205）
Foundation Items: The National Science and Technology Major Project (No.2012ZX03002002), The Science & Technology Project of CAAC (No.MHRD201009, No.MHRD201205)

分析，检测范围不够全面。文献[4]利用危险 API 调用和权限组合创建了 SVM 分类器，从而自动地将恶意软件分辨出来。文献[5]应用 k 均值（k-means）算法和微小批处理 k 均值算法（mini batch k-means）2 种聚类算法实现对恶意软件的分类。文献[4,5]虽然实现了对 Android 恶意软件的检测，但是检测精度都不够高。文献[6]利用深度学习算法实现了一个在线恶意软件检测工具 Droid Detector，实现了 Android 应用的在线检测分析，但是其算法复杂度较高，对计算机内存消耗较大。文献[7]提出了基于手机端和服务器端的协作恶意代码检测方案，文献[8]设计了一种 3 层混合系综算法（THEA, triple hybrid ensemble algorithm），并实现了检测工具 Androdect 进行恶意代码检测，但这 2 种方法在技术实现方面相对比较复杂。

综上所述，目前的检测研究成果在检测精度、检测效率和实现复杂度等方面还存在诸多不足。本文应用加权投票机制对随机森林（RF, random forest）算法进行改进，提出改进随机森林分类模型（IRFCM, improved random forest classification model）对 Android 应用程序进行逆向处理，并提取 Permission 信息及 Intent 信息作为特征属性，分别采用 IG 算法[9,10]和 ReliefF 算法[11,12]对特征属性进行优化选择，并建立分类器对 Android 应用进行检测分类，可有效提高对 Android 平台下的恶意软件检测效率。

2　随机森林算法

2.1　算法分析

RF 算法由 Leo 和 Adele 提出[13,14]，该算法结合了 bagging 以及由 Ho 和 Amit 提出的随机属性选择思想，是一个由多棵决策树分类器$\{h(x,\theta_k)\}$组成的集成学习分类器。其中，$\{\theta_k\}$是独立同分布的随机向量，k 表示决策树分类器的数量，每个决策树分类器根据输入的测试样本集 x 产生分类结果，最终通过投票原则确定测试样本类别。RF 算法的流程如图 1 所示。

该算法的具体步骤如下。

步骤 1　给定原始训练集 S，样本数量为 N，特征属性数量为 M。采用 bagging 抽样技术对 S 进行有放回抽样，共抽取 $n_{\text{try}}(n_{\text{try}}=N)$个样本形成训练子集。

步骤 2　从 M 个特征属性中随机选取 m_{try}（$m_{\text{try}}<<M$，通常取$\lfloor U_0(M)+1 \rfloor$）个属性，按照决策树生成算法（C4.5、CART 等）选择最优属性进行节点分裂，分裂过程是完全分裂不进行剪枝，从而对步骤 1 中抽取的训练子集生成一棵决策树。

步骤 3　重复步骤 1 和步骤 2 共 k 次，建立 k 棵决策树，生成随机森林。

步骤 4　应用生成的随机森林对测试样本集进行分类检测，汇总所有决策树共 k 个分类结果，采用简单投票原则计算出最终的分类结果。

2.2　算法特点

RF 算法有很多优点，首先，它是一种集成学习算法，通过组合若干单个分类器的分类结果，从而对测试样本的类别做出分类，该算法与单个分类器相比具有更好的分类效果和泛化能力；由于特征子集是随机选取的，因而该算法能够处理高维度数据且不必做特征选择；该算法的训练过程中决策树间相互独立，训练速度快且易于做成并行化方法。但是，RF 算法也存在不足，以下将针对其投票原则无法区分强分类器与弱分类器差异的缺陷进行改进。

3　随机森林算法的改进

投票决策过程是 RF 算法的重要组成部分，它决定了对测试样本的最终分类结果。RF 算法采用简单投票原则，对每个决策树赋予相同的权重，忽

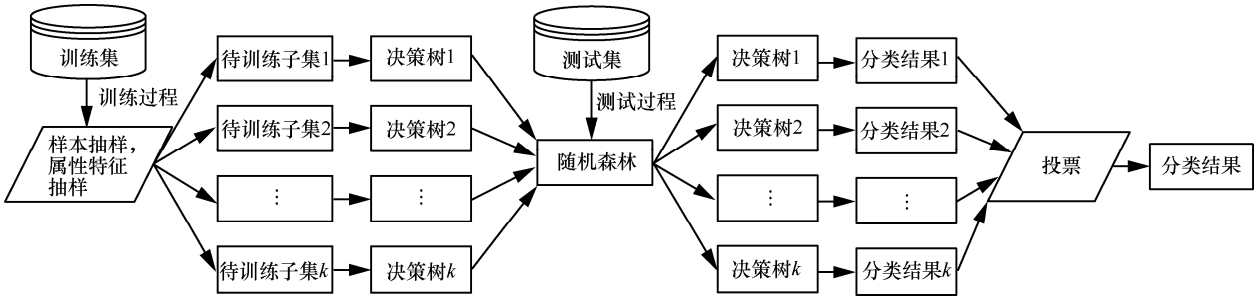


图 1　RF 算法流程示意

略了强分类器与弱分类器的差异，影响了随机森林分类器的整体分类性能。为此，本文采用加权投票原则对 RF 算法进行改进，形成改进随机森林（IRF, improved random forest）算法。

