

基于改进 SOINN 算法的恶意软件增量检测方法

张斌^{1,2}, 李立勋^{1,2}, 董书琴^{1,2}

(1. 信息工程大学, 河南 郑州 450001;
2. 河南省信息安全重点实验室, 河南 郑州 450001)

摘要: 针对基于批量学习的恶意软件检测方法存在检测模型动态更新困难、运算存储开销大的问题, 将改进的 SOINN 算法与有监督分类器有机结合, 利用 SOINN 算法的增量学习特性赋予恶意软件检测模型动态更新能力, 有效降低运算存储开销。首先对 SOINN 算法进行改进: 在 SOINN 算法竞争学习周期内, 根据全排列思想搜索所有样本输入次序下神经元的权重调节量, 计算所有权重调节量的平均值作为神经元最终权重调节量, 避免不同样本输入次序影响训练所得神经网络的稳定性, 使所得神经网络更能反映原始数据本质特征, 从而提高神经网络针对恶意软件检测的精度。然后采用非负矩阵分解和 Z-score 归一化对数据进行预处理, 将恶意软件行为特征向量从高维高数量级转换至低维低数量级, 在提高检测速度的同时有效降低高数量级维度对特征学习的不利影响, 进一步提高检测准确性。实验结果表明, 所提方法支持检测模型动态更新, 对未知新样本的检测准确率显著高于传统检测方法, 且运算存储开销更小。

关键词: SOINN 算法; 恶意软件检测; 神经网络; 增量学习; 入侵检测

中图分类号: TP393.08

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2019059

Malware detection approach based on improved SOINN

ZHANG Bin^{1,2}, LI Lixun^{1,2}, DONG Shuqin^{1,2}

1. Information and Engineering University, Zhengzhou 450001, China;
2. Key Laboratory of Information Security of Henan Province, Zhengzhou 450001, China

Abstract: To deal with the problems of dynamic update of detection model and high computation costs in malware detection model based on batch learning, a novel malware detection approach is proposed by combing SOINN and supervised classifiers, to reduce computation costs and enable the detection model to update dynamically with the assistance of SOINN's incremental learning characteristic. Firstly, the improved SOINN was given. According to the

收稿日期: 2019-01-08; **修回日期:** 2019-03-20

通信作者: 董书琴, dongshuqin377@126.com

基金项目: 河南省基础与前沿技术研究计划基金资助项目 (No. 2014302903); 信息工程大学新兴科研方向培育基金资助项目 (No. 2016604703)

Foundation Items: Henan Province Foundation and Frontier Technology Research Project (No. 2014302903); New Research Direction Cultivation Fund for Information Engineering University (No. 2016604703)

论文引用格式: 张斌, 李立勋, 董书琴. 基于改进 SOINN 算法的恶意软件增量检测方法[J]. 网络与信息安全学报, 2019, 5(6): 21-30.

ZHANG B, LI L X, DONG S Q, et al. Malware detection approach based on improved SOINN[J]. Chinese Journal of Network and Information Security, 2019, 5(6): 21-30.

whole alignment algorithm, search the adjusted weights of neurons under all input sequences in the learning cycle and then calculate the average value of all adjusted weights as the final result, to avoid SOINN's stability under different input sequences and representativeness of original data, therefore improve malware detection accuracy. Then a data preprocessing algorithm was proposed based on nonnegative matrix factor and Z-score normalization to transfer the malware behavior feature vector from high dimension and high order to low dimension and low order, to speed up and avoid overfitting and further improve detection accuracy. The results of experiments show that proposed approach supports dynamic updating of detection model and has a significantly higher accuracy of detecting unknown new samples and lower computation costs than tradition methods.

Key words: SOINN algorithm, malware detection, neural network, incremental learning, intrusion detection

1 引言

恶意软件是对怀有恶意目的或完成恶意功能软件的统称,对信息系统危害巨大^[1]。提高恶意软件的检测率有助于及时采取防御措施降低和消除恶意软件危害。随着网络攻击技术的发展,恶意软件入侵方式日益复杂、攻击行为愈加隐蔽,给恶意软件检测带来严峻挑战。

目前,恶意软件检测方法目前主要包括静态分析和动态分析。静态分析方法利用反编译等技术提取恶意软件的代码特征,再利用特征匹配等方式进行检测,部署简便且检测速度快,如控制流分析^[2]、信息流分析^[3]、操作码分析^[4]等。但随着攻击者在编码时越来越多地应用代码混淆、信息隐藏等手段,导致大量指令和函数被错误反编译,静态分析准确性有所降低。动态分析方法是指在虚拟环境中运行恶意软件实体,监控并分析其在运行过程中表现出的行为(如文件操作、网络通信等)来实施检测^[5-6],对于检测采用代码混淆等技术的恶意软件更具优势,主要包含以下两类方法:一类是基于行为序列分析的方法,通过监控并分析正常行为序列与异常行为序列的差异实施检测,如基于系统调用属性序列的检测^[7]、基于 shell 命令序列的检测^[8]和基于内核数据特征序列的检测^[9]等,此类方法普遍面临行为序列长度选择问题,序列长度选择过短可能增加误检率,过长又可能增加漏检率;另一类是基于行为特征学习的方法,利用特征工程构建恶意软件行为特征向

