基于 CS－ANN 的软件缺陷预测模型研究*

王海林[a]，于 倩[a,b]，李 彤[a,b]，郁 湧[a,b]，明 利[a]，孙金文[a]
（云南大学 a. 软件学院；b. 云南省软件工程重点实验室，昆明 650500）

摘 要： 为了提高软件缺陷预测的准确率，利用布谷鸟搜索（cuckoo search，CS）算法的寻优能力和人工神经网络（artificial neural network，ANN）算法的非线性计算能力，提出了基于 CS-ANN 的软件缺陷预测方法。此方法首先使用基于关联规则的特征选择算法降低数据的维度，去除了噪声属性；然后利用布谷鸟搜索算法寻找神经网络算法的权值，使用权值和神经网络算法构建出预测模型；最后使用此模型完成缺陷预测。使用公开的 NASA 数据集进行仿真实验，结果表明该模型降低了误报率，并提高了预测的准确率，综合评价指标 AUC（area under the ROC curve）、*F*1 值和 G-mean 都优于现有模型。

关键词： 软件缺陷预测；人工神经网络；布谷鸟搜索；软件质量；机器学习
中图分类号： TP311.53 **文献标志码：** A **文章编号：** 1001-3695（2017）02-0467-06
doi：10.3969/j.issn.1001-3695.2017.02.033

Research of software defect prediction model based on CS-ANN

Wang Hailin[a], Yu Qian[a,b], Li Tong[a,b], Yu Yong[a,b], Ming Li[a], Sun Jinwen[a]
（*a. School of Software, b. Key Laboratory for Software Engineering of Yunnan Province, Yunnan University, Kunming* 650500, *China*）

Abstract: To improve the accuracy of software defect prediction, this paper proposed a software defect prediction method based on CS-ANN, which took advantage of parameters optimization power of the cuckoo search（CS） and non-linear computing power of the artificial neural network（ANN）. The method firstly used the feature selection algorithm based on the association rules to reduce the dimension of the original data sets and remove the noise attributes existing in data, and found the weights of the neural network by using the CS algorithm. Then the method built the defect prediction model using the weights and neural network algorithm. Finally, this paper completed defect prediction by the model. The simulation experiment was performed by using the NASA datasets that were published. The results show that this model can reduce the false alarm rate and improve the prediction accuracy, moreover, comprehensive evaluation area under the ROC curve（AUC）, *F*1-measure and G-mean values are superior to existing models.

Key words: software defect prediction; artificial neural network; cuckoo search; software quality; machine learnin

0 引言

软件测试花费的工作量经常比其他任何软件工程活动都多[1]。软件缺陷预测模型旨在识别有缺陷倾向的软件模块，从而权衡质量保证（quality assurance，QA）活动。有了软件缺陷预测模型，QA 测试小组就可以集中关注软件模块的部分子集，从而能在软件发布之前的有限时间和资源下，尽可能多地找出软件中存在的缺陷，提高软件的质量，并可以减少软件发布后维护的花费[2]。

软件缺陷预测是使用先前的软件度量属性和错误数据，预测下一个将发布的软件模块是否有缺陷。在预测模型中，软件度量属性作为自变量，而错误数据作为因变量[3]。软件度量属性是在软件开发过程中收集的，输入模型中作为训练的软件度量和错误数据是先前软件开发记录的历史数据。运用软件度量预测软件缺陷最早是由 Porter 和 Selby 在 1990 年开始的，随后就出现了大量的基于度量的缺陷预测。建立软件缺陷预测模型使用的方法主要有统计方法、机器学习和一些混合技术。采用机器学习构建预测模型是目前学者们研究的热点，机器学习方法比传统的统计学习方法有很大的优越性[2]。

近十年以来，出现了大量的对软件缺陷预测的研究工作，文献[2,3]详细地描述和总结了软件缺陷预测研究使用的技术和方法。特别是在文献[3]中，作者对软件缺陷预测的研究趋向进行了准确详细的分析，指出机器学习将会在软件缺陷预测领域运用得越来越广泛。基于决策树[4]、支持向量机[5]、随机森林[6]和朴素贝叶斯[7]等机器学习算法，都是常用的构建软件缺陷预测模型的算法。

在现有的研究基础上，本文提出了基于人工神经网络的软件缺陷预测方法，并结合寻优算法和特征选择算法，提高了软件缺陷预测的准确率。

收稿日期：2016-03-28；修回日期：2016-05-05 基金项目：国家自然科学基金资助项目（61379032，61462091，61262025）；云南省教育厅科学研究基金资助项目（2015Z018）；云南大学博士科研启动项目（XT412004）

作者简介：王海林（1989-），男，河南周口人，硕士研究生，主要研究方向为软件缺陷预测、软件工程；于倩（1975-），女，河南安阳人，讲师，博士，主要研究方向为软件过程、软件工程（yuqian2001cn@163.com）；李彤（1963-），男，河北石家庄人，教授，博士，主要研究方向为软件工程、软件演化、信息安全；郁湧（1981-），男，副教授，博士，主要研究方向为软件过程、可信软件；明利（1989-），女，河南安阳人，硕士研究生，主要研究方向为软件演化、软件过程；孙金文（1989-），男，山西朔州人，硕士研究生，主要研究方向为需求工程、智能算法.

