2017 年 11 月 第 11 期
Vol. 38 No. 11 2017

深度信念网络的恶意代码分类策略研究

罗世奇[1],田生伟[1],孙 华[1],禹 龙[2]

[1](新疆大学 软件学院,乌鲁木齐 830008)
[2](新疆大学 网络中心,乌鲁木齐 830046)
E-mail:tianshengwei@163.com

摘 要: 恶意代码的分类是恶意代码分析领域研究的重要问题之一.为解决这一问题,提出深度信念网络(Deep Belief Networks,DBN)的恶意代码分类策略.首先,从样本集中提取恶意代码图像特征、指令语句中的频度特征;其次,为确保准确率的提高,将上述两类特征进行融合,训练深度信念网络模型中的限制玻尔兹曼机(Restricted Boltzmann Machine,RBM)和反向传导算法(Back Propagation,BP).实验结果表明,提出的深度信念网络模型对恶意代码的分类平均准确率可达 95.7%,明显高于传统浅层机器学习模型 KNN 的 94.5%.

关 键 词: 深度信念网络;恶意代码;限制玻尔兹曼机;分类
中图分类号: TP309 **文献标识码:** A **文 章 编 号:** 1000-1220(2017)11-2465-06

Research Strategy of Classify Malicious Code into Families on the Method of Deep Belief Networks

LUO Shi-qi[1],TIAN Sheng-wei[1],SUN Hua[1],YU Long[2]
[1](School of Software,Xinjiang University,Urumqi 830008,China)
[2](Network Center,Xinjiang University,Urumqi 830046,China)

Abstract: The classification of malicious code is one of the most important issues in the field of malicious code analysis. To solve this problem, the Deep Belief Networks (DBN) malicious code classification strategy is proposed. Firstly, extract the characteristics of malicious code images from the sample set and the frequency characteristics in the instruction statement. Secondly, to ensure the improvement of accuracy, combine the two kinds of features above, to train Boltzmann Machine, (RBM) and Back Propagation (BP). The experimental results show that the average accuracy rate of the proposed model is 95.7%, which is significantly higher than that of the traditional shallow machine learning model KNN's 94.5%.

Key words: deep belief networks; malicious code; restricted boltzmann machine; classify

1 引 言

随着互联网技术的应用与普及,目前的 Internet 正面临着严峻的安全挑战,网络安全事件也层出不穷.自 2000 年以来,RedCode、Nimda、Slammer、Blaster、Sasser、Zotob、Saodengbo、conficker、Stuxnet 等重大网络蠕虫对整个互联网造成严重的危害,与此同时,网络病毒的数目也急剧的增加,其通过各种方式疯狂的传播自身,木马程序变种不断,对整个互联网产生了严重的威胁.随着桌面平台和移动终端的广泛使用与普及,近年来,恶意代码在移动平台上迅猛发展,真正的给用户带来了巨大的危害.[1] 恶意代码的特征提取、分类是恶意代码分析领域研究的热点问题之一.

目前典型的反病毒(广义的病毒泛指所有的恶意代码)技术包括:特征值查毒法、校验和技术、启发式扫描技术、虚拟机技术、行为监控技术、主动防御技术.即基于静态分析的方法[2],基于动态分析的方法[3],以及基于静态和动态相结合的方法[4].但是动态分析运行时受到显著的攻击、过度攻击、不能直接应用于移动设备,属于手动制作的检测模式,不能用于新的恶意软件实例,应用于行业内部,不公开透明.静态分析只产生小的运行时间开销、效率最高、误报率低、非运行状态时就可实现检测、同时也减小对恶意代码系统的影响.由于静态检测的方法效率最高、误报率低,因此被广泛的应用于恶意代码检测之中,是恶意代码检测的主流方法.[5]

深度学习是机器学习研究中的一个新领域,通过模拟人脑机制进行数据解释,并利用无监督学习算法训练模型,近年来在数据分类处理中得到广泛的应用.[6] 已有的基于机器学习的恶意代码检测方法(如 KNN[7]),建模过程中函数简单,对复杂的函数和分类问题表达受限,泛化能力受到制约[8],分类效果不明显,准确率不高.为提高分类准确率,达到恶意代码更优的分类效果.基于深度信念网络的恶意代码分类策略,采用静态分析的方法,融合恶意代码纹理特征以及指令频度特征,作为深度信念网络的输入,输出数据与标签数据对

