doi: 10.7690/bgzdh.2025.12.006

基于改进欧几里得距离的恶意代码行为检测方法

毕凯峰，王　健
（南方电网数字电网研究院有限公司，广州 510000）

摘要：为解决传统恶意代码行为检测方法所需检测时间长、稳定性及其综合性能差的问题，提出一种基于改进欧几里得距离的恶意代码行为检测方法。分析恶意代码行为数据类型，计算恶意代码行为相似匹配度；借助支持向量机完成恶意代码行为数据的划分；确定恶意代码行为数据冗余度，完成数据的预处理；通过对欧几里得距离矩阵与恶意代码行为数据冗余度进行点乘运算，实现恶意代码行为的检测。实验结果表明：该方法检测速度快、精确度和稳定性高，具有良好的召回率及综合指标，能够提高恶意代码行为检测的有效性。

关键词：恶意代码；行为检测；相似匹配度；欧几里得距离；数据冗余度
中图分类号：TP393　**文献标志码：**A

Malicious Code Behavior Detection Method Based on Improved Euclidean Distance

Bi Kaifeng, Wang Jian
(*Digital Grid Research Institute*, *China Southern Power Grid*, *Guangzhou* 510000, *China*)

Abstract: To solve the problems of long detection time, poor stability, and overall performance of traditional malicious code behavior detection methods, a malicious code behavior detection method based on improved Euclidean distance is proposed. Analyze the data types of malicious code behavior and calculate the similarity matching degree of malicious code behavior; Using support vector machines to partition malicious code behavior data; Determine the redundancy of malicious code behavior data and complete data preprocessing; By performing dot multiplication on the Euclidean distance matrix and the redundancy of malicious code behavior data, malicious code behavior detection is achieved. The experimental results show that this method has fast detection speed, high accuracy and stability, and can effectively detect the behavior of malicious code to a certain extent. It has good recall rate and comprehensive indicators, which can improve the effectiveness of malicious code behavior detection.

Keywords: malicious code; behavior detection; similarity matching degree; Euclidean distance; data redundancy

0　引言

我国的网络安全形势严峻，工业控制系统已经成为国外敌对势力重点关注的目标，部分关键信息基础设施已处于被敌对势力掌控、敏感信息被窃取的“攻陷”状态。工业控制系统长期未安装补丁，对移动介质管控不严成为恶意代码入侵、扩散的重要突破口。

恶意代码自从被首次检测出来后，就成为黑客、不法分子、国外敌对势力等进行网络渗透的首选工具。国内外发生的多起重大网络安全事故均和恶意代码相关，定制化、复合型的木马病毒已经被制作成战争武器[1]。随着移动互联网、数字化、5G 等新技术的发展，针对恶意代码的攻防对抗已经成为网络空间战场主要力量[2]，针对恶意代码的快速、高效检测成为网络安全商业机构、研究部门乃至国家关注的重点内容；所以，对恶意代码行为进行检测的研究显得尤为重要。刘建松等[3]提出了一种基于行为关系网络的恶意代码检测方法，该方法通过调用 API、注册表访问和文件读写操作等行为记录来构建行为关系网络，再使用一种基于元图的方法来计算样本之间的相似度矩阵，并用一种自定义核的支持向量机模型来进行训练和预测。杨吉云等[4]提出了一种基于系统行为序列特征的 Android 恶意代码检测方法，该方法划分了程序运行发生的敏感 API 调用、文件访问、数据传输等系统活动的行为序列，基于马尔科夫链模型将系统行为序列转换为状态转移序列并生成状态转移概率矩阵，将状态转移概率矩阵和状态发生频率作为特征集对 SAEs 模型进行学习和训练，最后利用训练后的 SAEs 实现对 Android 恶意代码的检测。

上述 2 种方法均存在检测时间长、准确性低、

收稿日期：2024-10-16；**修回日期：**2024-11-20
第一作者：毕凯峰(1983—)，男，广东人，硕士。

稳定差的问题；为此，笔者提出一种基于改进欧几里得距离的恶意代码行为检测方法。

1　恶意代码行为分析

1.1　恶意代码行为相似匹配度计算

通过对恶意代码行为的相似匹配进行分析，可以为恶意代码的行为进行分类。对于恶意代码行为的相似匹配，需要将各个字段的哈希变换成一个多维的矢量，并要求该矢量对应多个特征，在不同特征下的矢量是唯一的；因此，对文档中所包含的矢量加权处理，就是对参数的特征权重处理。通过处理得到的文档矢量，即分词出现多数后，再进行特征匹配计算[5]。

