软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(6):1169−1179 [doi: 10.13328/j.cnki.jos.004430]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于高斯过程的缺陷定位方法*

陈理国, 刘 超

(北京航空航天大学 计算机学院,北京 100191)

通讯作者: 陈理国, E-mail: netclg@gmail.com, http://www.sei.buaa.edu.cn

摘 要: 在软件系统中,缺陷定位是缺陷修复的一个关键环节,如果能将缺陷自动定位到很小的范围,将会极大地降低缺陷修复的难度.基于高斯过程提出了一种缺陷定位方法(GPBL),即针对每个缺陷,向开发人员推荐这个缺陷可能存在于哪些源文件中,从而帮助开发人员快速修复缺陷.为了验证方法的有效性,采集了开源软件 Eclipse 和 Argouml 中的数据,实验结果表明,高斯过程缺陷定位的查全率和查准率平均分别为 87.16%和 78.90%.与基于 LDA 的缺陷定位方法进行比较,表明高斯过程更能准确定位缺陷的位置.

关键词: 缺陷定位;缺陷修复;缺陷报告;推荐方法;高斯过程

中图法分类号: TP311

中文引用格式: 陈理国,刘超.基于高斯过程的缺陷定位方法.软件学报,2014,25(6):1169−1179. http://www.jos.org.cn/1000-9825/4430.htm

英文引用格式: Chen LG, Liu C. Bug localization method based on Gaussian processes. Ruan Jian Xue Bao/Journal of Software, 2014,25(6):1169−1179 (in Chinese). http://www.jos.org.cn/1000-9825/4430.htm

Bug Localization Method Based on Gaussian Processes

CHEN Li-Guo, LIU Chao

(School of Computer Science and Engineering, BeiHang University, Beijing 100191, China)

Corresponding author: CHEN Li-Guo, E-mail: netclg@gmail.com, http://www.sei.buaa.edu.cn

Abstract: In software systems, bug localization is a key step in the bug fix process. By automatically narrowing down potential bug locations, the difficulty of bug fix is greatly reduced. In this paper, a bug localization method based on Gaussian processes, called Gaussian processes bug localization (GPBL) is proposed. This method can facilitate fixing bugs for the developers, by recommending source files that may contain bugs. In order to evaluate GPBL, the open-source software Eclipse and Argouml are employed as data sources. Experimental results show that GPBL can achieve 87.16% recall and 78.90% precision on average. In addition, GPBL can locate relevant buggy files more accurately compared with LDA-based bug localization methods.

Key words: bug localization; bug fix; bug report; recommendation method; Gaussian processes

为了提高软件的质量,通常通过测试尽可能地发现软件缺陷,当发现软件缺陷后,开发人员需要对软件进行调试,以发现缺陷存在的位置,这就是缺陷定位问题.开发人员为了发现缺陷存在的位置,一般通过设置断点、查看程序执行轨迹以及查看变量值等方法,然而,这些方法会消耗开发人员的大量时间.

为了能够帮助开发人员快速而又准确地发现缺陷存在的位置,在软件工程研究领域,一些研究人员提出了自动定位缺陷的方法[1−4],这些动态的缺陷定位方法需要测试用例集,然后应用测试用例执行程序,在程序运行过程中收集程序执行的信息,并通过程序执行信息来判断哪些程序语句可能存在缺陷.动态的缺陷定位方法严重依赖于测试用例集,在开源软件中往往没有测试用例集,而开源软件中存在缺陷报告、代码库.缺陷报告详细记录了历史缺陷的信息,代码库中记录了代码的变化历史.一些研究者使用信息检索的方法将缺陷作为查询,代

* 基金项目: 国家高技术研究发展计划(863)(2007AA010302)
收稿时间: 2012-11-16; 定稿时间: 2013-03-07

码作为检索的对象,计算缺陷与代码的相关性,与缺陷相关性越大的代码存在缺陷的可能性越大,通过这个方法来定位缺陷可能存在的位置.然而,由于缺陷通常是用自然语言描述,而代码是用程序语言描述,它们的语言空间不一样,用信息检索的方法来定位缺陷的效果往往不理想.

通过分析开源软件中缺陷库与代码提交日志,可以提取缺陷修复所对应修改的源文件关系,通过缺陷与源文件这种链接关系,可以建立缺陷与源文件之间的关系图.那么,缺陷的定位就对应预测缺陷与源文件之间的链接预测问题.在机器学习领域,高斯过程是链接预测问题的一种有效方法[5−7],基于高斯过程,我们提出了一种高斯过程缺陷定位方法(Gaussian processes bug localization,简称 GPBL).通过此方法,可以预测新产生的缺陷可能发生在哪些文件中.该方法不同于动态的缺陷定位方法,不需要程序的测试用例,也不用收集程序的执行信息,而且还可以准确地定位缺陷发生的位置.

由于高斯过程计算复杂度高,模型的训练速度慢.为了解决这个问题,本文针对高斯过程推理过程中涉及的矩阵计算对矩阵进行变换,降低了计算复杂度.另外,原始的实验数据中,正负样本严重不均衡,导致高斯过程预测效果差.针对这个问题,提出了一种数据采集算法.总之,本文的主要贡献包括:

(1) 提出了一种数据采集的方法;

(2) 提出了一种降低高斯过程推理过程中计算复杂度的方法;

(3) 描述基于高斯过程的缺陷定位方法的推理过程.

本文第 1 节介绍缺陷定位等相关工作.第 2 节首先描述方法的处理流程,然后对每个处理部分进行介绍,其中详细介绍高斯过程在缺陷定位中的应用、对高斯过程的推理过程以及算法进行详细的介绍.第 3 节是实验部分,使用开源软件 Eclipse 和 Argouml 中的数据,实验结果表明了基于高斯过程的缺陷定位方法的有效性,并对实验结果进行分析.最后总结全文并讨论进一步的研究工作.