收稿日期:2016-09-27 收修改稿日期:2016-11-28 基金项目:新疆自治区科技人才培养项目(QN2016YX0051)资助;国家社会科学基金项目(12CFX053)资助;2013 年度湖北省教育厅科学研究计划项目(B2013041)资助. 作者简介:**罗世奇**,男,1993 年生,硕士研究生,CCF 会员,研究方向为信息安全;**田生伟**(通信作者),男,1973 年生,博士,教授,研究方向为计算机技术、大数据处理;**孙 华**,女,1977 年生,博士,副教授,CCF 会员,研究方向为信息安全;**禹 龙**,女,1974 年生,硕士,教授,研究方向为计算机智能技术、计算机网络.

比,得出分类准确率.

2 算法模型

2.1 DBN 算法基础

2.1.1 限制玻尔兹曼机(RBM)

RBM 可被视作为一个无向图构成,是一种典型基于能量的模型(Energy-Based Model,EBM),其中一层为隐藏层,另外一层为可视层,其结构如图 1 所示.

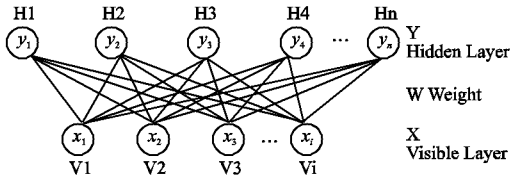

图 1 RBM 结构图

Fig. 1 RBM structure

其中,V = (V_1,V_2,V_3,V_i)表示可见层,用于观测数据,H = (H_1,H_2,H_3,H_n)表示隐藏层,用于特征提取;W 是两层之间的连接权值,通过无监督贪心算法获取,X 表示可见层各节点之间的偏移量,Y 表示隐藏层各节点之间的偏移量.

假设所有隐藏节点和可见节点均为二值变量,V_i 表示第 i 个可见节点的状态,H_j 表示第 j 个隐藏节点的状态. 针对一组给定状态为(V,H)的 RBM 模型而言,可见层输入 v 向量和隐含层输出向量 h 之间的能量函数值为:

$$E(v,h|\theta) = -\sum_{i\in visible} x_i v_i - \sum_{j\in hidden} y_j h_j - \sum_{i,j} v_i h_j w_{ij} \quad (1)$$

这两者的联合概率:

$$p(v,h|\theta) = \frac{1}{\sum_{v,h} e^{-E(v,h|\theta)}} e^{-E(v,h|\theta)} \quad (2)$$

但在实际问题当中,由于 RBM 定义的 V 分布非常重要,即联合概率的边缘分布 $P(V|\theta)$:

$$P(V|\theta) = \frac{1}{\sum_{V,H} e^{-E(V,H|\theta)}} \sum_{H} e^{-E(V,H|\theta)} \quad (3)$$

但是 RBM 的结构比较特殊,给定可见节点的状态的条件下,各个隐藏节点的状态相互条件独立,即已知第 i 个可见节点的激活概率或第 j 个隐藏节点的激活概率,其分布为:

$$P(V_i = 1|H;\theta) = f(\sum_j w_{ij} h_j + x_i) \quad (4)$$

$$P(H_j = 1|V;\theta) = f(\sum_i w_{ij} v_i + y_j) \quad (5)$$

其中 $f(x)$ 为 Sigmod 函数,$f(x) = \frac{1}{1+e^{-x}}$.

基于以上算法,RBM 采用层层迭代的方式进行训练,最后获得学习参数 $\theta = (w_{ij}, x_i, y_j)$,用来拟合给定的训练数据,学习参数通过在训练集合上的极大对数似然函数得到.

$$\theta^* = \arg\max L(\theta) = \arg\max_{\theta} \sum_{t=1}^{T} \log P(v^{(t)}|\theta) \quad (6)$$

其中,$P(V|\theta)$为似然函数,$v^{(t)}$ 表示第 t 个训练样本. 在文献[9]中提出的对比散度快速学习法,通过随机梯度上升法求解(3)(6)式,各参数更新如下:

$$-W \leftarrow W + \varepsilon(P(h=1|v)v^T - P(h^1=1|v)v^{1T})$$
$$-a \leftarrow a + \varepsilon(v - v^1)$$
$$-b \leftarrow b + \varepsilon(P(h=1|v)v^T - P(h^1=1|v)v^{1T}) \quad (7)$$

其中,ε 是学习率. 当重构的可视层 v^1 与原可视层 v 误差 $E = \|v^1 - v\|$ 小于某个阈值时,终止训练.

