基于卷积神经网络的代价敏感软件缺陷预测模型

邱少健[1] **蔡子仪**[1] **陆璐**[1]

(华南理工大学计算机科学与工程学院 广州 邮编 510000)[1]

摘 要 基于机器学习的软件缺陷预测方法受到软件工程领域学者们的普遍关注，通过缺陷预测模型可一定程度地分析软件中的缺陷分布，以此帮助软件质量保障团队发现软件中潜在的错误并合理分配测试资源。然而，多数缺陷预测模型是基于人工提取的软件度量特征进行缺陷预测的，而未考虑源码中潜在的语义特征，导致预测效果不理想。针对该问题，构建了基于三层卷积神经网络的代价敏感软件缺陷预测模型（CS-TCNN），目的在于从软件中挖掘源码的语义特征并将其利用于软件缺陷预测任务中，同时利用代价敏感方法处理预测任务中分类不均衡的问题。通过采用 PROMISE 软件缺陷数据库的仿真实验，分别将 CS-TCNN 与逻辑回归、深度置信网络等模型比较，结果表明 CS-TCNN 有效地提高了软件缺陷预测模型的性能。

关键词 软件缺陷预测，卷积神经网络，语义特征挖掘，代价敏感

中图法分类号 TP311 **文献标识码** A **DOI**

Cost-sensitive Convolutional Neural Network Model for Software Defect Prediction

QIU Shao-jian[1] CAI Zi-yi[1] LU Lu[1]

(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510000, China)[1]

Abstract Machine-learning-based software defect prediction methods are received widely attention from the researchers in the field of software engineering. The defect prediction model can analyze the defect distribution in the software, so as to help the software quality assurance team to detect potential software errors and allocate test resources reasonably. However, most current models are based on hand-crafted features for defect prediction, without considering the potential semantic features in the source code, resulting in poor prediction performance. To solve this problem, a cost-sensitive three-layer convolutional neural network (CS-TCNN) is constructed. Its purpose is to mine the semantic features of source code from software and use it in software defect prediction tasks. Cost-sensitive learning method is also be used to handle the imbalanced problem. By conducting experiments on PROMISE database, the comparison with traditional logistic regression and deep belief network models shows that CS-TCNN effectively improves the performance of software defect prediction model.

Keywords Software defect prediction，Convolutional neural network，Semantic feature mining，Cost-sensitive

1 引言

在软件的生命周期中，软件质量保障技术扮演着重要角色。软件中潜在的、未被发现的缺陷势必影响软件质量，尽早地发现缺陷并对其加以处理将对软件质量的保障具有重要的意义[1]。通过软件缺陷预测技术，可一定程度地预测软件系统中的缺陷分布，以此帮助质量保障团队客观地了解软件质量状态，从而有效地分配测试资源，评估软件是否达到了交付和使用的标准[2]。

基于此，许多研究人员密切关注缺陷预测技术并尝试通过机器学习方法检测软件中有缺陷倾向的模块或文件。这些软件缺陷预测方法是在获

到稿日期： 返修日期：
本文受自然科学基金（61370103）、广州产学研基金（201802020006）、中山产学研基金（2017A1014）资助。
邱少健（1990-），男，博士生，主要研究方向为软件可靠性及软件测试；蔡子仪（1994-），男，硕士生，主要研究方向为深度学习及软件可靠性；陆璐（1971-），男，博士，教授，主要研究方向为软件工程及软件测试，E-mail: lul@scut.edu.cn。

取手工软件度量特征的基础上进行的，软件特征选择的方法会直接影响缺陷预测的效率和准确性。Radjenovic D 等人[3]认为软件模块的复杂性与软件缺陷的分布具有直接的关系,以源码复杂性特征作为输入可预测软件的缺陷，其主要包括基于操作符和操作数的 Halstead 特征，基于依赖性的 McCabe 特征。Jureczko M 等人[4]针对面向对象软件的缺陷预测问题进行了软件特征提取，整理了如类中使用的方法个数、类在继承树中子类个数等特征。Yang X 等人[5]认为软件开发过程中所产生的信息也对缺陷预测有所帮助，其验证了通过软件开发的过程特征进行软件缺陷预测的有效性。刘望舒等人[6]认为缺陷预测数据集中含有的冗余和无关特征会影响缺陷预测模型的性能,因此提出一种基于聚类分析的特征选择方法。通过对软件源码特征的提取，软件缺陷预测方法利用历史缺陷数据构建如支持向量机、随机森林、逻辑回归等分类器，并利用以上分类器对新软件源码开展缺陷预测任务。预测结果将帮助软件质量保障团队找到可能包含缺陷的源码区域。