量并据此训练有监督分类模型实施检测,能够避免行为序列长度选择问题且具有较高准确率^[10-11],但是此类方法普遍以批量学习方式训练检测模型,存在模型更新困难和运算存储开销大的不足^[12]。

自组织增量式神经网络^[13](SOINN, self-organizing incremental neural network)是一种无监督聚类算法,通过对新样本的增量学习支持神经网络动态更新,而且增量学习时可自动保存已学知识避免再次学习历史样本,从而降低学习过程中的存储和运算开销。因此,本文将 SOINN 算法与有监督分类模型相结合,实现恶意软件检测模型的动态更新并降低运算存储开销。但是将 SOINN 算法引入恶意软件检测领域必须解决两个问题:一是 SOINN 算法对样本输入次序敏感,即同一样本集以不同次序训练 SOINN 可能会得出完全不同的神经网络,这将影响恶意软件检测精度;二是 SOINN 算法基于距离度量样本相似度并实施聚类,在处理恶意软件行为特征这种高维高数量级数据时运行速度较慢,且偏向于以高数量级维度区分样本而忽略低数量级维度对样本区分的作用,从而影响特征学习的准确性。

为解决上述问题,主要工作如下。

1) 提出 WM-SOINN (SOINN with weight modification) 算法,在 SOINN 竞争学习周期内,利用全排列思想搜索所有样本输入次序下神经元的权重调节量,计算所有权重调节量的平均值作为神经元最终权重调节量,避免个别样本输入次序影响训练所得神经网络的稳定性,使所得神经

网络更能反映原始数据本质特征,从而提高将神经网络应用于后续恶意软件检测的精度。

2) 采用非负矩阵分解^[14] (NMF, nonnegative matrix factor) 和 Z-score^[15]归一化方法处理训练数据,对恶意软件行为特征实施降维和数据标准化,在提高检测速度的同时降低高数量级维度对特征学习的不利影响,进一步提高 WM-SOINN 准确性。

3) 提出基于 WM-SOINN 的恶意软件增量检测方法,将具有增量更新特性的 WM-SOINN 与具有高准确率的有监督分类器有机结合,在保证高准确率情况下实现检测模型动态更新,并有效降低运算和存储开销。

2 改进 SOINN 算法

2.1 SOINN 算法

SOINN 属于在线学习的竞争型神经网络,与传统神经网络算法相比,SOINN 可以学习新样本包含的特征,并有效保留已有学习成果,实现神经网络动态更新。同时,SOINN 采用基于原型聚类的无监督学习思想,输出的神经网络可以反映原始数据特征,而 SOINN 的神经元动态调整和网络去噪机制可以保证神经网络的规模远小于原始数据,因此具有良好的数据压缩特性,用于增量学习时可以有效降低存储和运算开销。SOINN 算法的主要步骤如下^[13]。

1) 初始化单个竞争学习周期内的神经元集合 N , 神经元连接关系集合 $C \subseteq N \times N$ 。

2) 接收新的输入样本 ξ , 基于 L_2 范数查找 N 中与 ξ 最接近的神经元 s_1 和 s_2 (获胜神经元), 即

$$s_1 = \arg \min \| \xi - W_\tau \|, \tau \in N \quad (1)$$

$$s_2 = \arg \min \| \xi - W_\tau \|, \tau \in N \setminus \{s_1\} \quad (2)$$

其中, W_τ 表示神经元 τ 的权重。

3) 计算神经元 s_1 和 s_2 的相似度阈值 T_{s_1} 和 T_{s_2} , 对任意神经元 τ , 将 τ 的邻居神经元集合记为 N_τ , T_τ 的计算方法如下。

如果 $N_i \neq \emptyset$

$$T_\tau = \max \| W_\tau - W_j \|, j \in N_\tau \quad (3)$$

否则

$$T_\tau = \min \| W_\tau - W_j \|, j \in N \setminus \{\tau\} \quad (4)$$

判断是否生成新神经元, 若 $\| \xi - W_{s_1} \| > T_{s_1}$ 或 $\| \xi - W_{s_2} \| > T_{s_2}$ 成立, 则将 ξ 作为新生成神经元, 即 $N = N \cup \{\xi\}$, 并返回步骤 2), 否则继续执行。

4) 若 s_1 和 s_2 不存在连接关系, 则为其增加新连接, 即 $C = C \cup \{(s_1, s_2)\}$ 。

5) 更新获胜节点 s_1 所有边的年龄参数, 即 $\text{age}_{(s_1, i)} = \text{age}_{(s_1, i)} + 1$, $i \in N_{s_1}$, N_{s_1} 表示 s_1 的所有邻居神经元。

6) 更新获胜神经元权重。

$$W_{s_1} = W_{s_1} + \varepsilon(t)(\xi - W_{s_1}) \quad (5)$$

$$W_{s_2} = W_{s_2} + \varepsilon'(t)(\xi - W_{s_2}) \quad (6)$$

其中, $\varepsilon(t)$ 和 $\varepsilon'(t)$ 为学习率, $\varepsilon(t) = \frac{1}{t}$, $\varepsilon'(t) = \frac{t}{100}$, t 为神经元从训练开始截至目前的获胜次数。

