统计策略序列模式挖掘及其在软件缺陷预测中的应用

唐 磊[1] 李春平[1] 杨 柳[2]
(清华大学软件学院 北京 100084)[1] (中南大学软件学院 长沙 410075)[2]

摘 要 人类的生活越来越依赖于高可靠性和可用性的软件系统,软件缺陷一直是软件工程领域中研究最活跃的内容之一。在研究序列模式挖掘技术的基础上,介绍了软件缺陷预测的相关技术,设计了一种基于统计策略的序列模式挖掘算法的软件缺陷预测方案,实现了 InfoMiner 和 STAMP 两种模式挖掘算法、卡方检验特征选择和 SVM 等分类算法;构造了一个软件缺陷预测模型,实现了预测和发现软件系统中的未知缺陷的功能。实验结果表明,所提软件预测模型可以获得良好的预测结果,具有一定的使用价值和应用前景。

关键词 数据挖掘,序列模式,软件缺陷,信息增益,分类预测

中图法分类号 TP391 **文献标识码** A

Statistically Significant Sequential Pattern Mining Applying to Software Defect Prediction

TANG Lei[1] LI Chun-ping[1] YANG Liu[2]
(School of Software, Tsinghua University, Beijing 100084, China)[1] (School of Software, Central South University, Changsha 410075, China)[2]

Abstract Nowadays the human beings are more and more reliant on software systems which have high reliability and usability, and the technology of software defect prediction has been one of the most active parts of software engineering. This paper introduced the technology of software defect prediction on the basis of sequential pattern mining and designed a model for software defect prediction with the technology of mining statistically significant pattern. It described the architecture and detailed implementation of the algorithms named "InfoMiner" and "STAMP". The model using InfoMiner and STAMP to mine patterns, chi-square test to feature selection and SVM to classify can find unknown defects with high probability. Experimental results show that the model is able to get high prediction accuracy, so that it is valuable and has future prospects.

Keywords Data mining, Sequential pattern, Software defect, Information gain, Classification and prediction

随着信息产业和软件技术的发展,软件已经成为影响国民经济、军事、政治的重要因素,然而软件缺陷不可避免。许多研究者都致力于通过建模等方式来有效预测软件缺陷,并尽可能减少软件缺陷的产生,从而提高软件的质量。数据挖掘是知识发现(Knowledge Discovery)中的核心步骤,目的是从海量数据中提取出有效的、有用或潜在有用的、新颖的且可以被理解的知识。自 Agrawal 等人[1]首次提出序列模式挖掘(Sequential Pattern Mining, SPM)以来,人们一直对此领域保持高度的研究热情,例如基于 Bayesian Network 模型的和 Probabilistic Relational Model 的软件缺陷预测模型[2]。缺陷检测技术是软件质量保证的基础,因此可以把软件缺陷检测看作是对序列数据进行挖掘序列模式的一个过程,通过序列模式挖掘方法挖掘出特定模式后,再将其运用于软件缺陷预测中,这在软件的整个生命周期中有重要意义。

本文提出的基于统计策略主要就是利用统计学的方法,计算每个模式的信息熵和信息增益对其进行模式重要程度的评估,主要利用了 InfoMiner 算法和 STAMP 算法从训练集中挖掘出对分类有重大意义的模式。然后利用卡方检验进行特征选择,最后采用 SVM 分类器和提出的 inTimes 分类器对测试集进行分类预测。

1 相关理论基础

1.1 统计策略序列模式挖掘

基于统计策略指按照统计学的方法,计算每个模式的信息熵和信息增益对其进行模式重要程度的评估。本文将介绍两种基于统计策略的序列模式挖掘算法,一种是用于挖掘整个序列的 InfoMiner 算法,另一种是挖掘子序列的模式算法——STAMP 算法。