然而，传统的手工源码特征未能涵盖软件源码中丰富的语义特征，其可能导致训练后的模型预测性能不理想。通过对源码中语义特征的挖掘，使缺陷预测相关的语义信息得以被利用，将提高软件缺陷预测的性能[7]。近年来，深度学习作为自动特征生成的技术之一得到了较为充足的研究[8]。在软件缺陷预测领域，深度学习也可以被用于挖掘软件源码中复杂的非线性特征。Wang S 等人[9]利用多层受限玻尔兹曼机叠加成的深度置信网络(Deep Belief Network，DBN)自动提取项目源码的语义特征，用此特征组构建软件缺陷预测模型。甘露等人[10]同样利用 DBN，先进行特征集成与迭代，并对这些特征数据进行深度学习，构建了基于 DBN 的软件缺陷预测模型(DBNSDPM)。事实上，除了 DBN 之外，卷积神经网络(Convolutional Neural Network，CNN)作为在图像分类、文本语义分析等领域运用广泛[11]的神经网络之一也可用于软件源码语义特征的提取。Li 等人[7]利用 CNN 提取源码语义特征后将其与传统静态特征进行合并，构建了逻辑回归分类器对软件缺陷进行预测，其仿真实验验证了利用 CNN 构建的预测模型可以得到比 DBN 更好的效果。然而，上述用于软件缺陷预测的 CNN 模型中存在着语义特征挖掘不足（仅利用单层卷积层）和分类不均衡处理方法随机性较强（仅利用样本随机过采样方法）的问题，本文针对上述两点问题，构建了代价敏感的三层卷积神经网络模型（Cost-sensitive three-layer convolutional neural network，CS-TCNN），并利用 CS-TCNN 对实际软件数据集进行了实验仿真,取得了较好的结果。

2 基于卷积神经网络的代价敏感软件缺陷预测模型

卷积神经网络是一种用于处理类似网络结构数据的特殊神经网络，诸如时间序列和图像序列。卷积神经网络已经被证实在语音识别，图像分类，自然语言处理等领域表现出众。和传统的多层感知机相比卷积神经网络具备两个重要的特征：稀疏连接和参数共享。稀疏连接是指卷积神经网络在两层神经元之间采样局部连接的方式来利用空间的局部特性。参数共享是指一个模型中的多个函数使用相同的参数。

基于卷积神经网络的特点和优势，本文提出 CS-TCNN 模型利用了卷积神经网络从软件源码中提取语义特征并开展软件缺陷预测任务，CS-TCNN 模型主要包含以下 4 个步骤：

1） 源码语法解析和样本表征向量生成

2） 表征向量映射成整数向量

3） 根据代价敏感对整数向量进行权重赋值

4） 对 CS-TCNN 模型进行训练及缺陷预测

2.1 源码语法解析和样本表征向量生成

抽象语法树(Abstract Syntax Tree，AST)是高级编程语言源码的树状表示，源码中的每种结构可表示为树中的一个节点。值得一提的是，AST 不仅能表示源码当中的语法特征，其还隐藏了源码的语义特征。相关的工作[9]已经证明 AST