1) 在决策树的生成过程中，使用 bagging 方法从原始训练集（样本总数为 N）中有放回地抽取样本，形成一个样本集，因此，存在一些未被抽取到的样本。训练集中每个样本未能被抽取到的概率为 $p=(1-\frac{1}{N})^N$，可以证明，当 $N\to\infty$ 时，$p\approx0.368$，表明每个样本集的抽取过程中都会有趋近 37%的样本没有被抽取，这一部分样本称为 out-of-bag（OOB）数据。设 X 为测试样本集，x 为其中一个样本；T 为训练完成的决策树分类器集合，t 为当前决策树；C 为分类结果集合，c 为其中一个分类。

对当前决策树 t 而言，其 OOB 数据记为 O_t，并应用决策树 t 对 O_t 中的各个样本进行分类。由于 O_t 中的样本也是原始训练集中的，所以每个样本的真实类别是可知的。通过对比分类结果和样本真实类别，可以得到对 O_t 数据分类正确的样本数量，记为 O_{tr}。记 CR_t 为决策树 t 对 O_t 的分类正确率，则

$$CR_t=\frac{O_{tr}}{O_t} \tag{1}$$

由式(1)可见，CR_t 越大，说明决策树 t 的分类效果越好，属于强分类器；反之，决策树 t 的分类效果越差，属于弱分类器。

2) 将每棵决策树对 OOB 数据的分类正确率 CR_t 作为对应决策树的权重，将样本 x 通过随机森林分类器进行检测分类并经过加权统计，属于 c 类别的总得票数记为 S_c，则

$$S_c=\sum_{t=1}^{T}(T_{c,x}(x)CR_t) \tag{2}$$

其中，$T_{c,x}(x)$取值为 1 或 0，若样本 x 经过决策树的分类测试后结果为 c 类，取值为 1；若样本 x 经过决策树的分类测试后结果不为 c 类，取值为 0。