InfoMiner[3]是 Jiong Yang 等人提出的,以 Information 值去衡量一个模式的重要程度,累计序列中的所有项的 Information 值,并乘以模式在序列中的重复次数,得到信息增益(Information Gain),并以此来过滤候选模式作为判断意外模式的依据。一个序列可能随着时间的推移发生变化,很多模式可能只在一段时间内有效。如果考虑整个序列,模式的重要程度可能被冲淡,再加上在有的应用当中,用户很可能不仅仅对模式是否重要感兴趣,还对其模式在序列中的位置感

到稿日期:2012-07-27 返修日期:2012-11-08

唐 磊(1989-),男,硕士生,主要研究方向为计算机图形学、数据挖掘,E-mail:tanglei3shi@163.com;李春平(1965-),男,博士,副教授,主要研究方向为人工智能、数据挖掘、Web 信息资源搜索等;杨 柳(1979-),博士,讲师,主要研究方向为网格计算、语义 Web、智能算法。

兴趣。在 STAMP 算法[4]中，提出了一种基于利用信息损失及综合信息增益的概念来解决这个问题的方案。

1.2 特征选择算法

在处理分类的问题中，特征向量是一系列的特征值，这些特征值描述了样本数据属于某一类的特征信息，采取何种特征值来描述和区分特征对最后整个序列数据的分类预测结果非常重要。本文的研究中主要采用支持度特征值（以模式在序列数据中的支持度作为该模式的特征值）和信息增益特征值（以信息增益作为模式的特征值，某个模式的信息增益值越大，代表这个模式对分类越有用[5]）。

卡方检验是一种用途比较广泛的假设检验方法，特别是在分类资料统计中有广泛的应用[6]。卡方统计量能够衡量变量之间的关系密切程度。本文将为每一个特征计算一个分数，也就是卡方统计量，选取分数较高的特征参与机器学习进行训练，并用来进行分类预测。

1.3 分类预测算法

常见的分类算法有基于决策树的算法，如 ID3[7] 和 C4.5[8]、基于贝叶斯分类算法[9]和支持向量机（Support Vector Machine，SVM）分类算法等。本文在研究过程中提出了 inTimes 分类算法，因发现其准确率不高，随后采用著名的 SVM 分类。

inTimes 分类的基本思路是将训练集中的数据按照其分类标签分成两类，通过 InfoMiner 或者 STAMP 算法分别对这两类序列数据进行挖掘，得到针对不同类的一系列模式，扫描测试集，根据测试的序列属于某类挖掘的模式居多者来判定其属于该类。

支持向量机 SVM 使用一种非线性的映射，将原始训练数据映射到较高的维，通过新的较高的维搜索线性最佳分类超平面，使两类数据可以被该超平面分开，SVM 使用支持向量和边缘来发现该超平面[20]。

2 统计策略 SPM 算法设计实现

本文研究的软件缺陷分类预测模型分为模式挖掘、特征提取和分类预测 3 个阶段。其预测模型如图 1 所示，首先用 InfoMiner、STAMP 算法就软件缺陷历史数据进行模式挖掘，得到体现软件缺陷历史数据的序列模式。然后采取一定的特征提取算法从众多的序列模式中选取最能体现软件缺陷序列数据的特征值。最后结合软件缺陷数据，利用分类算法进行分类预测得到结果。

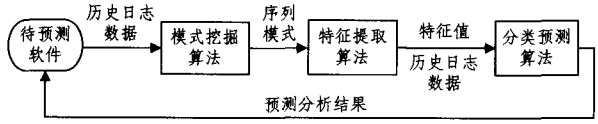


图 1 软件缺陷分类预测模型

2.1 InfoMiner 挖掘算法

InfoMiner 算法以投影子序列为基础，通过不断统计序列数据中相关事件的重复次数来计算候选模式的信息增益值，从而发现信息增益大于给定阈值的意外模式。在挖掘周期为 period 的模式时，在第 i 层递归中，将统计 period-i 个位置的事件的重复次数。InfoMiner 算法主要通过图 2 所示的流程来实现。

