CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 40 卷第 10 期 2018 年 10 月
Vol. 40, No. 10, Oct. 2018

文章编号:1007-130X(2018)10-1796-09

基于深度自编码网络的软件缺陷预测方法*

周　末,徐　玲,杨梦宁,廖胜平,鄢　萌
(重庆大学大数据与软件学院,重庆 401331)

摘　要:软件缺陷预测是提升软件质量的有效方法,而软件缺陷预测方法的预测效果与数据集自身的特点有着密切的相关性。针对软件缺陷预测中数据集特征信息冗余、维度过大的问题,结合深度学习对数据特征强大的学习能力,提出了一种基于深度自编码网络的软件缺陷预测方法。该方法首先使用一种基于无监督学习的采样方法对 6 个开源项目数据集进行采样,解决了数据集中类不平衡问题;然后训练出一个深度自编码网络模型。该模型能对数据集进行特征降维,模型的最后使用了三种分类器进行连接,该模型使用降维后的训练集训练分类器,最后用测试集进行预测。实验结果表明,该方法在维数较大、特征信息冗余的数据集上的预测性能要优于基准的软件缺陷预测模型和基于现有的特征提取方法的软件缺陷预测模型,并且适用于不同分类算法。

关键词:软件缺陷预测;特征降维;深度自编码网络;类不平衡
中图分类号:TP311.5　　**文献标志码**:A
doi:10.3969/j.issn.1007-130X.2018.10.011

Software defect prediction based on deep autoencoder networks

ZHOU Mo, XU Ling, YANG Meng-ning, LIAO Sheng-ping, YAN Meng
(School of Big Data & Software Engineering, Chongqing University, Chongqing 401331, China)

Abstract: Software defect prediction is an effective way for improving the quality of software, and the effect of software defect prediction is closely related to data sets' own characteristics. In regard of feature information redundancy and large dimension of data sets, combining with the powerful learning feature ability of deep learning, we propose a software defect prediction method based on deep autoencoder networks. This method firstly uses an unsupervised learning sampling method to do sampling for 6 open source projects data sets to solve class imbalance problem of datasets. We then build a deep autoencoder network model through training, which can reduce the dimension of data sets. The model uses three classifiers for connection and employs the training sets with reduced dimension to train the classifiers. Finally, we use the test sets to do prediction. Experimental results show that the proposed method outperforms the basic software defect prediction model and the software defect prediction model based on existing feature extraction methods under the circumstance of the data sets with large dimension and redundant feature information. Besides, it is adaptive to different classifiers.

Key words: software defect prediction; feature dimension reduction; deep autoencoder network; class imbalance

* **收稿日期**:2017-09-07;**修回日期**:2018-03-20
通信地址:401331 重庆市重庆大学大数据与软件学院
Address: School of Big Data & Software Engineering, Chongqing University, Chongqing 401331, P. R. China

1　引言

随着软件规模与复杂度日益增长,软件开发过程中不可避免地产生了缺陷数据,而含有缺陷的软件在运行时会造成不可预期的后果,严重时会给企业带来巨大的经济损失。在软件项目的开发周期中,缺陷被发现得越晚,则修复缺陷的代价就越高。软件缺陷预测[1]旨在软件开发初期,及时准确地预测出软件模块是否包含缺陷,以此来合理有效地分配有限的测试资源,降低软件维护成本,提高软件质量。

软件度量元(软件特征)[2]是影响软件缺陷预测精度的重要因素。在软件缺陷预测过程中,往往将不同数据源收集到的数据整合成数据集,数据不可避免地存在大量冗余特征,这些冗余特征会干扰到模型算法的学习效果,影响缺陷预测准确性。特征提取是软件缺陷预测的重要步骤,其目的是从特征集合中选择满足一定评价准则的最优特征子集,通过剔除不相关特征或冗余特征,实现降维[3]。

特征提取方法主要通过线性或非线性的方法对原始特征进行组合,将高维数据映射到低维空间,常见的方法有:主成分分析 PCA(Prinicipal Component Analysis)[4]、拉普拉斯特征映射方法 LE(Laplacian Eigenmaps)[5]、局部保持映射 LPP(Locality Preserving Projections)[6]等。特征选择是根据某些评价准则从原始样本的特征中选择使得评价准则最优的一个子集,一般分为过滤式[7]、封装式[8]和嵌入式[9-11]。刘芳等人[12]将主成分分析、混沌粒子群优化算法 CPSO(Chaotic Particle Swarm Optimization)和支持向量机 SVM(Support Vector Machine)相结合,提出了基于 PCA-ISVM 的软件缺陷预测模型,有效消除了软件数据中的冗余,获得了最优支持向量机参数,提升了软件缺陷预测模型性能。陈翔等人[13]提出了一种混合的两阶段特征选择方法 HFS(Hybrid Feature Selection)用于软件缺陷预测中,首先利用特征子集评估器移除冗余特征,然后基于特征排序评估器进一步移除无关特征,最后不仅能够选出更小规模的特征子集,而且有效提升了缺陷预测模型的性能。

上述特征降维方法都属于浅层模型,只能从大量的数据中提取或选择一些简单的特征,不能得到具有复杂结构的特征。近年来,深度学习作为机器学习中最有前景的方法之一,在很多研究领域中都被证明是一种十分有效的方法,尤其在语音识别[14,15]和图像识别[16-18]领域取得了巨大成功。然而,深度学习是否可以用来提升软件缺陷预测的性能目前很少有文献证明。本文针对软件缺陷预测过程中遇到的高维特征提取问题,结合深度学习对数据特征优化学习的能力,提出了一种基于深度自编码网络的软件缺陷预测方法。该方法分为两个阶段:特征提取阶段和机器学习阶段。第一阶段使用深度自编码网络进行数据特征提取,第二阶段用提取的特征和与其对应的标签训练分类器进行分类,典型的分类方法包括逻辑回归[19,20]、支持向量机[21,22]、人工神经网络[23,24]、贝叶斯方法[25,26],得到最终的软件缺陷预测模型。