1 相关工作

缺陷的定位是软件工程领域中被广泛研究的主题,其大致可分为动态和静态的方法,动态方法使用预先设计好的测试用例对程序进行测试,在程序运行过程中收集程序的执行信息,通过收集的信息来判断缺陷可能存在的位置.近年来,研究者们提出了基于统计的缺陷定位方法,Liblit 等人[1]提出 CBI 技术用于缺陷定位,通过有选择地收集软件执行信息,分析计算谓词与程序崩溃的相关性,用相关性判断缺陷可能发生的位置.Liu 等人[2]提出了 SOBER 方法用于缺陷定位,这种方法与 CBI 方法不同,它不是通过选择与程序失效的谓词来做缺陷定位,而是通过谓词在程序正确执行与错误执行中出现的不同模式进行缺陷的定位.周吴杰等人[3,4]提出了基于组合测试的缺陷定位方法.这些缺陷定位的动态技术效果依赖于测试用例,计算复杂度高,且定位的准确性还有待提高.

静态的缺陷定位多采用信息检索的方法,通过分析代码和缺陷报告的特征,用这些特征计算得出缺陷与代码的相关性.Marcus 等人[8]首次用信息检索中的潜在语义索引(latent sematic indexing,简称 LSI)模型做缺陷定位,利用余弦相似度计算缺陷与代码之间的相关性.Gregory 等人[9]使用向量空间模型计算缺陷与代码之间的相似度,通过用户对相似度列表的反馈信息来修正缺陷与代码的之间相似性,在 Eclipse 上的实验表明:由于反馈信息的加入,可以提高查全率 5%.后来,Lukins 等人[10,11]将潜在狄利克雷分配模型(latent Dirichlet allocation,简称 LDA)应用到缺陷定位中.LDA 模型将一个文档表示为多个主题的分布,而主题由多个词的分布组成,将缺陷报告作为查询,而将代码文件作为检索的对象,通过计算缺陷报告与代码的相关性来定位缺陷.在文献[11]中的实验结果表明:在缺陷定位上,LDA 模型优于 LSI 模型.近年来,研究者们利用缺陷之间的相似性来提高缺陷定位的效果.因为相似的缺陷可能出现在同一个位置,Zhou 等人[12]结合缺陷之间的相似性和缺陷与文件之间的相似性,将缺陷定位到文件级.在 Eclipse 上的实验结果表明:利用缺陷之间的相似性,可以提高查全率 7%左右.Davies 等人[13]利用缺陷之间的相似性提高基于信息检索方法的缺陷定位方法的效果,在开源软件 Argouml 上的实验结果表明:利用缺陷之间的相似性,可以提高缺陷定位的查全率大约 5%.

与动态的缺陷定位方法不同,静态的方法不需要测试用例,可以使用开源软件数据来检验方法的有效性.静

态的方法通常定位到代码文件级或者是程序方法级.基于高斯过程的缺陷定位方法属于缺陷定位的静态方法,将缺陷定位到文件级,即向缺陷修复人员推荐缺陷可能存在哪些文件中,缺陷修复人员通过重点关注这几个源文件,可以较快地诊断出缺陷产生的原因.将缺陷定位到文件级也是合理的,因为缺陷的修复通常不仅仅只关注缺陷所在的位置,而且实验结果表明,我们的方法能够较为准确地将缺陷定位到文件级.

2 基于高斯过程的缺陷定位方法

如图 1 所示,方法处理流程分为 3 部分:预处理模块、模型的训练模块、预测模块.下面详细描述各个部分的功能.

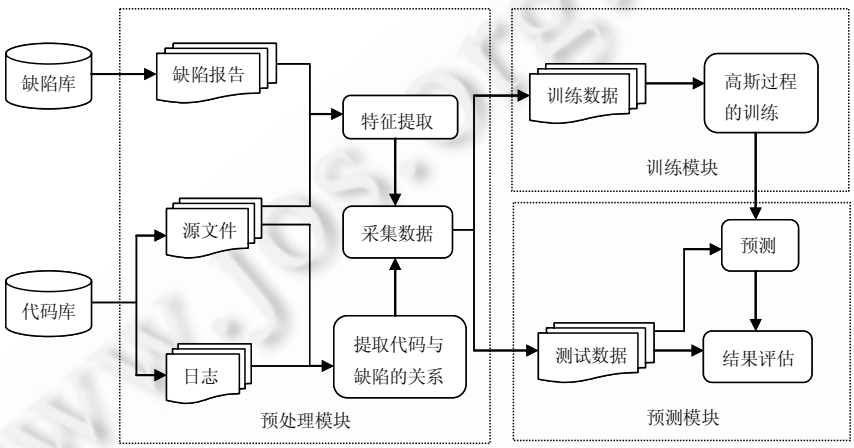

Fig.1 Flow of process

图 1 处理流程

2.1 预处理模块

预处理模块的功能是从缺陷库和代码库中提取模型的输入数据,包括:提取缺陷与源文件之间的关系、缺陷报告和源文件特征的提取以及采集数据.

2.1.1 缺陷与源文件关系的提取

定义 1(缺陷与源文件的关系图). 将每个缺陷与源文件抽象成图中的节点,二分图 $G=(U,V,E)$表示缺陷与源文件的关系,其中,U 表示缺陷节点集,V 表示源文件节点集,E 表示边集.如果缺陷 u 的修复需要修改文件 v,则在 G 中存在一条无向边(u,v).