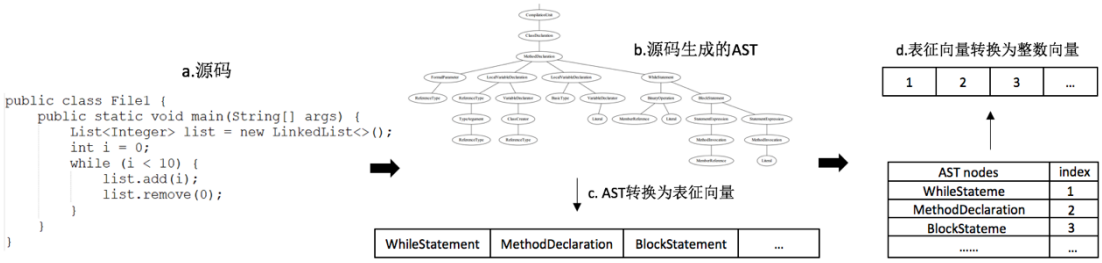

图 1 源码转换成整数向量的过程

Fig.1 The process of converting source code into integer vector

可用于源码完整性和缺陷的检测。为了能将一个源码文件表示成向量的形式用作神经网络的输入，本文使用开源的 *Python* 依赖包 *Javalang* 解析软件项目文件并生成对应的抽象语法树。遍历 AST 所有节点后可生成其对应的表征向量(Token Vector)。本文主要采用以下三种类别的 AST 节点作为语法树表征向量元素，同时表 1 列出本文记录的 AST 节点类型：

1）AST 中用于表示方法调用和类实例化的节点，并且用节点对应方法名或类名作为此节点在表征向量中的表征。

2）表示声明的节点。例如方法的声明（MethodDeclaration）、类型的声明（VariableDeclarator）等。节点的名称作为表征向量的表征。

3）控制流节点。例如条件控制(IfStatement)，循环控制(WhileStatement)。节点的名称用作表征向量的表征。

表 1 本文记录的 AST 节点类型

Table 1 The selected AST nodes

类别	节点类型
方法调用和类实例化节点	MethodInvocation SuperMethodInvocation ClassCreator
声明节点	PackageDeclaration,InterfaceDeclaration ClassDeclaration,ConstructorDeclaration MethodDeclaration,VariableDeclarator
控制流节点	IfStatement,WhileStatement DoStatement,ForStatement AssertStatement,BreakStatement ContinueStatement,ReturnStatement ThrowStatement,SynchronizedStatement TryStatement,SwitchStatement BlockStatement,TryResource CatchClause,CatchClauseParameter SwitchStatementCase,ForControl EnhancedForControl
其他节点	StatementExpression,FormalParameter BasicType,MemberReference SuperMemberReference,ReferenceType

2.2 表征向量映射成整数向量

由于 CNN 的训练需要的输入实数向量，因此 2.1 中提取的表征向量不能直接作为 CNN 模型训练的输入。为了解决这个问题，本文首先在整数和表征（Token）之间建立一个映射，并将表征向量转换为整数向量。本文使得每一个表征和一个大于 0 的整数标识符对应，此整数标识符的范围从 1 到软件中表征种类的数量，以这种方式进行映射，相同的表征在向量中保持相同的整数标识符。此外，由于 CNN 要求输入向量具有相同的长度，如果一个整数向量长度小于最大向量长度，则用整数 0 填充向量，使得所有整数向量的长度和最长表征向量一致。在将表征映射到整数的同时统计每一个表征出现的频数，依据文献[9]的处理方式，本文将出现频数小于 3 的表征的值设为 0。图 1 中表示了将一段示例源码转换成整数向量的过程。

2.3 根据代价敏感对整数向量进行权重赋值

在软件缺陷预测领域，数据分类不均衡问题是普遍存在且具有挑战性的。如果分类器在高度分类不均衡的数据集上训练的，预测模型往往对少数类别的样本分类能力较弱。较为常见数据分类不均衡问题的处理方法有基于二次采样技术[12]的随机欠采样（Random Under-sampling, RUS）、随机过采样（Random Over-sampling, ROS）和 SMOTE 等方法，然而简单的随机二次采样方法具有较强的随机性，而 SMOTE 方法因为涉及到样本之间距离的度量，对本文基于 AST 的实例之间的距离度量是否有效仍需进一步研究，因此本文探索另一种方法处理数据分类不均衡问题。在分类问题中，由于少数类别实例的稀缺性，很自然地