本文采用了 6 个数据集进行实验,包括不同的项目,分别为美国国家航空航天局 NASA(National Aeronautics and Space Administration)项目、PROMISE 库中的 Eclipse 项目以及 NetGene[27]库中的 Lucene 项目,这些数据集存在缺陷数据分布严重不平衡问题。这一直是软件缺陷预测领域的一大挑战,缺陷数据的比例过少会严重影响软件缺陷预测模型的预测性能。一般的研究方法多采用随机采样方法来划分数据集,这样就很容易出现缺陷数据分布不平衡的情况。有研究表明,软件度量元的复杂度越高,就越可能存在缺陷[28]。Nam 等人[29]指出,中位数可以作为度量元的阈值衡量度量元的复杂度。本文在此基础上提出了一种基于无监督学习和随机采样的采样方法,该方法确保了数据分布中缺陷数据的比例不会过低,从而提升了软件缺陷预测模型的性能。

本文的主要贡献包括:

(1)提出了一种基于无监督学习的采样方法,显著提升了预测模型的性能指标。

(2)提出了基于深度自编码网络的软件缺陷预测模型,实验表明该模型的性能指标优于不采用降维处理的软件缺陷预测模型以及采用一般降维方法的软件缺陷预测模型。

2　基于深度自编码网络的软件缺陷预测模型

2.1　软件缺陷预测模型建立

软件缺陷数据分布极不平衡,只有很少部分数据存在缺陷。如果采用随机采样方法,训练数据集中极有可能含有极少甚至没有缺陷数据,很难训练出好的预测模型,缺乏足够多的缺陷数据作为训练

样本是缺陷预测模型的一个瓶颈。通过分析软件度量元和软件缺陷潜在关联性,本文提出了基于无监督学习和随机采样的混合采样方法,该方法确保了训练集中包含一定比例的缺陷数据,减少数据类不平衡所带来的影响。

在此基础上对原始数据集的特征采用深度自编码网络进行特征提取,减少了特征的维度,去除了大量冗余信息,学习出了具有复杂结构的特征,能显著提升预测性能。详细步骤如图1所示。

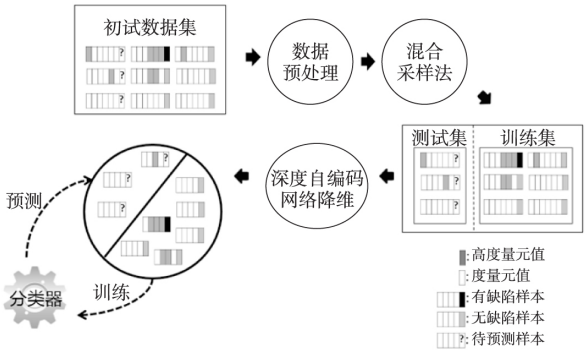

Figure 1 Building the software defect prediction model

图1 软件缺陷预测模型建立

具体步骤如下:

(1)数据预处理,包括缺省数据填充,数据标准化处理,这里使用Z-core法进行标准化,减少因属性之间数值差距过大对预测模型的影响,使得样本度量元的值分布在0～1。

(2)使用本文提出的一种基于无监督采样和随机采样结合的混合采样方法,从大量的不平衡软件数据集中采集训练数据集,使训练集中软件缺陷数据占一定比例,剩下的部分作为测试数据集。

(3)使用去标签的训练集训练深度自编码网络,提取出高层具有复杂结构的特征,然后将提取出的特征加上对应的标签,训练分类器。

(4)使用训练好的分类器对测试数据集进行预测。

2.2 采样方法

有研究表明度量元复杂度越高,存在缺陷的可能性越大[28]。通过分析缺陷数据和非缺陷数据中度量元的盒图(图2),我们发现,一般存在缺陷模块的度量元值要高于不含缺陷模块的度量元值。文献[29]指出,属性中位数可以作为阈值衡量属性的复杂度,我们在此假设基础上提出一个基于无监督学习和随机采样的混合采样方法,该方法确保样本包含一定的缺陷样本数据率,具体采样流程如图3所示。

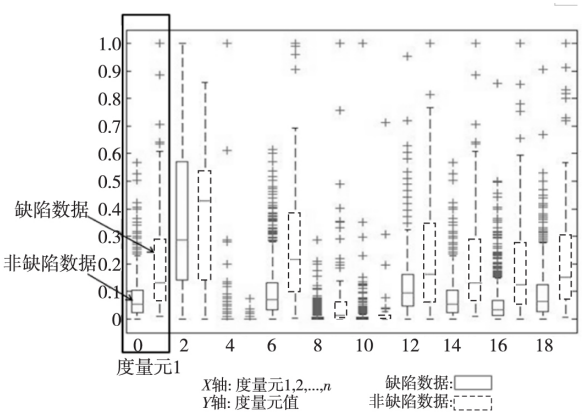

Figure 2 Distribution of metrics between defect data and non-defect data

图2 度量元值在缺陷数据和非缺陷数据中分布

图3中$I1$～$I7$代表数据样本,$x1$～$x6$代表每个样本的不同度量元的值。首先,初始化训练集,假设采样个数为N,采样方法主要步骤如下所示:

(1)计算每个度量元在所有样本中的值的中位数,即每个度量元的阈值,例如,度量元$x1$在所有样本中的值为{2,3,0,1,2,1,3},中位数就为2。

(2)对于每个样本,每个度量元的值与对应的阈值进行比较,计算出每条样本大于阈值的度量元值的个数,例如,样本$I1$,其中$x3$和$x5$大于与其对应的中位数,所以样本$I1$的高值度量元数量为2。