7) 每学习完 λ 个样本代表竞争学习周期结束, 此时移除超龄的边, 即如果 $\text{age}_{(i, j)} > \text{age}_{\max}$, $C = C \setminus \{(i, j)\}$, 其中 $\text{age}_{\max}$ 为预定义参数。执行神经元去噪操作, 即删除低密度区域的神经元。若竞争学习未结束, 则返回步骤 2)。

算法最后输出神经元集合 N 和连接关系集合 C 。

2.2 WM-SOINN 算法

SOINN 算法以顺序学习的方式处理输入样本, 而学习率是神经元获胜次数的函数而非一个定值, 因此当样本以不同次序训练 SOINN 时, 获胜神经元被增量调整的权重也不相同, 进而导致同一样本集以不同次序训练 SOINN 得出不同结果。为降低 SOINN 对输入数据次序的敏感性, 提出神经元权重修正方法: 在每个竞争学习周期内, 记忆对神经元实施过权重调整的所有样

本, 基于全排列思想搜索这些样本在不同输入次序下对神经元权重的调节量, 计算所有权重调节量的平均值作为该竞争学习周期结束时神经元的最终权重调节量。根据以上思路, 提出 WM-SOINN 算法如下。

Step1 对任意神经元 τ , 在单个学习周期开始时, 记 τ 的当前权重为 O_τ , 当前周期内使 τ 成为 s_1 并对其进行权重调整的输入样本集合为 A_τ , 初始化 $A_\tau = \emptyset$ 。

Step2 输入样本 ξ 进行 SOINN 竞争学习, 即传统 SOINN 算法的步骤 1)至步骤 6)。

Step3 判断 τ 是否成为 s_1 , 是则继续执行 Step4, 否则返回 Step2。

Step4 将当前样本 ξ 并入集合 A_τ , 即 $A_\tau = A_\tau \cup \{\xi\}$ 。

Step5 判断学习周期是否结束, 是则继续执行 Step6, 否则返回 Step2。

Step6 根据 A_τ 和 O_τ 修正该学习周期内 τ 的最终权重, 执行 SOINN 步骤 7), 返回 Step1。

Step6 中的神经元权重修正详细方法如下。

将 A_τ 中的样本个数记为 t' , 分别计算 t' 个样本与 τ 初始权重的差值 $d_1, d_2, \dots, d_{t'}$, 其中 $d_i = \xi_i - O_\tau$ ($\xi_i \in A_\tau, 1 \leq i \leq t'$)。根据 SOINN 思想, τ 成为 s_1 时的学习率 $\varepsilon(t) = \frac{1}{t}$ 。因为 t' 个样本共有 $t'!$ 种不同的排列, 所以当 t' 个样本以不同次序输入模型时, τ 在单个学习周期内的权重调整量有 $t'!$ 种。记 τ 在样本集一个特定输入次序时的权重调整量为 ΔW_k ($1 \leq k \leq t'!$), 则

$$\Delta W_k = \frac{d_{y_1}}{t'' - t' + 1} + \frac{d_{y_2}}{t'' - t' + 2} + \dots + \frac{d_{y_{t'}}}{t''} \quad (7)$$

其中, $y_1, y_2, \dots, y_{t'} \in \{1, 2, \dots, t'\}$, 且 $y_1 \neq y_2 \neq \dots \neq y_{t'}$, t'' 为从训练开始到此竞争学习周期结束时 τ 成为第一获胜神经元 s_1 并实施权重更新的次数。

为减小不同输入次序对 ΔW_k 的影响, 取 ΔW_k

的算术平均值 $\Delta \bar{W}$ 作为 τ 的最终权重调整量, 即

$$\begin{aligned} \Delta \bar{W} &= \frac{\sum_{k=1}^{t'!} \Delta W_k}{t'!} \\ &= \frac{(t'-1)! \sum_{i=1}^{t'} d_i \left(\frac{1}{t'' - t' + 1} + \frac{1}{t'' - t' + 2} + \dots + \frac{1}{t''} \right)}{t'!} \\ &= \frac{\sum_{i=1}^{t'} d_i \left(\frac{1}{t'' - t' + 1} + \frac{1}{t'' - t' + 2} + \dots + \frac{1}{t''} \right)}{t'} \end{aligned} \quad (8)$$

记 τ 在本周期的最终权重为 E_τ , 则

$$E_\tau = O_\tau + \Delta \bar{W} \quad (9)$$

由于第二获胜神经元 s_2 的学习率 $\varepsilon'(t) = \frac{t}{100}$, 分母较大, 调整量较小, 对模型影响较小, 因此可以不予修正。

3 恶意软件增量检测方法

3.1 恶意软件行为特征向量构建

恶意软件主要通过调用操作系统提供的应用程序接口(API, application programming interface)实现文件访问、网络通信等行为, 因此可以通过记录 API 调用情况监控恶意软件行为。为逃避杀毒软件查杀和安全人员分析, 恶意软件通常将自身的恶意行为分散到多个不同的进程实施。这些分散执行的进程主要包含两类: ①恶意软件主进程创建的子进程; ②恶意软件通过代码注入技术注入的受害进程(通常是系统进程和杀毒软件白名单进程, 如 svchost.exe 和 excel.exe)。同时监控主进程、子进程和被注入进程的 API 调用情况, 更有利于发现复杂、隐蔽的恶意软件。