与缺陷修复相关的代码变更的提交日志中会出现 fix,defect,change 等词,通过正则表达式匹配的方法提取这些关键词紧接后面的缺陷 ID 号,这样就知道哪些缺陷的修复对应修改了哪些文件.通过这些信息,可以建立缺陷与文件的关系图.如图 2 所示,缺陷 u_1 的修复同时修改了文件 v_1,v_3,v_4,而缺陷 u_2 的修复只修改了文件 v_1.有些缺陷的修复对应修改哪些文件的信息,通过正则表达式匹配的方法是不能获取的,一些研究者除了使用关键字匹配,还通过一些启发式规则分析缺陷与代码提交日志的相似性、变更的代码、代码变更的时间以及缺陷修复的时间[14]等信息来提取缺陷修复对应的代码变更.这些方法不可避免会错误地引入缺陷与文件之间的关系,为了保证所建立的缺陷与文件关系的正确性,只采用关键字匹配的方法.

2.1.2 缺陷报告和代码文件的特征提取

缺陷报告通常被存储到缺陷库中(如 bugzilla),在缺陷库中记录了缺陷的一些特征信息,在 bugzilla 中,每个缺陷报告包括缺陷的 id 号、缺陷的标题、缺陷的描述以及缺陷属于哪个模块等,这些信息都是用自然语言描述的文本信息,通过对这些文本信息进行分词、大小写转换、去掉词干、去掉停用词后,将每个缺陷报告表示为一个向量$(w_1,w_2,\ldots,w_m)$,向量中的每个分量 w_i 表示第 i 个词在该缺陷报告中占的权重,其计算方法采用信息检

索中常用的 tf-idf(term frequency-inverse document frequency)计算词权重的方法[15],具体计算见公式(1):

$$w_i = tf_i \times idf_i, tf_i = \frac{N_i}{N_{total}}, idf_i = \log\frac{D_{total}}{D_{w_i}} \tag{1}$$

其中,N_i,N_{total} 分别表示该缺陷报告中第 i 个词出现的次数和词的总数,D_{total}, D_{w_i} 分别表示缺陷报告的总数和第 i 个词在多少个缺陷报告中出现过.

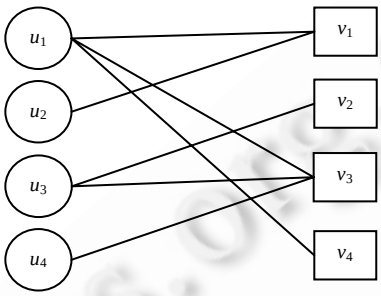

Fig.2 Relationship graph between bugs and source files

图 2 缺陷与文件之间的关系图

代码文件的特征提取与缺陷报告特征提取类似,通过访问代码的抽象语法树,首先提取代码文件中的标识符,包括类名、方法名以及调用的方法名.然后对这些标识符进行分割,例如:将类名 CompilationProgress 分割为两个词 Compilation 和 Progress.最后进行大小写转换,去掉词干,去掉停用词.对代码标识符处理之后,同样将每个代码文件表示为一个词向量,而向量中每个分量也是通过公式(1)计算得到.代码文件的特征提取时,没有提取代码的注释,因为提取了这些信息会导致代码的特征向量的维度显著提高,导致模型的训练时间长,而且注释是用自然语言描述,它的语言空间与代码的语言空间不同,实验结果表明:没有利用代码注释,模型的预测效果也相当好.

2.1.3 采集样本

通过第 2.1.1 节和第 2.1.2 节的方法提取了缺陷与文件的关系图 $G=(U,V,E)$以及图中每个节点表示的特征向量,通过这个关系图提取模型的训练样本集,每个样本表示为(u,v,y),其中,$u\in U$,$v\in V$.如果节点 u 和 v 之间有边相连,则 $y=+1$;否则,$y=-1$.当 $y=+1$ 时,这个样本称为正样本,否则为负样本.在实际的软件系统中,每个缺陷的修复只会修改几个源文件,分析 eclipse 中的数据表明,大多数缺陷的修复只会改动 1~2 个源文件,从而导致训练样本集中负样本数据过多.分析 eclipse 中 jdt.core 这个组件表明,负样本数占总样本数 90%以上.在这样正负样本严重不均衡的情况下,训练得出的模型会偏向于负样本,从而极大地影响模型的预测精度.可以通过随机采集一部分负样本,使得最后训练集中的正样本与负样本的个数均衡.然而,不同的样本对模型训练的贡献不一样,例如,有的源代码文件实现单一而又简单的功能,这类源代码出现缺陷的可能性非常小,几乎可以肯定是没有缺陷的.与这类源文件节点相应的负样本对模型的训练没有任何帮助,而对于那些功能复杂的源文件,是否存在缺陷是不确定的,将与这类源文件相应的负样本纳入训练集会有利于模型的训练.正如文献[16]所述,当一个样本的类别越模糊,其真实的类别对模型的训练越有帮助.鉴于此,提出了一种采集样本的方法——优先采集算法,使得样本集中的正负样本是均等的.实验结果表明:在采集后的样本集上训练模型是合理的,具体的采集样本的方法见算法 1 和算法 2.

算法 1. 优先采集算法.

输入:缺陷与源代码文件的关系图 $G=(U,V,E)$,

//U 为缺陷节点集,V 为源文件节点集,

//E 为边集;

输出:采集的样本集 D.

```
1.  初始化集合 S1={}, S2={}
2.  for each u∈U do
3.   for each v∈V do
4.     if (u,v)∈E then
5.       S1←(u,v,+1)   //正样本
6.     else
7.       S2←(u,v,−1)   //负样本
8.     end if
9.     //将正样本添加到样本集中
10.    D←S1
11.    //正样本的个数
12.    n=size(S1)
13.    D←prioritySelect(S2,n)   //见算法 2
14.  end for
15.  S1={}, S2={}
16. end for
17. return D
```

算法 2. 优先选择(prioritySelect).