它们预测结果的准确性在模型训练时比多数类别实例更具价值。因此，本文将代价敏感方法作为CS-TCNN 模型中处理类别不均衡问题的方法。与传统的二次采样技术不同，基于不均衡率的代价敏感方法使用类别误分成本作为权重赋予训练样本。本文在 CS-TCNN 模型中所使用的损失函数为交叉熵损失函数：

$$Loss_{Crossentropy} = -\sum_{i=1}^{n}[y_i * \log \hat{y}_i + (1 - y_i) * \log(1 - \hat{y}_i)]$$

给定训练集中有缺陷的样本个数为n_1，无缺陷样本的个数为n_0，将训练集中有缺陷样本权重ω_1和无缺陷样本权重ω_0为设置为其样本数占总样本数比例的倒数。

$$\omega_1 = (n_1 + n_0) / n_1 \quad ; \quad \omega_0 = (n_1 + n_0) / n_0$$

将ω_1和ω_0作为有无缺陷数据样本的代价权重代入交叉熵损失函数中，可得到 CS-TCNN 训练过程中的损失函数计算方式为：

$$Loss = -\sum_{i=1}^{n}[\omega_1 * y_i * \log \hat{y}_i + \omega_0 * (1 - y_i) * \log(1 - \hat{y}_i)]$$

2.4 CS-TCNN 模型训练及缺陷预测

如上述所讨论的，本文利用 CNN 的特征生成能力捕获源代码的潜在的语义特征。将携带权重的整数向量集合作为训练数据训练 CNN 模型（即训练 CNN 中神经元的权重和偏差）。CS-TCNN 所构建的是三层卷积神经网络，具体结构如图 2 示意图所示。CS-TCNN 包含一个嵌入层，三层卷积层和最大池化层的组合，全连接层，和最后一个输出层。除了输出层使用 Sigmoid 函数作为激活函数以外，CS-TCNN 中的其他层都使用 ReLU 作为激活函数。其中嵌入层[13]的作用是将输入的整数向量中的整数元素转换成向量，进而用一对向量的距离来表示两整数的相似度。卷积层和池化层的作用是为了捕获源码中的局部特征,本文 CS-TCNN 模型的三个卷积层参数保持一致。全连接层则是为了将卷积后提炼的特征展开，以便开展最后的输出分类任务。本文 CS-TCNN 在输出层采用逻辑回归作为最终分类器。

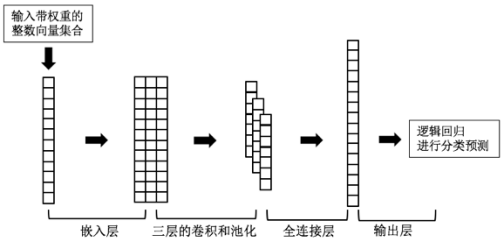

图 2 CS-TCNN 模型示意图

Fig.2 Schematic diagram of CS-TCNN model

3 实验设置与结果分析

3.1 数据集

为了评估 CS-TCNN 模型的性能，本文从 PROMISE 数据集[14]中选取了 8 个软件，按照版本划分共含 19 个项目，可组成 11 组的软件缺陷预测任务（在同个软件中，分别将旧版本数据作为训练集，将新版本数据作为测试集，如 Camel-1.2 数据作为训练集，Camel-1.4 中数据作为测试集）。表 2 显示了软件项目的基本信息，其包含软件名称，软件版本，实例数和缺陷率。在本文中，为了验证 CS-TCNN 的通用性，数据集由不同大小和缺陷率的软件项目组成。

表 2 实验数据集

Table 2 Datasets

软件项目	版本	平均实例数	平均缺陷率
camel	1.2，1.4，1.6	815	22.4%
jedit	4.2，4.3	350	17.3%
log4j	1.0，1.1	123	30.0%
lucene	2.0，2.2，2.4	260	56.0%
poi	2.5，3.0	352	54.2%
synapse	1.0，1.1，1.2	212	25.5%
velocity	1.5，1.6	213	57.4%
xalan	2.6，2.7	830	54.4%

3.2 评估方法

在本文的实验中，采用 AUC[15]和 MCC[16]两个在软件缺陷预测领域运用较为广泛的评估度量[17,18]来评定各模型的性能。它们的定义如下：

在二分类的软件缺陷预测任务中，可以获得四个测试数据结果：1）将真正有缺陷的实例分类为了有缺陷的（真阳性，TP）；将真正无缺陷的实例分类为有缺陷的（误报，FP）；将真正有缺陷的实例分类为无缺陷的（假阴性，FN）；将真正无缺陷的数据分类为无缺陷（真阴性，TN）。