3) 选出得票数最多类别 C_x 作为样本 x 的最终类别

$$C_x=\arg\max(S_c) \tag{3}$$

4　恶意软件分类模型

4.1　模型设计

改进随机森林分类模型（IRFCM, improved random forest classification model）结构如图 2 所示。

该模型对恶意软件的检测由特征向量提取和检测分类 2 个阶段组成。

1) 特征向量提取阶段包括逆向处理、特征提取、生成特征向量和特征属性优化等过程，主要目的是提取具有代表性的特征属性并生成优化特征向量集合。处理过程设计如下。

① 反编译样本集中的 apk 文件生成对应 AndroidManifest.xml 文件。

② 将 AndroidManifest.xml 文件的 Permission 信息及 Intent 信息作为特征属性，分别对标签项

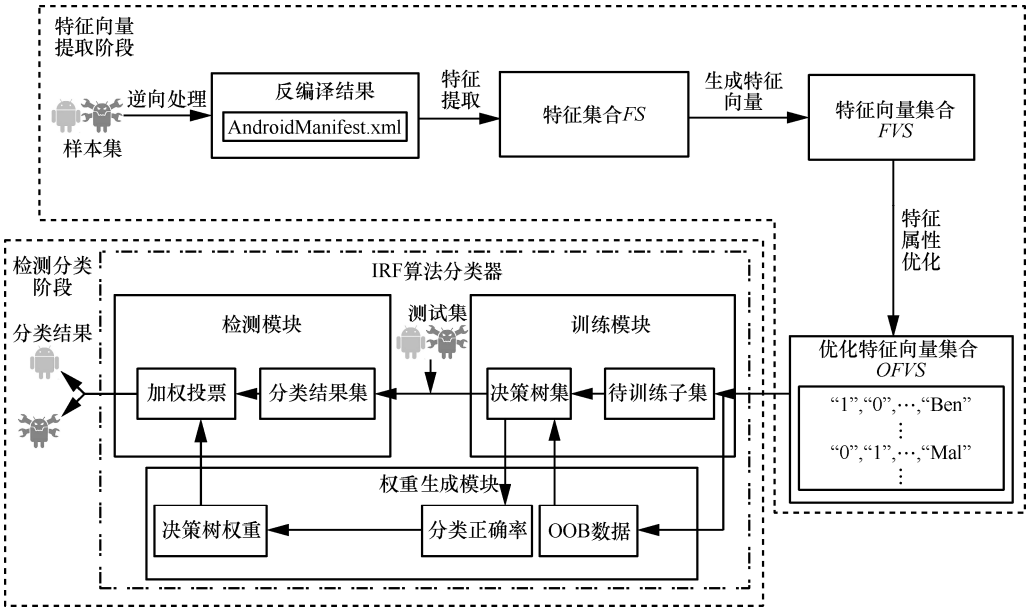


图 2　IRFCM 结构

<uses-permission>、<intent-filter><action>和<intent-filter><category>进行属性值提取，并根据所提取出的属性值，统计每个属性值出现的次数并排序，将使用频率高的属性值组合成为特征集合。

③ 根据特征集合，对比与每个 apk 文件提取出的属性值，生成相应特征向量集合。

④ 使用特征优化算法对生成的特征向量集合进行特征属性优化排序，根据排序结果选取更具有代表性的特征属性并生成优化特征向量集合。

2) 检测分类阶段使用 IRF 算法分类器，该分类器由训练、检测和权重生成模块组成，主要目的是训练分类器并对测试集进行检测分类。处理过程设计如下。

① 将特征向量提取阶段生成的优化特征向量集合输入训练模块，抽样一部分数据作为待训练子集训练分类器并生成训练后的决策树集，将剩余数据作为 OOB 数据。

② 权重生成模块将训练模块中生成的 OOB 数据输入训练后的决策树集，得出每棵决策树的分类正确率，作为每棵决策树的权重值。

③ 检测模块将测试集输入训练后的决策树集，得出分类结果集，并应用权重生成模块得到的权重值进行加权投票，得出最终分类结果。

4.2 特征向量提取

在本文研究中使用 Apktool[15]反编译工具对由 Android 应用软件组成的样本集进行逆向处理。目标文件经过反编译处理后，生成包含 Android Manifest.xml 文件的文件夹，并使用 Python 的 xml.dom 模块和 IO 模块对 AndroidManifest.xml 文件进行解析。具体步骤设计如下。

步骤 1　反编译 apk 文件。使用 Apktool 工具和 cmd 控制平台批量反编译样本集中的全部 apk 文件，并将结果输出到指定目录下。

步骤 2　抽取相应特征组成属性集合。步骤 1 对每个 apk 文件经过反编译生成的文件夹下新建 permission.txt、action.txt 和 category.txt 文档，用于存储该 apk 文件的属性值。解析文件夹中的 AndroidManifest.xml 文件，将选择的 3 种信息标签中的属性值提取出来，并分别存到对应的 permission.txt、action.txt 和 category.txt 文档中。

步骤 3　计算属性值数量并排序。对 Android 良性软件和恶意软件分别建立 sortper.txt、sortact.txt 和 sortcate.txt 文档，用于记录两者常用的属性值。统计每个 apk 文件中由步骤 2 输出的 txt 文档，分别计算出良性软件和恶意软件中使用频率较高的 permission、action 及 category 属性值，并将统计结果由高到低分别输出到对应的 sortper.txt、sortact.txt 和 sortcate.txt 文档中。

步骤 4　生成特征向量集合。根据步骤 3 中的排序结果，分别将良性软件和恶意软件使用频率较高的属性值组合为相应集合：良性软件 permission 集合 *bps*（benign permission set）、良性软件 action 集合 *bas*（benign action set）、良性软件 category 集合 *bcs*（benign category set）、恶意软件 permission 集合 *mps*（malware permission set）、恶意软件 action 集合 *mas*（malware action set）、恶意软件 category 集合 *mcs*（malware category set）。然后将这几个集合通过运算组合成特征集合 *FS*（feature set）

$$FS = (bps \mid mps) \mid (bas \mid mas) \mid (bcs \mid mcs) \tag{4}$$

其中，符号“|”为取并集运算。

利用匹配算法分别计算出样本集中每个 apk 文件是否含有 *FS* 集合中对应的元素，生成相应特征向量并组合成特征向量集合 *FVS*（feature vector set）。特征向量中的元素可以取值为“1”或“0”，“1”表示该 apk 应用含有对应属性，“0”则表示不含有对应属性。然后，在特征向量末尾添加一个标识位，标识位可以取值为“Ben”或“Mal”，“Ben”表示该应用为良性软件，“Mal”表示该 apk 应用为恶意软件。

步骤 5　特征属性优化。采用特征选择算法对步骤 4 生成的 *FVS* 中的特征属性进行优化排序，并根据排序结果重新组合形成优化特征向量集合（OFVS, optimized feature vector set）。