(3)根据(2)计算出的样本的高值度量元个数

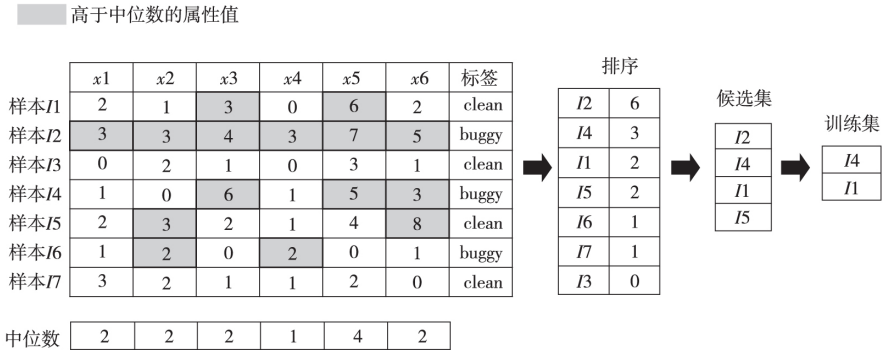

Figure 3 Hybrid sampling method

图3 混合采样法

进行排序。例如,图 3 中排序结果为{$I2,I4,I1,I5,I6,I7,I3$}。

(4)从排序结果中选取前 $2*N$ 个样本作为候选样本集。例如,图 3 中的候选集为{$I2,I4,I1,I5$}。

(5)从候选样本集中随机选取 N 个作为最终的采样结果。例如,图 3 中结果{$I4,I1$}。

2.3　深度自编码网络特征提取

自动编码器[30]从输入数据学习特征,优化特征。它是一种无监督的学习算法,利用反向传播算法,让目标值尽可能等于输入值,通过不停训练调整网络参数,得到每一层训练权重,则中间层就是原始输入的近似表达,以此实现无监督特征提取。

对于一个自动编码网络,数据特征提取分为编码和解码两个阶段。

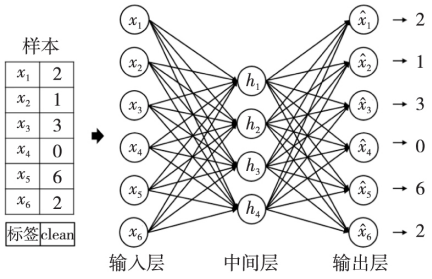

Figure 4　One layer autoencoder

图 4　单层自动编码器

(1)编码阶段:每一个输入样本为 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$,$x_1\sim x_n$ 代表每一个数据样本的不同度量特征值。例如,图 4 中 $x_1\sim x_6$ 代表一个样本的 6 个不同的度量元值{2,1,3,0,6,2}。对其进行 Z-score 标准化,使 $\boldsymbol{x}\in[0,1]^n$。通过编码函数 $h_\theta(\boldsymbol{x})$ 对输入样本进行编码($h_\theta\in[0,1]^m$),其中,编码函数定义为:

$$h_\theta(\boldsymbol{x})=f(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}) \tag{1}$$

$$f(\boldsymbol{z})=\frac{1}{1+\exp(-\boldsymbol{z})}\text{sigmoid} \tag{2}$$

其中,$W\in\mathbf{R}^{m*n}$ 代表权重矩阵,$\boldsymbol{b}\in\mathbf{R}^m$ 代表偏置向量,$\theta=\{\boldsymbol{W},\boldsymbol{b}\}$,$f(\boldsymbol{x})$ 采用 sigmoid 函数,其定义如公式(2)所示。编码阶段整体体现为图 4 的输入层到中间层的转换过程。例如,图中 $h_i=f(w_{i1}x_1+w_{i2}x_2+\cdots+w_{i6}x_6+b_i)$。其中,$i\in\{1,2,3,4\}$。

(2)解码阶段:通过解码函数对网络输入进行重构得到输出层 $\hat{\boldsymbol{x}}=(\hat{x}_1,\hat{x}_2,\cdots,\hat{x}_n)$($\hat{\boldsymbol{x}}\in[0,1]^n$)。解码函数为:

$$\hat{\boldsymbol{x}}=g_{\theta'}(\boldsymbol{h})=f(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}+\boldsymbol{b}') \tag{3}$$

其中,$\theta'=\{\boldsymbol{W}^{\mathrm{T}},\boldsymbol{b}'\}$,$\boldsymbol{W}^{\mathrm{T}}\in\mathbf{R}^{n*m}$,$\boldsymbol{b}'\in\mathbf{R}^n$ 代表偏置向量,参数 $\boldsymbol{h}$ 需要传入的是编码阶段的结果,$f(\boldsymbol{x})$ 仍然是 sigmoid 函数。解码阶段体现为图 3 的中间层到输出层的转换过程。这是一个对称的网络结构。该网络模型的参数 $\boldsymbol{W},\boldsymbol{b},\boldsymbol{b}'$ 通过目标函数:

$$J(\theta,\theta')=\frac{1}{n}\sum_{i=1}^{n}L(x^{(i)},\hat{x}^{(i)}=\frac{1}{n}\sum_{i=1}^{n}L\{(x^{(i)},g_{\theta'}[h_\theta(x^{(i)})]\} \tag{4}$$