为加快算法的计算速度和对矢量进行进一步压缩，必须求出 2 个矢量间的 Haming 间距，其表达式是：

$$F(\boldsymbol{x},\boldsymbol{y})=\sum_{m}^{k=1}x_k\oplus y_k。\tag{1}$$

式中：$\oplus$为恶意代码行为加运算符号；x,y 分别为 2 个恶意代码行为二进制向量。当矢量的 1 维数据大于 0 时，它的特征值就是 1，反之就是 0。该步骤可以迅速获得全空间的所有矢量信息，从而避免了矢量空间中各向异性的繁琐运算。利用这种技术，各分析报表能够产生自身的特征，并对其进行分析，从而得到其相似性。

通过调用序列集，将特征库中恶意代码行为的特征分别进行匹配，按照式(2)计算两者的相似匹配度，公式为：

$$\mathrm{sim}=F(\boldsymbol{x},\boldsymbol{y})\frac{\sum_{p=1}^{m}\mathrm{match}(h_p)\times r_p}{\sum_{p=1}^{m}r_p}。\tag{2}$$

式中：h_p 为恶意代码行为的第 p 个匹配的特征；r_p 为某恶意代码行为的第 p 个匹配特征对应的权值；match(•)为恶意代码行为数学函数；h_p 为恶意代码行为的匹配结果。如果匹配成功，则说明调用的序列集与特征库中的恶意代码行为的特征相似程度较高；相反，它会降低，从而判定恶意代码行为类型。

1.2　恶意代码行为数据划分

在上述相似匹配度计算的基础上，对恶意代码行为特征数据进行划分。在恶意代码行为特征数据划分中，由于外部变量的影响，需要划分的行为特征较多。为了便于后续恶意代码行为的检测，笔者借助支持向量机划分恶意代码行为数据，在此基础上，实现恶意代码行为数据的异常检测[6-7]。

支持向量机是一种能够在有限的资料中发现最优资料并进行高效划分的恶意代码行为方法。能在设定的区域内发现最具类别的超平面，并将 2 个不同的资料划分，其资料划分原则如图 1 所示。

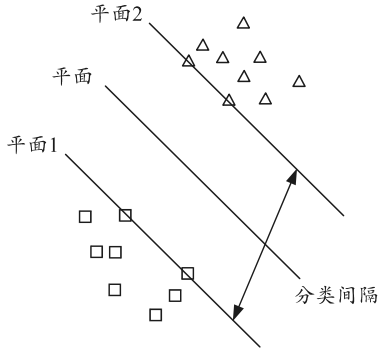

图 1　支持向量机数据划分原理

图 1 中，三角形和正方形表示 2 种不同的恶意代码行为数据，其中，数据的平面划分表示数据采集时的间隔，而各平面间的分类间隔是恶意代码行为数据划分的间隔，从而确保了真实的划分结果在一个可信区间内。

为了实现划分结果的有效性，在将恶意代码行为进行分类时，必须要满足一定的限制条件，即：

$$\left.\begin{aligned}K(i)w_y+h_p\geqslant 1,\beta_e=1\\K(i)w_y+h_p<1,\beta_e=-1\end{aligned}\right\}。\tag{3}$$

式中：w_y 为划分的恶意代码行为数据样本；$K(i)$为恶意代码行为的划分比重；β_e 为恶意代码行为数据划分的边界范围。

将式(3)进行恒定转换：

$$\beta_e(K(i)w_y+h_p)\geqslant 1-\phi。\tag{4}$$

式中 ϕ 为恶意代码行为数据中的松弛变量，且 $\phi\geqslant 0$。

为了实现在设定范围内的恶意代码行为数据，在满足上述限制条件的基础上，求得恶意代码行为数据划分函数的最小值 $R(K(i))$为：

$$R(K(i))=\min(\frac{1}{2}\|K(i)\|^2+E\sum\nolimits_{y=1}^{n}\phi)。\tag{5}$$

式中 E 为恶意代码行为数据复杂度。

在恶意代码行为数据划分中，为了解决凸优化问题引入拉格朗日乘数法，获取恶意代码行为数据划分的最大范围，在满足上述限制条件后，根据最优分类函数获取恶意行为异常数据，即：