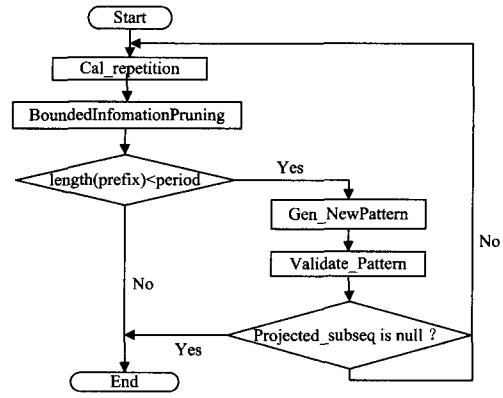


图 2 InfoMiner 算法流程图

其算法描述如图 3 所示，算法入口将传入两个参数，一个是模式的前缀（prefix），另外一个是前缀的投影子序列（items），随着算法不断进行，前缀长度不断增加，当达到模式周期长度（period）或者投影子序列为空时停止。Projected_Subsequence 用于计算某前缀的投影子序列，Cal_Repetition 中扫描投影子序列，依次记录每个事件出现的次数。BoundedInfomationPruning 通过计算序列数据中以给定前缀开头的最大信息量和最小重复次数 min_rep，来排除掉小于 min_rep 的候选事件。Validate_Pattern 中只需要用候选模式的信息量乘以重复次数得到信息增益，再与给定的信息增益阈值相比较即可。

```
Algorithm:info_Mining
Input:前缀:prefix,prefix 的投影子序列:items
Output:满足条件的序列模式
info_Mining(prefix,items)
  //根据前缀 prefix 在 items 中计算出重复次数
  repetition= cal_Repetition(prefix,items)
  //根据边界信息增益性质裁剪掉不符合条件的事件元素
  BoundedInfomationPruning(prefix,items,repetition)
  for i ← 0..period-length(prefix)
  begin
    for each candidate event j in the pruned repetition[i] //裁剪后选候选事件
      begin
        newPrefix = prefix + ['*']*i + j //添加新的候选事件生成新前缀
        //在得到的新前缀后面添加'*'构成长度为 period 的候选模式
        candPattern = gen_Candidate_pattern
        //根据重复次数计算信息增益,验证候选模式
        Validate_Pattern(candPattern,repetition[i][j])
        newItems = Projected_Subsequence(items,newPrefix)//算投影子序列
        if ∃newItems //如果存在新项生成,递归产生新模式
        then info_Mining(newPrefix,newItems)
      end
  end
```

图 3 InfoMiner 算法

以序列 S=⟨1,3,4,5,1,4,3,3,2,6,3,2,1,4,3,3,1,3,3,5,1,3,4,5,2,3,3,5,1,3,4,5,2,6,5,2,2,6,2,2⟩为例，考虑周期 l=4，计算以⟨2⟩为前缀的投影子序列，得到{(2,6,3,

2),(2,3,3,5),(2,6,5,2),(2,6,2,2)}。

如图 4 所示,扫描一遍此投影子序列,例如候选模式中第 2 个位置有事件 6 和 3,支持度分别为 3 和 1,则重复次数 *repetition* 分别是 2 和 0。

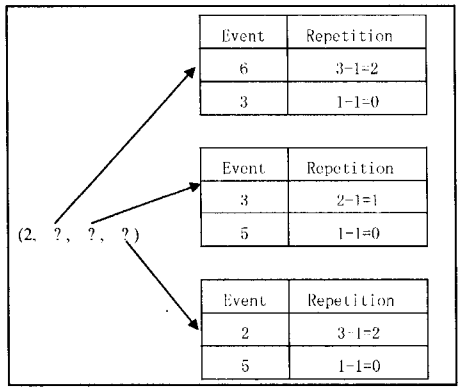


图 4　前缀(2)的投影子序列重复次数

上例中假设阈值 $Min\_Gain=4.5$,首先计算 Max_Info。从表 1 可以看出,投影子序列(2,6,5,2)的信息量最大,为 4.301。

表 1　前缀(2)的投影子序列信息量

投影子序列	信息量
(2,6,3,2)	0.898(2+0.672+1.446=3.914
(2,3,3,5)	0.898+0.672(2+1.059=3.301
(2,6,5,2)	0.898(2+1.059+1.446=4.301
(2,6,2,2)	0.898(3+1.446=4.14