根据监控到的 API 调用情况构建恶意软件行为特征向量, 流程如图 1 所示。

Step1 在虚拟环境中执行待监测样本, 防止恶意样本对真实系统造成破坏。

Step2 利用 API 监控工具监控主进程、子进

程和注入进程的 API 调用情况。

Step3 根据监控结果构建行为特征向量, 其中, API_i (i 为正整数) 字段取值为自然数, 表示第 i 种 API 的调用次数, Child 和 Injected 字段取值为布尔类型, 分别表示是否创建子进程和是否有注入进程。

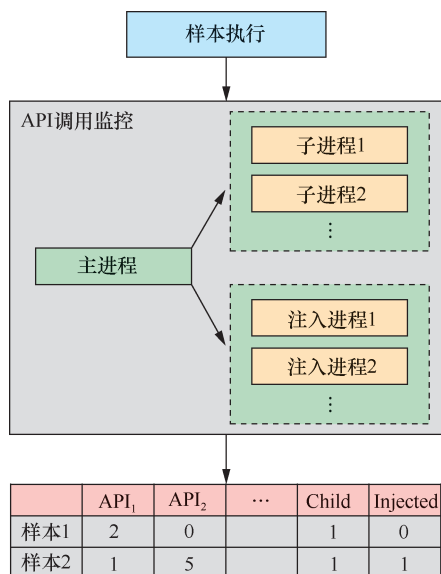


图1 恶意软件行为特征向量构建流程

3.2 数据降维与标准化

恶意软件种类繁多、功能各异, 利用特征工程构建的恶意软件行为特征向量往往维度过高, 且不同维度的取值存在数量级差异。为提高 WM-SOINN 对恶意软件行为特征向量的处理速度和特征学习的准确性, 设计恶意软件行为特征向量预处理算法, 通过数据降维提升 WM-SOINN 运算速度, 再实施数据标准化减小训练数据不同维度的数量级差异, 进而提高 WM-SOINN 准确性。

API 调用频数具有非负性, 因此恶意软件行为特征向量构成的特征空间矩阵也具有非负性, 而 NMF 可通过矩阵分解实现降维, 并且降维得到的低维矩阵仍然具有非负性, 与原矩阵物理含义相符, 便于理解和解释。在降维之后利用 Z-score 方法实施数据标准化, 使矩阵各维度平均

值为 0、标准差为 1, 有效减少数据不同维度存在的数量级差异。

恶意软件行为特征向量预处理算法步骤如下。

Step1 记恶意软件行为特征向量集合为 $X = \{x_0, x_1, \dots, x_m\}$, 对于任意 $x_i \in X$ ($0 \leq i \leq m$), x_i 共有 n 个维度, 将特征向量依次排列得到矩阵 $V_{m \times n}$ 。

Step2 构建 $\hat{V}_{m \times n} = W_{m \times l} \times H_{l \times n}$, 使 $V_{m \times n} \approx \hat{V}_{m \times n}$, 其中 $W_{m \times l} \geq 0$ 且 $H_{l \times n} \geq 0$ 。

Step3 利用欧式距离定义损失函数 $lost$ 用于度量 $V_{m \times n}$ 与 $\hat{V}_{m \times n}$ 的相似程度, 即 $lost = \|V_{m \times n} - \hat{V}_{m \times n}\|^2$ 。

Step4 求解 $W_{m \times l}$ 和 $H_{l \times n}$, 使 $\minimize \|V_{m \times n} - W_{m \times l} \times H_{l \times n}\|^2$, 其中 $W_{m \times l} \geq 0$ 且 $H_{l \times n} \geq 0$ 。

Step5 对于求得的矩阵 $W_{m \times l}$, 将其第 i 行第 j 列 ($0 \leq i \leq m$, $0 \leq j \leq l$) 的取值记为 $y_i[j]$,

计算第 j 列的平均值 $\mu_j = \frac{\sum_{i=0}^m y_i[j]}{m}$, 标准差

$$\sigma_j = \sqrt{\frac{1}{m} \sum_{i=0}^m (y_i[j] - \mu_j)^2}.$$

Step6 将第 i 行第 j 列的新取值设定为 $y_i^*[j] = \frac{y_i[j] - \mu_j}{\sigma_j}$, 输出处理完成后的矩阵 $W'_{m \times l}$ 。

上述步骤中, Step4 属于最优化问题求解, 具体方法见文献[14]。 $W_{m \times l}$ 可理解为 m 个样本在 l 维“新特征空间”中的取值, 而 $H_{l \times n}$ 为“新特征空间”与原始 n 维特征空间的映射关系。因为新旧特征空间的映射关系 $H_{l \times n}$ 固定, 所以用 $W_{m \times l}$ 可反映样本与原始 n 维特征空间的关系。而人为设定的 l 值通常远小于 n 值, 将完成数据标准化后的 $W'_{m \times l}$ 代入后续计算即可提高 WM-SOINN 运算速度和准确率。

