高技术通讯 2022 年 第32 卷 第8 期:789-800

doi:10.3772/j.issn.1002-0470.2022.08.002

基于逻辑回归监督学习的大样本日志异常检测优化方法①

申罕骥②*** 付 翔*** 李 俊③*

(*中国科学院计算机网络信息中心 北京 100190)

(**中国科学院大学 北京 100049)

(***北京蓝天前沿科技创新中心 北京 100085)

摘 要 传统基于日志的异常检测方法依赖于人工分析,适用于数据量小的系统,而对于复杂且庞大的日志系统,其检测效率往往很低,无法满足要求。随着机器学习的发展,检测手段发生了根本的转变,检测效率及性能也大幅提高。对于同一个日志系统,针对不同的日志预处理方法及机器学习算法,尤其对日志模板及特征的提取目前还没有统一的成熟模型,导致最后得到较大差异的检测准确率、性能等指标。本文基于监督学习方法提出大样本日志异常检测优化方法,将数据集进行日志解析得到精确的日志模板,再进行日志序列的向量化处理,使用逻辑回归监督学习算法进行分类训练与测试,结合不同的测试指标来选取最佳的参数,最终得到最优模型。实验结果证明,经此方法获取的模型能够达到较优的检测结果。

关键词 监督学习;大样本;日志处理;异常检测

0 引 言

随着信息时代的转变,信息系统的数据量呈现爆炸式增长,而基于分布式系统,由于其组件的异构性、开放性(允许增加或替换组件)、安全性、可伸缩性(用户数量增加时能正常运行的能力)、故障处理及组件的并发性,越来越受到业界重视,其产生的日志数据使传统的人工日志检测及分析方法已不可接受。检测手段的日益更新,例如机器学习的应用,使得从大量的日志中检测异常日志变得可行,检测效率及准确率等得到不同程度的提高。

本文的主要贡献和创新点是:(1)针对大数据平台日志数据集特点,提出一种新的思路对数据特征分布不平衡的日志进行一系列处理,设计了基于逻辑回归监督学习的大样本日志异常检测优化方法。该方法易理解、处理流程较简单,能广泛应用于大数据平台日志。(2)针对大样本日志异常检测效率问题,通过正则表达式、向量化处理和逻辑回归监督学习算法实现对异常检测模型的训练和优化,并采用真实数据集和不同的测试指标对模型进行了参数训练,实验验证此方法获取的模型能够达到较优的检测结果。(3)经实验对比,本文最终的模型方法在召回率、精确率、*F*1 值方面呈现较好的结果。

1 国内外研究现状

1.1 日志解析技术

目前日志模板抽取研究主要有两种类型技术路线:基于聚类思想和基于启发式的方法。

1.1.1 基于聚类思想的日志解析

聚类分析的目的是通过分析日志特点,把日志

① 国家自然科学基金(61672490,61602436)资助项目。

② 男,1987 年生,博士生;研究方向:大数据应用,互联网安全;E-mail:shenhanji@cnic.cn。

③ 通信作者,E-mail:lijun@cnic.cn。

(收稿日期:2021-08-16)

划分为不同的分组,组内中的日志相似度越大越好,组间的日志相似度越小越好,聚类属于无监督学习。Lighari 和 Hussain[1] 提出了一种离线和在线日志解析的新方法 LPV(log parser based on vectorization),与现有的日志解析方法相比,LPV 具有很好的性能。其方法是先将日志信息转化为向量,通过向量之间的距离度量两条日志信息之间的相似度,然后通过向量的聚类对日志信息进行聚类,并从聚类结果中提取日志模板。在线日志解析为日志模板分配了一些平均向量,这样传入的日志消息和每个日志模板之间的相似度也可以通过两个向量之间的距离来衡量。

Ren 等人[2] 提出了一种在图形处理器(graph processing unit,GPU)上利用独特的层次索引结构计算加权编辑距离的并行方法。该方法使用 LKE(log key extraction)可以减少处理大规模日志所需的时间。实验表明,利用 GPU 计算加权编辑距离的 LKE 解析器在 HDFS 数据集和海洋信息数据集上具有较高的效率和准确性。文献[3-5]研究了一系列的日志分析方法,其中文献[4]提出的 IPLoM(iterative partitioning log mining)方法把每条日志转换成词对的集合,基于具有公共词对的数量进行聚类,根据专家领域知识人工指定类簇的个数,对每一类进行模板提取,该方法不受日志格式的限制。Ning 等人[6] 提出的 HLAer(heterogeneous log analyzer)是一个异构的日志分析系统,首先采用层次聚类把异构日志依据格式信息进行分类并索引,然后针对每一种日志类型进行自然语言处理中常用的分词处理,并构建日志间的距离函数,应用基于密度的方法进行聚类,进而提取日志模板。

1.1.2 基于启发式的日志解析

基于启发式的日志模板抽取方法是根据日志的格式信息或日志中的词信息得到适合日志的启发式算法并提取模板。Du 和 Li[7] 提出的 Spell(a structured streaming parser for event logs using an longest common subsequence)方法是基于流模式的最长公共子序列匹配的思想提取出日志模板,解决了日志模板的在线提取问题。实验结果表明,该方法和其之前的离线算法相比,在准确度和效率上都有明显提升。He 等人[8] 提出的 Drain(a fixed depth tree based online log parsing)方法是以流的工作方式,在领域知识的基础上对相同类型日志建立正则表达式,对原始日志预处理,根据日志长度把日志分组,并把日志长度作为树结构的第一层节点。对于新的一条原始日志,首先预处理,然后遍历第一层节点,找到其所属的长度分组,根据预处理后日志的第一个标识符遍历树结构的第二层。以此类推,树的叶子节点是当前具有相似日志结构的日志集合,遍历到叶子节点层时,计算新日志和已知日志模板的相似度,比较其和事先定义阈值的大小,确定所属分组。