由 $Min\_Rep=\lceil Min\_Gain/Max\_Info \rceil$ 有 $Min\_Rep=2$,得到满足该条件的两个候选模式(2,6,*,*)和(2,*,*,2),再进行下一步的模式验证阶段。经过此步的计算,还得到了新前缀(2,6),其投影子序列为{(2,6,3,2),(2,6,5,2),(2,6,2,2)},同理进行 Cal_repetition 处理,得到另一个候选模式(2,6,*,2),同样将进行模式验证阶段。

前面已经得到了以(2)为前缀的两个候选模式(2,6,*,*)和(2,*,*,2),加上以(2,6)为前缀的候选模式(2,6,*,2),这 3 个候选模式的 *repetition* 值都是 2。计算候选模式(2,6,*,*)的信息增益 $Info\_Gain(2,6,*,*)=(Info(2)+Info(6))\times 2=(0.898+1.446)\times 2=4.688>Min\_Gain=4.5$。因此(2,6,*,*)为满足条件的序列模式,同理可得(2,6,*,2)满足条件,(2,*,*,2)不满足条件,予以排除。

2.2　STAMP 挖掘算法

大多数以统计为理论基础来挖掘序列模式的算法,都只考虑到模式出现的次数,而忽略了其出现的位置,这就影响了其他完全匹配模式的公平性。STAMP 算法不仅弥补了这一缺陷,而且能挖掘序列中一个子集(子序列)的显著模式。

STAMP 算法的实现主要是通过 Cal_OIS 模块、GenCandE、Counting_MIG、Gen_Pattern 模块以及 Validate_Pattern 模块。

(1)Cal_OIS 模块

OIS 即 Optimal Information Surplus,计算出 OIS 的值是整个算法的基础,下面给出其计算方法。

假设 $infoloss(a_k,l,j)$ 代表事件 a_k 在序列的位置 j 处的信息损失,这个值与连续两个相同事件之间的距离(*distance*)有关,计算方法如式(1)所示(其中 l 表示周期长度)。

$$infoloss=\left\lceil \frac{(distance-l)}{l} \right\rceil \times Info(a_k) \tag{1}$$

与之对应的 $infogain(a_k,l,j)$ 代表事件 a_k 在序列的位置 j 处的信息增益,因为每一次事件 a_k 的出现都对整个综合信息增益有所贡献,即 *infoGain* 的值就等于 a_k 的信息量。综合信息增益值就可以用 *infoGain* 减去 *infoloss* 得到。OIS 取值如式(2)所示,其中,$f(a_k,l,0)=0$。

$$\begin{cases} OIS(a_k,l,j+1)=\max\{0,f(a_k,l,j+1)-Info(a_k)\} \\ f(a_k,l,j+1)=\max(0,f(a_k,l,j)-infoloss(a_k,l,j+1))+infogain(a_k,l,j+1) \end{cases} \tag{2}$$

例如在序列 $S=\langle 1,2,7,4,9,2,4,2,2,4,9,7,4,2,2,6,9,2,4,2,1,4,9,7,6,6,2\rangle$ 中,不妨设 $Min\_GIG=2.1$,其事件的 *Information* 值分别为:$Info(1)=1.45$,$Info(2)=0.61$,$Info(4)=0.84$,$Info(7)=1.23$,$Info(6)=1.23$,$Info(9)=1.07$。则事件 2 的 OIS 值计算如表 2 所列(表 2 中 x 代表 *infoloss*,y 代表 *infogain*)。

表 2　事件 2 的 OIS 值(周期长度 $l=3$)