3.3 基于 WM-SOINN 的恶意软件增量检测

WM-SOINN 属于无监督聚类学习算法, 直接用于恶意软件检测需要人为设定检测阈值, 通过判断待测样本与神经元的距离是否大于检测阈值而得出结论。这种检测方法存在两点不足: 一是

检测阈值的设定存在很强主观性, 阈值过大会增加漏检率, 过小又会增加误检率; 二是恶意软件与正常软件在特征空间的重合度极高, 这种基于距离阈值实施检测的精度有限。

将 WM-SOINN 与有监督分类模型结合实施恶意软件检测, 可以避免人为设定检测阈值带来的主观性, 同时利用有监督学习提高检测精度。方法的基本思想是通过 WM-SOINN 分类学习恶意软件与正常软件样本, 将学习所得的神经元用于训练有监督分类模型实施检测, 再利用检测结果对神经网络和检测模型实施增量更新。具体步骤如下。

Step1 对初始带标签样本集进行降维和标准化处理, 根据正负标签划分为正常训练样本 N -samples 与恶意训练样本 M -samples。

Step2 利用 WM-SOINN 分别学习 N -samples 和 M -samples 得到正常样本的神经网络 N -WM-SOINN 和恶意样本的神经网络 M -WM-SOINN。

Step3 将 N -WM-SOINN 中的正常神经元 N_{normal} 和 M -WM-SOINN 中的恶意神经元 $N_{\text{malicious}}$ 同时用于训练有监督分类器得到检测模型 model 。

Step4 对无标签待测样本 ξ 预处理后输入 model 进行检测, 生成带标签样本 ξ_{tag} 。

Step5 每检测完 θ 个无标签样本, 再利用生成的 θ 个带标签样本进行增量学习, 将其中的正常样本输入 N -WM-SOINN, 异常样本输入 M -WM-SOINN, 进行增量学习。

Step6 利用 N -WM-SOINN 和 M -WM-SOINN 增量学习后得到的新正常神经元 N'_{normal} 和新恶意神经元 $N'_{\text{malicious}}$ 再次训练有监督分类器更新 model , 重复执行 Step4 至 Step6, 实现增量式恶意软件检测。

上述步骤中, Step1 和 Step2 通过 WM-SOINN 学习初始样本特征并生成神经元, 减少了后续训练有监督分类器时所需的样本数。Step3 和 Step4

为有监督分类器训练与检测过程。Step5 将检测结果反馈至 WM-SOINN, 利用检测结果及时更新神经网络。Step6 将 WM-SOINN 增量学习后获得的新知识传递给有监督分类器, 最终实现检测模型的增量更新。在此过程中, θ 表示增量学习周期, 一般情况下, θ 值越小表明检测模型的更新速度越快, 学习新知识的能力越强, 但频繁的模型更新操作会增大系统运行开销, 实际应用时应结合检测性能要求和系统环境灵活设置 θ 。

4 实验

实验主要验证 WM-SOINN 与常用有监督分类器结合后的检测性能、算法运算效率和增量学习能力。

4.1 实验环境与数据集

在 Windows 平台上利用 Python 3.6.4 进行实验, 编程工具为 PyCharm, CPU 为 Intel i5-6300HQ, 内存为 8 GB。实验数据集为机器学习数据仓库 UCI 中的 “Dynamic Features of Virus Share Executables Data Set”, 该数据集包含 107 888 个软件样本的动态行为特征, 被广泛应用于恶意软件动态分析领域。数据集中的每一条样本记录均有 484 个特征维度, 其中第一个维度表示样本的评分 (0~1.0, 评分越高表示样本的危害越大), 其余 483 个维度对应样本的动态行为特征。

随机选取 20 000 个评分高于 0.75 的样本作为恶意样本, 10 000 个评分低于 0.25 的样本作为正常样本, 去除非 API 冗余特征后再经过 NMF 降维和 Z-score 标准化, 最终得到 30 000 个实验样本。

采用下列指标进行评价。

TP (true positive): 被正确判定为恶意的恶意样本数。

FP (false positive): 被错误判定为恶意的正常样本数。

TN (true negative): 被正确判定为正常的正常

样本数。

FN (false negative): 被错误判定为正常的恶意样本数。

FPR (false positive rate): 假阳性率, 即被错误判定的正常样本占有所有正常样本的比例。

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

Precision: 查准率, 在所有被判定为恶意的样本中真实的恶意样本所占的比例。

$$Precision = \frac{TP}{TP + FP}$$

Accuracy: 准确率, 所有被准确判定的样本占样本总量的比例。

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

其中, Precision 和 Accuracy 可以代表算法的准确性, 值越大表明算法越准确; FPR 可以代表算法的误检率, 值越小代表算法造成的误判越少。

4.2 检测性能

采用 K 折交叉验证的方式检验 WM-SOINN 分别与 3 类有监督分类器 (支持向量机 SVM、朴素贝叶斯 NB 和随机森林 RF) 结合后的恶意软件检测性能。随机分割样本集为 K 等份, 每次选取其中的 $K-1$ 份用于训练, 余下的 1 份用于检测, 最终检测结果取 K 次实验的平均值。实验结果如表 1 所示, 其中 SOINN-SVM 表示 SOINN 与 SVM 结合而构造的分类器, 其余算法名称的含义类似。若无特殊说明, NMF 降维时 $l=15$, SVM 的核函数为径向基函数 (RBF, radial basis function), NB 为 BernoulliNB, RF 包含 10 棵决策树且不对树的深度做限制, K -Means 算法的聚类数目为 20, SOINN 与 WM-SOINN 的 $age_{\max}=50$ 、 $\lambda=200$ 。