1.2 日志异常检测算法

逻辑回归是广泛应用于分类的统计模型,通过训练得到逻辑函数,该函数可以计算问题中所有状态的概率,概率最大的状态即所属分类。近年来,国内外关于逻辑回归的研究成果主要集中在数据预处理、标签噪声问题、算法的优化(如梯度下降法、坐标下降法及自适应随机梯度下降法)及并行化实现。Farshchi 等人[9] 采用基于回归的分析技术来查找操作活动日志与操作活动对云资源的影响之间的相关性,然后将相关模型用于导出声明规范,该声明规范可用于对运行中的操作及其对资源的影响进行运行时验证。在实验时注入了随机故障,该方法有效地对随机注入的故障发出了警报,能用于对云应用操作进行异常检测。

决策树(decision tree)常用于数据领域的分类和回归,是一种流行的机器学习手段。Rochmawati 等人[10] 使用临床症状数据集,利用 J48 and Hoeffding Tree 决策树对症状进行分类,并取得较好结果。Gavankar 和 Sawarkar[11] 提出了一种新的算法 Eage 决策树,该算法在训练时构造一个单一的预测模型,该模型考虑了测试数据中未知属性值的所有可能性。它很自然地解决了决策树归纳法中测试数据未知值的处理问题。Sahu 等人[12] 提出了一种贪婪的启发式二值化策略,以配分函数作为可分性测度。与大多数主要的多类支持向量机(support vector machines, SVM)分类器相比,该方法具有较高的分类精度和较少的计算开销。Erfani 等人[13] 提出了

一种混合模型,用无监督的深度置信网络(deep belief networks, DBNs)提取通用的基础特征,然后从DBNs学习的特征中训练一类SVM。混合模型不会造成精度损失,且具有可伸缩性。Lin等人[14]设计了LogCluster聚类方法用来识别在线系统问题。Azevedo等人[15]使用聚类算法检测卫星系统中的异常。Wurzenberger等人[16]引入了日志数据增量聚类的半监督概念,它不依赖于日志的语法和语义,通用性较强,为基于日志数据流的在线异常检测解决方案奠定了基础。Rehman等人[17]通过对主成分分析(principal component analysis, PCA)、稀疏PCA、核PCA和增量PCA等不同特征提取算法性能的评价,寻找最优的特征提取算法。这些算法与机器学习模型相结合,提高了高频预测的精度。通过对克利夫兰心衰数据库的分析,评价了这些综合模型的性能。该实验结果表明,核主成分分析算法与线性判别分析模型相结合,稀疏主成分分析算法与高斯朴素贝叶斯(Gaussian naive Bayes, GNB)模型相结合,可获得91.11%的高频分类准确率。

2 架构设计及实现

2.1 系统框架流程图

本文设计基于逻辑回归监督学习的大样本日志异常检测优化方法,系统框架流程如下。

(1)对日志使用正则表达式对所有日志进行解析,得到一系列日志模板。

(2)对每个日志模板,使用独热(One-Hot)编码进行映射,并且在任意时候,其中只有一位有效。

(3)将所有原始日志以其BlockID为关键字进行分类,相同的BlockID聚合成日志序列,再利用日志编码将日志序列进行向量化处理,得到一个完整的向量。

(4)将取得的向量进行标准化处理,用逻辑回归模型进行训练,并使用正则化,以使模型能进行特征向量的一定取舍,使训练结果有更佳的性能以及更好的泛化能力。

(5)对训练所得的模型用测试数据进行检测,以检测本次方法。

系统架构如图1所示。

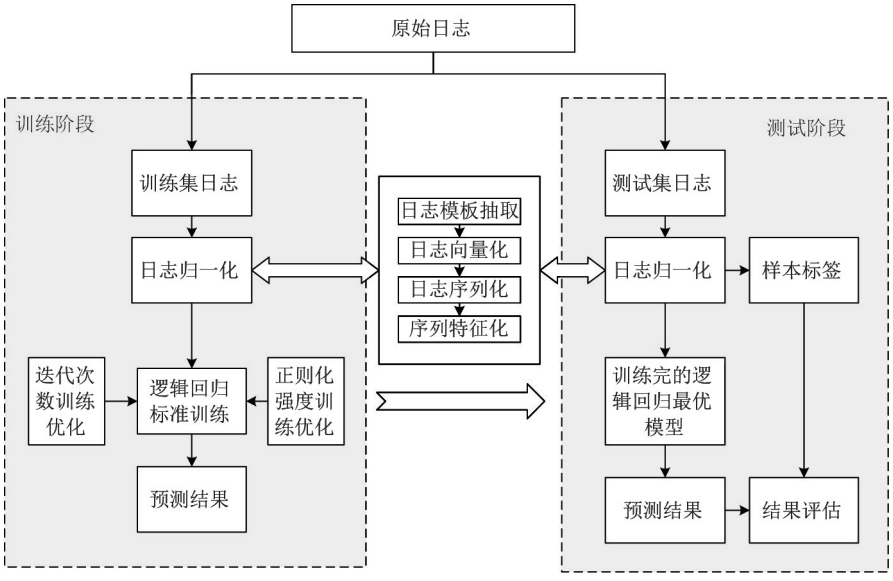

图1 基于逻辑回归监督学习的大样本日志异常检测模型

2.2 日志解析

2.2.1 原始日志

选用Amazon EC2平台上的203节点集群收集的日志数据集,该HDFS数据集属于大规模分布式系统产生的日志,是Amazon公开的生产环境日志,具有一定的研究价值,可作为日志异常检测的方法及模型的参考标准。该日志数据集HDFS包含38.7 h的11 175 629条原始日志记录,大小为1.6 GB,包含