4.3 检测分类

检测分类阶段包括训练和检测 2 个模块，每个模块的设计如下。

1) 训练模块的作用是生成决策树集和对应权重值，具体过程为：对特征提取阶段生成的特征向量集合进行抽样，形成待训练子集和 OOB 数据，训练待训练子集生成决策树集，将 OOB 数据输入决策树集中得出分类正确率，并将该值作为当前决策树的权重。训练模块流程如图 3(a)所示。

2) 检测模块的作用是对测试样本集进行检测分类并对分类效果进行评价。为了有效度量分类器的分类效果，定义一组检测效果指标。

指标 1　真正性（TP, true positive）：实际为恶意软件，被识别为恶意软件。

指标 2　假正性（FP, false positive）：实际为良

性软件，被识别为恶意软件。

指标 3 真负性（TN, true negative）：实际为良性软件，被识别为良性软件。

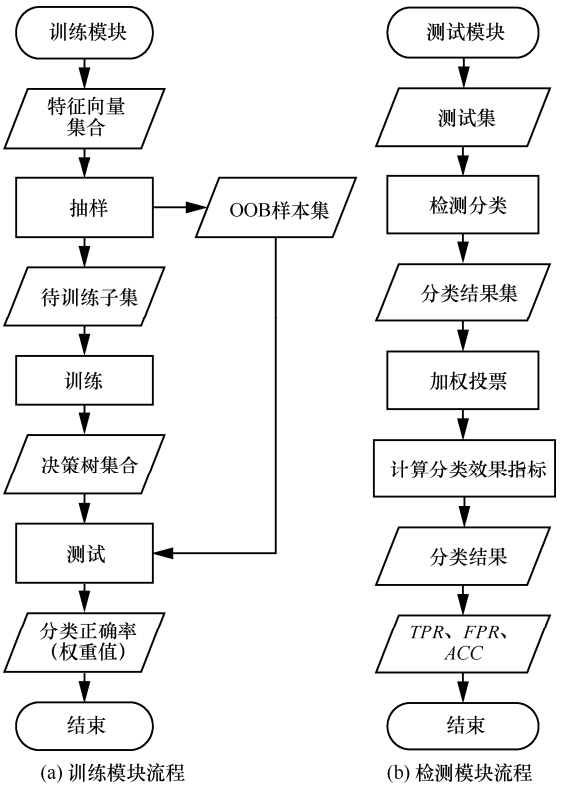
指标 4 假负性（FN, false negative）：实际为恶意软件，被识别为良性软件。

指标 5 真正率（TPR, true positive rate）= $\frac{TP}{TP+FN}$，表示识别出的恶意软件占实际恶意软件的比例。

指标 6 假正率（FPR, false positive rate）= $\frac{FP}{FP+TN}$，表示所识别出的恶意软件占实际良性软件的比例。

指标 7 分类精度（ACC, accuracy）= $\frac{TP+TN}{TP+TN+FP+FN}$，用来衡量总体分类精度，该值越高则分类效果越好。

具体过程为：将测试集输入决策树集合中得到分类结果集，并应用权重生成模块中生成的决策树权重对分类结果进行加权投票，得出最终分类结果，对比该 apk 应用的实际类别，计算出 *TPR*、*FPR* 和 *ACC* 值来评价分类器的分类效果。检测模块流程如图 3(b)所示。



(a) 训练模块流程 (b) 检测模块流程

图 3 训练模块和检测模块流程

5 恶意软件检测实验

5.1 实验样本和环境配置

在测试实验中，良性 Android 应用来源于 Google 官方应用市场 Android Market 和国内第三方应用市场。从上述应用市场下载了 674 个良性软件，共计 15.4 GB。对于恶意软件，文献[16]创建了 Android 恶意软件基因组项目（Android malware genome project）并已收集了许多 Android 恶意软件供 Android 移动安全研究使用。本文由该项目获取了 1 260 个恶意软件，共计 1.53 GB。

将上述 2 类 Android 应用软件集合合并组成实验样本集合，共计 1 934 个 Android 软件。实验的软硬件环境配置如表 1 所示。

表 1 实验环境配置

硬件配置	软件环境
Intel Core i7-3770 CPU @ 3.40 GHz	Windows7 64 位操作系统
4.0 GB RAM	Weka3.8

5.2 特征向量提取及优化

将实验样本中良性软件和恶意软件的 apk 文件分别放到指定目录下，按照前文所述的 IRFCM 中特征向量提取阶段的步骤，通过对 apk 文件的反编译和解析，分别得到良性软件和恶意软件中使用频率较高的标签项<uses-permission>、<intent-filter><action>和<intent-filter><category>的属性值信息，其中，部分排序结果如图 4 所示。