由表 1 可知, 原始的 SVM、NB 和 RF 中准确性最高的是 RF, 其 Precision 和 Accuracy 超过 0.99、FPR 小于 0.01; 其次是 SVM, 其 Precision

表 1 算法恶意软件检测性能对比

算法	K 值	FPR	Precision	Accuracy
SVM	5	0.077	0.924	0.931
	10	0.063	0.938	0.944
NB	5	0.085	0.911	0.891
	10	0.079	0.918	0.904
RF	5	0.008	0.992	0.994
	10	0.001	0.999	0.999
K -Means-SVM	5	0.076	0.922	0.914
	10	0.057	0.943	0.940
K -Means-NB	5	0.361	0.658	0.667
	10	0.336	0.691	0.709
K -Means-RF	5	0.037	0.963	0.964
	10	0.023	0.977	0.979
SOINN-SVM	5	0.075	0.926	0.930
	10	0.080	0.922	0.935
SOINN-NB	5	0.378	0.685	0.722
	10	0.329	0.727	0.773
SOINN-RF	5	0.030	0.970	0.975
	10	0.014	0.986	0.988
WM-SOINN-SVM	5	0.051	0.949	0.950
	10	0.047	0.953	0.952
WM-SOINN-NB	5	0.295	0.744	0.782
	10	0.288	0.752	0.792
WM-SOINN-RF	5	0.005	0.994	0.994
	10	0.007	0.993	0.995

不低于 0.92、Accuracy 不低于 0.93、FPR 不超过 0.08; 效果较差的是 NB, 其 Precision 和 Accuracy 不超过 0.92、FPR 超过 0.08。与经典 K -Means 算法相比, SOINN 和 WM-SOINN 能够自动确定聚类数目, 减少人为设定聚类数目导致的误差, 并且检测性能普遍优于经典的 K -Means 算法。在批量学习时, SOINN 和 WM-SOINN 与 SVM、RF 结合之后只对分类器的准确性和误检率造成了极

小的影响,以 RF 为例,其 Precision 不低于 0.986、Accuracy 不低于 0.988、FPR 不超过 0.014。实验结果还表明,WM-SOINN 与分类器结合后的 Precision 和 Accuracy 高于 SOINN 与分类器结合后的同类指标,而 FPR 也更低,证明 WM-SOINN 训练得出的神经网络更加稳定,输出的神经元也更能代表原始样本,神经元权重修正方法有效。

4.3 运算效率

首先分析 SOINN 与 WM-SOINN 的时间复杂度。SOINN 中的计算以线性为主,时间复杂度为 $O(n)$ (n 为样本个数),仅有获胜神经元查找步骤需要较为复杂的排序操作,本文实验选取冒泡排序算法,时间复杂度为 $O(m^2)$ (m 为神经元个数)。WM-SOINN 中加入的神经元权重修正算法同样为线性,因此 WM-SOINN 整体计算过程为线性,时间复杂度为 $O(n)$ 。SOINN 中的神经元动态调整和网络去噪机制可以保证神经元个数通常远低于样本个数,即 $m \ll n$,所以 SOINN 和 WM-SOINN 中的计算仍以线性为主。分别选取规模为 2 000、5 000、10 000、20 000 的数据集进行实验,比较 SOINN 与 WM-SOINN 的运算时间,结果如图 2 所示。

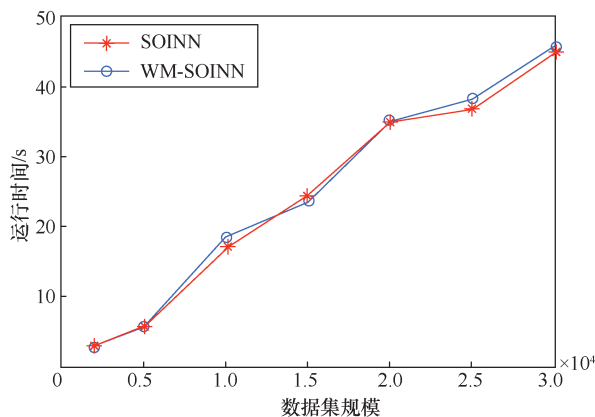


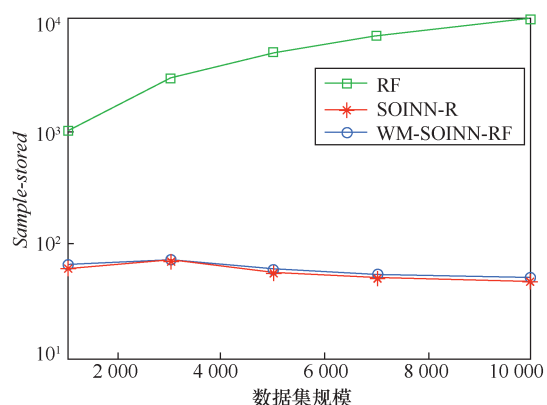
图 2 算法运行时间对比

由图 2 可知,对于同等规模数据集,增加了神经元权重修改操作的 WM-SOINN 与原始 SOINN 的运行时间基本相等,证明 WM-SOINN 不会造成更大的时间开销。此外,无论 SOINN 还