输入:样本集 *S*,选择样本的个数 *n*;

输出:采集的样本集 *D*.

```
1.  初始化集合 D={}, degreeSum=0, wheel={}
2.  for each (u,v)∈S do
3.   degreeSum+=degree(v)   //将文件节点的度累加
4.  end for
5.  for each (u,v)∈S do
6.    i=0
7.    wheel[i]=degree(v)/degreeSum
8.    i++
9.  end for
10. for i=1:n do
11.   r=rand()   //产生 0~1 之间的随机数
12.   for j=1:length(wheel) do
13.     if r<wheel[j] then
14.       D←S[j]
15.       break
16.     end if
17.   end for
18. end for
19. return D
```

2.2 训练模块

训练模块的功能是以训练样本作为输入对高斯过程进行推理,这部分是方法的核心.本节详细给出了高斯过程的推理过程,并针对推理过程的时间复杂度高的问题提出了解决方法.

通过第 2.1 节的方法采集的数据集中,一部分数据作为训练样本,假设训练样本集为

$$D=\{(u_i,v_j,y_{ij})|i=1,2,\ldots,m;j=1,2,\ldots,n\},$$

其中,样本输入数据 $u_i\in U=R^{D_1}$,$v_j\in V=R^{D_2}$,样本输出数据集 $Y=\{y_{ij}|y_{ij}=\pm 1\}$.

在高斯过程模型中,假设存在一个隐含函数 $f:(U\times V)\to R$,且 $F=\{f(u_i,v_j)\}_{i=1,2,\ldots,m;j=1,2,\ldots,n}$ 服从均值为 0、协方差为 $\Sigma\otimes\Omega$ 的高斯过程 $GP(0,\Sigma\otimes\Omega)$,其中,$\Sigma:U\times U\to R$ 为集合 U 上的协方差函数(也称为核函数),$\Omega:V\times V\to R$ 为集合 V 上的协方差函数.

为了方便起见,设置 $f_{ij}=f(u_i,v_j)$,相对于这个隐含函数的条件概率 $p(y_{ij}|f_{ij})=\Phi(y_{ij}f_{ij})$,$\Phi(\cdot)$通常为高斯累积函数或者是逻辑函数,它是预先设定的似然函数.在本文中,$\Phi(x)=\int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{\frac{(x-u)^2}{2\sigma^2}}\mathrm{d}x$ 为高斯累积函数.

基于高斯过程的缺陷定位的直观含义可以表述为:每个缺陷报告节点和源文件节点对,对应一个隐含函数值,隐含函数值完全决定了节点对之间是否有链接关系,且隐含函数值是服从高斯分布,且协方差函数与缺陷报告节点和源文件节点对应的向量相关.假设节点 u_1 与 u_2 相似,v_1 与 v_2 相似,则当 u_1 与 v_1 之间有链接,则 u_2 与 v_2 之间也可能有链接.

由于 $F\sim GP(0,\Sigma\otimes\Omega)$,则 F 的概率密度函数可表示为

$$p(F|U,V)=(2\pi)^{-\frac{mn}{2}}|\Sigma\otimes\Omega|^{-\frac{1}{2}}\exp\left[-\frac{1}{2}F(\Sigma\otimes\Omega)^{-1}F^T\right] \tag{2}$$

利用贝叶斯公式可得 F 的后验概率:

$$p(F|U,V,Y)=\frac{p(F|U,V)p(Y|F)}{p(Y|U,V)} \tag{3}$$

其中,$p(Y|U,V)=\int p(F|U,V)p(Y|F)\mathrm{d}F$.由训练样本独立可得:$p(Y|F)=\prod p(y_{ij}|f_{ij})$.

高斯过程的训练过程,是为了求 F 的后验概率 $p(F|U,V,Y)$,然而直接利用公式(3)求 F 的后验概率是不可计算的(intractable),可以用近似计算的方法来求这个后验概率,常用的高斯近似推理方法包括 Laplace 近似、变分贝叶斯以及期望传播(expectation propagation,简称 EP)[17].尽管在一些数据集上表明 EP 表现出较好的效果,也有一些研究者[5,7]认为,当后验概率 $p(F|U,V,Y)$ 是多峰时不能使用 Laplace 近似推理,我们依然采用 Laplace 近似推理.Laplace 近似具有推理过程简单、代码实现较容易等优点.实验结果也表明,使用 Laplace 方法是可行的,具体的推理过程如下:

Laplace 方法用一个高斯分布 $q(F|U,V,Y)$ 近似表示后验概率 $p(F|U,V,Y)$:

$$q(F|U,V,Y)=N(\hat{F},A^{-1}) \tag{4}$$

其中,$\hat{F}=\arg\max_F\log(p(F|U,V,Y))$,$A=-\nabla\nabla\log(p(F|U,V,Y))|_{F=\hat{F}}$.为了求 $\hat{F}$,对公式(3)两边取对数,得:

$$\log(p(F|U,V,Y))=\log(p(F|U,V))+\log(p(Y|F))-\log(p(Y|U,V)) \tag{5}$$

由于 $\log(p(Y|U,V))$ 的取值与 F 无关,为了最大化 $\log(p(F|U,V,Y))$,只需最大化:

$$\varphi(F)=\log(p(F|U,V))+\log(p(Y|F)) \tag{6}$$

结合公式(2),对公式(6)分别求关于 F 的一阶、二阶导数,得到:

$$\nabla\varphi(F)=-F(\Sigma\otimes\Omega)^{-1}+\nabla\log(p(Y|F))=-FK^{-1}+\nabla\log(p(Y|F)) \tag{7}$$

$$\nabla\nabla\varphi(F)=-K^{-1}+\nabla\nabla\log(p(Y|F)) \tag{8}$$

其中,

$$K=(\Sigma\otimes\Omega),W=-\nabla\nabla\log(p(Y|F)) \tag{9}$$

由 $p(Y|F)=\prod p(y_{ij}|f_{ij})$ 得 W 是一个对角矩阵,为了使 $\varphi(F)$ 取最大值,$\varphi(F)$ 的一阶导数必定为 0,所以由公式(7)得:

$$\hat{F}=K\nabla\log(p(Y|\hat{F})) \tag{10}$$

为了求公式(10)中的 $\hat{F}$,可以通过牛顿法进行求解,具体计算过程见算法 3.