的数据块有 575 061 个,其中异常数据块有16 838 个。HDFS 日志为每个工作执行分配唯一的 ID 号,以每个块 ID 划分为一个时域窗口,每个唯一的块 ID 将日志划分为一组日志序列,该序列对应一系列块操作,例如分配、写入、复制、删除等。

2.2.2 日志特征

(1)日志格式为日期 + 时间 + 时间参数 + 日志级别(INFO 和 WARN) + 数据操作者 + 具体信息。具体示例为 081110 123628 11303 INFO dfs. DataNode $ PacketResponder: Received block blk _ 88601410 04676523018 of size 67108864 from /10. 250. 13. 240。

(2)每条日志都有数据块的号码,如 blk _ 8860141004676523018。

(3)有些数据块日志数量极少,仅有 2 条日志 3418 个数据块。

(4)日志中出现的词的字数统计分布极其不平衡,日志信息中单词频繁度的不平衡特征会影响传统文本挖掘技术对日志进行研究的效果。如基于频繁项日志挖掘算法和聚类算法。

(5)日志记录的语法结构非常弱,但是字段及其参数之间仍然有相当的逻辑关系和语义关系。

(6)日志冗余信息较多,许多日志异常检测算法利用此特点,根据不同的窗口定义方法,得到以相同日志出现次数为特征的特征矩阵。

(7)相同源的日志数据有相同的结构,不同源的日志数据有不同的结构,大多数现有的日志数据聚类算法隐式地或明确地利用了日志结构的这一特点。

(8)日志还有明显的时序特征,时序数据由时间戳、标签和指标三要素组成。时间戳表明数据发生的时间;标签为属性信息,表明数据属于的设备/模块,不随着时间变化;指标为统计数据、状态指标。

2.2.3 日志预处理

(1)日志解析

日志解析通常作为后续日志分析任务的第一步,是训练机器学习模型的必要数据预处理步骤。将文本日志消息解析为结构化格式可以有效地搜索、过滤、分组、计数和进行复杂的日志挖掘。

数据清洗:通过对大量的日志事件进行观察发现,一些标记变量通常都是以数字、IP 地址、端口号、块大小等形式出现,或者这些标记中包含了大括号、中括号、圆括号,再或者标记中有下划线、斜线、反斜线等。这些标记在事件消息中非常容易辨认,可以使用规则定义的方法进行识别。因此,通过定义显式的规则表达来描述这些典型的标记变量,并将这些标记变量使用空来表示。这一步骤之后,剩下的标记为候选的标记常量。基于以上日志的特点及日志分析的目的,本次预处理只提取具体信息部分,日志开头的日期时间及日志级别均不作提取,且对于每条日志的信息部分,本次用正则表达式来进行解析。

算法:以每条日志为单位,将其中的关键字(非关键字及数字等除外)保留,以作为依据判断为某个特征语句。而其中的非关键字,即日志中的日期、时间、级别直接删除,IP 地址、端口号、块大小等带数字信息标记为变量,以“ * ”代替。示例如下:

dfs. DataNode $ DataXceiver: Receiving block * src: * dest: *

(2)日志模板提取

提取日志模板是处理海量系统日志十分有效的方法。其对所有日志进行处理,得到一个不含任何参数信息的通用日志,并进行去重处理,最后得出相互独立的日志,且每种格式代表不同种类的日志消息,这些日志就称为日志模板,由这些模板可以组成各种各样的日志。

算法:根据上一步的日志解析方法,将所有日志解析成不带参数的通用日志,然后将得到的日志进行去重处理,最后得到的日志即为需要的日志模板。

优势:由于该算法使用正则表达式,能准确提取出日志的关键字,且手动去重解析后的日志比其他算法(如基于聚类思想和基于启发式)的准确率高。在相对不复杂的日志结构中,是一个比较理想的选择。

算法伪码如算法 1 ~ 算法 3 所示。

用正则表达式即可提取出日志模板,得到一个非重复的日志模板,如表 1 所示。

算法 1　用正则表达式替换日志中的变量

```
输入:HDFS _ 1. log
输出:HDFS _ REP. log
        read log file
        while 1:
            read all lines
            if no lines:
                break
            for every line:
                for i in range(3):
                    delete digits in front of line
                    delete backspace
                delete level of message(uppercase)
                delete backspace
            convert line into array with backspace split
            for i in range(length of array):
                if   chars including digits or blockID
                    replace with' * '
            convert array into string
        write string into log file
```

算法 2　清除日志之间空格行

```
输入:HDFS _ REP. log
输出:HDFS _ null. . log(with null line)
open log file and new a log file
  for text infr. readlines():
    if line:
      write line into new log file
```

算法 3　去除重复行

```
输入:HDFS _ null. log
输出:HDFS _ unique. . log(with unique line)
open log file and new a log file
read log file
for line in file:
      delete enter char
      if   line not in new log file:
            write line into new log file
```