2.1.2 反向传导算法(Back Propagation,BP)

BP 神经网络是有监督多层前馈网络,其网络结构包括输入层、隐藏层、输出层,其结构如图 2 所示.

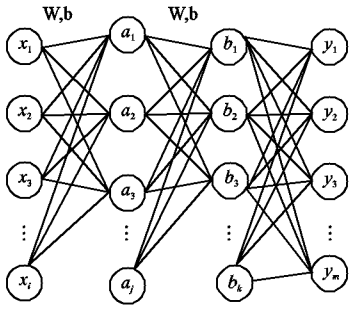

图 2 BP 结构图

Fig. 2 BP structure

其中输入 Input 为 X = {x_1,x_2,x_3,…,x_i},输出 Output 为 Y = {y_1,y_2,y_3,…,y_i}.

2.2 基于深度信念网络(DBN)的恶意代码分类策略

2.2.1 DBN

深度信念网络的概念是由 Hinton 等人于 2006 年在 Science 上提出来的,是指通过组合较低层的特征形成较高层的

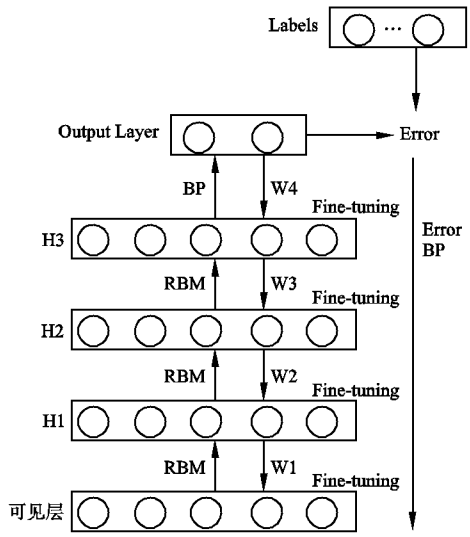

图 3 DBN 结构图

Fig. 3 DBN structure

具有代表性的特征,用来挖掘数据特征的优化分布,DBN 作为一种深度学习模型,成功用于图像处理以及自然语言处理领域[10,11]. 从结构上来看,DBN 是由多层 RBM 和一层 BP 构成,如图 3 所示.

2.2.2 基于 DBN 的恶意代码分类策略

本文深度信念网络训练步骤具体如下:

A)带标签的恶意代码进行预处理;

B)将 A)中预处理的数据作为输入,充分训练第一层 RBM;

C)固定第一层的 RBM 的权值和偏移量,使用其隐藏节

点,作为第二层 RBM 的输入向量;

D)充分训练第二层 RBM 后,将第二层 RBM 堆叠在第一层 RBM 上方;

DBN 输入:$V=[v_1,v_2,\cdots,v_k]=[feature_1,feature_2]$

$feature_1=imgfeatures.csv=[pixnum_1,pixnum_2,\cdots,pixnum_n]$

$feature_2=n-gramfeature.csv=[frequency_1,frequency_2,\cdots,frequency_m]$

E)重复执行第 C)步直到训练完所有的 RBM;

F)最后一个 RBM 的输出作为第一层的输入,初始化 W 和 b;

G)采用 BP 对整个网络进行调整.

3　恶意代码实验检测方案

我们从微软开源数据集中采集大量的训练和测试样本,用于后期的特征学习、分类预测,接下来进行特征提取,最后进行分类,得出准确率.

3.1　实验数据源

实验数据来自微软开源的恶意代码数据[1],数据集中总共有 9 类恶意代码,每个数据集样本包含了两个文件,一个是十六进制表示的. bytes 文件,另外一个是利用 IDA 反汇编生成的. asm 文件.

