计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190100047

边缘计算构架下基于孤立森林算法的 DoS 异常检测

陈 佳 欧阳金源 冯安琪 吴 远 钱丽萍
浙江工业大学信息工程学院 杭州 310023
(jiachen_zjut@163.com)

摘 要 随着网络技术的快速发展,网络攻击带来了极大的负面影响,因此网络安全问题亟待解决。针对网络攻击中的拒绝服务(Denial of Service,DoS)攻击,提出了一种基于边缘计算框架的孤立森林网络异常检测方法。该方法根据每个边缘节点的特性实现对模型训练任务的合理分配,有效地提高了边缘节点的利用效率;同时,利用边缘计算的特点实现了对云中心模型训练任务的分流,从而更好地减少系统的耗时,减轻云中心的任务负担。为了验证所提方法的有效性,对 10%-KDDCUP99 网络数据集进行预处理,并提取部分数据用于实验。实验结果表明,与支持向量机(Support Vector Machine,SVM)和多层感知器(Multi-Layer Perceptron,MLP)方法相比,所提方法将系统建立时间分别缩短了 90%和 60%,且得出的曲线下面积(Area Under Curve,AUC)可达 0.9 以上,这证明该方法能够在确保较高异常检测性能条的件下有效减少异常检测系统的建立时间。

关键词:异常检测;边缘计算;孤立森林;DoS 攻击;数据预处理
中图法分类号 TP309.2

DoS Anomaly Detection Based on Isolation Forest Algorithm Under Edge Computing Framework

CHEN Jia,OUYANG Jin-yuan,FENG An-qi,WU Yuan and QIAN Li-ping
College of Information Engineering,Zhejiang University of Technology,Hangzhou 310023,China

Abstract With the rapid development of network technology,network attacks have brought huge negative impacts,so network security issues need to be resolved urgently. Aiming at denial of service (DoS) attacks in networks,an anomaly detection method for isolated forest based on edge computing framework was proposed. According to the characteristics of each edge node,the method realizes the reasonable distribution of the model training tasks and effectively improves the utilization efficiency of edge nodes. Meanwhile,the characteristics of edge computing are utilized to realize the offloading of model training tasks from cloud center,so as to better reduce the time consumption of the system and reduce the burdenof the cloud center. In order to verify the effectiveness of the proposed method,the 10%-KDDCUP99 network dataset is preprocessed and partial data used for experiments. Experimental results show that compared with the Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP) methods,time consumption of proposed method is reduced by 90% and 60% respectively,and area under curve (AUC) can reach more than 0.9,which indicates that the method can effectively reduce the system time consumption and ensure a high detection performance.

Keywords Anomaly detection,Edge computing,Isolation forest,DoS attack,Data preprocessing

1 引言

随着网络技术的不断进步,网络安全问题成为了制约互联网发展的重要因素,尤其是各种形式的网络攻击(如网络病毒、口令入侵和拒绝服务攻击等)对网络的稳定性造成了很大的负面影响。目前,网络异常检测方法主要有两种:误用检测通过对目标行为与事先定义的行为库进行匹配来判断网络异常;异常检测将偏离正常行为的网络数据认定为异常行为,能够发现未知的攻击。其中,异常检测是更有效的网络异常检测方法,被广泛应用于网络入侵检测系统[1-3]。

目前,研究人员在异常检测领域进行了大量研究,提出了多种方法[4-13],如基于统计的方法[4]和基于机器学习的方法[5]等。文献[6]在多个不同数据集上评估了 14 种不同的异常检测算法。文献[7]针对大数据环境下的 DDoS 攻击检测问题,设计了一种融合聚类和智能蜂群算法的 DDoS 攻击检测系统。文献[8]提出了一种基于非限制 α 稳定一阶模型和统计假设检验的网络流量异常检测方法。文献[9]开发了一种基于知识的警报验证方法,并设计了一种基于多类

到稿日期:2019-01-07 返修日期:2019-06-03 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61572440);浙江省自然科学基金(LR16F010003,LR17F010002)
This work was supported by the National Natural Science Foundation of China (61572440) and Natural Science Foundation of Zhejiang Province,China (LR16F010003,LR17F010002).
通信作者:钱丽萍(lpqian@zjut.edu.cn)