2.3　特征处理

2.3.1　日志向量化处理

算法:将 54 条日志用一个 54 个行列的单位矩阵表示,矩阵每行表示一条日志,且单位矩阵中的每个向量是正交关系,即每个向量相互独立,所以每条日志也是相互独立。

表 1　本数据集日志模板

数据操作模块	数据操作
dfs. FSNamesystem	BLOCK * NameSystem. allocateBlock: * . blk _ *　*
	BLOCK * NameSystem. addStoredBlock: blockMap updated: * is added to blk _ * size *
	BLOCK * ask * to replicateblk _ * to datanode(s) *
	BLOCK * NameSystem. delete: blk _ * is added to invalidSet of *
	BLOCK * ask * to replicateblk _ * to datanode(s) * *
	BLOCK * NameSystem. addStoredBlock: addStoredBlock request received for blk _ * on * size * But it does not belong to any file.
	BLOCK * NameSystem. addStoredBlock: Redundant addStoredBlock request received for blk _ * on * size *
	BLOCK * Removing blockblk _ * from neededReplications as it does not belong to any file.
dfs. DataNode $ DataXceiver	Receiving block * src: * dest: *
	Received blockblk _ * src: /ip: * dest: / * of size *
	* :Got exception while servingblk _ * to /ip: * Served blockblk _ * to /ip
	writeBlockblk _ * received exception java. io. IOException: Could not read from stream
	writeBlockblk _ * received exception java. io. IOException: Block blk _ * is valid, and cannot be written to.
	writeBlockblk _ * received exception java. net. SocketTimeoutException
	writeBlockblk _ * received exception java. io. IOException: Connection reset by peer
	writeBlockblk _ * received exception java. io. EOFException
	writeBlockblk _ * received exception java. nio. channels. ClosedByInterruptException
	writeBlock * received exception java. io. InterruptedIOException: Interruped while waiting for IO on channel java. nio. channels. SocketChannel[connected * * * millis timeout left.
	writeBlock * received exception java. net. SocketTimeoutException: * millis timeout while waiting for channel to be ready for read. ch : java. nio. channels. SocketChannel[connected * *

	writeBlock * received exception java. net. SocketTimeoutException: * millis timeout while waiting for channel to be ready for write. ch : java. nio. channels. SocketChannel [connected * *
	writeBlockblk _ * received exception java. net. NoRouteToHostException: No route to host
	writeBlockblk _ * received exception java. io. IOException: Broken pipe
	writeBlockblk _ * received exception java. io. IOException: Interrupted receiveBlock
dfs. DataNode $Packet Responder	Received blockblk _ * of size * from /ip
	PacketResponderblk _ * 2 Exception java. io. EOFException
	PacketResponder * * Exception java. net. SocketTimeoutException: * millis timeout while waiting for channel to be ready for read. ch : java. nio. channels. SocketChannel [connected * *
	PacketResponder * * Exception java. io. InterruptedIOException: Interruped while waiting for IO on channel java. nio. channels. SocketChannel[connected * * * millis timeout left.
	PacketResponderblk _ * 1 Exception java. io. IOException: The stream is closed
	PacketResponderblk _ * 2 Exception java. nio. channels. ClosedByInterruptException
	PacketResponderblk _ * 0 Exception java. io. IOException: Connection reset by peer
	PacketResponderblk _ * 1 Exception java. io. InterruptedIOException: Interruped while waiting for IO on channel java. nio. channels. SocketChannel[closed]. 59 990 millis timeout left.
	PacketResponderblk _ * 2 Exception java. io. IOException: Broken pipe
	PacketResponder * for block blk _ * terminating
	PacketResponder * for block blk _ * Interrupted.
dfs. DataNode	* Starting thread to transfer blockblk _ * to * , *
	* Starting thread to transfer blockblk _ * to *
dfs. DataNode $ DataTransfer	* :Failed to transferblk _ * to * got java. io. IOException: Connection reset by peer
	* :Transmitted blockblk _ * to / *
dfs. FSDataset	Unexpected error trying to delete blockblk _ * . BlockInfo not found in volumeMap.
	Deleting blockblk _ * file /mnt/hadoop/dfs/data/current/ * /blk _ *
	Reopen Blockblk _ *
dfs. DataBlockScanner	Adding an already existing blockblk _ *
	Verification succeeded forblk _ *
dfs. PendingReplicationBlocks $ PendingReplicationMonitor	PendingReplicationMonitor timed out block blk _ *
dfs. DataNode $ BlockReceiver	Exception inreceiveBlock for block blk _ * java. io. EOFException
	* :Exception writing blockblk _ * to mirror *
	Exception inreceiveBlock for block * java. io. InterruptedIOException: Interruped while waiting for IO on channel java. nio. channels. SocketChannel [connected * * * millis timeout left.
	dfs. DataNode $ BlockReceiver: Exception in receiveBlock for block * java. net. SocketTimeoutException: * millis timeout while waiting for channel to be ready for write. ch : java. nio. channels. SocketChannel [connected * *
	Exception inreceiveBlock for block blk _ * java. nio. channels. ClosedByInterruptException
	Exception inreceiveBlock for block blk _ * java. io. IOException: Broken pipe
	Receiving empty packet for block *
	Changing block file offset of block * from * to * meta file offset to *
	Exception in receiveBlock for block blk _ * java. io. IOException: Connection reset by peer

算法 4 日志向量化计算

```
输入:HDFS _ unique. log
输出:Array(54 * 2)
set a unit matrix(54 dimensions)
set an array of 54 * 2 dimensions
for k in range(54): assign row vector of the unit matrix to 1st column of 54 * 2
open log formwork file
While 1:
  read log formwork file
  If non-line:
    break
  for line in lines:
    delete '\n' in line
    Add line to 2nd column of the array(54 * 2)
```