图 4 展示了实验中样本数据 0ACDbR5M3ZhBJajygTuf. bytes 的信息.

```
004010F0  B4 00 00 40 00 02 20 25 08 00 00 00 00 00 00 00
00401100  08 00 00 50 00 08 40 50 00 02 06 22 08 85 30 00
00401110  00 80 00 80 60 00 09 00 04 20 00 00 00 00 00 00
00401120  00 82 40 02 00 11 46 01 4A 01 8C 01 E6 00 86 10
00401130  4C 01 22 00 64 00 AE 01 EA 01 2A 11 E8 10 26 11
00401140  4E 11 8E 11 C2 00 6C 00 0C 11 60 01 CA 00 62 10
00401150  6C 01 A0 11 CE 10 2C 11 4E 10 8C 00 CE 01 AE 01
00401160  6C 10 6C 11 A2 01 AE 00 46 11 EE 10 22 00 A8 00
00401170  EC 01 08 11 A2 01 AE 10 6C 00 6E 00 AC 11 8C 00
```

图 4　bytes 文件信息

Fig. 4　bytes file information

图 5 展示了实验中样本数据 0ACDbR5M3ZhBJajygTuf. asm 的信息.

3.2　实验数据源处理

实验中,从开源数据库中读取 9 类恶意代码的标签文件 trainLabels. csv,每类恶意代码随机抽取 M 个样本数据,生成训练(测试)子集 subtrainLabels. csv.

得到训练(测试)子集 subtrainLabels. csv 之后,通过 excel 处理重复的 ID,使训练(测试)数据具有唯一性.

```
.text:00402C5F 49                    dec     ecx
.text:00402C60                 06    push    es
.text:00402C61 80 75 F4        6D    xor     byte ptr [ebp+var_C],6Dh
.text:00402C65
.text:00402C65                 loc_402C65:  ;CODE XREF: sub_402C24+35j
.text:00402C65 E8 41 04        00 00 call    sub_4030AB
.text:00402C6A 83 C4 0C              add     esp,0Ch
.text:00402C6D 8B E5                 mov     esp,ebp
.text:00402C6F 5D                    pop     ebp
```

图 5　asm 文件信息

Fig. 5　asm file information

```
trainlabels = pd.read_csv('trainLabels.csv')
fids = []
opd = pd.DataFrame()
for clabel in range(1,10):
    mids = trainlabels[trainlabels.Class == clabel]
    mids = mids.reset_index(drop = True)
    rchoice = [rs.randint(0,len(mids)-1) for i in range(M)]
    print rchoice
    rids = [mids.loc[i].Id for i in rchoice]
    fids.extend(rids)
    opd = opd.append(mids.loc[rchoice])
print len(fids)
opd = opd.reset_index(drop = True)
print opd
opd.to_csv('subtrainLabels.csv',encoding = 'utf-8',index = False)
```

3.3　实验数据特征抽取

3.3.1　. bytes 文件中的有效纹理特征提取

根据同一类型的恶意代码在纹理上存在一定的相似性,基于 DBN 的恶意代码分类策略,将. bytes 文件的纹理特征以灰度的方式展示出来,利用图像之中的纹理特征对恶意代码进行分类. 每一个. bytes 文件的字节范围在 00-FF 中间,可以对应灰度值 0-255,本文中的策略先将. bytes 文件的十六进制数据先转化为矩阵,然后转化为有效的特征 $feature_1$. [12]

```
    fh = numpy.array([int(hexst[i:i+2],16) for i in range(0,len(hexst),2)])  #按字节分割
    fh = numpy.reshape(fh[:rn*width],(-1,width))  #根据设定的宽度生成矩阵
    im = Image.fromarray(getMatrixfrom_bin(filename,512)) #转换为图像
    f.seek(startindex,0)
    content = f.read(pixnum) #选取. asm 特征数目
    hexst = binascii.hexlify(content)
```

3.3.2　. asm 文件中的有效特征提取

基于 DBN 方法的恶意代码分类策略,利用自然处理中 n-gram 的思想,将 n-gram 思想应用于. asm 文件的指令文件

[1] https://www.kaggle.com/c/malware-classification/

的特征提取之中.[13]

```
def getOpcodeNgram(ops,n=??): ##?? 为 n 的取值
        opngramlist = [tuple(ops[i:i+n]) for i in range(len
        (ops)-n)]
        opngram = Counter(opngramlist)
        return opngram
```

通过计算.asm 文件中指令 n-gram 的频度 f,作为 $feature_2$,然后进行输入.