$$\chi(v)=\sin g(\sum\nolimits_{y=1}^{n}h_y\beta_e(w,w_y)+z)。\tag{6}$$

式中：h_y 为新的拉格朗日乘数；z 为非零系数。

当划分的恶意代码行为数据性质不可分时，必须采用最优的分类函数对恶意代码行为进行有效划分，即：

$$\mu(x)=\sin g(\sum_{y=1}^{n}h_y\beta_e\xi(w,w_y)+K(i))。\tag{7}$$

式中：ξ为恶意代码行为的核函数；$\mu(x)$为恶意代码行为数据划分的决策函数。

在恶意代码行为数据划分中，借助支持向量机确定数据划分范围，设定划分的限制条件，通过拉格朗日乘子，确定恶意代码行为数据划分的最大范围，通过最优分类函数完成恶意代码行为数据的划分。

1.3 恶意代码行为数据冗余度处理

经上述恶意代码行为数据划分后，由于划分的恶意代码行为数据在检测过程中存在矛盾冲突，需要对数据的冗余进行处理[8-9]。首先应判定划分的恶意代码行为数据是否存在冗余，其判定公式为：

$$\beta_i=\mu(x)\frac{\sum(I-\bar{Q})(Q-\bar{Q})}{(n-1)\partial_I\partial_Q}。\tag{8}$$

式中：n 为恶意代码行为数据的个数；$\bar{Q}$为恶意代码行为数据的平均值；$\partial_I\partial_Q$为恶意代码行为数据不同属性的标准差。

$$\partial_I=\sqrt{\sum(I-\bar{I})^2/(n-1)}；\tag{9}$$

$$\partial_Q=\sqrt{\sum(Q-\bar{Q})^2/(n-1)}。\tag{10}$$

式中 I 和 Q 之间属性呈正相关。

在恶意代码行为数据中，随着 I 值的逐渐增加，Q 值也会越来越高，该数值越大则表示该恶意代码行为数据涵盖的属性类型越多。当该值增加到最大程度时，会剔除此恶意代码行为数据的属性冗余度；而如果恶意代码行为数据属性间相互独立，就不存在冗余度问题，并将其保存下来。通过式(11)对恶意代码行为数据冗余度进行处理，即：

$$\lambda'=(\lambda-\bar{I})/\partial_I。\tag{11}$$

式中 λ 为规格化的恶意代码行为数据。

在恶意代码行为数据冗余度处理中，确定恶意代码行为数据冗余度，获取恶意代码行为数据的不同属性，将恶意代码行为数据中不相互独立的数据进行冗余度处理，完成数据的冗余度处理[10]。

1.4 恶意代码行为异常检测

在上述基础上对恶意代码行为进行异常检测，恶意代码行为异常检测是利用欧几里得距离矩阵，并结合特征数据冗余处理对系统的操作情况判断是否存在恶意代码行为异常的现象。其中，恶意代码行为异常检测模型如图 2 所示。

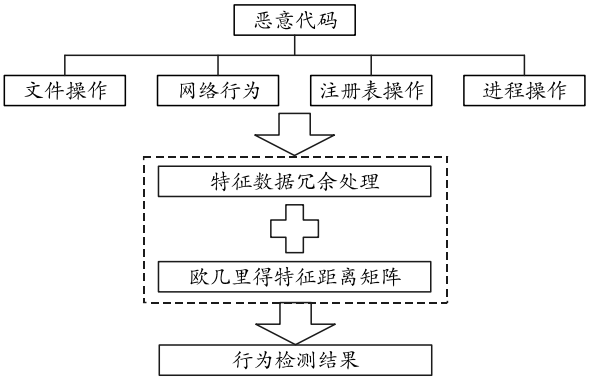

图 2 恶意代码行为异常检测模型

为避免特征权重处理结果对真实结果的影响较大，笔者在恶意代码行为异常检测过程中，引入特征相似度矩阵作为修正参数[11]。

特征相似度矩阵利用特征间欧几里得距离计算得出[12-13]。以一条数据为例，该数据恶意代码行为对应的欧几里得矩阵计算为：

$$\vec{X}_i=X_{i,1},X_{i,2},\cdots,X_{i,n}(n>0)。\tag{12}$$

已知各类别恶意代码行为数据特征均值为：

$$\vec{X}_{\mathrm{avg}_i}=X_{\mathrm{avg}_{i,1}},X_{\mathrm{avg}_{i,2}},\cdots,X_{\mathrm{avg}_{i,n}}(n>0)。\tag{13}$$