对于标签项<uses-permission>，无论良性软件还是恶意软件大部分都需要用到网络相关的权限来连接和使用网络，如 INTERNET、ACCESS_NETWORK_STATE 等属性值。不同的是，恶意软件更多地用到短消息相关权限，如 READ_SMS、WRITE_SMS 和 SEND_SMS 等属性值，存在一类典型的恶意软件，其恶意行为是在用户完全不知情的情况下发送付费短信或定制增值业务，会给用户带来额外开销。

对于标签项<intent-filter><action>和<intent-filter><category>，几乎所有的软件都用到 MAIN 属性值和 LAUNCHER 属性值，MAIN 决定了一个应用程序先启动哪个组件，LAUNCHER 决定应用程序是否显示在程序列表里。

根据上述属性排序结果选取使用频率较高的

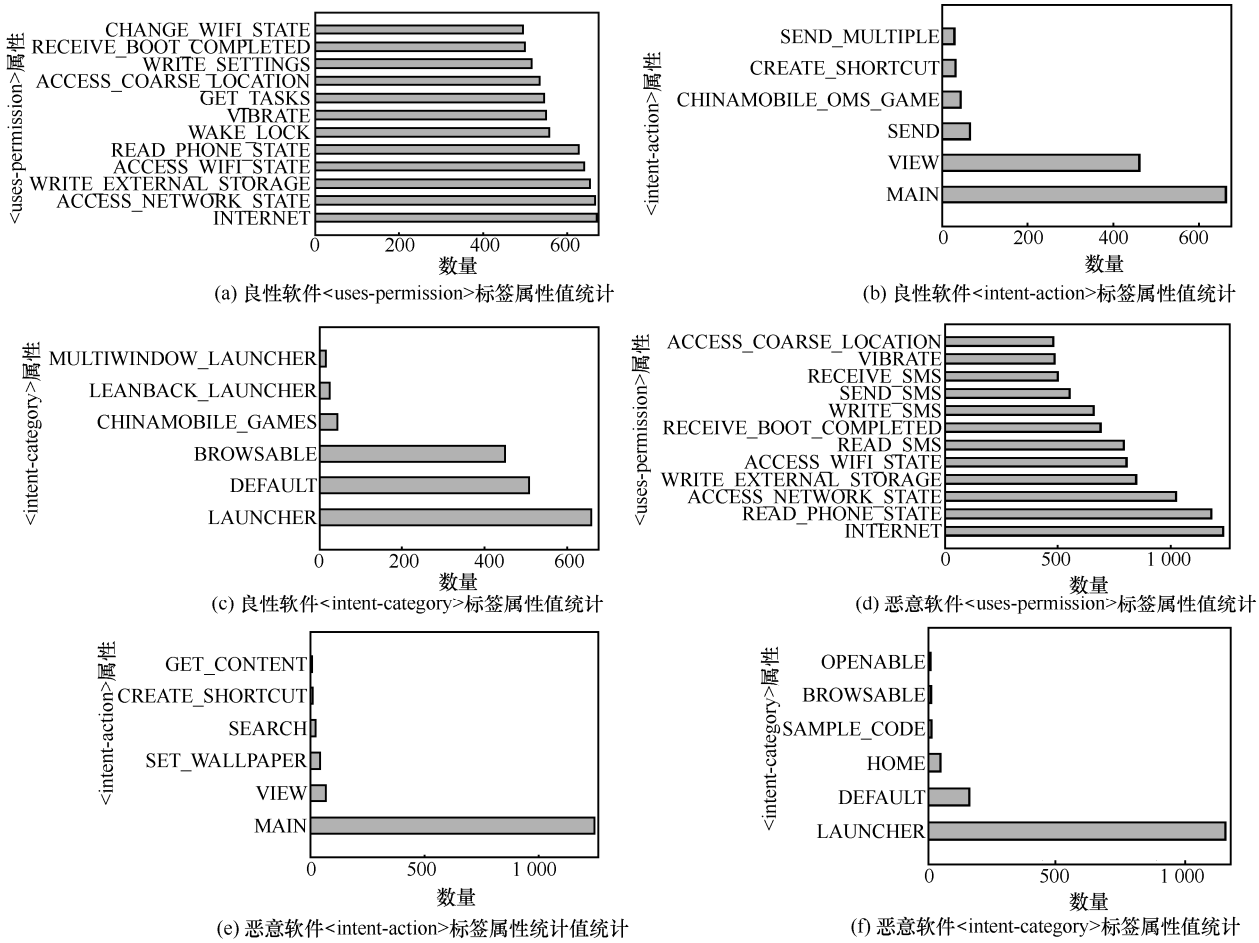


图 4　Android 应用常用属性值统计

属性，本文选取了<uses-permission>标签项中的 31 个属性值、<intent-filter><action>标签项中的 16 个属性值和<intent-filter><category>标签项中的 16 个属性值，共计 63 个属性值作为特征属性。