1 背景和相关工作

软件缺陷预测的目标是提高软件质量和软件测试效率。软件缺陷预测使用的数据是在历史开发过程中收集的，McCabe[3]和Abran[3]方法是软件工程界使用最多的软件度量方法。McCabe是通过分析程序结构来计算软件模块的复杂度，Abran是根据程序中可执行代码的操作符和操作数的数量来计算模块的复杂度，两种度量都能有效反映软件缺陷的分布。

机器学习方法在各个领域应用得越来越广泛，尤其是基于人工神经网络(artificial neural network，ANN)的深度学习，成为机器学习研究中的一个新领域。人工神经网络在软件缺陷预测领域的研究也越来越多。文献[10]指出软件缺陷预测模型主要是基于软件度量属性与缺陷倾向之间的关系构建的，但这种关系是复杂且非线性的。人工神经网络算法具有很强的容错能力，并且有很强的处理非线性关系的能力[11]，可以构建非常复杂的函数和非线性的关系，所以使用神经网络构建软件缺陷预测模型具有极大的优势。文献[12~14]使用了神经网络算法构建软件缺陷预测模型，但是这些模型使用传统的方法寻找神经网络的权值，难以找到全局最优值。

一些启发式优化方法也被用来构建软件缺陷预测模型，如遗传算法[15]、粒子群算法[16]、人工蚁群算法[17]。启发式算法能够直观有效地解决复杂计算问题，在寻找问题最优解方面表现出了很好的性能。布谷鸟搜索(cuckoo search，CS)算法是一种新的启发式智能优化算法，相比其他优化算法，布谷鸟搜索算法有简单易行和参数少等优点，并且实验证明布谷鸟搜索算法的性能要远远优于遗传算法和粒子群算法。文献[18]也指出使用布谷鸟算法寻找神经网络的最优权值能够获得很好的结果。

另一方面，机器学习算法也受到数据质量的影响，数据不相关和冗余或者有噪声数据这些都是影响模型性能的因素。然而在软件开发过程中收集的软件度量属性难免不会引入这些因素[19]。

本文结合以上算法提出了基于CS-ANN的软件缺陷预测方法，同时使用了基于关联规则的特征选择算法对数据进行预处理。布谷鸟搜索算法用于寻找神经网络的最优权值，CFS能达到数据降维和处理噪声数据的效果，这样能有效处理ANN寻找最优权值的困难和质量差的数据对模型性能的影响，从而提高软件缺陷预测模型的性能。

2 基于CS-ANN的软件缺陷预测方法

2.1 特征选择算法

基于关联规则的特征选择算法是一种过滤器模式的特征选择方法。它根据各个属性的预测性能以及组间的关联寻找一个好的属性子集。通过CFS进行特征选择，能够有效地降低数据维度和处理噪声属性，提高机器学习算法的预测能力。CFS算法在性能上优于特征加权算法(ReliefF)和主成分分析(PCA)等特征选择算法[20]。本文重点关注的是模型的构建，对于使用的数据预处理方法不再详细介绍，文献[20]对CFS的原理进行了详尽的介绍。

2.2 人工神经网络

人工神经网络(artificial neural network，ANN)是一种模仿生物神经网络的结构和功能的计算模型[21]。ANN由大量的节点相互连接而成，每个节点代表一种特定的输出函数，称为激励函数(activation function)。两节点之间有加权值，称之为权值(weight)。ANN的学习过程就是通过迭代不断调整这些权值。一个神经网络系统由输入层、隐含层和输出层三部分组成。数据从输入层输入，每个节点乘以各自的权值，最后把这些相乘的结果相加，把这些相加得到的结果输入激励函数f，f计算出的结果t作为下一层的输入值再向与之关联的节点传播，这即是模仿生物信号在神经元之间的传播原理。使用神经网络作预测时，把最后一层的输出结果与设定的阈值进行比较，大于此阈值的数据分类为正类，小于此阈值的数据分类为负类。图1描述了ANN模型的基本结构。

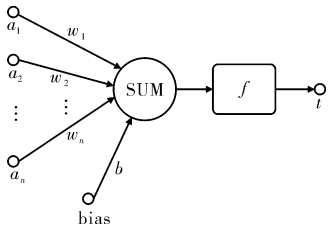

图1 人工神经网络模型的基本结构

ANN的激励函数可以是线性或非线性的。可以用特定的满足模型要解决的待解决问题的激励函数。第i层的输出值可以使用式(1)得到。

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right) \tag{1}$$

其中：y_i是一个节点的输出值；n是这个节点的总输入量；w_{ij}是这个节点与第j个输入直接的权值；x_j是第j个输入；b_i是偏移量；f_i是第i层的激励函数。本文使用的激励函数如式(2)所示。

$$f(z) = \frac{1}{1+e^{-z}} \tag{2}$$

神经网络的目标是通过迭代不断调整每一层的权值，最小化代价函数值(cost-function)。本文使用的损失函数如式(3)所示。

$$J(w) = -\frac{1}{m}\left[\sum_{i=1}^{m}\left(y_0^i \times \log f(wx^i) + (1-y_1^i) \times \log(1-f(wx^i))\right)\right] + \frac{\lambda}{2\times m}\sum_{l=1}^{n-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(w_{ij}^l)^2 \tag{3}$$