k-最近邻分类器的智能警报过滤器。文献[10]提出了新的系统框架,在滥用、异常和基于混合网络的入侵检测系统(Intrusion Detection Systems,IDS)中应用随机森林的数据挖掘算法进行入侵检测。文献[11]提出了一种改进的数据异常检测方法。该方法基于模拟退火算法选择精度高且有差异的隔离树来优化森林,同时去除冗余的隔离树,改进了隔离森林的构建过程。然而,对于数据急剧上升所带来的问题,中心式系统已经难以应付,因此研究者开始利用边缘计算来解决海量数据下的异常检测问题。文献[12]针对物联网中实时采集的传感数据总体质量低下的问题,提出了基于边缘计算的传感数据异常实时检测算法。文献[13]利用网络功能虚拟化和软件定义的网络架构,为移动边缘计算提出了针对分布式拒绝服务攻击的协同防御框架。而在众多异常检测算法中,Liu等[14]首次提出的孤立森林算法通过样本在多个二叉树算得的路径长度来判断异常。该算法适用于维度不高的数据,算法效果好,时间效率高且复杂度低[14],因此受到了许多学者的关注[15-17]。文献[15]采用孤立森林算法来预测软件代码更改中的错误。文献[16]探讨了利用孤立森林算法检测大型云中心异常的可行性,并提出了一种将时间序列信息编码为额外属性的方法。文献[17]通过孤立森林算法对从无线传感网络收集的数据进行了有效检测。

现有的许多网络检测方法依然存在很多不足,这主要是由于网络数据存在以下几个问题:1)网络数据量庞大,且每条网络数据都包含大量属性,因此为每一条数据打上标签是极为困难的;2)随着网络环境的变化,新的网络异常不断涌现,系统无法识别所有新出现的异常;3)随着网络应用的不断发展,数据量将会呈指数式增长,中心式系统显得力不从心;4)现实网络环境中,很难获取足够的异常数据用于训练,采集的数据往往是不均衡的,导致系统性能降低[18]。

针对目前孤立森林中心式系统的不足和系统建立时间长等缺点,本文提出了基于边缘计算的孤立森林方法。该方法的优势在于:利用边缘计算思想将孤立森林算法的训练任务分散到移动边缘计算(Mobile Edge Computing,MEC)节点,并行化构建系统。该方法的特点是:根据 MEC 节点的特性实现对模型训练任务的合理分配,有效地提高了对 MEC 节点的利用效率;同时,利用边缘计算实现了对云中心模型训练任务的分流,从而更好地减少了系统的耗时。所提方法在保证较高检测性能的同时,能有效应对网络数据的快速增长,减轻云中心的压力,显著减少检测系统建立的耗时。

2 系统框架与建模

2.1 系统框架网络

面对网络数据量的快速增长,传统的中心式系统已经无法满足快速处理数据的需求,而边缘计算通过在数据源附近建立节点来减少数据传输时延,分流计算中心的任务,有效解决了这一问题。根据文献[19],图 1 给出了边缘计算的相应框架。由图 1 可知,边缘计算框架分为 3 层:云中心层,边缘层和设备层。本文模型重点关注云中心层和边缘层,如图 2 所示。

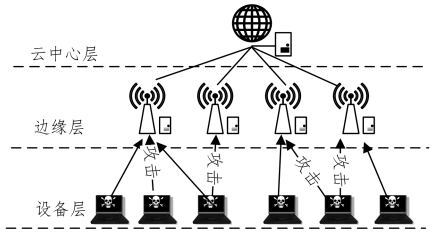

图 1 网络异常检测的系统框架

Fig. 1 System framework of network anomaly detection

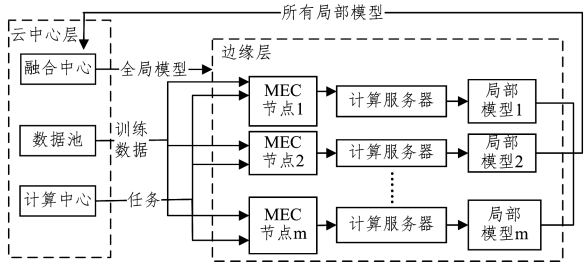

图 2 网络异常检测的系统模型

Fig. 2 System model of network anomaly detection

云中心层包括三大部分:融合中心、数据池和计算中心。融合中心用于融合所有 MEC 节点生成的局部模型;数据池会根据任务给 MEC 节点下发相应的训练数据;计算中心获取节点信息并进行任务调度。边缘层中的每个 MEC 节点都包含一个计算服务器,它拥有一定的计算能力和存储能力,用于训练数据的存储和局部模型的训练。具体地,云中心层进行任务分配、数据下发和模型融合;边缘层进行局部模型训练和数据检测。

2.2 系统建模

基于边缘计算的孤立森林算法需要在 MEC 节点训练孤立树(iTree),但不同节点的计算能力、传输带宽和任务处理能力等不同,需要由云中心对任务进行分配。本节将详细说明基于边缘计算的孤立森林算法,其中使用的关键变量及其定义如表 1 所列。

表 1 关键变量及其定义

Table 1 Key variables and their definitions

变量名称	变量定义
$\mathcal{M}$	MEC 节点集合
M	MEC 节点个数
R	MEC 节点的 CPU 处理速度集合
r_m	节点 m 的 CPU 处理速度
B	MEC 节点的传输带宽集合
b_m	节点 m 的传输带宽
x_m	在节点 m 上要训练的 iTree 数量
y_m	在节点 m 上每棵 iTree 的样本量
d_m	节点 m 所需的数据量
D_{total}	云中心可以提供的数据量
l_m	节点 m 上任务的计算量
$L_m^{\max}$	节点 m 的最大计算能力
$Y_{\min}$	系统中每棵 iTree 样本量的最小值
$X_{\min}$	系统中 iTree 数量的最小值
t_m	节点 m 完成任务的时间
t_{total}	系统完成时间
C	任务分配的组合解集合