3.3.3 数据特征融合

$feature_1$ 是通过.bytes 文件提取出的恶意代码的纹理特征,$feature_2$ 是通过.asm 文件提取出的恶意代码指令频度的特征.基于机器学习方法的恶意代码分类策略通过融合 $feature_1$ 与 $feature_2$,作为分类算法特征 V 进行深度信念网络的输入.

Labels = subtrainLabels.csv

$feature_1 = imgfeatures.csv = [pixnum_1, pixnum_2, \cdots, pixnum_n]$

$feature_2 = n-gramfeature.csv = [frequency_1, frequency_2, \cdots, frequency_m]$

$substrain_1 = pd.merge(feature_1, feature_2, on = ID)$

$substrain_2 = pd.merge(substrain_1, Labels, on = ID)$

$V = [v_1, v_2, \cdots, v_k] = [feature_1, feature_2]$

3.4 实验环境及需要的 Python 模块

表 1 实验环境

Table 1 Experimental environment

参 数	数 值
操作系统	Windows 7 64 位
CPU	Intel(R) Core(TM) i5-4200M CPU
内存	8G/ DDR3/1333MHz
硬盘	2T HDD
Java 版本	1.7.0_80
Python	2.7.3

Python 2.7.3 下,选择合适的开源库以及正确的版本对于实验的调试和结果有很重要的影响,表 2 列举了实验中需要的一些 Python 模块.

3.5 实验中的变量以及生成的数据

实验中的变量以及生成的数据如表 3 所示.

4 实验分析

在本实验中为了保证结果的真实可靠性,根据算法策略分别选取了 1000、1405、2000、3178、3626 个样本作为实验数据,然后从实验数据中随机选取 60% 的数据样本作为训练集,剩余 40% 的作为测试集.

本文设置的迭代次数为 200 次,学习率为 0.26,权重衰减参数为 0.8.

```
dbn = DBN([trainX.shape[1],9], learn_rates = 0.26,
learn_rate_decays = 0.8, epochs = 200, verbose = 1)
knn = neighbors.KNeighborsClassifier(n_neighbors = 9,
weights = 'uniform', algorithm = 'auto', metric = 'pyfunc',
func = dtw_dist)
```

.bytes 纹理特征、.asm 指令频度特征、DBN 模型中网络输入维度特征 V 以及网络深度皆可对实验结果产生影响.

表 2 实验需要的 Python 模块

Table 2 Python module for the experiment

名 称	介 绍	版 本
numpy	Python 的一种开源的数值计算扩展	1.9.2
pandas	基于 Numpy 构建的含有更高级数据结构和工具的数据分析包	0.16.0rc1 (0.11.1)
PIL	Python 图像处理库(PIL)	1.1.7
Scikit-learn	用于机器学习的 Python 模块	0.16.1
scipy	python 用于数学计算的工具	0.15.1
twisted	事件驱动的网络框架	15.4.0
six	–	1.7.3
pip	python 包管理工具	8.1.2
wheel	python 生成包格式文件安装工具	0.29.0
datutil	日期时间模块的扩展	2.2
pyparsing	–	2.0.1
setuptools	安装第三方 python 包工具	0.16.1
pytz	python 时区设置模块	2016.6.1
nolearn	Python 的神经网络和机器学习库	nolearn – 0.6.0
theano	Python 深度学习框架	0.8.2
gdbn	Python 深度学习 DBN 模型	0.2

表 3 实验中生成的数据及实验中的变量

Table 3 Data generated in the experiment and the variables in the experiment

	说 明	文件名/备注
实验中生成的数据	提供的训练集标注	trainLabels.csv
	随机生成训练子集	subtrainLabels.csv
	ASM 文件图像纹理特征	imgfeature.csv
	n-gram 特征	n-gramfeature.csv
实验中的变量	决策树数目	m_estimators
	.bytes 纹理特征	pixnum
	.asm 频度特征	frequency
	测试集比例	Test_size
	n-gram 特征	n-gramfeature.csv 中 n 的值

4.1 .bytes 纹理特征 $feature_1$ 选择

实验中,选取合适的.bytes纹理特征对于实验结果有着重要的影响.随着.bytes 纹理特征数目的增加,准确率呈现下降的趋势,如表 4 和图 6 所示,实验中.bytes 纹理特征选择 1500 维.