进行训练优化,L 为损失函数,$L(\boldsymbol{x},\hat{\boldsymbol{x}})=\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|$。最终得到一个稳定的网络结构,在这个网络中,$\hat{\boldsymbol{x}}\approx\boldsymbol{x}$,例如图 3,$\hat{x}_1\sim\hat{x}_6$ 表示为输出,值为{2,1,3,0,6,2},和输入层度量元值一样。输出层尽可能复现了输出层的每个软件数据样本的度量元,则中间层的数据就可以作为输入的一种近似表示,这种表示就是特征。由于整个过程没有数据样本标签(有缺陷或无缺陷)参与,所以这是一种无监督特征提取方法。

单层的自编码网络虽然已经具备一定的特征提取能力,但是这样提取的特征还不足以使得分类器具有很好的分类性能。深度学习是模拟人的神经系统对信息的分级处理过程,这是一个不断迭代、不断抽象的过程。高层特征是底层特征的组合,特征表示也更抽象,更能表现语义或者意图。而抽象层面越高,存在的冗余信息就越少,就越利于分类。因此我们运用深度自编码网络[31]来对特征进行进一步的抽象和提取,进一步提升分类效果。

深度自编码网络[31]是由多个单层自动编码器组成的神经网络,前一个自动编码器的中间层作为下一个自动编码器的输入,采用逐层贪婪训练法进行训练,获取最终的深度自编码网络参数。深度自编码网络训练过程如图 5 所示。

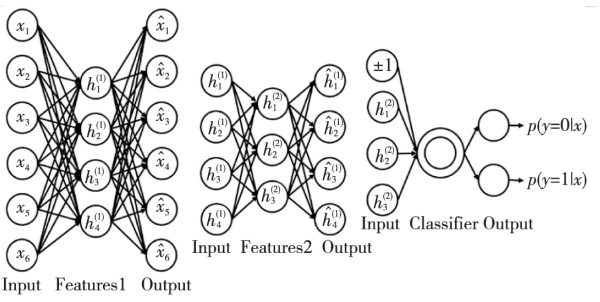

Figure 5　Process of training deep autoencoder networks

图 5　深度自编码网络训练过程

(1)第一层自动编码器的输入样本采用了机器学习中常用的数据预处理方法 Z-score 进行归一化处理,样本的度量元 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$,$\boldsymbol{x}\in[0,1]^n$。按照上述单层自动编码的训练方式训练第一层自动编码器,得到第一层自动编码器的初始网络参数,然后将第一层自动编码器的中间层作为输入

传入第二层自动编码器。

(2)第二层自动编码器,以同样的无监督方法进行训练,得到该层的网络参数。在多层自动编码器进行网络参数初始化之后,在顶层添加一个分类器,然后加上每个数据样本对应的标签(本方法针对的项目具有大量历史数据集并且自身带有标签,有缺陷或无缺陷),用有监督学习的方法对整个网络进行训练,例如,图 5 是一个包含两层自动编码器的深度自编码网络的训练过程。

(3)最后将输入层和中间的特征层全部连接起来,然后连接上分类器,得到最终的深度自编码网络,整体框架如图 6 所示。

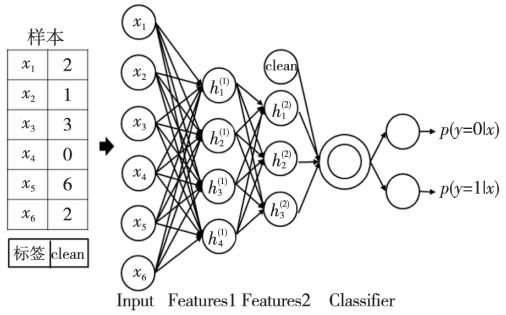

Figure 6 Overall framework of the deep autoencoder network

图 6 深度自编码网络整体框架

图 6 是一个包含两个中间特征层的深度自编码网络结构框架,网络的输入是 6 个度量元值,利用深度自编码网络,最后的特征向量降为三维,把这三个最终的特征加上对应的标签,传入分类器,就可以对分类器进行训练。用所有的训练集训练好模型后,就可以用这个最终的深度自编码网络模型进行软件缺陷预测,将测试样本以特征向量的形式作为输入,最后会得到 $y=0$(无缺陷)和 $y=1$(有缺陷)的概率值。概率值较大的作为最终的预测结果。

3 实验与结果

本节评估深度自编码器的有效性。实验环境是:Intel(R) Core i7-4700MQ CPU,8 GB RAM,Windows 7(64-bit)。我们首先介绍实验数据集和评估标准,然后提出两个研究问题,并通过实验结果来进行分析论证。

RQ1 相对于不使用降维手段的基准软件缺陷预测模型,本文提出的基于深度自编码网络的缺陷预测模型是否能提升性能,并且对不同分类算法是否具有普遍性?

RQ2 相对于基于其他一般的降维方法,即基于特征提取和特征选择方法的软件缺陷预测模型,本文提出的软件缺陷预测模型是否能有效提升预测性能,并且对不同分类算法是否具有普遍性?

3.1 数据集及评估指标

实验数据集来自几个常用的开源项目,包括:NASA、PROMISE 库中的 Eclipse 项目以及 NetGene[27]库中的 Lucene 数据集。NASA 数据集的度量元主要包括 Halstead 复杂度[32]、McCabe 环路复杂度[33]以及代码行数;Eclipse 项目数据集主要从代码复杂度和抽象语法树方面设计度量元;Lucene 数据集主要包括 network 和 change genealogy 度量元[27],是经过人工验证的去噪后的数据集。表 1 统计了数据集具体信息。数据集的特征数量包括了少量(21)、中等(155)和大量(465)三个级别。

Table 1 Data sets

表 1 数据集

数据集	来源	特征数	实例数	缺陷实例数	缺陷率/%
kc1	NASA	21	2 109	326	9.8
pc1	NASA	21	1 109	77	6.9
Eclipse2.0	PROMISE	155	6 927	975	14.1
Eclipse2.1	PROMISE	155	7 888	854	10.8
Eclipse3.0	PROMISE	155	10 593	1 568	14.8
Lucene	NetGene	465	1 671	346	10.7