恶意代码行为欧几里得距离：

$$d_s(\vec{X}_{i,}\vec{X}_{\mathrm{avg}_i})=\sqrt{\sum_{k}^{n}(X_{i,k}-X_{\mathrm{avg}_{i,k}})^2}(n>0)。\tag{14}$$

最终得到如下所述的恶意代码行为特征距离相似度矩阵：

$$\begin{bmatrix} d_1(\vec{X}_1,\vec{X}_{\mathrm{avg}_1}) & \cdots & d_k(\vec{X}_1,\vec{X}_{\mathrm{avg}_k}) \\ \vdots & & \vdots \\ d_1(\vec{X}_n,\vec{X}_{\mathrm{avg}_1}) & \cdots & d_k(\vec{X}_n,\vec{X}_{\mathrm{avg}_k}) \end{bmatrix}。\tag{15}$$

式中：k 为恶意代码行为数据分类的总类别数；$d_k(\vec{X}_n,\vec{X}_{\mathrm{avg}_k})$为恶意代码行为数据 X_n 与第 k 类恶意代码行为数据特征均值 $\vec{X}_{\mathrm{avg}_k}$ 的欧几里得距离值。

最终检测结果可由欧几里得距离矩阵与恶意代码行为数据冗余度处理后的结果点乘后得出。

假设传送器输出网络恶意代码行为结果是 f_i，检测器输出的网络恶意代码行为是 p_j，而传送器传输的结果不受检测器中网络恶意代码行为所影响，则网络恶意代码行为的阈值是：

$$\Delta\varpi=\lambda'(\frac{1}{f_i}\sum_{a=1}p_j^a+\lambda_a)。\tag{16}$$

式中：$\Delta\varpi$ 为网络恶意代码行为阈值；a 为可操作恶意代码测试样本；p_j^a 为在可操作恶意代码测试样本 a 下的网络恶意代码行为信息；λ_a 为在可操作恶意代码测试样本 a 下的网络恶意代码行为脆弱性检测状态[14-15]。根据上述结果构建网络恶意代码行为检测模型，其计算公式为：

$$w(l)=\lambda'(\sqrt{2}\sum_{b=1}\varsigma_b^t+\lambda_a\Delta\varpi)。\tag{17}$$

式中：$w(l)$为网络恶意代码行为检测模型；b 为网络恶意代码行为的数量；ς_b^t 为网络恶意代码行为的脆弱性状态。

将网络恶意代码行为特征值与脆弱性评估结果进行对比，两者之间的差值与阈值相比，如果差值小于阈值，则表明网络恶意代码行为正常，反之异常。

2　实验分析

2.1　实验方案设计

为了验证本文中基于改进欧几里得距离的恶意代码行为检测方法的整体性能，设计对比分析实验，具体实验方案如下：

步骤 1：实验数据及环境。在进行实验前，首先要设计好实验环境，为实验做充分的准备，同时给出具体的研究对象，也就是实验数据。

步骤 2：实验性能指标。根据实验性能指标，给出其具体的计算公式。

步骤 3：性能分析。通过上述的性能指标来反映不同检测方法的性能，进而验证所提方法的有效性和可行性。

该实验为对比分析实验，实验过程中采用本文中方法、文献[3]的基于行为关系网络的恶意代码检测方法及文献[4]的一种基于系统行为序列特征的 Android 恶意代码检测方法进行对比分析。