为了选取具有更高区分度并消除尽可能多的无关和冗余特征属性，本文采用 IG 算法和 ReliefF 算法分别对特征属性进行优化排序。

IG 算法通过计算特征的熵值与其条件熵的差值得到该特征的 IG 值，该值越大表明相关程度越高。ReliefF 算法主要根据特征在同类样例中以及相近的不同类样例中的区分能力来评价特征与类别的相关度并输出特征对应的 ReliefF 值，ReliefF 值越大表示该特征的分类能力越强。采用上述算法的排序结果如表 2 和表 3 所示。表 2 和表 3 仅列出了 63 个特征属性经过排序后排名在前 35 的特征属性。

由表 2 和表 3 可以看出，IG 算法和 ReliefF 算法排序结果中的特征属性大多相同，有 26 个相同属性，但是排序差异较大，如表 2 中排名第 1 的特征属性“SYSTEM_ALERT_WINDOW”在表 3 中排名第 6，而表 3 中排名第 1 的特征属性“WRITE_SMS”在表 2 中仅排到第 16。其原因是：IG 算法使用了所有样例的统计属性，对噪声的敏感度较低，但容易出现过拟合；而 ReliefF 算法对数据类型没有限制，能够处理高维数据集，但不能去除冗余特征，因此，2 种算法的排序结果会有差异。

根据排序结果，分别生成 IG 算法和 ReliefF 算法对应的特征属性数量为 5、10、15、20、25、30 和 35 的特征向量集合。

5.3　IRF 算法分类实验

在 IRF 算法分类实验中，其输入是根据 2 种特征优化算法生成的不同数量的特征向量集合。在实验中采用 10 折交叉验证的方法，将特征向量集合分成 10 份，轮流将其中 9 份作为训练集，剩余 1 份作为测试集，并对 10 次实验结果取均值，实验结果如图 5 和图 6 所示。

表 2　IG 算法特征属性排序

排名	特征属性	IG 值
1	SYSTEM_ALERT_WINDOW	0.477
2	BROWSABLE	0.424
3	WRITE_SETTINGS	0.380
4	VIEW	0.336
5	GET_TASKS	0.291
6	DEFAULT	0.281
7	CAMERA	0.272
8	MOUNT_UNMOUNT_FILESYSTEMS	0.251
9	CHANGE_NETWORK_STATE	0.239
10	RECORD_AUDIO	0.199
11	WAKE_LOCK	0.168
12	VIBRATE	0.131
13	READ_EXTERNAL_STORAGE	0.130
14	CHANGE_WIFI_STATE	0.119
15	ACCESS_COARSE_LOCATION	0.117
16	WRITE_SMS	0.117
17	READ_LOGS	0.107
18	WRITE_EXTERNAL_STORAGE	0.105
19	ACCESS_WIFI_STATE	0.102
20	ACCESS_FINE_LOCATION	0.092
21	WRITE_APN_SETTINGS	0.069
22	ACCESS_NETWORK_STATE	0.063
23	READ_SMS	0.062
24	DISABLE_KAYGUARD	0.055
25	INSTALL_PACKAGES	0.046
26	SEND	0.046
27	CHINAMOBILE_OMS_GAME	0.035
28	CHINAMOBILE_GAMES	0.035
29	WRITE_CONTACTS	0.034
30	RECEIVE_BOOT_COMPLETED	0.027
31	SEND_MULTIPLE	0.024
32	LEANBACK_LAUNCHER	0.020
33	MULTIWINDOW_LAUNCHER	0.013
34	RECEIVE_SMS	0.011
35	CREATE_SHORTCUT	0.009

表 3　ReliefF 算法特征属性排序

排名	特征属性	ReliefF 值
1	WRITE_SMS	0.309
2	READ_SMS	0.301
3	MOUNT_UNMOUNT_FILESYSTEMS	0.298
4	CHANGE_WIFI_STATE	0.292
5	CALL_PHONE	0.250
6	SYSTEM_ALERT_WINDOW	0.246
7	WRITE_SETTINGS	0.246
8	GET_TASKS	0.223
9	SEND_SMS	0.217
10	BROWSABLE	0.217
11	RECEIVE_BOOT_COMPLETED	0.214
12	READ_LOGS	0.214
13	WRITE_APN_SETTINGS	0.204
14	DEFAULT	0.197
15	WRITE_CONTACTS	0.183
16	RECEIVE_SMS	0.181
17	DISABLE_KAYGUARD	0.178
18	VIEW	0.166
19	ACCESS_FINE_LOCATION	0.163
20	CAMERA	0.163
21	CHANGE_NETWORK_STATE	0.159
22	ACCESS_COARSE_LACATION	0.156
23	READ_CONTACTS	0.155
24	INSTALL_PACKAGES	0.154
25	ACCESS_WIFI_STATE	0.145
26	READ_EXTERNAL_STORAGE	0.139
27	VIBRATE	0.129
28	WRITE_EXTERNAL_STORAGE	0.129
29	RESTART_PACKAGES	0.124
30	WAKE_LOCK	0.099
31	SET_WALLPAPAER	0.084
32	ACCESS_NETWORK_STATE	0.083
33	RECORD_AUDIO	0.054
34	READ_PHONE_STATE	0.039
35	HOME	0.024