实验在表 1 所示的数据集中采用十折交叉法[34]验证。将数据集平均分成十份,轮流将其中九份作为训练集,一份作为测试集进行验证,在软件工程研究中十折交叉法是一种应用非常广泛的评估方法。

验证所提方法有效性的性能评估采用目前主流的四个评价指标:查全率(*recall*)、查准率(*precision*)、*F*1-*score*、*AUC*(Area Under ROC Curve)。这些性能指标的评估结果基于表 2 的混淆矩阵。

Table 2 Confusion matrix

表 2 混淆矩阵

	预测为有缺陷	预测为无缺陷
实际有缺陷	*TP*(true positive)	*FP*(false positive)
实际无缺陷	*TN*(true negative)	*FN*(false negative)

查全率指正确预测的缺陷模块占真实所有缺陷模块的比例,计算如式(5)所示:

$$recall = \frac{TP}{TP + FN} \tag{5}$$

查准率指正确预测的缺陷模块占预测为缺陷

的所有模块的比例,计算如式(6)所示:

$$precision = \frac{TP}{TP + FP} \tag{6}$$

*F*1-*score* 为查全率和查准率的调和平均数,综合了查全率和查准率的结果,可以综合评价软件缺陷模型的性能,计算如式(7)所示:

$$F1\text{-}score = \frac{2 \times recall \times precision}{recall + precision} \tag{7}$$

AUC,基于 ROC(Receiver Operating Characteristic)曲线,综合考虑了不同的分类阈值,以此来评估软件缺陷预测模型的性能,适用于评估分类不平衡情况下的软件缺陷模型。本实验采用的数据集缺陷率最高为 14.8%,是典型的数据不平衡数据。

3.2　实验与结果分析

在深度自编码网络的软件缺陷预测实验中,根据本文提出的基于无监督和随机采样的混合采样方法,采样个数 N 必须从 $2 * N$ 的候选集随机采样,因此采样率不能大于 0.5,而如果采样率过小,训练集会出现缺陷数据占比过大的情况,使得训练集分布不平衡,影响软件缺陷预测性能。因此,本实验设置采样率为 $u=0.4$,采用十折交叉验证法,重复实验 20 次,取 20 次实验结果的平均值作为最终实验结果。分类器采样三种常用的分类算法,分别是 Softmax、支持向量机(SVM)[21] 和逻辑回归 LR(Logistic Regression)[19]。

本实验用于对比的基准方法包括三种:

(1)不使用任何特征提取方法,直接使用原始数据集训练缺陷预测模型。

(2)使用常见的特征提取方法 PCA 提取特征,然后训练缺陷预测模型。

(3)使用一种混合的特征选择方法 HFS 提取特征,然后训练缺陷预测模型。

主成分分析法 PCA[4] 是一种常见的特征提取算法,目的是通过线性变换寻找一个能够保留样本差异性的低维投影空间,使得少量主成分包含大量信息。具体实现参照文献[4]。特征选择方法 HFS 是文献[13]提出的一种混合了特征子集评估器和特征排序评估器的特征选择方法,能够提升特征质量,达到优于一般特征选择方法的预测性能。整体实验设计框架如图 7 所示。

对于第一个研究问题 RQ1,实验采用表 1 中的 6 个数据集进行实验,基准模型(Basic)是指不采用特征提取方法直接分类的方法,与本文提出的基于深度自编码网络的软件缺陷预测模型(简称 DA)在使用三种不同分类器 Softmax、SVM[21]、LR[19] 的情况下进行对比实验。实验结果如表 3～表 5 所示,字体加粗的数字代表结果更优的数值。

Table 3　Result comparison between the proposed model and the basic models under softmax classifier

表 3　分类器为 softmax 时本文模型与和基准模型的对比结果

数据集	*recall*		*precision*		*F*1-*score*		*AUC*	
	Basic	DA	Basic	DA	Basic	DA	Basic	DA
kc1	0.095	**0.165**	**0.301**	0.195	0.144	**0.179**	0.742	**0.789**
pc1	**0.152**	0.083	0.412	**0.542**	**0.222**	0.144	**0.798**	0.721
Eclipse2.0	**0.452**	0.421	0.083	**0.358**	0.140	**0.387**	0.521	**0.678**
Eclipse2.1	0.512	**0.673**	0.102	**0.413**	0.170	**0.512**	0.514	**0.721**
Eclipse3.0	0.132	**0.310**	0.236	**0.563**	0.169	**0.400**	0.572	**0.686**
Lucene	0.314	**0.547**	0.364	**0.430**	0.337	**0.482**	0.601	**0.753**

Table 4　Result comparison between the proposed model and the basic models under SVM classifier

表 4　分类器为 SVM 时本文模型与基准模型的对比结果

数据集	*recall*		*precision*		*F*1-*score*		*AUC*	
	Basic	DA	Basic	DA	Basic	DA	Basic	DA
kc1	**0.196**	0.158	**0.451**	0.325	**0.273**	0.213	0.771	**0.842**
pc1	**0.276**	0.197	0.075	**0.402**	0.118	**0.264**	0.698	**0.824**
Eclipse2.0	0.375	**0.512**	0.282	**0.438**	0.322	**0.472**	0.641	**0.704**
Eclipse2.1	0.362	**0.611**	0.210	**0.470**	0.266	**0.531**	0.527	**0.673**
Eclipse3.0	0.127	**0.491**	0.327	**0.502**	0.183	**0.496**	**0.618**	0.587
Lucene	0.256	**0.663**	0.401	**0.462**	0.313	**0.545**	0.792	**0.769**