表 4 bytes 纹理特征对于实验结果的影响

Table 4 Effect of.bytes texture features on experimental results

.bytes Data features	data = 1000, n = 3, f > 500	data = 1405, n = 3, f > 500
800	0.835	0.907
1000	0.915	0.957
1500	**0.9225**	**0.9608**
2500	0.895	0.9448
3000	0.885	0.932
4000	0.85	0.923

4.2 .asm 指令频度特征 $feature_2$ 选择

实验中,.asm 指令的频度特征也对于实验结果有着重要的影响,随着指令的频度特征的增加,分类准确率呈现先上升后下降的趋势,在频度选取为 500 的时候分类效果最佳,如表 5 和图 7 所示. 实验中,指令的频度特征选择 500 作为特征进行输入.

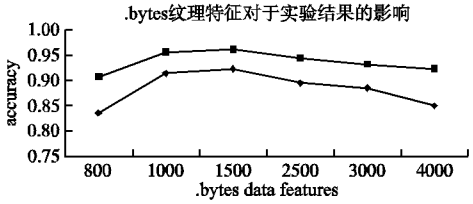

图 6　bytes 纹理特征对于实验结果的影响

Fig. 6　Effect of. bytes texture features on experimental results

表 5　指令的频度特征对于实验结果的影响

Table 5　Effect of frequency features on experimental results

frequency	data = 1000, n = 3, features1 = 1500	data = 1405, n = 3, features1 = 1500
100	0.8975	0.957
200	0.83	0.951
300	0.8875	0.957
500	**0.9225**	**0.9608**
1000	0.8925	0.9501
1500	0.9025	0.955

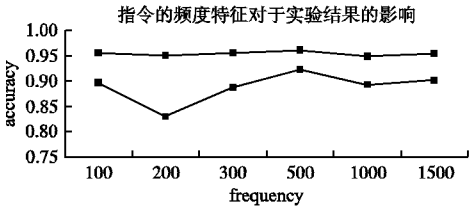

图 7　指令的频度特征对于实验结果的影响

Fig. 7　Effect of frequency features on experimental results

4.3 DBN 层数的选择

DBN 的层数同样对于实验结果以及准确率的提升高有着重要的影响,如表 6、图 8 所示.

表 6　DBN 层数对于实验结果的影响

Table 6　Effect of DBN Layer features on experimental results

DBN Layer	data = 1000, n = 3, f > 500, $features_1$ = 1500	data = 1405, n = 3, f > 500, $features_1$ = 1500
2	**0.9225**	**0.9608**
3	0.1975	0.281
4	0.17	0.258

根据. bytes 纹理特征、. asm 指令的频度特征以及 DBN 的层数的选择,实验中,我们选取. bytes 纹理特征为 1500 维,指令的频度特征选择为 500,DBN 层数选择为 2 层.

4.4 DBN 与 KNN 对比实验

为进一步验证本文提出的基于 DBN 的检测方案更适用于恶意代码的检测,将 DBN 模型与 KNN 模型分别用于恶意代码检测,然后进行对比,对比实验结果如表 7 和图 9 所示.

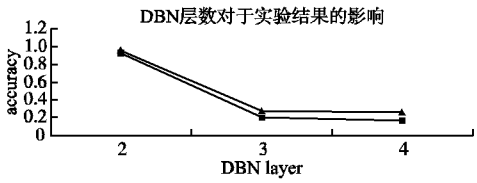

图 8　DBN 层数对于实验结果的影响

Fig. 8　Effect of DBN Layer features on experimentalresults

表 7　分类准确率对比

Table 7　Classification accuracy comparison

Data Num	DBN	KNN
1000	0.9225	0.9
1405	0.9608	0.948
2000	0.961	0.9562
3178	0.9716	0.95833
3626	0.9724	0.9648
average	**0.95766**	**0.94546**