模型中,MEC 节点集合 $\mathcal{M}=\{1,2,\cdots,m,\cdots,M\}$,各节点的处理速度集合 $R=\{r_1,r_2,\cdots,r_m,\cdots,r_M\}$,各节点的带宽集合 $B=\{b_1,b_2,\cdots,b_m,\cdots,b_M\}$。

孤立森林算法的检测性能是由 iTree 的数量和每棵 iTree 的样本量直接决定的。性能约束条件如下：

$$\sum_{m=1}^{M} x_m \geqslant X_{\min}, m=1,2,\cdots,M \tag{1}$$

其中，$\{x_m\}$是在节点 m 上要训练的 iTree 数量，$X_{\min}$是系统中 iTree 数量的最小值。式(1)表明：如果算法要达到一定的性能，所有 MEC 节点训练的 iTree 总量须大于 $X_{\min}$。

$$y_m \geqslant Y_{\min}, m=1,2,\cdots,M \tag{2}$$

其中，$\{y_m\}$是在节点 m 中每棵 iTree 的样本量，$Y_{\min}$是系统中每棵 iTree 样本量的最小值。式(2)表明：如果要达到一定的性能，每棵 iTree 的样本量须大于 $Y_{\min}$。

孤立森林算法的训练任务中包含$\{x_m\}$和$\{y_m\}$两种参数，云中心会计算任务所需要的数据量以及任务计算量，并根据节点情况分配任务。

$$d_m = x_m \times y_m, m=1,2,\cdots,M \tag{3}$$

其中，d_m 是 MEC 节点 m 所需的数据量，即云中心需要下发给节点 m 的数据量。

$$l_m = \left(\frac{y_m(y_m+1)}{2}-1\right)\cdot x_m, m=1,2,\cdots,M \tag{4}$$

当 MEC 节点上每棵 iTree 的样本量为$\{y_m\}$时，一棵 iTree 的最大计算量为$\frac{y_m(y_m+1)}{2}-1$。式(4)中，l_m 为节点 m 上$\{x_m\}$棵 iTree 的任务计算量。

$$\sum_{m=1}^{M} d_m \leqslant D_{\text{total}}, m=1,2,\cdots,M \tag{5}$$

其中，D_{total}是云中心可以提供的数据量。式(5)表明：发送给所有节点的数据总量不能超过云中心数据总量。

$$l_m \leqslant L_m^{\max}, m=1,2,\cdots,M \tag{6}$$

其中，$L_m^{\max}$ 是节点 m 的最大计算能力。式(6)表明：每个节点处理的计算量不能超过其上限。

当所有 MEC 节点的局部模型训练完成时，局部模型将被上传至云中心融合成全局模型；云中心将全局模型发送至所有节点后，系统建立完毕。由于模型上传与下发的耗时很短，这里暂不考虑其耗时。将所有 MEC 节点完成任务的耗时视为系统建立时间。

$$t_m = \frac{l_m}{r_m}+\frac{d_m}{b_m}, m=1,2,\cdots,M \tag{7}$$

其中，t_m 为 MEC 节点 m 完成任务的耗时，主要由两部分组成，即训练数据从云中心传输给节点 m 的耗时$\frac{l_m}{r_m}$以及节点 m 进行局部模型训练的耗时$\frac{d_m}{b_m}$。r_m 是节点 m 的 CPU 处理速度，b_m 是节点 m 的传输带宽。

$$t_{\text{total}} = \max(t_m), m=1,2,\cdots,M \tag{8}$$

其中，t_{total}为检测系统建立的耗时，应取 MEC 节点中耗时最长的时间。

结合约束条件式(1)、式(2)、式(5)和式(6)，建立以最小化系统耗时为目标的函数：

$$\min\ t_{\text{total}} = \max(t_m) \tag{9}$$

$$\text{s.t.}\begin{cases}\sum_{m=1}^{M} d_m \leqslant D_{\text{total}}\\ l_m \leqslant L_m^{\max}\\ \sum_{m=1}^{M} x_m \geqslant X_{\min}\\ y_m \geqslant Y_{\min}\end{cases}$$

变量：$\{x_m\}$，$\{y_m\}$

其中，$m=1,2,\cdots,M$。

第一个约束条件说明数据下行量不能超过云中心数据总量；第二个约束条件说明 MEC 节点处理的计算量不能超过其上限；第三个约束条件说明要使系统达到一定的性能，所有 MEC 节点训练的 iTree 总量须大于 $X_{\min}$；第四个约束条件说明要使系统达到一定的性能，每棵 iTree 的样本量$\{y_m\}$必须大于 $Y_{\min}$。