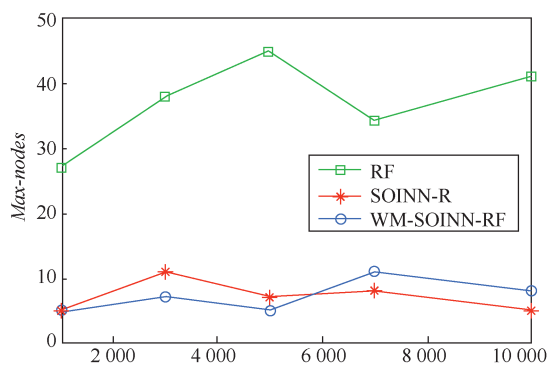
是 WM-SOINN,其运行时间均随着数据集规模的增加而近似线性增加,与时间复杂度分析结果相吻合。

下面以 SOINN、WM-SOINN 与 RF 结合为例比较算法在增量学习时的空间开销,分别选取规模为 1 000、3 000、5 000、7 000、10 000 的数据集进行实验,结果如图 3 所示。其中 Sample-stored 表示增量学习需要存储的样本数,也是神经网络输出神经元的数目,值越小表示需要的存储空间越少;Max-nodes 表示 RF 中决策树的最大节点数,Max-depth 表示 RF 中决策树的最大深度,在检测率相同的前提下,Max-nodes 和 Max-depth 越小表示检测模型需要的存储空间越小,同时表明检测模型的泛化能力越强,可以避免出现过拟合。

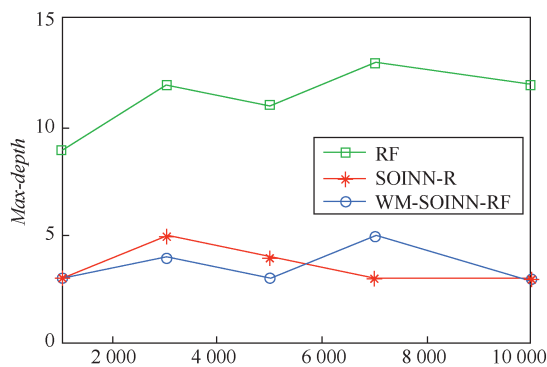
从 Sample-stored 指标可知,如果直接利用 RF 进行增量学习,则单次学习完成后需要存储所有样本,而与 SOINN 或 WM-SOINN 结合之后,只需要存储神经网络学习所得的极少量神经元即可进行后续增量学习。从图 3(a)可以看出,对于同一数据集,在样本分布均匀的情况下,神经网络学习所得的神经元数目是相对稳定的,并不会因数据集规模的扩大而大量增加,甚至可能减少。并且对于本实验所用数据集,当数据集规模大于 10^4 时,数据压缩比率超过 0.99。由表 1 可知 RF、WM-SOINN-RF 具有同等检测性能,SOINN-RF 检测性能略差,但是通过比较 Max-nodes 和 Max-depth 指标可以看出,经过 SOINN 或 WM-SOINN 压缩数据后再训练得到的 RF,与利用原始数据直接训练得到的 RF 相比具有更小的规模,Max-nodes 平均减少 78.5%,Max-depth 平均减少 63.4%。比较 SOINN-RF 和 WM-SOINN-RF 的 Sample-stored 指标可以发现两者具有同等的数据压缩能力,而 4.2 节实验已经表明 WM-SOINN-RF 的检测性能更优。上述结论表明本文方法可以显著降低训练数据和检测模型的存储开销。



(a) Sample-stored指标



(b) Max-nodes指标



(c) Max-depth指标

图3 算法空间开销对比

4.4 增量学习能力

以 WM-SOINN-RF 为例检验算法增量学习能力。首先利用 2 000 个样本训练初始检测模型，再利用余下的 28 000 个样本进行检测与增量学习，通过比较不同增量学习周期 θ 下检测模型的检测性能来分析算法增量学习能力。实验中记录每个学习周期完成后检测模型的检测性能，最终取所有学习周期检测性能的算术平均值，结果如表 2 所示。

表2 算法增量学习性能

周期	<i>FPR</i>	<i>Precision</i>	<i>Accuracy</i>
∞	0.248	0.762	0.770
1 000	0.187	0.827	0.854
500	0.154	0.856	0.880
200	0.105	0.899	0.915
100	0.049	0.952	0.960