算法 3. 用牛顿法求 $\hat{F}$.

输入:最大迭代次数 max*It*, $\hat{F}$ 的初始值 *initValue*;

输出: $\hat{F}$.

1. x_1=*initValue*, *i*=1
2. 依据公式(7)、公式(8)分别计算$\nabla\varphi(x_i)$,$\nabla\nabla\varphi(x_i)$
   //在求 K^{-1} 时,通过求$\Sigma^{-1}\otimes\Omega^{-1}$来减少计算量
3. 如果$\nabla\varphi(x_i)$=0 或者 *i*=max*It*,则 $\hat{F}=x_i$,停止
4. $x_{i+1}=x_i-\dfrac{\nabla\varphi(x_i)}{\nabla\nabla\varphi(x_i)}$
5. *i*=*i*+1,跳到步骤 2 执行

通过算法 3 求得 $\hat{F}$ 后,根据公式(9)可得 *W*,而 $q(F|U,V,Y)=N(\hat{F},(K^{-1}+W)^{-1})$,$(K^{-1}+W)$是一个 *mn*×*mn* 阶的矩阵,直接求这个矩阵的逆,计算复杂度为 $O(m^3n^3)$,而这个矩阵的逆决定了整个高斯过程模型训练的计算复杂度.为了解决这个问题,提出了一种方法,具体计算过程如下:

假定Σ,Ω均为线性核函数,即$\Sigma(i,j)=\langle u_i,u_j\rangle$,$\Omega(i,j)=\langle v_i,v_j\rangle$,在缺陷定位这个问题上,线性核函数是合理的,实验结果可以说明这一点.由于 $K=\Sigma\otimes\Omega$,那么 $K=GG^T$,其中,$G^T=[u_i\otimes v_j]$,由矩阵求逆引理[18]可得:

$$(K^{-1}+W)^{-1}=K-KW^{1/2}B^{-1}W^{1/2}K \tag{11}$$

其中,$B=I+W^{1/2}KW^{1/2}$.由 woodbury matrix identity 得:

$$\begin{aligned}B^{-1}&=(I+W^{1/2}KW^{1/2})^{-1}\\&=(I+W^{1/2}GG^TW^{1/2})^{-1}\\&=(I+(W^{1/2}G)(W^{1/2}G)^T)^{-1}\\&=I-(W^{1/2}G)(I+(W^{1/2}G)^T(W^{1/2}G))^{-1}(W^{1/2}G)^T\end{aligned} \tag{12}$$

由于$(I+(W^{1/2}G)^T(W^{1/2}G))$是一个 $D=D_1D_2$ 维的方阵,而 $D<<mn$,极大地降低了 B^{-1} 的计算复杂度,也降低了$(K^{-1}+W)^{-1}$ 的计算复杂度.求得 $\hat{F}$,$(K^{-1}+W)^{-1}$ 后可得后验概率:

$$p(F|U,V,Y)\approx q(F|U,V,Y)=N(\hat{F},(K^{-1}+W)^{-1}) \tag{13}$$

2.3 预测模块

假设需要预测一个样本(u_*,v_*,y_*)的输出值 y_*,先求概率:

$$p(y_*=+1|U,V,Y,u_*,v_*)=\int\Phi(f_*)\,p(f_*|U,V,Y,u_*,v_*)\mathrm{d}f_* \tag{14}$$

其中,$\Phi(f_*)$为高斯累积函数.由于 *F* 与 f_*的联合分布:

$$\begin{bmatrix}f_*|u_*,v_*\\F|U,V,Y\end{bmatrix}\sim N\left(0,\begin{bmatrix}K((u_*,v_*),(u_*,v_*)) & K((u_*,v_*),(U,V))\\K((U,V),(u_*,v_*)) & (K^{-1}+W)^{-1}\end{bmatrix}\right) \tag{15}$$

其中,

$$K((U,V),(u_*,v_*))=\Sigma(U,u_*)\otimes\Omega(V,v_*),\Sigma(U,u_*)=\begin{bmatrix}\langle u_1,u_*\rangle\\\langle u_2,u_*\rangle\\\cdots\\\langle u_m,u_*\rangle\end{bmatrix},\Omega(U,u_*)=\begin{bmatrix}\langle v_1,v_*\rangle\\\langle v_2,v_*\rangle\\\cdots\\\langle v_n,v_*\rangle\end{bmatrix},$$

$$K((u_*,v_*),(U,V))=\Sigma(u_*,U)\otimes\Omega(v_*,V),K((u_*,v_*),(u_*,v_*))=\Sigma(u_*,u_*)\otimes\Omega(v_*,v_*).$$

由于高斯分布的条件分布仍是高斯分布,所以由公式(15)可得:

$$f_*|U,V,F,u_*,v_*\sim N(K((u_*,v_*),(U,V))(K^{-1}+W)F,K((u_*,v_*),(u_*,v_*))-K((u_*,v_*),(U,V))(K^{-1}+W)K((U,V),(u_*,v_*))) \tag{16}$$

$$p(f_*|U,V,Y,u_*,v_*)=\int p(f_*|U,V,F,u_*,v_*)\,p(F|U,V,Y)\mathrm{d}F \tag{17}$$