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
event	1	2	7	4	9	2	4	2	2	4	9	7	4	2	2	6	9	2	4	2	1	4	9	7	6	6	2
			distance=4						distance=5																distance=7		
$x(2,3,j)$						-0.61									-0.61												-1.22
$y(2,3,j)$			0.61			0.61		0.61	0.61						0.61	0.61			0.61		0.61						0.61
$f(2,3,j)$			0.61			0.61		1.22	1.83						1.83	2.44			3.05		3.66						3.05
$OIS(2,3,j)$			0			0		0.61	1.22						1.22	1.83			2.44		3.05						2.44

(2)GenCandE 模块

Gen_CandE,即生成候选事件,将采取下面的计算过程。

在上一步中已经得到了 OIS 的值,对于周期长度为 l,假设 E_l 表示出现在有效模式 P 中的所有事件的集合,有效的定义即其模式的综合信息增益值 *GIG* 大于给定的阈值。问题转换成求出包含在模式 P 中的所有事件,其 OIS 值之和大于给定阈值。E_l 的计算方法如下。

假设 $T(l,s)$ 表示在第 s 段(序列以周期长度 l 分段)结束之前包含的 OIS 为正的事件集合,即 $T(l,s)=\{a_k|OIS(a_k,l,j)>0\}$,其中 j 是在 s 段之前 ak 最后一次出现的位置。例如上例中,所有事件 OIS 值如表 3 所列,在第 3 段(*position*=7,8,9)中,$T(3,3)=\{2,4\}$,一个事件的 OIS 值在某一段中可能不存在也可能被改变了多次,本文的研究中,总是取离该段结束最近的一次作为结果。例如表 3 中的第 3 段,$OIS(2)=1.22$,第 4 段 $OIS(2)$ 的值也为 1.22,都是取的 *position*=9 处的值。

表 3　候选事件产生 $l=3$,$Min\_GIG=2.1$

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
event	1	2	7	4	9	2	4	2	2	4	9	7	4	2	2	6	9	2	4	2	1	4	9	7	6	6	2
seg	seg1			seg2			seg3			seg4			seg5			seg6			seg7			seg8			seg9		
OIS	1	0																			0						
	2		0			0	0.61	1.22					1.22	1.83			2.44		3.05								2.43
	4			0			0.84			1.68			2.52							2.52			3.36				
	6															0										0	1.23
	7		0								0													0			
	9			0					0								0							0			
T(3, seg)							{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2, 6}	
E(3, seg)							{}			{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2, 6}	
P(3							{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2}			{4, 2, 6}	

$E(l,s)$ 表示在第 s 段结束之前所有事件的集合,其中这些事件的 OIS 值之和必须大于给定的综合信息增益阈值。例如在表 3 中,E(3,3)={},因为在第 3 段中,$OIS(4)+OIS(2)=2.06<Min\_GIG=2.1$。同理,因为在第 4 段中,$OIS(4)+OIS(2)=2.70$,因此 $E(3,4)=\{4,2\}$。

取所有 $E(l,s)$ 的并集即可得到 E_s。

(3)Counting_MIG 模块

这一步即是通过扫描序列计算包含第(2)步产生的候选事件的单一模式中的最大信息增益(Maximum Information Gain)。其计算方法如式(3)所示。

$$\begin{cases} b[j+1]=\max\{0,b[j]\}+temp[j] \\ c[j+1]=\max\{b[j+1],c[j]\} \end{cases} \tag{3}$$

其中，$temp[j]$为单一模式在第 j 段的综合信息增益，如果匹配，则取 $Info(e)$，否则取 $-Info(e)$。通过初始化 $b[0]=c[0]=0$，扫描一遍序列即可得到。MIG 的值定义为 $MIG=\max\{0,c[j]-Info(e)\}$，$j$ 为序列中的最后一段。