3　基于边缘计算的孤立森林算法

3.1　孤立森林算法的原理

本文将结合边缘计算对孤立森林算法进行改进。为了方便理解，本节对孤立森林算法的定义和工作原理进行简单说明，相关理论推导和细节描述见文献[14]。

孤立森林算法对异常的定义是“少量和不同”：异常数据在整个数据中所占的比例很小；异常数据与正常数据有很大区别[14]。因此，异常数据更容易被区分。孤立森林算法是一种集成学习算法，根据集成学习思想，需要多个基分类器来提升算法性能，所以孤立森林要达到一定的性能，需要依靠一定数目的基分类器，即一定数量的 iTree。

孤立森林(iForest)由多棵 iTree 融合而成。iTree 与二叉搜索树有相同的结构[20]，每个节点都是叶子节点或者包含两个节点的内部节点。iTree 通过随机选取数据属性 q 和划分值 p 来分割数据集，递归构造节点直到满足下列条件之一：1)达到限定树高；2)只有一个样本；3)所有样本相同。

iTree 的构造步骤如下：

1)将相应的数据集放入 iTree 的根节点。

2)随机取一个属性 q，并在该属性中随机取一个划分值 p(介于属性 q 的最大值和最小值之间)。

3)将属性 q 中值小于 p 的样本放在当前节点的左子节点，其余的样本放在右子节点。

4)对子节点递归步骤 2)和 3)，构造新的子节点，直到满足终止条件。

3.2　基于边缘计算的孤立森林算法

基于边缘计算的孤立森林算法需要将 iTree 的训练放置在具有不同能力的 MEC 节点上并行化完成，以达到快速建立系统的目的，因此合理分配训练任务至关重要。本系统中的任务分配具有巨大的组合解空间，传统算法难以有效求解，而模拟退火(Simulate Anneal，SA)[21]算法可以高效地求解此类问题，是一种理想的算法，所以我们采用 SA 算法进行任务分配。但 SA 算法无法直接用于本文问题，须先对其稍做改变，具体的算法工作流程如图 3 所示。

组合解空间定义：云中心将训练任务分配给各个 MEC 节点，组成解空间 $C=\{\{x_1,y_1\},\cdots,\{x_m,y_m\},\cdots,\{x_M,y_M\}\}$，即 MEC 节点 1 到 M 的任务分别为$\{x_1,y_1\}$到$\{x_M,y_M\}$。

新组合解空间定义：随机选取两个任务序列$\{x_i,y_i\}$和$\{x_j,y_j\}$，交换两个序列对应的位置，则原解空间 $C=\{\{x_1,y_1\},\cdots,\{x_i,y_i\},\cdots,\{x_j,y_j\},\cdots,\{x_M,y_M\}\}$变为新解空间 $C=\{\{x_1,y_1\},\cdots,\{x_j,y_j\},\cdots,\{x_i,y_i\},\cdots,\{x_M,y_M\}\}$。