由公式(16)、公式(17)得均值:

$$\begin{aligned}\mu &= E(p(f_*|U,V,Y,u_*,v_*))\\ &= \int E(f_*|U,V,F,u_*,v_*)p(F|U,V,Y)\mathrm{d}F\\ &= \int K((U,V),(u_*,v_*))K((u_*,v_*),(u_*,v_*))^{-1}Fp(F|U,V,Y)\mathrm{d}F\\ &= K((U,V),(u_*,v_*))K((u_*,v_*),(u_*,v_*))^{-1}E(p(F|U,V,Y))\end{aligned} \tag{18}$$

由公式(4)得知 $E(p(F|U,V,Y))\approx\hat{F}$,所以 $E(p(f_*|U,V,Y,u_*,v_*))=K((U,V),(u_*,v_*))K((u_*,v_*),(u_*,v_*))^{-1}\hat{F}$,方差:

$$\begin{aligned}\sigma^2 &= Var(p(f_*|U,V,Y,u_*,v_*))\\ &= E((f_*-E(f_*|U,V,F,u_*,v_*))^2)+E((E(f_*|U,V,F,u_*,v_*)-E(f_*|U,V,Y,u_*,v_*))^2)\\ &= K((u_*,v_*),(u_*,v_*))-K((u_*,v_*),(U,V))K^{-1}K((U,V),(u_*,v_*))+\\ &\quad K((u_*,v_*),(U,V))K^{-1}(K^{-1}+W)^{-1}K^{-1}K((U,V),(u_*,v_*))\end{aligned} \tag{19}$$

因此,$(f_*|U,V,Y,u_*,v_*)$为均值μ、方差σ^2的高斯分布,由此可以求得:

$$p(y_*=+1|U,V,Y,u_*,v_*)=\Phi\left(\frac{\mu}{\sqrt{1+\sigma^2}}\right)=\int_{-\infty}^{\frac{\mu}{\sqrt{1+\sigma^2}}}\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{\frac{(x-\mu)^2}{2\sigma^2}}\mathrm{d}x \tag{20}$$

当 $p(y_*=+1|U,V,Y,u_*,v_*)\geqslant\theta$(阈值$\theta\in(0,1)$)时,$y_*$预测值为+1,否则为−1.

3 实验及结果分析

为了评估方法的有效性,我们设计了两组实验:第 1 组实验验证采集样本算法的有效性;第 2 组实验通过与基于 LDA 的缺陷定位方法进行比较,展示高斯过程缺陷定位方法的有效性.评估指标采用查全率(recall)和查准率(precision),计算方法如下:

$$Recall=\frac{TP}{TP+FN},Precision=\frac{TP}{TP+FP} \tag{21}$$

其中,*TP* 表示正样本预测成正样本的个数,*FN* 表示正样本预测成负样本的个数,*FP* 表示负样本预测成正样本的个数.

3.1 数据集

为了评估高斯过程缺陷定位的有效性,使用第 2.1.1 节说明的方法收集了 Eclipse 3.0~3.2 平台上的 3 个组件 jdt.core,swt,gef 以及 argouml 开源项目的缺陷报告和源文件作为实验数据,去掉未被修复以及不能找到对应修改的源文件的缺陷,表 1 展示了每组数据集中包含的缺陷数、源文件数以及缺陷与源文件关系图中边的个数,比如,jdt.core 的缺陷数为 1 107,源文件数为 975,缺陷与源文件关系图中边的个数为 3 124.

Table 1 Description of datasets

表 1 数据集描述

数据集	缺陷数	源文件数	缺陷与源文件关系图中边的个数
jdt.core	1 107	975	3 124
swt	392	955	1 105
jef	244	260	432
argouml	478	712	1 196

3.2 实验1

缺陷与源文件关系图中每条边对应一个正样本,而没边的缺陷报告节点和源文件节点对应一个负样本.正如表 1 所示,在 jdt.core 关系图中边的个数是 3 124,而节点对总数是 1 079 325,因此正样本数只占总样本数的 0.29%.其他数据集中正样本也是占总样本数的比例很小,在这种负样本数远多于正样本数时,会使模型的训练失效,从而将所有的新样本预测为负样本,因此对样本集进行一次采集.为了验证优先采集方法的合理性,分别使用优先采集方法和随机采集方法采集相同个数的样本,然后使用这两组样本集分别对高斯过程进行训练,最后在相同的测试集上做预测.

预测结果的评价标准采用查全率和查准率(见公式(21)),阈值θ的变化会导致查全率和查准率的变化.将θ设置为不同的取值,分别计算预测结果的查全率和查准率,结果如图 3 所示.