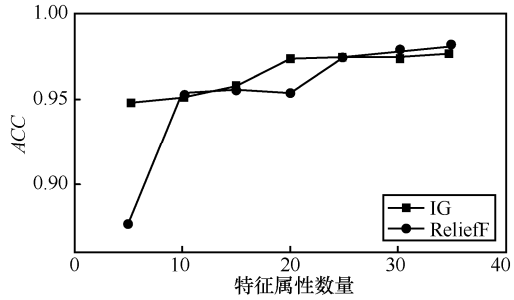


图 5　不同数量特征属性 *ACC* 比较

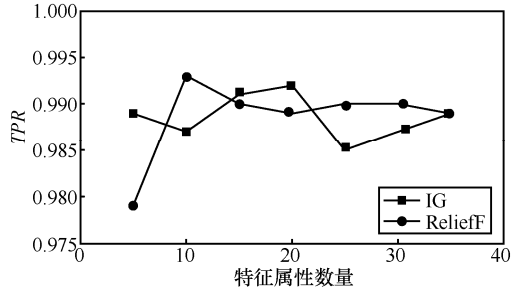


图 6 不同数量特征属性 *TPR* 比较

由图 5 和图 6 可知，改进后的 RF 分类器具有较好的检测分类效果，*ACC* 值基本高达 95%以上；在选取不同数量的特征属性时，IG 算法较 ReliefF 算法更加稳定；在特征属性数量大于 20 后，*ACC* 值较高并趋于平稳。

5.4　对比实验与分析

根据 4.3 节中的 IRF 算法分类实验结果，选取 IG 算法作为特征优化算法，并选取特征向量数量为 20 的特征向量集合作为对比实验的输入。将 IRF 算法集成到 Weka 中，同时使用 Weka 中集成的数据挖掘算法：NB、J48 和 RF，实验同样采用 10 折交叉验证的方法，实验结果如表 4 和表 5 所示。

表 4　　不同分类算法的分类效果对比

指标	NB	J48	RF	IRF
真正类 *TP*	1 233	1 240	1 250	1 254
假正类 *FP*	101	50	40	31
真负类 *TN*	573	624	634	643
假负类 *FN*	27	20	10	6
真正类 *TPR*	0.979	0.984	0.992	0.995
假正类 *FPR*	0.150	0.074	0.059	0.046
分类精度 *ACC*	0.934	0.964	0.974	0.981

由表 4 和表 5 可知，IRF 算法的分类精度高达 98%，优于传统 RF 算法，并且明显高于其他几类数据挖掘算法；在建模时间上，IRF 算法的处理时间与其他几类数据挖掘算法相比较长。首先，RF 算法本身是一种集成学习算法，与单个分类器相比，具有更好的分类效果。其次，IRF 算法对 RF 算法的投票原则进行了改进，对强分类器赋予较高的权重值，而对弱分类器赋予较低的权重值，导致分类效果提高，但建模时间相对增加。

表 5　　不同分类器的建模时间对比

算法	建模时间/s
NB	0.01
J48	0.06
RF	0.22
IRF	0.24

文献[7]的检测方案和文献[8]中的检测工具 Androdect 均采用静态检测与动态检测相结合的方法。本文所提出的 IRFCM 属于静态检测，与上述 2 种方法相比，IRFCM 的设计与实现较为简单且具有较高的检测效果，分类精度高达 98.1%，高于文献[8]的 Androdect 分类精度 94.24%。由于上述 2 个分类精度数据是在不同的实验环境下获得的，该对比的说服性相对较弱，故在下一步研究工作中将开展相同环境下的对比实验。

6　结束语

针对 Android 平台恶意软件泛滥以及现有检测算法检测精度不够高等问题，提出了 IRFCM 检测模型，一方面，对 RF 算法简单投票原则进行改进，提出了一种加权投票机制，对每棵决策树赋予不同权重，区分出强分类器与弱分类器的差异；另一方面，在特征属性的选择上选取 Permission 信息以及 Intent 信息，增强了对 Android 应用的区分度。通过对比实验结果表明该分类模型拥有较高的检测分类精度。