基于这四个结果，可以定义检测概率（PD）和误报概率（PF）。PD 表示真正有缺陷的实例与所有缺陷实例的数量相比的概率。PF 表示假阴性实例与所有无缺陷实例的数量相比的概率。他们的计算方式如下：

$$PD = \frac{TP}{TP + FN}$$

$$PF = \frac{FP}{FP + TN}$$

ROC 图是一种二维图，其中 PD 绘制在 y 轴上，PF 绘制在 x 轴上。ROC 图可以用来显示成本和收益之间的相对折衷。在本文的实验中，使用 ROC 曲线下面积（the area under the ROC curve, AUC）来评估预测模型。根据 ROC 的含义，本文寻找具有高 PD 和低 PF 的模型。换句话说，具有更高的 AUC 值说明预测模型更好。

马修斯相关系数（Matthews correlation coefficient, MCC）通过考虑所有真假的有缺陷数据和无缺陷数据来衡量预测结果和真实结果之间的关系。它的返回值是[-1,1]，其中 1 表示完美的正相关，-1 表示完美的负相关。MCC 按以下公式计算：

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

值得一提的是，与软件缺陷预测研究中其他广泛使用的指标（如 F-measure[7]和 G-mean[19]）比较，AUC 和 MCC 度量同时考量了 TN，且他们对分类不平衡问题不太敏感，具有一定的优势。

3.3 实验设置

实验将本文提出的 CS-TCNN 与如下几个方法进行比较：

1）逻辑回归（Logistic regression，LR）：基于 PROMISE 数据集中提供的 20 个手工特征构建的逻辑回归分类器。

2）DBN 方法：在源代码上使用深度置信网络提取软件源码中的语义特征，进而构建逻辑回归分类器对软件项目进行缺陷预测。在实现 DBN 时，使用与文献[9]中相同的网络和参数，即含 10 个隐藏层，每个隐藏层中含 100 个节点。

3）CNN 方法：在源代码上使用一层卷积神经网络(CNN)提取缺陷预测的语义特征的缺陷预测方法。在实现 CNN 时，使用经验参数[7]设置如下：嵌入层的维度设为 k，batch 大小为 32，训练的 epoch 为 15。卷积核的大小为 20，数量为 50，隐藏层单元的数量为 100。

4）TCNN 方法：本文中提出的具有三层卷积神经网络的方法，参数同 CNN 一致。

为了公平比较，本文为 DBN、CNN、TCNN 和 CS-TCNN 提供相同的整数向量集合作为输入。值得一提的是，由于 AST 数值向量并非特征向量，在做数据预处理时基于距离的不均衡问题处理方法并不能直接被使用，因此，上述方法在处理数据不平衡问题中，除了 CS-TCNN 以外均使用样本过采样（Over-sampling）方法进行数据预处理，在 CS-TCNN 方法执行时，基于本文 2.3 所述代价敏感方法对不同类别的样本权重进行赋值。此外，由于样本过采样方法及卷积神经网络训练涉及一定的随机性，因此实验中每个方法执行 50 次，并取其平均值作为模型比较时的参考依据。

3.4 实验结果及分析

本文采用 AUC 和 MCC 两个指标对 LR，DBN，CNN，TCNN 和 CS-TCNN 五个方法在 11 组预测任务的预测性能进行度量，图 3 和图 4 分别记录了 AUC 和 MCC 的结果，每行最优结果被加粗。

从图 3 和图 4 中可以观察到，无论是 AUC 还是 MCC，在 11 对项目组合中，CS-TCNN 模型的效果大多数预测任务中好于 LR，DBN，CNN 和 TCNN。使用 CS-TCNN 对 11 组任务进行缺陷预测的 AUC（MCC）在 0.609~0.741（0.04~0.508）之间，平均值为 0.679（0.286）。平均值较于 LR，DBN，CNN，TCNN 分别高出了 4.7%（24.3%），4.8%（21.3%），4.9%（16.6%）和 1.8%（7.4%），说明 TCNN 在结合了代价敏感的方法所得到的 CS-TCNN 在这 11 对软件缺陷预测任务中具备更好的性能。