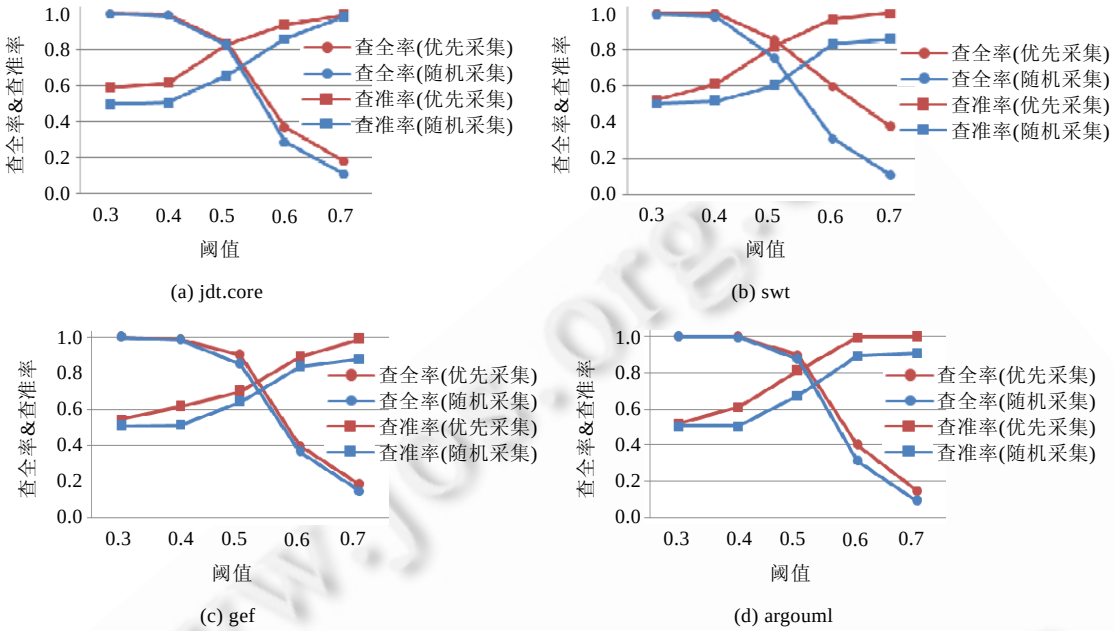

Fig.3 Comparison of two sampling methods

图 3 采集方法对比

从图 3 中可以看出,随着θ值的增加,查全率曲线逐渐下降而查准率曲线逐渐上升.从查全率和查准率变化的曲线可以看出:在θ取值相同的情况下,每个数据集上,优先采集方法的查全率和查准率均高于随机采集方法;特别是在查准率方面,优先采集方法明显优于随机采集方法.如:当θ取值为 0.5 时,在数据集 swt 上,随机采集方法的查准率是 0.63,而优先采集方法的查准率是 0.816,提高了 18%左右;在数据集 Argouml 上,随机采集方法的查准率是 0.672 2,而优先采集的查准率是 0.812 9,提高了 14%左右.从图 3 也可以看出:θ取值为 0.5 时,查全率和查准率的取值均比较高;而θ<0.4 时,查准率过低;θ>0.6 时,查全率过低.因此在实验 2 中,θ取值设置为 0.5.

优先采集方法优于随机采集方法的原因在于:优先采集在选择负样本时,针对每个缺陷优先选择缺陷数多的源文件组成的节点对作为负样本,因为这些缺陷数多的源文件相对于缺陷数少的源文件是否与当前的缺陷相关更具有不确定性.正如文献[16]所述:若一个样本的类别越模糊,其真实类别对模型的训练越有帮助,所以优先选择缺陷数多的源文件的策略更有利于模型的训练.

3.3 实验2

一些研究人员利用信息检索模型[8−10],如向量空间模型、潜在语义模型、概率潜在语义模型和潜在狄利克雷分配模型(LDA)做缺陷的定位.与高斯过程缺陷定位(GPBL)类似,这些方法也是将缺陷和源代码抽象成一个词向量.在基于信息检索的缺陷定位方法中,LDA 是最新的方法,并且最新研究工作[11]表明,基于 LDA 模型的缺陷定位优于其他的信息检索模型.为了表明高斯过程优于 LDA 模型,使用文献[11]中采用的 LDA 工具,参数设置也与它保持一致,在数据集上做实验.

为了展示缺陷相似性对高斯过程缺陷定位结果的影响,对每个数据集分两种情形分别做实验:

- 情形 1:首先计算每对缺陷之间的相似度,从中挑选出高相似度的缺陷对;然后,从这些高相似度缺陷中选出一定数量的缺陷;最后,将这些缺陷从原数据集中的缺陷集中删除,剩下的缺陷作为训练数据;
- 情形 2:与情形 1 过程类似,不同的是从原数据集中删除的是相似度低的缺陷.

比较情形 1 和情形 2 可以看出,情形 1 减少了高相似性的缺陷对,而情形 2 减少了低相似性的缺陷对.为了实验的公平性,在两种情形中删除的缺陷数是相同的,均是原缺陷数的 30%,且之间没有相同的缺陷.两种情形的实验结果见表 2.

Table 2 Comparison of Bug Localization Based on LDA and GPBL

表 2 基于 LDA 模型的缺陷定位与高斯过程缺陷定位(GPBL)结果对比

数据集	情形 1				情形 2			
	GPBL		LDA		GPBL		LDA	
	查全率	查准率	查全率	查准率	查全率	查准率	查全率	查准率
jdt.core	0.781 5	0.805 7	0.547	0.618	0.845 1	0.859 8	0.603	0.573
swt	0.801 6	0.736	0.816	0.705	0.850 9	0.827 7	0.831	0.673
gef	0.813 8	0.662 4	0.786	0.594	0.883 1	0.733 2	0.801	0.615
argouml	0.858 6	0.792	0.764	0.629	0.910 7	0.836 3	0.785	0.657

表 2 显示结果表明:在情形 2,高斯过程缺陷定位的查全率和查准率均高于情形 1,但是两种情形的结果差距不大(5%~10%之间).例如:在数据集 jdt.core 中,情形 2 的查全率是 0.845 1,查准率是 0.859 8;而在情形 1 中查全率是 0.781 5,查准率是 0.805 7.在其他数据集中也有类似的结果,说明随着数据集中缺陷间相似度增加,高斯过程缺陷定位的查全率和查准率会增加.

从表 2 可以看出,在情形 1 的数据集 swt 中,LDA 的查全率略高于高斯过程缺陷定位.原因在于:高斯过程缺陷定位需要利用缺陷之间的相似度,当缺陷之间的相似度偏低时,高斯过程缺陷定位的效果会下降;而 LDA 模型不利用缺陷之间的相似度,因此缺陷之间相似度的高低不会对它产生显著的影响.而在其他 3 个数据集中,LDA 缺陷定位的查全率和查准率均低于高斯过程.例如:在情形 1 的数据集 argouml 上,高斯过程的查全率为 0.858 6,查准率为 0.792;而 LDA 模型的查全率为 0.764,查准率为 0.629.基于高斯过程的缺陷定位的效果明显优于基于 LDA 模型,其原因在于:缺陷是用自然语言描述的,而程序是用程序语言来描述,这两种语言的语义空间不同,用 LDA 模型计算缺陷报告与代码的相关性,当缺陷报告的用词与代码的用词不一致时,用 LDA 计算得出的缺陷报告与缺陷所在的代码文件的相关性很低;而基于高斯过程模型的缺陷定位,计算的是缺陷报告与缺陷报告的相似性以及文件与文件之间的相似性,相似性的计算是在同一个语义空间,这样在同一个语义空间中计算得出的相关性比较准确.