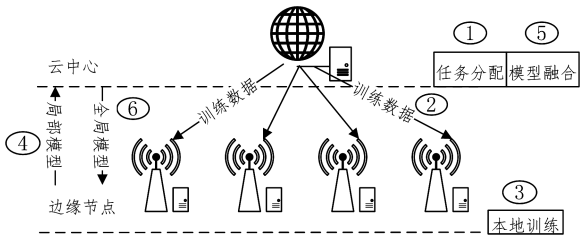

图 3 改进 SA 算法的工作流程

Fig. 3 Workflow of improve SA algorithm

根据图 3,下面给出相应的具体实现步骤。

1)云中心确定任务集合,然后通过模拟退火算法将任务分配给 MEC 节点。

2)云中心将任务所需的训练数据下发给各 MEC 节点。

3)所有 MEC 节点接收到数据后同时进行训练,建立局部孤立森林模型。

4)所有 MEC 节点将局部模型上传到云中心。

5)云中心将所有局部模型进行融合,形成全局孤立森林模型。

6)云中心将全局模型下发到各个 MEC 节点。

基于边缘计算的孤立森林算法的代码实现如算法 1 所示。

算法 1 基于边缘计算的孤立森林算法

1. 初始化参数:设置初始值 $T_i=1\,000$,下降因子 $\alpha=0.99$,稳定值 $T_s=1$,孤立森林集合 IF=Ø;
2. 任务随机分配给 MEC 节点,同时生成满足 iTree 数量约束(1)、每棵 iTree 样本量约束(2)的任务集;
3. 产生的初始组合解假定为 $C'=\{\{x_1',y_1'\},\cdots,\{x_m',y_m'\},\cdots,\{x_M',y_M'\}\}$($C'$为临时解)
4. 根据式(7)和式(8)计算耗时 $t_{total}(C')$
5. While $T_i>T_s$ do
6. 根据新组合解空间的定义产生新的解空间 $C''=\{\{x_1'',y_1\}'',\cdots,\{x_m'',y_m''\},\cdots,\{x_M'',y_M''\}\}$($C''$为临时解)
7. If C''满足约束(4)和(6) then
8. 根据式(7)和式(8)计算 C''的耗时 $t_{total}(C'')$
9. End if
10. 计算 $\Delta t_{total}=t_{total}(C')-t_{total}(C'')$
11. If $\Delta t_{total}>0$ then
12. 更新临时解和最优解 $C_{opt}=C'',C'=C''$
13. Else
14. 更新最优解 $C_{opt}=C'$
15. 随机产生一个数 $r(0<r<1)$,r 服从均匀分布
16. 计算概率 $p=\exp\{-\Delta t_{total}/T_i\}$
17. If $p>r$ then
18. 更新临时解和最优解 $C_{opt}=C'',C'=C''$
19. End if
20. End if
21. $T_i=\alpha * T_i$
22. End while
23. 输出分配方案 $C_{opt}=\{\{x_1,y_1\},\cdots,\{x_m,y_m\},\cdots,\{x_M,y_M\}\}$
24. 根据最优解 C_{opt}将任务和数据分配给所有 MEC 节点
25. For M 个 MEC 节点 do
26. 设置树高限制 $l=ceiling(\log_2(y_m))$(ceiling 向上取整)
27. For MEC 节点的$\{x_m\}$棵树 do
28. 当前树高 e=0
29. If $e>l$ 或只有一个数据 then
30. return 叶子节点
31. Else
32. 随机选择一个属性 q,并在此属性中随机选择一个分割值 p(介于属性 q 的最大值和最小值之间)
33. 划分训练数据,将属性 q 中值小于 p 的数据放在该节点的左叶子节点,不大于 p 的数据放在右叶子节点
34. e=e+1
35. 在新生成的叶子节点上递归重复步骤 27-步骤 32,并保存 iTree
36. End if
37. 将生成的 iTree 加入 IF 集合
38. End for
39. End for

通过前文对基于边缘计算的孤立森林算法的详细说明,可以发现其主要分为训练任务分配和模型训练两个部分。假设边缘节点和任务数量为 n,最大迭代次数为 k,则训练任务分配部分的时间复杂度为 $O(kn)$;假设耗时最大的 MEC 节点任务规模为(x,y),即需要建立 x 棵 iTree,每棵 iTree 样本量为 y,节点样本数量为 d,则模型训练部分的时间复杂度为 $O(dx\log(y))$。因此,算法的整体复杂度为 $O(kn+dx\log(y))$。

4 仿真与结果分析

4.1 实验环境和 KDD 数据集

硬件环境:64 位 Core i5,2.5 GHz CPU,8 GB RAM,500 GB ROM。采用 KDDCUP99 数据集[22]来验证本文方法的有效性。KDDCUP99 是入侵检测和异常检测常用的经典数据集之一,研究中一般使用它的 10% 抽样 10%-KDDCUP99 数据集,其中包含 4 种攻击类型(DoS,Probe,R2L 和 U2R),每条数据包含了 41 种特征属性(7 个离散型,34 个连续型)和 1 种类标识。10%-KDDCUP99 数据集信息如表 2 所列。

表 2 10%-KDDCUP99 数据集信息

Table 2 Dataset information of 10%-KDDCUP99

攻击类型	训练数据集		测试数据集	
	数量	百分比/%	数量	百分比%
Normal	97278	19.69	60593	19.49
Probe	4107	0.83	4166	1.34
DoS	391458	79.24	229853	73.9
U2R	52	0.01	228	0.07
R2L	1126	0.23	16189	5.2

4.2 数据预处理

KDDCUP99 数据集由于数据量庞大,数据属性众多,会产生大量冗余,因此直接使用 KDDCUP99 数据集会影响检测算法的性能。为了避免这些问题,采用文献[23]中的方法对原始数据集进行相应的预处理。

原始数据集中包含 4 898 431 条数据,其中攻击数据有 3 925 651 条(80.1%),这样庞大的数据量会耗费极大的资源,

同时对于真实环境和异常检测系统来说异常比例过高，需要压缩数据量来减少异常数据占比。我们从 10%-KDDCUP99 中提取 DoS 攻击中的 neptune，back，pod 和 teardrop 攻击，制造了 4 个子数据集，例如“Normal-neptune”数据集中包含正常数据和 neptune 攻击数据。对每种子数据集进行特征降维，选择与攻击类型相关性最高的属性进行研究[23]。各个数据集的信息如表 3 所列。

表 3　实验数据集信息

Table 3　Information of experiment dataset

数据集名称	异常占比/%	属性	训练集		测试集	
			正常	异常	正常	异常
Normal-neptune	2.4	6	80000	2000	17278	432
Normal-back	2.2	6	78417	1764	19560	440
Normal-pod	0.27	6	78279	179	19577	53
Normal-teardrop	1	6	80784	816	20196	204