2.2　实验环境

本文中实验平台是在 Matlab 仿真环境下完成的，操作系统为 MacOS10.10.5，处理器采用的是 2.7 GHz Intel Core i5，内存是 8 GB 1 867 MHz DDR3，磁盘为 256 GB SSD。本文中数据集由 24 种不同类型的 12 000 条数据组成，每条信息经过特征抽取后，产生了 1 000 多个维度的 01 特征矢量，包括 3 000 个训练样本和 7 000 个测试样本，并选取 4 种典型样本进行分析，在这些数据中，样本的正负比是 4:1。由此可以看出，恶意程序在文件操作、注册表操作等方面存在着显著的恶意特征，而正常程序在进程、网络行为方面基本上不具备恶意特征。

2.3　实验指标

为突出所设计方法可有效检测出恶意代码的行为，实验中共设定 5 个实验指标，具体实验指标及计算方式如下：

指标 1：恶意代码行为准确率指标。该指标是指在实际测试中，检测过程中选择的指标以及迭代次数过程中呈现的求取结果的准确度，该值以百分比的形式呈现，该值越高说明检测的效果越好，该指标为：

$$A=\frac{r_i}{r_l}\times 100\%。\tag{18}$$

式中：r_i 为恶意代码行为检测结果的计算值；r_l 为恶意代码行为检测结果的真实值，该值通过专家调查法得出。

指标 2：恶意代码行为召回率指标。该指标是指在实际测试中得到分类的结果，被预测真实的正例数据占全部真实正例数据的比率，高的召回率意味着检测的效果越好，该指标为：

$$\mathrm{Recall}=\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})。\tag{19}$$

式中：TP 为恶意代码行为真实的正例数据，也就是正例数据被分类为正例数据的情况；FN 为恶意代码行为真实的反例数据，也就是正例数据被分类为反例数据的情况。

指标 3：恶意代码行为综合指标 F_1 值。该指标衡量的是精度与召回率，在通常情况下，两者成反比，相互矛盾，所以在此采用 F_1 值来衡量两者之间的影响，是两者之间调和的平均值，F_1 值越大则说明检测方法效果更好，该指标为：

$$F_1=2\times\frac{\mathrm{Precision}\times\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}。\tag{20}$$

式中 Precision 为恶意代码行为检测的精准度。

指标 4：恶意代码行为检测时间指标。该指标主要是指在保证检测质量的基础上，恶意代码检测所需要的时间。本次实验中根据实验迭代次数变化，在可控范围内测试需要的时间开销。

指标 5：恶意代码行为检测稳定性指标。该指标是指在受到外部环境的作用下，各系统因素所呈现的一种稳定的状况。在此次实验中，在可控的条件下，根据实验迭代次数变化，对所提出的算法进行稳定性实验测试。

2.4 实验结果

2.4.1 恶意代码行为检测准确性对比

为验证本文中基于改进欧几里得距离的恶意代码行为检测方法的有效性，将文献[3]方法、文献[4]方法作为对比算法，与本文中方法进行准确性对比实验，结果如表 1 所示。

表 1 不同方法准确率对比结果 %

迭代次数	本文中方法	文献[3]方法	文献[4]方法
10	98.97	93.00	90.32
20	96.67	94.81	92.01
30	96.85	93.32	88.23
40	98.56	94.56	92.35
50	98.32	94.69	85.69
60	98.56	94.89	91.23
70	97.69	94.78	92.24
80	96.89	94.67	89.89
90	97.63	93.56	89.96
100	98.54	93.85	90.89

根据表 1 中的数据可知：本文中方法检测恶意代码行为的准确率更高，最高值为 98.97%，最低值为 96.67%；而文献[3]方法检测恶意代码行为的准确率最高为 94.89%，文献[4]方法检测恶意代码行为的准确率最高为 92.35%，二者均低于本文中方法。由此可知，本文中方法具有较高的准确性，能有效地检测恶意代码行为，具有一定的实用性。