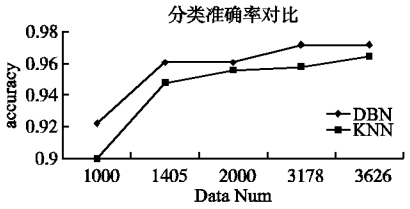

图 9　分类准确率对比

Fig. 9　Classification accuracy comparison

5　结束语

结合深度学习思想,利用融合恶意代码静态特征与深度信念网络分类模型相结合的方法. 创新之处在于,该方法不仅充分考虑了. bytes 文件纹理特征信息,还考虑了. asm 文件指令的特征,进行了特征融合.

实验结果表明,基于深度信念网络的算法与恶意代码多特征的融合,分类准确率明显高于浅层机器学习模型 KNN,并且随着样本数目的增加,准确率不断升高,通过实验结果来看,其平均分类准确率高达 95.7%,高于 KNN 模型的 94.5%.

References:

[1] Zhang Huan-guo, Du Rui-ying, Fu Jian-ming, et al. Information security engineer tutorial [M]. Beijing: Tsinghua University Press, 2016:467.

[2] Seshagiri P, Vazhayil A, Sriram P. AMA: static code analysis of web page for the detection of malicious scripts ☆[J]. Procedia Computer Science, 2016:768-773.

[3] Li Peng, Wang Ru-chuan. Malicious code dynamic analysis based on self similar characteristics[J]. Journal of Nanjing University of Posts & Telecommunications, 2012, 32(3):86-90.

[4] eufl P, Ferk M, Fitzek A, et al. Malware detection by applying knowledge discovery processes to application metadata on the Android Market (Google Play)[J]. Security & Communication Net-

works,2016,9(5):389-419.
[5] Li Ting, Dong Hang, Yuan Chun-yang, et al. Description of Android malware feature based on dalvik instructions[J]. Journal of Computer Research and Development,2014,51(7):1458-1466.
[6] Bengio Y, Yao L, Alain G, et al. Generalized denoising auto-encoders as generative models[C]. Advances in Neural Information Processing Systems,2013:899-907.
[7] Liu Xiao-ming. Anomaly detection of malicious Android applications based on k-nearest neighbor[D]. Beijing: Beijing Jiaotong University,2016.
[8] Yoshua Bengio. Learning deep architectures for AI [M]. Now Publishers Inc,2009.
[9] Hinton G E. Training products of experts by minimizing contrastive divergence [J]. Neural Computation (S0899-7667),2002,14(8): 1771-1800.
[10] Ch'Ng S I, Seng K P, Ang L M, et al. Block-based deep belief networks for face recognition[J]. International Journal of Biometrics, 2012,4(2):130-143.
[11] Yu D, Deng L. Deep learning and its applications to signal and information processing [Exploratory DSP][J]. IEEE Signal Processing Magazine,2011,28(1):145-154.
[12] Han Xiao-guang, Qu Wu, Yao Xuan-xia, et al. Research on malicious code variants detection based on texture fingerprint [J]. Joural on Communication,2014,35(8):125-136.
[13] Abouassaleh T, Cercone N, Ke? elj V, et al. N-gram-based detection of new malicious code[C]. International Computer Software & Applications Conference-Workshops & FAST. IEEE Computer Society,2004:41-42.

附中文参考文献:

[1] 张焕国,杜瑞颖,傅建明,等. 信息安全工程师教程[M]. 北京: 清华大学出版社,2016:467.
[3] 李 鹏,王汝传. 基于自相似特性的恶意代码动态分析技术[J]. 南京邮电大学学报(自然科学版),2012,32(3):86-90.
[5] 李 挺,董 航,袁春阳,等. 基于 Dalvik 指令的 Android 恶意代码特征描述及验证[J]. 计算机研究与发展,2014,51(7):1458-1466.
[7] 刘晓明. 基于 KNN 算法的 Android 应用异常检测技术研究[D]. 北京:北京交通大学,2016.
[12] 韩晓光,曲 武,姚宣霞,等. 基于纹理指纹的恶意代码变种检测方法研究[J]. 通信学报,2014,35(8):125-136.