4.3　仿真结果及分析

为了评估异常检测系统在边缘计算构架下的有效性及检测效果，文中设计了两大类实验：1）对任务分配算法进行有效性验证；2）对异常检测算法进行性能验证。其中，对于任务分配算法，选择了随机分配（在满足约束条件下，任务随机分配给 MEC 节点处理）和降序分配（在满足约束条件下，最大的任务分配给处理速度最快的 MEC 节点，直到所有任务分配完毕）进行比较；对于异常检测算法，选择了支持向量机（SVM，通过计算不同样本的分界面来进行分类）和多层感知机（MLP，通过多层神经网络的训练对样本进行分类）进行比较。为了保证实验的准确性，在多组数据集上对每种算法进行 10 次试验并求取平均值。

现实中样本存在分布不均衡的问题，使得传统的度量标准如准确率不能恰当地反映分类器的性能[24]，因此使用曲线下面积（AUC）来评估分类器的性能。*AUC* 值是当前分类器性能根据得分将正样本排在负样本前面的概率，较大的 *AUC* 值表明分类器性能更好。

在实验 1（节点和任务数量）中，将任务和 MEC 节点数量设置为 1～50 个，进行一对一匹配。SA 的初始温度为 1000，稳定温度为 1，冷却因子为 0.99。MEC 节点的带宽 b 服从[10，20]随机分布，处理速度 r 服从[1000，3000]随机分布，处理上限服从[3500000，7000000]随机分布。不同 MEC 节点和任务的数量下 3 种算法的耗时对比如图 4 所示。随着节点和任务增多，3 种算法的耗时都在增加，但本文算法耗时更少。因此，采用 SA 算法建立异常检测系统消耗的时间更少。

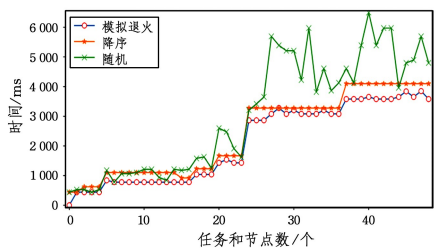

图 4　不同分配算法的对比

Fig. 4　Comparison of different distribution algorithms

在实验 2（带宽）中，为了了解 MEC 节点在不同带宽下对算法的影响，令带宽 b 服从[100，200]随机分布，其他参数设置与实验 1 相同。从图 5 中可以看出，当带宽较大时，降序分配算法与本文算法开始接近。这是由于当带宽增加时，数据传输所花费的时间在总耗时中的占比越来越小，MEC 节点的 CPU 处理速度占据主导地位，导致本文算法将处理时间作为主要优化目标，这与降序分配算法的优化目标接近。

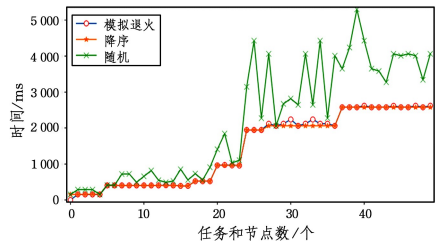

图 5　大带宽下算法效果的对比

Fig. 5　Comparison of algorithm effects under large bandwidth

在实验 3（迭代次数）中，为了了解迭代次数对算法的影响，在不同迭代次数下进行了实验。实验设置 20 个任务和 30 个 MEC 节点，进行一对一匹配；其他参数的设置与实验 1 相同。从图 6 中可以看出，随着迭代次数的增加，耗时逐渐减少，当迭代次数接近 500 时，结果趋于稳定。

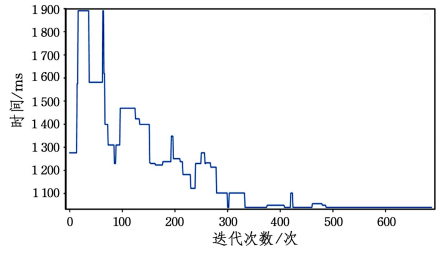

图 6　不同迭代次数下的算法效果

Fig. 6　Algorithm effect under different iterations

在实验 4（CPU 处理速度）中，将 MEC 节点的 CPU 处理速度 r 设置为 [10000，30000]随机分布，其他参数设置与实验 1 相同。从图 7 中可以看出，当 MEC 节点的处理速度普遍较快时，本文算法要明显优于其他算法。这是由于当处理速度加快时，数据计算所花费的时间在总耗时中的占比减小，本文算法则将传输时间作为主要的优化目标。

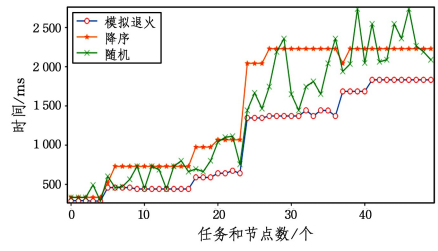

图 7　高 CPU 速度下算法效果的对比

Fig. 7　Comparison of algorithm effects under high CPU speed

在实验 5（iTree 数量）中，选取 Normal-back 数据集作为示例进行展示。其中每棵 iTree 的样本数{y_m}设置为 180，分别测试 iTree 数量{x_m}从 1 到 1000 棵的 *AUC* 值。从图 8 和图 9 中可以发现，在 Normal-back 数据集中，当 iTree 数量达到 800 棵时，结果趋于稳定，同时随着 iTree 的增加耗时呈线性增长。