2.4.2 恶意代码行为检测召回率对比

为验证基于改进欧几里得距离的恶意代码行为检测方法的有效性，对比了 3 种不同方法的召回率，并给出了对比结果，如图 3 所示。

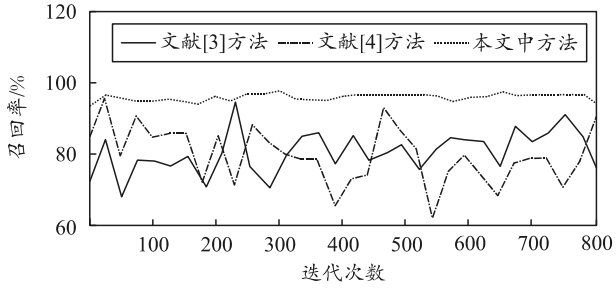

图 3 不同方法召回率对比结果

分析图 3 可知：在多次实验迭代中，只有所提的基于改进欧几里得距离的恶意代码行为检测方法的召回率在 90%以上，另外 2 种对比方法召回率的变化波动较大，且均低于所提方法。由此可见，所提的基于改进欧几里得距离的恶意代码行为检测方法具有较高的召回率，精度高，效果较好。

2.4.3 恶意代码行为检测 F_1 值对比

以 F_1 为指标对不同检测方法展开性能验证及对比分析，结果如表 2 所示。

表 2 不同方法 F_1 结果对比

迭代次数	文献[3]方法	文献[4]方法	本文中方法
20	0.968 4	0.974 5	0.985 8
40	0.968 2	0.973 1	0.985 7
60	0.968 9	0.973 5	0.988 6
80	0.968 5	0.973 2	0.986 5
100	0.968 3	0.969 8	0.988 9

从表 2 的结果可以看出：文献[3]方法的综合评价指标 F_1 最高为 0.968 9，文献[4]方法的综合评价指标 F_1 最高为 0.974 5，与其他 2 种方法相比，本文中所提方法的综合评价指标 F_1 均在 0.985 7 以上。为此，该方法能够对恶意代码行为进行有效检测。

2.4.4 恶意代码行为检测时间对比

为证明基于改进欧几里得距离的恶意代码行为检测方法的实效性，通过实验测试了恶意代码行为的检测时间，并给出了 3 种不同方法的对比结果，如图 4 所示。

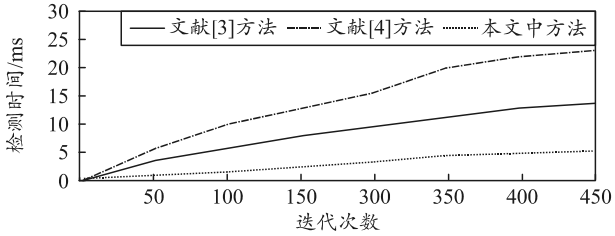

图 4 不同方法检测时间对比

根据图 4 可知：在恶意代码行为检测时间的对比实验中，由于迭代数不断增大，所需的检测时间也会发生改变。当迭代次数为 100 时，文献[3]、文献[4]方法的检测时间都在 5 ms 以上，而本文中方法的检测时间在 5 ms 以下；当迭代次数为 400 时，文献[3]方法的检测时间在 15 ms 以下，文献[4]方法的检测时间在 20 ms 以上，而本文中方法所需检测时间仍处于 5 s 以下，明显少于其他 2 种对比方法。由此可知，本文中方法更具应用优势，说明该方法的时效性更强。

2.4.5 恶意代码检测稳定性对比

在此基础上，对不同检测方法展开稳定性验证及对比分析，结果如图 5 所示。

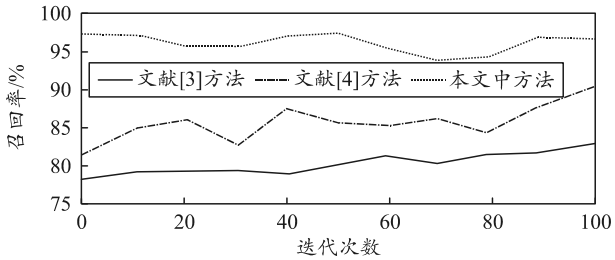

图 5 不同方法检测稳定性对比

分析图 5 可知：当实验重复的数量越来越多时，在恶意代码行为检测中，不同方法表现出了不同的稳定性。文献[3]方法在恶意代码行为检测过程中的稳定性在 75%～85%之间，相较于文献[3]方法，文献[4]方法的稳定性较高，但最大也未超过 90%，而本文中基于改进欧几里得距离的恶意代码行为检测方法在检测过程中的稳定性均在 90%以上，是 3 种方法中最高的。