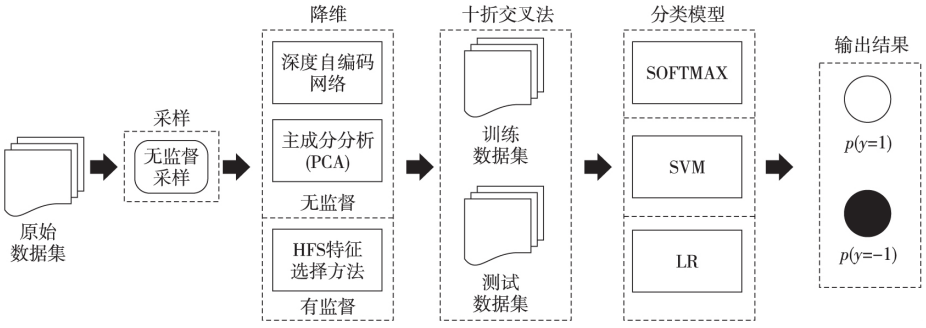

Figure 7　Experiment frame

图 7　实验框架图

Table 5 Result comparison between the proposed model and the basic models under LR classifier

表 5 分类器为 LR 时本文模型与基准模型的对比结果

数据集	*recall*		*precision*		*F1-score*		*AUC*	
	Basic	DA	Basic	DA	Basic	DA	Basic	DA
kc1	0.093	**0.116**	**0.249**	0.152	**0.135**	0.132	0.664	**0.714**
pc1	0.104	**0.301**	**0.293**	0.218	0.154	**0.253**	**0.674**	0.621
Eclipse2.0	**0.328**	0.285	0.217	**0.484**	0.261	**0.359**	0.567	**0.783**
Eclipse2.1	0.268	**0.407**	0.267	**0.568**	0.268	**0.474**	0.635	**0.832**
Eclipse3.0	0.149	**0.459**	0.301	**0.532**	0.199	**0.493**	**0.732**	0.704
Lucene	0.282	**0.394**	0.459	**0.441**	0.349	**0.416**	**0.784**	0.739

从表 3～表 5 的实验结果可以看出,对于 NASA 数据集 kc1 和 pc1,本文提出的方法在三种分类器下的各种评估指标上的性能效果都没有明显提升,很多项指标的结果甚至不如基准模型。例如,在使用 Softmax 作为分类器的情况下,对 pc1 的查全率、*F1-score* 和 *AUC* 的结果都不如基准模型。通过分析我们可以发现,NASA 数据集 kc1 和 pc1 的度量元只有 21 个属于少量级别,halstead 度量元和 McCabe 度量都是科学度量法,属于高层的特征,其中的冗余信息很少,所以我们使用深度自编码网络对其提取特征后,实验效果并不理想,甚至不如原始特征。而具有中等数量度量元的 Eclipse 数据集和具有大量度量元的 Lucene 数据集在本文提出的缺陷预测模型中的预测性能在三种分类器中都远好于基准模型,这四个数据集采用 Softmax 作为分类器的情况下,DA 相对于基准模型 *F1-score* 平均值提升了 118%,*AUC* 平均值提升了 28.5%;在 SVM 作为分类器的情况下,DA 相对于基准模型 *F1-score* 平均值提升了 88.6%;*AUC* 平均值提升了 6%;在 LR 作为分类器的情况下,DA 相对于基准模型 *F1-score* 平均值提升了 61.7%,*AUC* 平均值提升了 12.5%,可以看出性能提升十分明显。因为该数据集中含有冗余信息,通过本文的深度自编码网络能够提取到具有复杂结构的更高层的表示特征,利用这种特征能更好地训练分类器,同时,也说明本文方法具有一定的普遍适应性,对多种分类器都能有效提升性能。

对于第二个研究问题 RQ2,实验采用表 1 中的 6 个数据集,分别使用特征提取方法 PCA[4] 和一种混合的特征选择方法 HFS[13],然后和本文提出的基于深度自编码网络的降维方法(DA),在使用三种不同分类器 Softmax、SVM[21]、LR[19] 的情况下进行对比实验。

实验结果如表 6～表 8 所示,字体加粗的数字代表结果更优的数值。

Table 6 Result comparison between the proposed method and other dimension reduction methods under Softmax classifier

表 6 分类器为 Softmax 时本文方法与其他降维方法的对比结果

数据集	*recall*			*precision*			*F1-score*			*AUC*		
	PCA	HFS*2	DA	PCA	HFS*2	DA	PCA	HFS*2	DA	PCA	HFS*2	DA
kc1	0.152	**0.217**	0.165	**0.313**	0.247	0.195	0.204	**0.231**	0.179	0.683	0.662	**0.789**
pc1	**0.283**	0.152	0.083	0.210	0.229	**0.542**	**0.241**	0.183	0.144	0.603	**0.736**	0.721
Eclipse2.0	**0.451**	0.379	0.421	0.135	**0.374**	0.358	0.208	0.377	**0.387**	**0.795**	0.782	0.678
Eclipse2.1	0.372	0.421	**0.673**	**0.489**	0.173	0.413	0.423	0.245	**0.512**	0.695	0.701	**0.721**
Eclipse3.0	0.126	0.246	**0.301**	0.408	0.466	**0.563**	0.193	0.322	**0.392**	0.683	0.632	**0.686**
Lucene	0.493	0.316	**0.574**	0.348	0.259	**0.430**	0.408	0.285	**0.492**	**0.801**	0.772	0.753
Average							0.280	0.274	**0.351**	0.710	0.714	**0.725**

Table 7 Result comparison between the proposed method and other dimension reduction methods under SVM classifier