4 总 结

本文基于高斯过程提出了一种缺陷定位的方法;针对训练数据集中正负样本不均衡这个问题,提出了一种采集样本的算法;而且针对高斯过程计算复杂度高的缺点,提出了一种减少计算量的方法.实验结果表明,这些方法是有效的,与基于 LDA 模型的缺陷定位进行比较表明,高斯过程缺陷定位明显优于 LDA 模型.

高斯过程是一种机器学习的方法,它的预测效果与训练数据的质量有关.如果缺陷的书写不符合规范,代码的标识符的命名很随意,高斯过程缺陷定位的效果会受到影响.另外,训练数据需要达到一定的规模,否则高斯过程缺陷定位的效果也会受到影响.

核函数是高斯过程模型的重要部分,本文只是针对线性核函数情况下降低高斯过程的计算复杂度,而对于其他核函数的情况或者向量维度高的情况下,现在还没有办法降低高斯过程的计算复杂度,所以提高高斯过程模型的训练速度是下一步工作的重点.本文的方法可以集成到缺陷管理库中,作为缺陷管理库的一个组件,当新提交一个缺陷时,通过向缺陷修复人员推荐缺陷可能存在哪些文件中,可以帮助开发人员快速修复缺陷,所以下一步准备将方法集成到缺陷管理工具中.

References:

[1] Liblit B, Naik M, Zheng AX, Aiken A, Jordan MI. Scalable statistical bug isolation. ACM SIGPLAN Notices, 2005,40(6):15−26.

[2] Liu C, Yan XF, Fei L, Han JW, Midkiff SP. SOBER: Statistical model-based bug localization. ACM SIGSOFT Software Engineering Notes, 2005,30(5):286−295.

[3] Zhou WJ, Zhang DP, Xu BW. An adaptive algorithm of locating fault interactions based combinatorial testing. Chinese Journal of Computers, 2011,34(8):1509−1518 (in Chinese with English abstract).

[4] Zhou WJ, Zhang DP, Xu BW. Locating error interactions based on partial covering array. Chinese Journal of Computers, 2011, 34(6):1126−1136 (in Chinese with English abstract).

[5] Yu K, Chu W, Yu SP, Tresp V, Xu Z. Stochastic relational models for discriminative link prediction. Advances in Neural Information Processing Systems, 2007,19:1553−1560.

[6] Zhou ZH, Yang Q. Machine Learning and Applications. Beijing: Tsinghua University Press, 2011 (in Chinese).

[7] Chu W, Sindhwani V, Ghahramani Z, Keerthi SS. Relational learning with Gaussian processes. Neural Informaiton Processing Systems, 2007,19:289−296.

[8] Poshyvanyk D, Guéhéneuc YG, Marcus A, Antoniol G, Rájlich V. Combining probabilistic ranking and latent semantic indexing for feature identification. In: Proc. of the 14th IEEE Int'l Conf. on Program Comprehension. Athens: Institute of Electrical and Electronics Engineers Computer Society, 2006. 137−148.

[9] Gay G, Haiduc S, Marcus A, Menzies T. On the use of relevance feedback in IR-based concept location. In: Proc. of the IEEE Int'l Conf. on Software Maintenance. Edmonton: IEEE Computer Society, 2009. 351−360.

[10] Lukins SK, Kraft NA, Etzkorn LH. Source code retrieval for bug localization using latent dirichlet allocation. In: Proc. of the 15th Working Conf. on Reverse Engineering. Antwerp: IEEE Computer Society, 2008. 155−164.

[11] Lukins SK, Kraft NA, Etzkorn LH. Bug localization using latent dirichlet allocation. Information and Software Technology, 2010, 52(9):972−990.

[12] Zhou J, Zhang HY, Lo D. Where should the Bugs be fixed. In: Proc. of the Int'l Conf. on Software Engineering. Zurich: IEEE Computer Society, 2012. 14−24.

[13] Davies S, Roper M, Wood M. Using bug report similarity to enhance bug localization. In: Proc. of the 19th Working Conf. on Reverse Engineering. IEEE Computer Society, 2012. 125−134.

[14] Wu RX, Zhang HY, Kim SH, Cheung SC. Relink: Recovering links between bugs and changes. In: Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering. Szeged: Association for Computing Machinery, 2011. 15−25.

[15] Manning CD, Raghavan P, Schutze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2009.

[16] Mitchell TM. Machine Learning. McGraw-Hill, 1997.

[17] Minka TP. A family of algorithms for approximate Bayesian inference [Ph.D. Thesis]. Massachusetts Institute of Technology, 2001.

[18] Williams CKI, Rasmussen CE. Gaussian Processes for Machine Learning. Cambridge: MIT Press, 2006.

附中文参考文献:

[3] 周吴杰,张德平,徐宝文.基于组合测试的软件故障定位的自适应算法.计算机学报,2011,34(8):1509−1518.

[4] 周吴杰,张德平,徐宝文.基于部分覆盖表的错误交互定位方法.计算机学报,2011,34(6):1126−1136.

[6] 周志华,杨强.机器学习及其应用.北京:清华大学出版社,2011.

陈理国(1980－),男,湖北黄冈人,博士生,主要研究领域为软件工程,数据挖掘.
E-mail: netclg@gmail.com

刘超(1958－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件测试.
E-mail: liuchao@buaa.edu.cn