例如上例中，候选事件 2、4、6 在周期中每个位置的 MIG 值分别如表 4 所列。

表 4　MIG 值

event	1	2	3
2	0	0	1.22
4	3.36	0	0
6	0	0	0

(4)Gen_Pattern 模块

这一步将利用第(3)步计算出的每个候选事件最大信息增益值生成候选模式。例如在单一模式$(*,\cdots,p_j,\cdots,*)$在 j 处的 MIG 值为 $MIG(p_j)$，那么复杂模式 $P=(p_1,p_2,\cdots,p_l)$的 $MIG(P)$的值就是各个单一模式累加。Min_GIG 为给定的综合信息增益阈值。当 P 满足式(4)所示的条件时，P 就为候选模式，将进入下一个阶段的模式验证。

$$\sum_{j=1}^{j=l} MIG(P_j)\geqslant Min\_GIG \tag{4}$$

以上例为准，穷举表 4 所得到的“候选”模式$(4,*,2)$，上例中，长度为 3，在 $position=2$ 处，没有一个事件的 GIG 大于 0，即用 * 代替。其 $MIG(4,*,2)>Min\_GIG$，现在$(4,*,2)$成了真正意义的候选模式，将进行下一个阶段的模式验证。

(5)Validate_Pattern 模块

在此阶段扫描序列，并计算子序列的综合信息增益值，如果大于给定的阈值，则该模式为所求。如图 5 所示，在验证候选模式 $P=(4,*,2)$时，存在子序列 $S'=\langle 1,2,7,4,9,2,4,2,2,4,9,7,4,2,2\rangle$，使得 $GIG(4,*,2)=-(Info(4)+Info(2))+(Info(P))+(Info(P))-(Info(2))+(Info(P))=2\times Info(4)+Info(2)=2.29>Min\_GIG=2.1$。

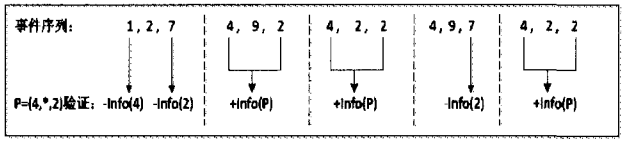


图 5　候选模式$(4,*,2)$的验证

因此候选模式$(4,*,2)$为其中一个满足条件的模式。

3　实验结果对比分析

本实验采用的数据集来源于 kdd09 中的文献[11]所提供的数据，其数据可以从 http://www.mysmu.edu/faculty/davidlo/kdd09.htm 下载得到，数据被分为两类且各类约占总体数据的 50%。下面将从序列模式挖掘结果和分类预测准确率两个方面对比 InfoMiner 算法和 STAMP 算法。

3.1　序列模式挖掘结果对比

从挖掘模式数量上来看，对 InfoMiner 算法在给定阈值下挖掘的模式数量随着周期的增加而增加，当达到某个峰值后又会随周期增加而减少，而 STAMP 算法挖掘的模式数量较稳定。如图 6 所示，数据集采用 schedule/f1，InfoMiner 算法信息增益阈值 Min_Gain 设置为 5000，STAMP 综合信息增益阈值设置为 0。

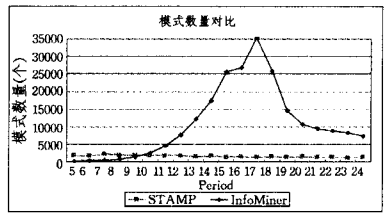
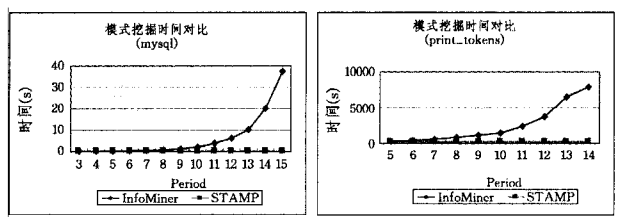


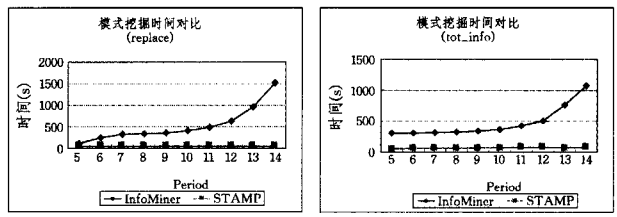
图 6　模式挖掘数量结果对比

从挖掘模式所花费的时间上讲，InfoMiner 算法中，其挖掘模式所耗费的时间随着周期的增加呈指数增长趋势，而 STAMP 算法挖掘所耗费的时间与周期没有明确的关系。