表 7 分类器为 SVM 时本文方法与其他降维方法的对比结果

数据集	*recall*			*precision*			*F1-score*			*AUC*		
	PCA	HFS*2	DA	PCA	HFS*2	DA	PCA	HFS*2	DA	PCA	HFS*2	DA
kc1	0.167	**0.305**	0.158	**0.423**	0.363	0.325	0.240	**0.332**	0.213	0.673	0.581	**0.842**
pc1	**0.232**	0.219	0.197	0.306	0.371	**0.402**	0.264	**0.275**	0.264	0.680	0.752	**0.821**
Eclipse2.0	0.427	0.485	**0.512**	0.358	0.402	**0.438**	0.390	0.440	**0.472**	**0.722**	0.634	0.704
Eclipse2.1	0.325	**0.621**	0.611	**0.511**	0.429	0.470	0.397	0.507	**0.531**	**0.686**	0.521	0.673
Eclipse3.0	0.362	0.454	**0.491**	0.317	0.421	**0.502**	0.338	0.437	**0.496**	0.639	**0.674**	0.587
Lucene	0.479	0.456	**0.663**	0.253	0.363	**0.462**	0.331	0.404	**0.442**	0.715	0.683	**0.769**
Average							0.327	0.399	**0.403**	0.686	0.641	**0.733**

Table 8　Result comparison between the proposed method and other dimention reduction methods under LR classifier

表 8　分类器为 LR 时本文方法与和其他降维方法的对比结果

数据集	*recall*			*precision*			*F1-score*			*AUC*		
	PCA	HFS*2	DA	PCA	HFS*2	DA	PCA	HFS*2	DA	PCA	HFS*2	DA
kc1	0.092	**0.251**	0.116	**0.373**	0.263	0.152	0.148	**0.257**	0.132	**0.716**	0.634	0.714
pc1	0.263	0.153	**0.301**	**0.337**	0.301	0.218	**0.295**	0.203	0.253	**0.639**	0.563	0.621
Eclipse2.0	**0.395**	0.273	0.285	0.352	0.385	**0.484**	**0.372**	0.320	0.359	0.750	0.743	**0.783**
Eclipse2.1	0.363	0.312	**0.407**	0.469	0.521	**0.568**	0.409	0.390	**0.474**	0.527	0.686	**0.832**
Eclipse3.0	0.256	0.263	**0.459**	0.374	0.406	**0.532**	0.304	0.319	**0.493**	0.693	0.594	**0.704**
Lucene	0.364	**0.406**	0.394	0.405	**0.474**	0.441	0.383	**0.437**	0.416	0.748	**0.763**	0.739
Average							0.319	0.321	**0.355**	0.679	0.664	**0.732**

通过表 6～表 8 的实验对比,在三种分类方法下,对于 NASA 数据集 kc1 和 pc1,相对于其他两种方法,本文方法效果并不理想,原因也是这两个数据集特征过少并且都属于高层特征。对于特征数量为中等级别和大量级别的 Eclipse 的 3 个数据集和 Lucene 数据集,在查全率和查准率的单项指标上,本文方法相对于其他两类方法是稍有优势,在某几个数据集上的指标不如其他两类方法,例如分类器为 Softmax 的情况下,对数据集 Eclipse2.0 的查全率不如 PCA,查准率又不如 HFS,但是指标的平均值都要高于其他两类方法。整体来看,对于评估指标 *F1-score* 平均值,在分类器为 Softmax 的情况下,DA 相对于 PCA 提升了 25.4%,DA 相对与 HFS 提升了 28.1%;在分类器为 SVM 的情况下,DA 相对于 PCA 提升了23.2%,DA 相对与 HFS 提升了 1%;在分类器为 LR 的情况下,DA 相对于 PCA 提升了 11.3%,DA 相对于 HFS 提升了 11%,可以看出在三种分类器下,平均预测结果是有显著提升的。对 *AUC* 的预测结果同 *F1-score* 一样,在整体上有明显提升。这说明本文所提方法能应用于软件缺陷预测,相对与常用的特征提取和特征选择方法有一定提升,并且对于多种分类方法都适用,具有一定的普遍适用性。

4　结束语

针对软件缺陷预测领域中数据集特征提取及维数过大、特征信息冗余等问题,本文利用深度自编码网络强大的特征学习优化能力,提出了一种基于深度自编码网络的软件缺陷预测方法。针对缺陷数据不平衡的问题,提出了一种基于无监督学习的采样方法。实验结果表明,在数据维度较小,并且冗余特征较少的两个 NASA 数据集中,该模型的预测性能提升效果不是很明显;而在数据维度较大的 PROMISE 数据集和 NetGen 数据集中,该模型无论是相对于基准模型还是 PCA 的降维方法,尤其针对 *F1-score*,预测效果都有明显的提升,并且对于三种不同的分类器,都能取得这样的效果。这样的实验结果表明了基于深度自编码网络的软件缺陷预测模型的有效性,并且说明其对于不同的分类器具有一定的适应能力。

参考文献:

[1] Li Yong, Huang Zhi-qiu, Fang Bing-wu, 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)

[2] Wang Qing, Wu Shu-jian, Li Ming-shu. Software defect prediction [J]. Journal of Software, 2008, 19(7): 1565-1580. (in Chinese)

[3] Gao Ke-han, Khoshgoftaar T M, Wang Huang-jing, et al. Choosing software metrics for defect prediction: An investigation on feature selection techniques [J]. Software-Practice And Experience, 2011; 41(5): 579-606.

[4] Turk M, Pentland A. Eigenfaces for recognition [J]. Journal of Cognitive Neuroscience, 1991, 3(1): 71-86.

[5] Belhumeur P N, Hespanha J P, Kriegman D. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(7): 711-720.

[6] Lu J, Plataniotis K N, Venetsanopoulos A N. Face recognition using LDA-based algorithms[J]. IEEE Transactions on Neural Networks, 2003, 14(1): 195-200.