表 3 五个方法的 AUC 实验结果

Table 3 The AUC results of 5 methods

训练集->测试集	LR	DBN	CNN	TCNN	CS-TCNN
Camel-1.2->Camel-1.4	0.649	0.638	0.687	**0.712**	0.695
Camel-1.4->Camel-1.6	0.601	0.628	0.645	**0.666**	0.657
jedit-4.2->jedit-4.3	0.656	0.677	0.614	0.656	**0.688**
log4j-1.0->log4j-1.1	0.734	0.702	0.713	0.738	**0.741**
lucene-2.0->lucene-2.2	**0.634**	0.625	0.602	0.628	0.609
lucene-2.2->lucene-2.4	0.583	0.617	0.612	0.62	**0.635**
poi-2.5->poi-3.0	0.666	0.623	0.662	0.679	**0.696**
synapse-1.0->synapse-1.1	0.601	0.601	0.554	0.567	**0.625**
synapse-1.1->synapse-1.2	0.635	**0.692**	0.622	0.638	0.670
velocity-1.5->velocity-1.6	0.657	0.642	0.700	0.712	**0.720**
xalan-2.6->xalan-2.7	0.718	0.687	0.71	0.724	**0.736**
AVG	0.649	0.648	0.647	0.667	**0.679**

表 4 五个方法的 MCC 实验结果

Table 4 The MCC results of 5 methods

训练集->测试集	LR	DBN	CNN	TCNN	CS-TCNN
Camel-1.2->Camel-1.4	0.239	0.233	0.293	**0.337**	0.307
Camel-1.4->Camel-1.6	0.172	0.201	0.309	**0.335**	0.322
jedit-4.2->jedit-4.3	0.108	0.12	0.122	0.147	**0.149**
log4j-1.0->log4j-1.1	0.468	0.466	0.479	0.498	**0.508**
lucene-2.0->lucene-2.2	0.265	0.225	0.264	0.274	**0.282**
lucene-2.2->lucene-2.4	0.163	0.238	0.202	0.217	**0.251**
poi-2.5->poi-3.0	0.327	0.237	0.296	0.331	**0.360**
synapse-1.0->synapse-1.1	0.192	**0.228**	0.096	0.113	0.218
synapse-1.1->synapse-1.2	0.261	**0.350**	0.233	0.25	0.305
velocity-1.5->velocity-1.6	0.298	0.273	0.364	0.385	**0.401**
xalan-2.6->xalan-2.7	0.036	0.021	0.037	0.039	**0.040**
AVG	0.230	0.236	0.245	0.266	**0.286**

综上所述，通过解决现有软件缺陷预测方法中语义特征挖掘不足的问题，在本文的 11 对软件缺陷预测任务的实验中，无论在平均 AUC 度量指标还是平均 MCC 度量指标中，TCNN 的值都比 LR、DBN 和 CNN 更高，结合了代价敏感的 CS-TCNN 在 TCNN 性能提升的基础上又有提高。由此可见，在软件缺陷预测技术研究领域引入基于三层卷积神经网络的代价敏感软件缺陷预测模型 CS-TCNN 能提高缺陷预测的准确性。

结束语 本文以软件源码语义特征的生成和应用为动机，提出基于三层卷积神经网络的软件缺陷预测模型，同时针对软件缺陷预测任务中普遍存在的分类不均衡问题，利用代价敏感方法调整了神经网络数据输入的权重。最终地，提出了基于三层卷积神经网络的代价敏感软件缺陷预测模型 CS-TCNN。通过与 LR，DBN，CNN 方法的实验对比，可以说明 CS-TCNN 能提高软件缺陷预测的准确性。在未来的工作中，将探索改进现有 CS-TCNN 模型的网络结构，同时在本文卷积神经网络使用的经验参数基础上，进一步地研究 CS-TCNN 中各参数调节和设置方法。此外，针对 AST 整数向量不可直接度量样本距离导致很多已有基于距离度量的分类不均衡处理方法无法直接使用的问题，提出合理的解决方案。