如图 7 所示，所取数据集是各个数据集 f1－f5 的平均结果，其中对于 mysql 数据集，InfoMiner 算法中 topK 值为 100，其他数据集 topK 值取 500，STAMP 算法中综合信息增益阈值都设置为 0。在数据集 mysql 中，因其序列长度不是太长，当模式的周期长度增加到 14 时，InfoMiner 算法也能在 40s 之内挖掘出所有模式，当数据集长度太长时，如图 7(b)所示，数据集 print_tokens 的序列长度最长达到了 5490，而练集中序列数目也达到了 4974，InfoMiner 算法挖掘模式所耗费的时间将近 2 小时。不管针对哪个数据集，STAMP 算法挖掘模式的时间相对都比较稳定，不会因为模式周期增加而发生明显的波动。



(a) 模式挖掘对比(dataset=mysql)　(b) 模式挖掘对比(dataset=print_tokens)



(c) 模式挖掘对比(dataset=replace)　(d) 模式挖掘对比(dataset=tot_info)

图 7　模式挖掘时间结果对比

3.2　分类预测结果对比

从整体来看，对 InfoMiner 算法挖掘出的模式进行分类预测取得的最高准确率要高于 STAMP 算法，但随着周期的变化，结果波动较大，而 STAMP 算法则比较平稳。以数据集西门子 tot_info/f3，f4，f5 作为实验对象，采用卡方检验过滤特征值，按照 4 种分类方法进行实验，其结果取均值，如图 8 所示。当周期较短的时候，InfoMiner 算法的准确率较高，随着周期的增加，其准确率会减少；而 STAMP 算法相对比较稳

(下转第 188 页)

2007,17(41):125-127

[8] Ushby J. Noninterference, Transitivity, and Channel-Control Security Policies[R]. Computer Science Laboratory, SRI International, 2005

[9] Haigh J T, Yong W D. Extending the noninterference model of MLS for SAT [J]. IEEE Transaction On Software Engineering, 1987,2(13):141-150

[10] Goguen J A, Meseguer J. Security policies and security models [C]// Proc. of the 1982 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, April 1982:11-20

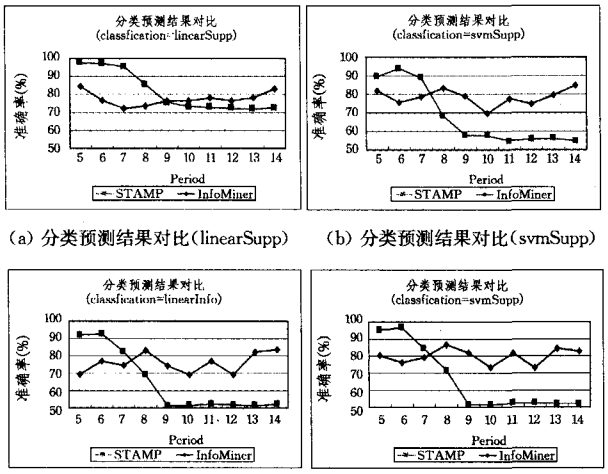
[11] 赵佳,沈昌祥,刘吉强,等. 基于无干扰理论的可信链模型[J]. 计算机研究与发展,2008,45(6):974-980

[12] 张帆,陈曙,桑永宣,等. 完整性条件下无干扰模型[J]. 通信学报,2011,32(10):11-19

[13] Bellard F. QEMU documentation [OL]. http://wiki.qemu.org/Main_Page,2011

(上接第 167 页)

定,会在一个较小的区间内波动。



(a) 分类预测结果对比(linearSupp) (b) 分类预测结果对比(svmSupp)

(c) 分类预测结果对比(linearInfo) (d) 分类预测结果对比(svmInfo)