本文提出的基于改进随机森林算法的 Android 恶意软件检测方法还存在一些有待改进的地方：1) 仅针对 AndroidManifest.xml 文件中的 2 类标签信息进行检测分析，对于其他特征，如 API 特征、指令特征等的检测效果还有待进行实验验证和分析；2) 对 apk 文件批量反编译耗时较长，可采用并行方法提高处理效率；3) 实验样本集还不够完备，良性样本的数量与恶意样本的数量尚不够均衡；4) 在相同实验平台和环境下开展 IRFCM 与 THEA 的分类对比实验，对本文方法进行进一步验证和分析。

此外，为了进一步提高检测效率，未来可以考虑将数据处理和分类算法改进为并行化处理方式，并在更大的样本数据空间以及其他一些特征信息方面上开展检测分类实验研究分析。

参考文献：

[1] 张怡婷, 张扬, 张涛, 等. 基于朴素贝叶斯的 Android 软件恶意行为智能识别[J]. 东南大学学报:自然科学版, 2015，45(2):224-230.
ZHANG Y T, ZHANG Y, ZHANG T, et al. Intelligent identification of malicious behavior in Android applications based on naive Bayes[J]. Journal of Southeast University: Natural Science Edition, 2015, 45(2): 224-230.

[2] 张锐, 杨吉云. 基于权限相关性的 Android 恶意软件检测[J]. 计算机应用, 2014, 34(5):1322-1325.
ZHANG R, YANG J Y. Android malware detection based on permission correlation[J]. Journal of Computer Applications, 2014,34(5): 1322-1325.

[3] 许艳萍, 伍淳华, 侯美佳, 等. 基于改进朴素贝叶斯的 Android 恶意应用检测技术[J]. 北京邮电大学学报, 2016, 39(2):43-47.
XU Y P, WU C H, HOU M J, et al. Android malware detection technology based on improved naive Bayesian[J]. Journal of Beijing University of Posts and Telecommunications, 2016, 39(2):43-47.

[4] LI W, GE J, DAI G. Detecting malware for Android platform: an svm-based approach[C]// IEEE, International Conference on Cyber Security and Cloud Computing. New Jersey, USA: IEEE, 2015: 464-469.

[5] FEIZOLLAH A, ANUAR N B, SALLEH R, et al. Comparative study of *k*-means and mini batch *k*-means clustering algorithms in Android malware detection using network traffic analysis[C]// International Symposium on Biometrics and Security Technologies. New Jersey, USA: IEEE, 2014: 193-197.

[6] YUAN Z, LU Y, XUE Y. Droid detector: Android malware characterization and detection using deep learning [J]. Tsinghua Science & Technology, 2016, 21(1):114-123.

[7] 文伟平, 梅瑞, 宁戈, 等. Android 恶意软件检测技术分析和应用研究[J]. 通信学报, 2014, 35(8):78-85.
WEN W P, MEI R, NING G, et al. Malware detection technology analysis and applied research of Android platform[J]. Journal on Communications, 2014, 35(8): 78-85.

[8] 杨欢, 张玉清, 胡予濮, 等. 基于多类特征的 Android 应用恶意行为检测系统[J]. 计算机学报, 2014, 37(1):15-27.
YANG H, ZHANG Y Q, HU Y P, et al. A malware behavior detection system of Android applications based on multi-class features[J]. Chinese Journal of Computers, 2014, 37(1): 15-27.

[9] FEIZOLLAH A, ANUAR N B, SALLEH R, et al. A review on feature selection in mobile malware detection[J]. Digital Investigation, 2015, 6(13):22-37.

[10] SHARMA A, DASH S K. Mining API calls and permissions for android malware detection[M]. Cryptology and Network Security. Berlin, Germany: Springer International Publishing, 2014: 191-205.

[11] YANG X L. Malicious detection based on ReliefF and boosting multi-dimensional features[J]. Journal of Communications, 2015, 10(11): 910-917.

[12] ROBNIKŠIKONJA M, KONONENKO I. Theoretical and empirical analysis of ReliefF and RReliefF[J]. Machine Learning, 2003, 53(1): 23-69.

[13] BREIMAN L. Random forest [J]. Machine Learning, 2001, 5(1):5-32.

[14] ALAM M S, VUONG S T. Random forest classification for detecting android malware[C]// Green Computing and Communications. 2013: 663-669.

[15] 丰生强. Android 软件安全与逆向分析[M]. 北京：人民邮电出版社, 2013.
FENG S Q. Android software security and reverse analysis[M]. Beijing: PTPRESS, 2013.

[16] JIANG X，ZHOU Y. Dissecting Android malware: characterization and evolution [C]// IEEE Symposium on Security & Privacy. New Jersey, USA: IEEE, 2012: 95-109.

作者简介：

杨宏宇（1969-），男，吉林长春人，博士，中国民航大学教授，主要研究方向为网络信息安全。

徐晋（1991-），男，安徽合肥人，中国民航大学硕士生，主要研究方向为移动安全。