每个日志模板与一个单位向量形成一一对应的关系,且单位向量两两正交,所以日志之间无关联,即不会相互影响。

获得的文件包括日志模板及每个模板的向量。

算法5　数组写入文件

```
输入:Array(54 * 2)
输出:feature _ vector. csv
Open feature vector file
write array(54 * 2) into feature vector file
```

2.3.2　样本特征向量化

算法:将所有日志依据不同的 BlockID 创建一个 log 文件(如没有该文件),相同的 BlockID 则放入相同的文件中,然后再根据每个文件,读出所有的日志。根据上述中的日志解析方法解析出日志后,对比日志模板中的向量表,找出该条日志的向量,并相加于该 BlockID 的向量,如此循环,直到日志读取完毕,得到的向量即为该 BlockID 的向量。

算法伪码如算法6和算法7所示。

算法6　生成每个 block 的向量

```
输入:blockID. log(575 061 log files)
输出:Array(575 061 * 2)
create array with 575 061 * 2 dimensions
read the names of files in log folders divided byblockID
for i in range(575 061):
  write names into 1st column in array(575 061 * 2)
for i in range(575 061):
  open log files divided byblockID
  While 1:
    read line in log file
    If non-line:
      break
    for line in lines:
        for i in range(3):
          delete digits in front of log
          delete backspace
        delete level info(UPPERCASE)
        delete backspace
        write logs into arrays
        for j in range(len(arrs)):
          if  string includingdigites:
            replace string with '*'
    for k in range(len(arrs)):
        delete '\n' in strings in arrays
      write array into strings
    for l in range(54):
      if line in feature vector:
          Add feature vector to 2nd column inarray
(575 061 * 2)
```

算法7　数组写入文件

```
输入:Array (575 061 * 2)
输出:blockID _ VECTORS. csv
create block vectors file with csv formt:
write arrays (575 061 * 2) into csv file
```

得到每个 BlockID 与其日志序列相对应的向量表,该表可作为后续模型训练直接使用的数据样本。

2.3.3　日志标签处理

算法:将每个 block 的标签加到对应的每个 BlockID 的额外向量中,即数据 + 标签,其中最后一维为标签。

算法伪码如算法8所示。

算法8　日志标签处理算法

```
输入:blockID _ VECTORS. csv
输出:blockID _ VECTORS _ labels. csv
write blockID's vectors into pandas array
add one column into pandas array
write values into 1st list(list1)
write label file into 2nd list(list2)
Create a csv file
for i in range(575061):
  for j in range(575061):
    if blockID equals for list2 in list1:
      Write vector divided by backspace into an array
      Add label value in list2 into last column in list1
      Write list1 into a new csv file          if  string in-
cluding digites:
          replace string with '*'
      for k in range(len(arrs)):
          delete '\n' in strings in arrays
        write array into strings
      for l in range(54):
        if line in feature vector:
          Add feature vector to 2nd column in array
(575 061 * 2)
```

得到一个数据文件,该文件不含表头,但包含每个样本值及对应的标签值。同时,该文件可作为后续监督学习模型的直接输入数据样本。

2.4 异常检测

2.4.1 样本选择

本次训练按照经典的比例选择训练和测试样本。

训练样本:575 061 ×75% =431 300。

测试样本:575 061 ×25% =143 761。

2.4.2 数据处理技术

(1)数据标准化处理

此样本中有些特征的方差过大,会主导目标函数从而使参数估计器无法正确地学习其他特征,所以需要对数据作标准化处理。标准化处理有两个好处,一个是提升模型精度,标准化/归一化使不同维度的特征在数值上更具比较性,提高分类器的准确性;另一个是提升收敛速度,对于线性模型,数据归一化使梯度下降过程更加平缓,更易正确地收敛到最优解。

其算法适用于本身服从正态分布的数据,所以本次使用标准差标准化算法对向量进行预处理,用来去均值和方差归一化,且针对每一个特征维度,而不是针对样本。标准化的缩放通过方差和每个点都相关,即每个点都作出了贡献。

(2)数据正则化处理

逻辑回归算法中添加多项式项后,可以对非线性数据进行分类。添加多项式项后,模型变得复杂,容易出现过拟合现象。在算法中加入正则化项能在一定程度上避免过拟合问题。见式(1)~(6)。

$$z = \boldsymbol{w}_1 x_1 + \boldsymbol{w}_2 x_2 + \cdots + \boldsymbol{w}_n x_n \tag{1}$$

$$\boldsymbol{w} = \boldsymbol{w}_1 + \boldsymbol{w}_2, \cdots, \boldsymbol{w}_n \tag{2}$$

回归函数:

$$y = \frac{1}{1 + e^{-\boldsymbol{w}^{\mathrm{T}} x}} \tag{3}$$

选择 L2 正则化方法时,代价函数:

$$L_2 = \min_{w,c} \frac{1}{2} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{w} + C \sum_{i=1}^{n} \log(\exp(-y_i(\boldsymbol{X}_i^{\mathrm{T}} \boldsymbol{w} + c)) + 1) \tag{4}$$

选择 L1 正则化方法时,代价函数:

$$L_1 = \min \| \boldsymbol{w} \|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(\boldsymbol{X}_i^{\mathrm{T}} \boldsymbol{w} + c)) + 1) \tag{5}$$