参考文献

[1] LIU H，HAO K G. Cause Analysis Method of Software Defect [J]. Computer Science，2009，36(1):242-243. (in Chinese)
刘海，郝克刚.软件缺陷原因分析方法[J].计算机科学，2009，36(1):242-243.
[2] PETERS F, MENZIES T, MARCUS A. Better cross company defect prediction[C]// Proceedings of the 10th IEEE Working Conference on Mining Software Repositories, 2013:409-418.
[3] RADJENOVIC D, HERICKO M, TORKAR R, et al. Software fault prediction metrics: A systematic literature review[J]. Information and Software Technology, 2013, 55(8):1397-1418.
[4] JURECZKO M, MADEYSKI L. Towards identifying software project clusters with regard to defect prediction[C]// Proceedings of the 6th

International Conference on Predictive Models in Software Engineering, 2010:9.
[5] YANG X, LO D, XIA X, et al. TLEL: A two-layer ensemble learning approach for just-in-time defect prediction[J]. Information and Software Technology, 2017, 87:206-220.
[6] LIU W S, CHEN X, GU Q, et al. A cluster-analysis-based feature-selection method for software defect prediction[J]. SCIENTIA SINICA Informationis, 2016, 46(9):1298-1320.(in Chinese)
刘望舒, 陈翔, 顾庆, 等. 软件缺陷预测中基于聚类分析的特征选择方法[J]. 中国科学: 信息科学, 2016, 46(9):1298-1320.
[7] LI J, HE P, ZHU J, et al. Software defect prediction via convolutional neural network[C]// Software Quality, Reliability and Security, 2017 IEEE International Conference on. IEEE, 2017:318-328.
[8] FISCHER A， IGEL C. An introduction to restricted Boltzmann machines[C]//Iberoamerican Congress on Pattern Recognition. Springer, Berlin, Heidelberg, 2012:14-36.
[9] WANG S, LIU T, TAN L. Automatically learning semantic features for defect prediction [C]. International Conference onSoftware Engineering, 2016:297-308.
[10] GAN L, ZANG L, LI H. Deep Belief Network Software Defect Prediction Model[J]. Computer Science, 2017, 44(4):229-233. (in Chinese)
甘露, 臧洌, 李航. 深度信念网软件缺陷预测模型[J]. 计算机科学, 2017, 44(4):229-233.
[11] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[C]// Advances in neural information processing systems. 2012:1097-1105.
[12] CHAWLA N V, BOWYER K W, HALL L O, et al. Smote: synthetic minority over-sampling technique[J]. Journal of Artificial Intelligence Research, 2002, 16(1):321–357.
[13] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]// Advances in neural information processing systems. 2013:3111-3119.
[14] BOETTICHER G. The PROMISE repository of empirical software engineering data[J]. http://promisedata.org/repository, 2007.
[15] FAWCETT T. An introduction to ROC analysis[J]. Pattern recognition letters, 2006, 27(8):861-874.
[16] BALDI P, BRUANAK S, CHAUVIN Y, et al. Assessing the accuracy of prediction algorithms for classification: an overview[J]. Bioinformatics, 2000, 16(5):412-424.
[17] LI Y, HUANG Z, FANG B, et al. Using Cost-Sensitive Classification for Software Defects Prediction[J]. Journal of Frontiers of Computer Science and Technology, 2014, 8(12):1442-1451. (in Chinese)
李勇, 黄志球, 房丙午, 等. 代价敏感分类的软件缺陷预测方法[J]. 计算机科学与探索, 2014, 8(12):1442-1451.
[18] SONG Q, GUO Y, SHEPPERD M. A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction[J]. IEEE Transactions on Software Engineering, 2018.
[19] XIONG J, GAO Y, WANG Ya. Software Defect Prediction Model Based on Adaboost Algorithm[J]. Computer Science, 2016, 43(7):186-190. (in Chinese)
熊婧, 高岩, 王雅瑜. 基于 Adaboost 算法的软件缺陷预测模型[J]. 计算机科学, 2016, 43(7):186-190.

作者联系电话：15817190456
作者联系邮箱：qiushaojian@outlook.com