[7] Almuallim H, Dietterich T. Learning with many irrelevant features [C]// Proc of the 9th National Conference on Artifical Intelligence, 1991: 547-552.

[8] Kohavi R, John G. Wrapper for feature subset selection [J]. Artificial Intelligence, 1997, 97(1-2): 273-324.

[9] Breiman L, Friedman J H. Classification and regression trees [J]. Cambridge: Wadsforth Publishing Co, 1984.

[10] Quinlan J R. Learning efficient classification procedures and their application to chess and games [M]. Berlin: Springer Berlin Heidelberg, 1983: 463-482.

[11] Quinlan J R. C4.5:Programs for machine learning [M]. San Francisco:Morgan Kaufmann Publishers Inc,1992.

[12] Liu Fang,Gao Xing,Zhou Bing,et al. Software defect prediction model based on PCA-ISVM [J]. Computer Simulation,2014,31(3):397-401. (in Chinese)

[13] Chen Xiang,He Cheng,Wang Yu,et al. HFS:Hybrid feature selection approach for software defect prediction [J]. Application Research of Computers,2016,33(6):1758-1761. (in Chinese)

[14] Lee H,Pham P,Largman Y,et al. Unsupervised feature learning for audio classification using convolutional deep belief networks[C]// Proc of Advances in Neural Information Processing Systems 22(NIPS 2009),2009:1096-1104.

[15] Hamel P,Eck D. Learning features from music audio with deep belief networks[C]// Proc of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010),2010:339-344.

[16] Taylor G,Hinton G,Roweis S. Modeling human motion using binary latent variables[C]// Proc of Advances in Neural Information Processing Systems 19 (NIPS 2006),2007:1345-1352.

[17] Nair V,Hinton G. 3D object recognition with deep belief nets[C]// Proc of Advances in Neural Information Processing Systems 22(NIPS 2009),2009:1339-1347.

[18] Li Jun,Chang He-you,Yang Jian. Sparse deep stacking network for image classification[C]// Proc of the 29th AAAI Conference on Artificial Intelligence,2015:1-7.

[19] Gyumothy T,Ferenc R,Siket I. Empirical validation of object-oriented metrics on open source software for fault prediction [J]. IEEE Transactions on Software Engineering,2005,31(10):897-910.

[20] Zhou Yu-ming,Leung H. Empirical analysis of object-oriented design metrics for predicting high and low severity faults [J]. IEEE Transactions on Software Engineering,2006,32(10):771-89.

[21] Xing Fei,Guo Ping,Lyu M R. A novel method for early software quality prediction based on support vector machine [C]// Proc of the 16th IEEE International Symposium on Software Reliability Engineering,2005:213-222.

[22] Elish K O,Elish M O. Predicting defect-prone software modules using support vector machines [J]. Journal of Systems and Software,2008,81(5):649-60.

[23] Arar Ö F,Ayan K. Software defect prediction using cost-sensitive neural network [J]. Applied Soft Computing,2015,33(C):263-277.

[24] Khoshgoftaar T M,Allen E B. Neural networks for software quality prediction [J]. Computational Intelligence in Software Engineering,1998,16:33-63.

[25] Pai G J,Duang J B. Empirical analysis of software fault content and fault proneness using Bayesian methods[J]. IEEE Transactions on Software Engineering,2007,33(10):675-686.

[26] Bouguilan,Wang J H,Hamzaa B. A Bayesian approach for software quality prediction [C]// Proc of the 2008 4th International IEEE Conference on Intelligent Systems,2008:1149-1154.

[27] Herzig K,Just S,Rau A,et al. Predicting defects using change genealogies [C]// Proc of the 2013 IEEE 24th International Symposium on Software Reliability Engineering,2014:118-127.

[28] Mebzies T,Greewald J,Framk A. Data mining static code attributes to learn defect predictors [J]. IEEE Transactions on Software Engineering,2007,33(1):2-13.

[29] Nam J,Kim S. CLAMI:Defect prediction on unlabeled datasets [C]// Proc of the 30th IEEE/ACM International Conference on Automated Software Engineering,2015:452-463.

[30] Bengio Y,Lamblin P,Popovici D,et al. Greedy layer-wise training of deep networks [C]// Proc of the 21st Annual Conference on Neural Information Processing System(NIPS 2007),2007:1096-1103.

[31] Vincent P,Larochelle H,Lajoie I,et al. Stacked denoising autoencoders:Learning useful representations in a deep network with a local denoising criterion[J]. Journal of Machine Learning Research,2010,11(3):3371-3408.

[32] Halstead M H. Elements of software science(operating and programming system series) [M]. New York:Elsevier Science Inc,1977.

[33] McCabe T J. A complexity measure [J]. IEEE Transactions on Software Engineering,1976,4(SE-2):308-320.

[34] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection[C]// Proc of the 14th International Joint Conference on Artificial Intelligence,1995:1137-1143.

附中文参考文献:

[1] 李勇,黄志球,房丙武,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.

[2] 王青,伍书剑,李明树. 软件缺陷预测技术[J]. 软件学报,2008,19(7):1565-1580.

[12] 刘芳,高兴,周冰,等. 基于PCA-ISVM的软件缺陷预测模型[J]. 计算机仿真,2014,31(3):397-401.

[13] 陈翔,贺成,王宇,等. HFS:一种面向软件缺陷预测的混合特征选择方法[J]. 计算机应用研究,2016,33(6):1758-1761.

作者简介:

周末(1993-),男,重庆人,硕士生,研究方向为软件工程。**E-mail**:357668386@qq.com

ZHOU Mo, born in 1993, MS candidate, his research interest includes software engineer.