选用弹性网络函数时,代价函数:

$$L = \min \frac{1-\rho}{2} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{w} + \rho \| \boldsymbol{w} \|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(\boldsymbol{X}_i^{\mathrm{T}} \boldsymbol{w} + c)) + 1) \tag{6}$$

其中,ρ 控制 L1 与 L2 的强度,C 值表示惩罚值。

假定目标 y_i 在测试时应属于集合[-1,1],可以发现弹性网络函数在 ρ =1 时与 L1 正则化等价,在 ρ =0 时与 L2 正则化等价。

C 值是对错误预测的惩罚值。C 值越大,分类器的准确性越高,但容错率会越低,泛化能力变差。相反,如果 C 值较小,分类器则具有较好的容错率,其泛化能力较好。

正则化的目的是 L1 正则化将系数 w 的 L1 范数作为惩罚项加到损失函数上,由于正则项非零,这就迫使那些弱的特征所对应的系数变成 0。因此 L1 正则化往往会使学到的模型很稀疏(系数 w 经常为 0),这个特性使得 L1 正则化成为一种很好的特征选择方法。L2 正则化将系数向量的 L2 范数添加到损失函数中。由于 L2 惩罚项中系数是二次方的,这使得 L2 和 L1 有着诸多差异,最明显的一点是 L2 正则化会让系数的取值变得平均。对于关联特征,这意味着它们能够获得更相近的对应系数。

总的来说,L2 正则化可以防止模型过拟合(overfitting),但一定程度上,L1 也可以防止过拟合。如果特征量很大,数据维度很高,倾向于使用 L1 正则化;如果目的只是为了防止过拟合,选择 L2 就足够了。

3 实 验

3.1 实验环境与实验数据集

实验环境采用华为服务器 FusionServer 2288H V5(CentOS 7.6 系统,Intel gold 6248 处理器,384 GB DDR4 内存),数据集采用 Amazon EC2 平台上的 HDFS 原始日志数据集,再随机抽取其中一部分数

据(所属数据块有 142 730 个)进行实验。所抽取的数据中异常数据块的比例与整个数据块一致,以保证该数据训练的模型尽可能反映整个数据集的特征。

3.2 实验指标

(1)精确度(precision)是正确预测为正的占全部预测为正的比例,即 $TP / (TP + FP)$。

(2)召回率(recall)是正确预测为正的占全部实际为正的比例,即 $TP / (TP + FN)$。

(3)$F1$-score 是精确率和召回率的调和平均数,即 $2 \times precision \times recall / (precision + recall)$。

(4)准确率(accuracy)是正确预测占所有样本的比例,即 $(TP + TN) / (TP + TN + FP + FN)$。

(5)受试者工作特性(receiver operating characteristic,ROC)曲线是反映敏感性和特异性的综合指标。

(6) AUC(area under curve)为 ROC 曲线下的面积大小,它能够量化地反映基于 ROC 曲线衡量出的模型性能。

3.3 实验方案

本次通过 Amazon EC2 平台上的 HDFS 原始日志数据集,使用 Python 开发语言,采用 scikit-learn 模型库进行实验。根据引言中的系统架构图,从原始日志经过数据解析、特征提取、模型训练后得到相关参数,并比较模型测试指标得到参数的最优解。其实验步骤如下。

(1)将日志解析后生成模板。

(2)模板向量化及特征提取。

(3)日志序列向量化。

(4)标签数字化处理及加入数据样本中。

(5)数据标准化处理。

(6)使用不同的正则化强度、迭代次数来进行参数的优化。

(7)得出使得参数最优化的模型。

本次比较不同的 C 值(正则化强度相关参数)及迭代次数与准确率、召回率(精确度)、耗时及 AUC 的关系,从中找到使精确度与耗时得到平衡的最佳参数值。

3.4 实验结果

3.4.1 迭代次数选择

训练模型时,通常迭代次数越多,得到的模型越好。但迭代次数和时间消耗是一对矛盾体,二者不可兼得。此时需要在模型好坏与时间消耗上找到一个平衡点,既能得到尽可能好的模型,又使消耗的时间较少。