从表 2 可以看出，当 $\theta = \infty$ ，即检测模型在初始批量训练完成后不再学习新样本中的知识时，对剩下的 28 000 个样本实施检测，*Precision* 和 *Accuracy* 不超过 0.77 且 *FPR* 为 0.248。随着学习周期的缩短，检测模型的准确性逐渐提高，误检率逐渐下降。当 $\theta = 100$ ，即每检测 100 个样本便进行一次检测模型增量更新，*Precision* 达到 0.952、*Accuracy* 达到 0.96、*FPR* 降至 0.049。该实验表明算法具有良好的增量学习能力，与有监督分类器结合可以显著提高检测模型对未知新样本的检测准确性。

5 结束语

本文将 SOINN 引入恶意软件检测领域并进行改进，将改进后的 WM-SOINN 算法与有监督分类器结合构建恶意软件增量检测模型，在保证检测精度的同时实现检测模型的动态更新，并能显著减少运算和存储开销，为恶意软件检测提供了新的解决方法。从检测性能、运算效率和增量学习能力 3 个角度进行实验分析，证明了所提方法的有效性。下一步将研究 WM-SOINN 算法与其他分类器相结合的方法，提高算法的适应性。

参考文献：

- [1] 刘剑, 苏璞睿, 杨珉, 等. 软件与网络安全研究综述[J]. 软件学报, 2018(1): 42-68.
LIU J, SU P R, YANG M, et al. Software and cyber security—a survey[J]. Journal of Software, 2018(1): 42-68.
- [2] SONG F, TOUILI T. Efficient malware detection using model-checking[C]//FM: Formal Methods. 2012: 418-433.

- [3] YIN H, SONG D. Privacy-breaching behavior analysis[M]. Automatic Malware Analysis. 2013: 27-42.
- [4] TODERICI A H, STAMP M. Chi-squared distance and metamorphic virus detection[J]. Journal in Computer Virology, 2013, 9(1): 1-14.
- [5] EGELE M, SCHOLTE T, KIRDA E, et al. A survey on automated dynamic malware-analysis techniques and tools[J]. ACM Computing Surveys, 2008, 44(2): 1-42.
- [6] 陈志锋, 李清宝, 张平, 等. 基于聚类分析的内核恶意软件特征选择[J]. 电子与信息学报, 2015, 37(12): 2821-2829.
CHEN Z F, LI Q B, ZHANG P, et al. Signature selection for kernel malware based on cluster analysis[J]. Journal of Electronics & Information Technology, 2015, 37(12): 2821-2829.
- [7] 李珍, 田俊峰, 杨晓晖. 基于系统调用属性的程序行为监控[J]. 计算机研究与发展, 2012, 49(8): 1676-1684.
LI Z, TIAN J F, YANG X H. Program behavior monitoring based on system call attributes[J]. Journal of Computer Research and Development, 2012, 49(8): 1676-1684.
- [8] 田新广, 段泳毅, 程学旗. 基于 shell 命令和多重行为模式挖掘的用户伪装攻击检测[J]. 计算机学报, 2010, 33(4): 697-705.
TIAN X G, DUAN M Y, CHENG X Q. Masquerade detection based on shell commands and multiple behavior pattern mining[J]. Chinese Journal of Computers, 2010, 33(4): 697-705.
- [9] 陈志锋, 李清宝, 张平, 等. 基于数据特征的内核恶意软件检测[J]. 软件学报, 2016, 27(12): 3172-3191.
CHEN Z F, LI Q B, ZHANG P, et al. Data characteristics-based kernel malware detection[J]. Journal of Software, 2016, 27(12): 3172-3191.
- [10] GREENGARD S. Cyber security gets smart[J]. Communications of the ACM, 2016, 59(5): 29-31.
- [11] SADEGHI A, BAGHERI H, GARCIA J. A taxonomy and qualitative comparison of program analysis techniques for security assessment of android software[J]. IEEE Transactions on Software Engineering, 2016.
- [12] 杨雅辉, 黄海珍, 沈晴霓, 等. 基于增量式 GHSOM 神经网络模型的入侵检测研究[J]. 计算机学报, 2014(5): 1216-1224.
YANG Y H, HUANG H Z, SHEN Q N, et al. Research on intrusion detection based on incremental GHSOM[J]. Chinese Journal of Computers, 2014(5): 1216-1224.
- [13] 邱天宇, 申富饶, 赵金熙. 自组织增量学习神经网络综述[J]. 软件学报, 2016, 27(9): 2230-2247.
- QIU T Y, SHEN F R, ZHAO J X. Review of self-organizing incremental neural network[J]. Journal of Software, 2016, 27(9): 2230-2247.
- [14] LEE D D, SEUNG H S. Learning the parts of objects by non-negative matrix factorization[J]. Nature, 1999, 401(6755): 788-791.
- [15] WANG W, ZHANG X, GOMBAULT S, et al. Attribute normalization in network intrusion detection[C]//International Symposium on Pervasive Systems, Algorithms, and Networks. 2010: 448-453.

[作者简介]



张斌 (1969—), 男, 河南南阳人, 信息工程大学教授、博士生导师, 主要研究方向为网络空间安全。



李立勋 (1994—), 男, 四川都江堰人, 信息工程大学硕士生, 主要研究方向为动态目标防御。



董书琴 (1990—), 男, 河北邢台人, 信息工程大学博士生, 主要研究方向为网络安全态势感知。