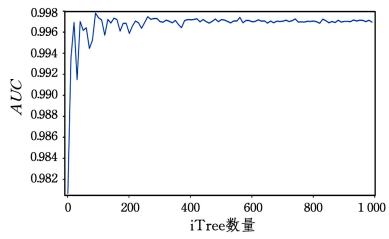

图 8 不同 iTree 数量下的 *AUC*

Fig. 8 *AUC* under different numbers of iTrees

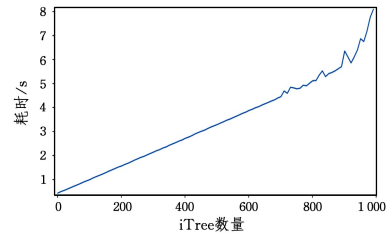

图 9 不同 iTree 数量下算法的耗时

Fig. 9 Time consumption under different numbers of iTrees

实验 6(每棵 iTree 样本数量)以 Normal-back 数据集作为实例进行展示。其中,iTree 的数量$\{x_m\}$设置为 800,分别测试每棵 iTree 样本数量$\{y_m\}$从 1 到 1024 下的 *AUC* 值。从图 10 和图 11 中可以发现,在 Normal-back 数据集中,每棵 iTree 的样本数量达到 180 左右时效果最好,同时随着 iTree 的增加耗时逐渐增长。

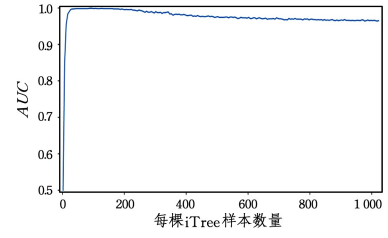

图 10 不同的单棵 iTree 样本数下算法的 *AUC*

Fig. 10 *AUC* under different iTree samples

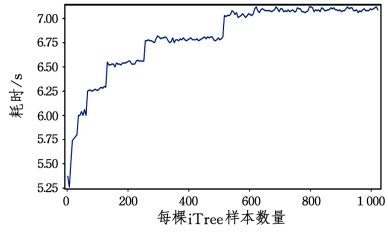

图 11 不同的单棵 iTree 样本数下算法的耗时

Fig. 11 Time consumption under different iTree samples

多组实验仿真表明,不同的数据集中 Y_{min} 和 X_{min} 的取值不同,各个数据集的最优参数取值如表 4 所列。

表 4 各个数据集中参数的最优取值

Table 4 Optimal parameter of each dataset

数据集	*AUC*	时间/s	X_{min}	Y_{min}
Normal-back	1	6.23	800	180
Normal-pod	0.964	4.6	60	900
Normal-teardrop	0.9	2.4	150	150
Normal-neptune	1	2.5	100	128

将不同的检测算法与本文算法相比较,SVM 核函数选择 RBF 核函数,gamma 值为 0.1;MPL 隐层设置为 10 * 10 * 10。实验结果如表 5 所列。

表 5 不同算法的实验结果

Table 5 Experimental results of different algorithms

数据集	本文算法		SVM		MPL	
	AUC	时间/s	*AUC*	时间/s	*AUC*	时间/s
Normal-back	1	6.23	0.99	449	0.93	16.42
Normal-pod	0.964	4.6	0.953	210	0.868	21.25
Normal-teardrop	0.9	2.4	0.92	154	0.87	30.66
Normal-neptune	1	2.5	0.98	892	0.99	17.36

从表 5 可以看出,尽管 SVM 在部分数据集中具有较高的性能,但它花费的时间过长;MLP 虽然耗时较少,但检测性能不足;而本文提出的检测方法在保证高性能的同时,大大缩短了系统建立所花费的时间。

结束语 本文针对网络攻击中的 DoS 攻击提出了一种基于边缘计算框架的孤立森林网络异常检测方法。该方法通过 SA 算法在边缘计算框架下将任务分配给不同的 MEC 节点来执行,MEC 节点利用孤立森林算法并行地进行局部孤立森林模型训练。该方法能够快速、高效地运行检测系统。实验结果表明,基于边缘计算框架的孤立森林网络异常检测方法优于其他检测方法,其能在保证较高精度的情况下减少耗时。在异常数据划分方面,孤立森林算法只能通过人为选择异常得分阈值来判断异常数据。但是,人为选择的阈值通常不是最优值,而不恰当的阈值可能会造成误检或漏检,在接下来的工作中将考虑如何改进异常数据划分的方法。

参 考 文 献

[1] PEDRO G T, JESÚS D V, M GABRIEL M F, et al. Anomaly-based network intrusion detection: Techniques, systems and challenges [J]. Computers & Security, 2009, 28(1/2): 18-28.