本节通过比较不同的迭代次数,查看各个不同性能指标对变化。

图 2 和图 3 分别是准确率、召回率、精确率及 $F1$ 值随着迭代次数变化的关系图。

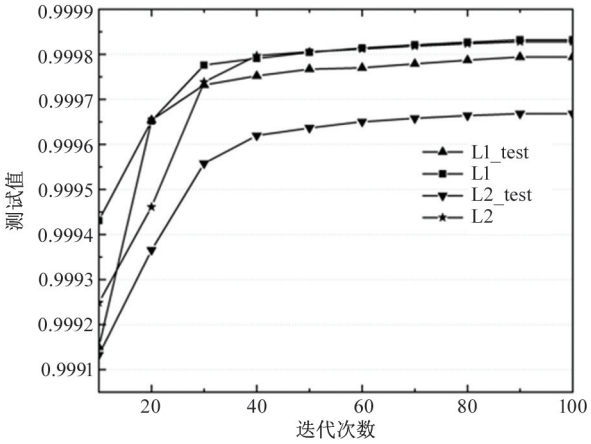

图 2 准确率与迭代次数的关系图

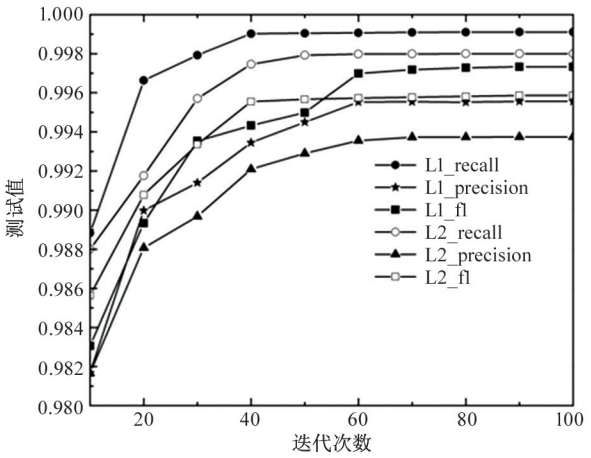

图 3 召回率、精确率及 $F1$ 值与迭代次数的关系图

实验结果表明,当迭代次数超过 80 次时,模型在 L1 和 L2 正则化方法下均能达到 99.9% 以上的准确率,$F1$ 值达到 99.73% 以上,性能指标情况见表 2。此时训练得到的逻辑回归模型参数较好。

3.4.2 正则化强度选择

逻辑回归的代价函数中,加入正则化项后,其代价函数中参数 C 为正则化强度的倒数,为正数,其

表 2　迭代次数 80 次以上时性能指标情况

性能指标	L1 正则化	L2 正则化
迭代次数	>80	>80
准确率(训练集)	99.9832%	99.9682%
准确率(测试集)	99.9794%	99.9109%
召回率	99.9109%	约 100%
精确率	99.5560%	99.4681%
*F*1 值	99.7331%	99.7333%
AUC 值	0.999 949	0.999 974
ROC 值	0.999 297	0.999 974

值更小代表更强的正则化。

本节通过比较不同的正则化强度相关参数,查看各个不同性能指标对变化。

图 4 和图 5 分别是准确率、召回率、精确率及 *F*1 值随着 *C* 值变化的关系图。

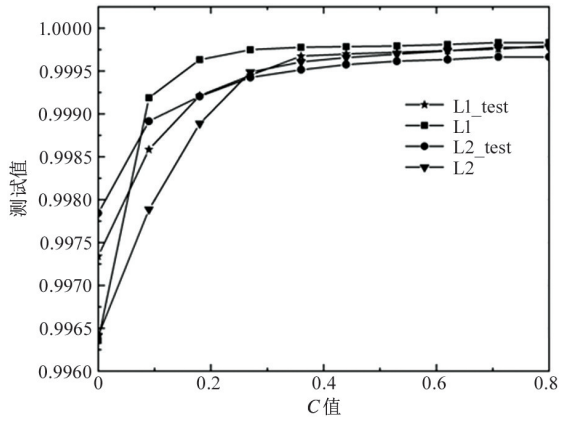

图 4　准确率与 *C* 值的关系图

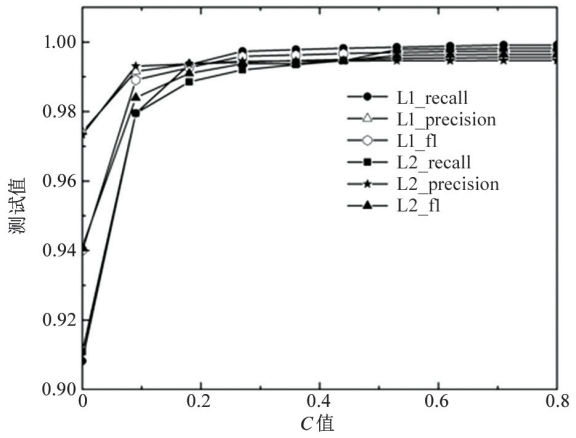

图 5　召回率、精确率及 *F*1 值与迭代次数的关系图

实验结果表明,当 *C* 值约为 0.8 时,模型在 L1 和 L2 正则化方法下均能达到 99.9% 以上的准确率,*F*1 值达到 99.64% 以上,性能指标情况见表 3。此时训练得到的逻辑回归模型参数较好。

表 3　*C* 值约 0.8 时性能指标情况

性能指标	L1 正则化	L2 正则化
C 值	约 0.8	约 0.8
准确率(训练集)	99.9832%	99.9776%
准确率(测试集)	99.9794%	99.9664%
召回率	99.9109%	99.8217%
精确率	99.5560%	99.4671%
*F*1 值	99.7331%	99.6441%
AUC 值	0.999 380	0.999 978
ROC 值	0.999 248	0.999 974

综上实验可知,当迭代次数为 80 次,*C* 值约为 0.8 时,本模型训练所需时间与模型的性能均能得到较理想的结果。

3.4.3　算法对比

针对已训练的模型参数,选取 PCA、DeepLog 算法进行对比实验,其中 DeepLog 选用 top-2 判断为预测正确。实验结果见图 6,通过实验发现本文训练的逻辑回归监督学习方法在召回率、精确率、*F*1 值方面呈现较好的结果。

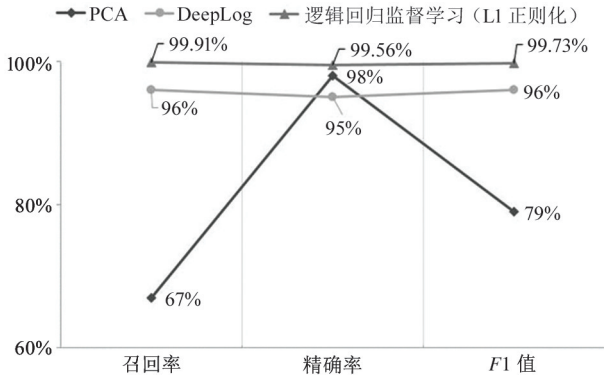

图 6　PCA、DeepLog 与逻辑回归监督学习算法对比

4　结 论

针对大数据平台日志数据集特点,本文提出一种新的思路对数据特征分布不平衡的日志进行一系列处理。即用正则表达式准确提出日志模板,将日志作为一个词向量,以 BlockID 作为时间窗口域,将