实验结果表明：本文中检测方法所需检测时间最短，准确率高达 98.97%，召回率达到 90%以上，其综合评价指标 F_1 值均在 98%以上，具有较高的准确性及稳定性，能够有效检测出网络中的恶意代码行为，具有一定的可行性。

3　结束语

近年来，由于网络安全问题日益普遍，恶意代码行为检测方法也越来越受到国家的重点关注。当前传统的恶意代码行为检测指标体系设计方法已无法满足现阶段网络安全的发展需要。针对现有传统的恶意代码行为检测方法存在的不足，设计一种欧几里得判别矩阵与特征数据冗余处理相结合的可满足网络安全恶意代码行为检测的方法。结果表明，笔者提出的基于改进欧几里得距离的恶意代码行为检测方法能快速、稳定、准确地实现恶意代码行为的检测分析，具有广泛的应用前景。

笔者提出的基于改进欧几里得距离的恶意代码行为检测方法，适用于能够转化为多维矢量表示的恶意代码行为数据，要求数据包含反映行为特征的多个维度，并需经过适当预处理以提高检测效率。该方法在 Matlab 仿真环境下表现出色，但实际应用中可能需一定计算资源支持大规模数据处理。该方法也存在局限性，如动态特性考虑不足可能影响快速变化网络环境下的检测效果，性能依赖于特征选择且泛化能力需进一步验证。未来研究将针对这些局限性，特别是处理网络动态特性和提高泛化能力方面进行改进，以提供更全面有效的恶意代码行为检测解决方案，推动网络安全技术发展。

参考文献：

[1] 王博, 蔡弘昊, 苏旸. 基于 VGGNet 的恶意代码变种分类[J]. 计算机应用, 2020, 40(1): 162−167.

[2] 喻民, 姜建国, 李罡, 等. 恶意文档检测研究综述[J]. 信息安全学报, 2021, 6(3): 54−76.

[3] 刘建松, 张磊, 方勇. 基于行为关系网络的恶意代码检测方法[J]. 四川大学学报(自然科学版), 2022, 59(2): 77−83.

[4] 杨吉云, 陈钢, 鄢然, 等. 一种基于系统行为序列特征的 Android 恶意代码检测方法[J]. 重庆大学学报, 2020, 43(9): 54−63.

[5] 杨轶, 苏璞睿, 应凌云, 等. 基于行为依赖特征的恶意代码相似性比较方法[J]. 软件学报, 2011, 22(10): 2438−2453.

[6] 朱翔宇, 张健, 高铖, 等. 基于虚拟化环境恶意代码检测系统的设计与实现[J]. 天津理工大学学报, 2020, 36(1): 12−17, 24.

[7] 马丹, 万良, 程琪芩, 等. Attention-CNN 在恶意代码检测中的应用研究[J]. 计算机科学与探索, 2021, 15(4): 670−681.

[8] 杨萍, 舒辉, 康绯, 等. 一种基于语义分析的恶意代码攻击图生成方法[J]. 计算机科学, 2021, 48(S1): 448−458, 463.

[9] 段玉莹, 王凤英. 基于级联与深度信念网络的恶意代码分层检测[J]. 计算机工程与设计, 2020, 41(7): 1815−1820.

[10] 赵晓君, 王小英, 张咏梅, 等. 基于恶意代码行为分析的入侵检测技术研究[J]. 计算机仿真, 2015, 32(4): 277−280.

[11] 李辉, 王欢, 王冬秀. 基于改进深度森林的抗恶意代码攻击算法仿真[J]. 计算机仿真, 2022, 39(1): 353−357.

[12] 魏高山, 俞叔刚, 咸洁敏, 等. 基于深度学习的恶意软件检测技术研究[J]. 信息技术, 2022, 46(9): 100−105.

[13] 袁天梦. 基于改进欧氏距离协调发展评估模型的电网投资决策算法[J]. 机械设计与制造工程, 2021, 50(5): 47−51.

[14] 田锋, 周安民, 刘亮, 等. ARS: 基于文件行为的勒索软件主动防御技术研究[J]. 四川大学学报(自然科学版), 2021, 58(2): 97−105.

[15] 傅依娴, 芦天亮, 马泽良. 基于 One-Hot 地 CNN 恶意代码检测方法[J]. 计算机应用与软件, 2020, 37(1): 304−308, 333.