[2] TAN A P, CHEN H, WU B Q. Network Intrusion Intelligent Detection Algorithm Based on AdaBoost [J]. Computer Science, 2014, 41(2): 197-200.

[3] CHEN J Y, XU X Y, SU M M. Research on Network Attack Detection Based on Self-adaptive Immune Computing [J]. Computer Science, 2018, 45(S1): 364-370.

[4] WANG C, VISWANATHAN K, LAKSHMINARAYAN C, et al. Statistical techniques for online anomaly detection in data centers [C] // Proceedings of the 12 IFIP/IEEE International Symposium on Integrated Network Management. IEEE, 2011: 385-392.

[5] DING Z G, DU D J, FEI M R. An isolation principle based distributed anomaly detection method in wireless sensor networks [J]. International Journal of Automation and Computing, 2015, 12(4): 402-412.

[6] KILLOURHY K S, MAXION R A. Comparing anomaly-detection algorithms for keystroke dynamics [C] // 2009 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN). IEEE Computer Society, 2009: 125-134.

[7] YU X S, HAN D Z, DU Z X. DDoS Attack Detection System Based on Intelligent Bee Colony Algorithm [J]. Computer Scien-

ce,2018,45(12):123-129.

[8] FEDERICO S W,JUAN I A P,PABLO C D L H,et al. Anomaly Detection in Network Traffic Based on Statistical Inference and alpha—Stable Modeling [J]. IEEE Transactions on Dependable & Secure Computing,2011,8(4):494-509.

[9] MENG W Z,LI W J,KWOK L F. Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection [M]. Security and Communication Networks,2015,8(18):3883-3895.

[10] ZHANG J,ZULKERNINE M,HAQUE A. Random-Forests-Based Network Intrusion Detection Systems [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,2008,38(5):649-659.

[11] XU D,WANG Y J,MENG Y L,et al. Improved Data Anomaly Detection Method Based on Isolation Forest[J]. Computer Science,2018,45(10):155-159.

[12] ZHANG Q,HU Y P,JI C,et al. Edge Computing Application: Real-Time Anomaly Detection Algorithm for Sensing Data [J]. Journal of Computer Research and Development,2018,55(3):524-536.

[13] LI H,WANG L. Online orchestration of cooperative defense against DDoS attacks for 5G MEC[C]// Wireless Communications and Networking Conference. IEEE,2018:1-6.

[14] LIU F T,TING K M,ZHOU Z H. Isolation Forest[C]// Proceeding of the 2008 Eighth Eighth IEEE International Conference on Data Mining. IEEE Computer Society,2008:413-422.

[15] HE Y,ZHU X,WANG G,et al. Predicting Bugs in Software Code Changes Using Isolation Forest[C]// IEEE International Conference on Software Quality. IEEE,2017:296-305.

[16] CALHEIROS R,RAMAMOHANARAO K,BUYYA R,et al. On the effectiveness of isolation-based anomaly detection in cloud data centers [J]. Concurrency and Computation: Practice and Experience,2017:e4169.

[17] DING Z,DU D,FEI M. An isolation principle based distributed anomaly detection method in wireless sensor networks [J]. International Journal of Automation and Computing,2015,12(4):402-412.

[18] HE H B,GARCIA E A. Learning from Imbalanced Data [J]. IEEE Transactions on Knowledge & Data Engineering,2009,21(9):1263-1284.

[19] SHI W S,SUN H,CAO J,et al. Edge Computing—An Emerging Computing Model for the Internet of Everything Era [J]. Journal of Computer Research & Development,2017,54(5):907-924.

[20] BRUNO R P. Data Structures and Algorithms with ObjectOriented Design Patterns in Java[M]. Wiley,1999.

[21] INGBER L. Simulated annealing:Practice versus theory [J]. Mathematical & Computer Modeling:An International Journal,1993,18(11):29-57.

[22] WU J S,ZHANG W P,MA Y. Data analysis and study on KDD-CUP99 data set [J]. Computer Applications and Software,2014(11):321-325.

[23] ADETUNMBI A,ADEOLA S,DARAMOLA O. Analysis of KDD'99 Intrusion Detection Dataset for Selection of Relevance Features [J]. Lecture Notes in Engineering & Computer Science,2010,2186(1):1371-1379.

[24] TREBAR M,STEELE N. Application of distributed SVM architectures in classifying forest data cover types [J]. Computers and Electronics in Agriculture,2008,63(2):119-130.

CHEN Jia, born in 1995, postgraduate. His main research interests include internet of things and network resource scheduling.

QIAN Li-ping, born in 1981, Ph.D, professor, Ph.D supervisor, is member of China Computer Federation. Her main research interests include wireless communication, deep space communication, cognitive radio network and smart grid.