属于每个 BlockID 的日志对应的向量相加得到该 Block 的日志序列化向量;然后使用数据标准化处理算法将样本数据进行归一化,再选择逻辑回归及正则化进行模型训练;最后用测试样本对训练后的模型进行测试,得到使得模型最优的参数及最终的模型。该方法易理解、处理流程较简单,能广泛应用于大数据平台日志。

参考文献

[1] LIGHARI S N, HUSSAIN D. Hybrid model of rule based and clustering analysis for big data security[C]// International Conference on Latest Trends in Electrical Engineering and Computing Technologies, Karachi, Pakistan, 2017:1-5

[2] REN X, ZHANG L, XIE K, et al. A parallel approach of weighted edit distance calculation for log parsing[C]// 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology, Beijing, China, 2019: 101-104

[3] MAKANJU A, ZINCIR-HEYWOOD A N, MILIOS E E, et al. Spatio-temporal decomposition, clustering and identification for alert detection in system logs[C]// Proceedings of the 27th Annual ACM Symposium on Applied Computing, New York, USA, 2012: 621-628

[4] MAKANJU A, ZINCIR-HEYWOOD A N, MILIOS E E. System state discovery via information content clustering of system logs[C]//2011 6th International Conference on Availability, Reliability and Security, Vienna, Austria, 2011: 301-306

[5] DU M, LI F. Spell:online streaming parsing of large unstructured system logs[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2018, 31(11): 2213-2227

[6] NING X, JIANG G, CHEN H, et al. HLAer: a system for heterogeneous log analysis[J]. *SDM Workshop on Heterogeneous Learning*, 2014: 1-22

[7] DU M, LI F F. Spell: streaming parsing of system event logs[C]//2016 IEEE 16th International Conference on Data Mining, Barcelona, Spain, 2016: 859-864

[8] HE P, ZHU J, ZHENG Z, et al. Drain:an online log parsing approach with fixed depth tree[C]//2017 IEEE International Conference on Web Services, Honolulu, USA 2017: 33-40

[9] FARSHCHI M, SCHNEIDER J G, WEBER I, et al. Experience report: anomaly detection of cloud application operations using log and cloud metric correlation analysis [C]//2015 IEEE 26th International Symposium on Software Reliability Engineering, Gaithersbury, USA, 2015: 24-34

[10] ROCHMAWATI N, HIDAYATI H B, YAMASARI Y, et al. Covid symptom severity using decision tree[C]//2020 3rd International Conference on Vocational Education and Electrical Engineering, Surabaya, Indonesia, 2020: 1-5

[11] GAVANKAR S S, SAWARKAR S D. Eager decision tree [C]//2017 2nd International Conference for Convergence in Technology, Mumbai, India, 2017: 837-840

[12] SAHU S K, PUJARI A K, KAGITA V R, et al. GP-SVM: tree structured multiclass SVM with greedy partitioning[C]//2015 International Conference on Information Technology, Bhubaneswar, India, 2015: 142-147

[13] ERFANI S M, RAJASEGARAR S, KARUNASEKERA S, et al. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning [J]. *Pattern Recognition*, 2016, 58: 121-134

[14] LIN Q, ZHANG H, LOU J G, et al. Log clustering based problem identification for online service systems[C]// 2016 IEEE/ACM 38th International Conference on Software Engineering Companion, Austin, USA, 2016: 102-111

[15] AZEVEDO D R, AMBRÓSIO A M, VIEIRA M. Applying data mining for detecting anomalies in satellites[C]// 2012 9th European Dependable Computing Conference, Sibiu, Romania, 2012: 212-217

[16] WURZENBERGER M, SKOPIK F, LANDAUER M, et al. Incremental clustering for semi-supervised anomaly detection applied on log data[C]//Proceedings of the 12th International Conference on Availability, Reliability and Security, New York, USA, 2017: 1-6

[17] REHMAN A, KHAN A, ALI M A, et al. Performance analysis of PCA, sparse PCA, kernel PCA and incremental PCA algorithms for heart failure prediction[C]//2020 International Conference on Electrical, Communication, and Computer Engineering, Istanbul, Turkey, 2020: 1-5

Large sample log anomaly detection optimization method based on logistic regression supervised learning

SHEN Hanji[*,**], FU Xiang[***], LI Jun[*]

([*] Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] Blue Sky Frontier Science and Technology Innovation Center, Beijing 100085)

Abstract

The traditional anomaly detection method based on log relies on manual analysis, which is suitable for the system with small amount of data, but for the complex and large log system, its detection efficiency is often very low and unsuitable. With the development of machine learning, fundamental changes have taken place in detection methods, and detection efficiency and performance have also been greatly improved. For the same log system, there is no unified mature model for different log preprocessing methods and machine learning algorithms, especially for the extraction of log templates and features, which leads to relatively different detection accuracy, performance and other indicators. Based on supervised learning method, a large sample logging anomaly detection method is put forward, the accurate data set is obtained after log parse template, then the vectorization of logging sequence is processed, logistic regression supervise learning algorithms are used to classify training and testing, different test indexes are combined to select the best parameters, finally the optimal model is obtained. Experimental results show that the model obtained by this method can achieve better detection results.

Key words: supervised learning, large sample, log processing, anomaly detection