图 8 分类预测结果对比

对所有数据集进行实验,InfoMiner 算法分类结果、STAMP 算法分类结果以及 KDD09 原始分类结果的最终对比如图 9 所示。

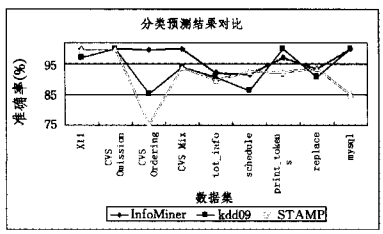


图 9 分类预测结果对比(InfoMiner,STAMP,KDD09)

从图 9 中可以看出,InfoMiner 算法的预测准确率高于 STAMP 和 KDD09,对于序列长度相对较短的模拟数据集 mysql,InfoMiner 算法的预测准确率能达到 100%;对于长序列数据集 replace 和 schedule,STAMP 算法结果与 InfoMiner 持平,且都高于 KDD09,而其他数据集 STAMP 结果是 3 种算法最差的。

实验结果表明,InfoMiner 和 STAMP 算法在挖掘模式上各具特点,InfoMiner 算法随着周期的增加,所耗费时间呈指数增长,而 STAMP 算法比较稳定。InfoMiner 算法挖掘出模式能够良好体现序列数据的特征,虽然挖掘模式的效率比较低,但其分类预测的结果很好,几乎超过了文献[11]的结果;而 STAMP 算法主要专注于序列数据中的部分周期模式,因此这方面其预测效果比 InfoMiner 差。

结束语 本文在研究了基于统计策略的两种算法 InfoMiner 和 STAMP 算法的基础上,运用卡方检验来测试模式与序列之间的相关度,并用支持度或信息增益值作为描述序列数据的特征,利用基于软件系统的历史数据和已知缺陷数据,构造了一个对已知缺陷进行分类的分类器,并以此来预测发现未知的软件缺陷。

从实验结果可以看出,利用 SVM 分类器可以取得较好的准确率,但在其训练过程所耗费的时间也占用了较大的比例。因此本文的进一步研究可以从改进序列模式挖掘算法效率以及改进分类算法效率或者利用其他分类算法两方面着手。

参 考 文 献

[1] Agrawal R, Srikant R. Mining sequential patterns [C]// Proceedings of the Eleventh International Conference on Data Engineering. Washington DC, USA: IEEE Computer Society, 1995: 3-14

[2] Chen Yuan, Shen Xiang-heng, Du Peng, et al. Research on Software Defect Prediction Based on Data Mining[C]// The 2nd International Conference on Computer and Automation Engineering. Singapore: ICCAE, 2010:563-567

[3] Yang Jiong, Wang Wei, Yu P S. Infominer: mining surprising periodic patterns [C]// Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'01). New York, USA: ACM, 2001:395-400

[4] Yang Jiong, Wang Wei, Yu P S. STAMP: on discovery of statistically important pattern repeats in long sequential data[C]// Proceedings of the Third SIAM International Conference on Data Mining(SDM'03). San Francisco, CA, USA: SIAM, 2003: 224-238

[5] 张小康. 基于数据挖掘和机器学习的恶意代码检测技术研究[D]. 合肥:中国科学技术大学,2009

[6] 周聚. 基于网络信息审计的文本过滤的研究与实现[D]. 苏州:苏州大学,2010

[7] 杨明,张载鸿. 决策树学习算法 ID3 的研究[J]. 微机发展,2002,12(5):6-9

[8] Quinlan J R. C4.5: Programs for Machine Learning[M]. San Francisco: Morgan Kaufmann Publishers, 1993

[9] 眭俊明,姜远,周志华,等. 基于频繁项集挖掘的贝叶斯分类算法[J]. 计算机研究与发展,2007,44(8):1293-1300

[10] Han Jia-wei, Kamber M. Data Mining: Concepts and Techniques [M]. San Francisco: Morgan Kaufmann Publishers, 2006

[11] Lo D, Cheng Hong, Han Jia-wei, et al. Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach [C]// Proceedings of the 15th ACM SIGKDD (KDD'09). New York, USA: ACM, 2009:557-565