其中：m是训练数据集的总数；y_0和y_1分别表示负类和正类的值，本实验取0和1，分别代表无缺陷模块和有缺陷模块；$f(x)$是式(2)所示的激励函数；方程式的最后一项是正则化项，为了防止产生过拟合(over-fitting)现象，$\lambda \geqslant 0$是系数，用于权衡经验风险和模型复杂度，实验设置$\lambda = 1$；n是神经网络的总层数；w_{ij}^l是第l层的第i个节点与第$l+1$层的第j个节点之间的权值。

在机器学习领域，常用的寻找最优权值的方法有梯度下降法(gradient descent)、共轭梯度法(conjugate gradient)和BFGS法。但是这些方法难以找到全局的最优值，在实际实验中难以取得理想的结果，所以本文采用布谷鸟搜索算法来寻找最优权值。

2.3 布谷鸟搜索算法

布谷鸟搜索算法是基于布谷鸟寻找鸟窝放置鸟蛋的行为，并结合了一些鸟类或者果蝇的Lévy飞行行为而提出的一种智

能优化算法[22]。

CS 算法结合了局部随机过程和全局搜索过程。局部随机过程可以通过式(4)来描述。

$$x_i^{(t+1)} = x_i^{(t)} + \alpha s \oplus H(p_a - \varepsilon) \otimes (x_j^{(t)} - x_k^{(t)}) \tag{4}$$

其中：$x_i^{(t)}$ 表示第 i 个布谷鸟的第 t 个鸟窝(解决方案)，$x_j^{(t)}$ 和 $x_k^{(t)}$ 是随机打乱 $x_i^{(t)}$ 生成的两个序列；$\alpha > 0$ 是全局步长，它与求解问题的规模有关，多数情况下，可以使用 $\alpha = 1$，本文设置 $\alpha = 1$；s 是随机生成的局部游走步长；$H(x)$ 是 Heaviside 函数，当 $x < 0$ 时，$H(x) = 0$；当 $x > 0$ 时，$H(x) = 1$；当 $x = 0$ 时，$H(x) = 0.5$；⊕表示矩阵对应元素相乘；p_a 是在算法中设置的鸟窝主人发现外来鸟蛋的概率，文献[22]指出大多数的问题可以设置 $p_a = 0.25$，本文设置 $p_a = 0.25$；ε 是随机数；⊗表示矩阵相乘。

另一方面，全局随机过程可以按 Lévy 飞行过程执行。Lévy 飞行可以描述为一个运动的实体能够偶尔迈出异常大的步子，它的运动方向是随机的。Lévy 飞行的体征是小步的移动很多，但是间或有很大步的位移，这样就使得实体不会重复在一个地方搜索。运用这种飞行的特征，CS 算法能够在全局内搜索最优值。CS 算法中 Lévy 飞行使用式(5)执行。

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Lévy}(\lambda) \tag{5}$$

$$\text{Lévy} \sim u = t^{-\lambda} \tag{6}$$

式(5)中的 $x_i^{(t)}$ 与 α 与式(4)中的含义相同，式(5)本质上是随机游走的数学描述。一般地，随机游走是一个马尔可夫链，下一个状态/位置仅仅取决于当前的位置(以上方程式的首项)和转移概率(方程式的第二项)。Lévy 飞行的随机步长是从 Lévy 分布获取的，Lévy 分布如式(6)所示。它具有无限的方差和均值，而且相对于线性关系它的值增加得较快，这也是通过 Lévy 飞行非常有效的原因。

从数学上实现 Lévy 飞行最有效和直接的方式是通过 Mantegna 算法，在算法中步长 s 可以通过式(7)计算。

$$s = \frac{u}{|v|^{\frac{1}{\beta}}} \quad 0 < \beta < 2 \tag{7}$$

其中：u 和 v 服从正态分布。

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_u^2) \tag{8}$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \sigma_v = 1 \tag{9}$$

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt \tag{10}$$

$\beta = \frac{3}{2}$ 时是莱维分布的一个特殊形式，本文实验也取这个值，由式(9)(10)可计算出 σ_u 和 σ_v 的值，进而可以生成服从式(8)分布的随机数 u 和 v；然后根据式(7)计算出步长；最后根据式(5)，已知 $x_i^{(t)}$ 即可计算出 $x_i^{(t+1)}$ 的值。

2.4　构建基于 CS-ANN 的软件预测模型

CS-ANN 预测模型结合了人工神经网络和布谷鸟搜索算法，通过人工神经网络算法计算预测值以及损失函数值，然后使用布谷鸟搜索算法不断迭代，在设置的范围内寻找神经网络的最优权值，布谷鸟算法寻找的鸟窝即是神经网络算法的权值。CS-ANN 模型分为两个阶段，第一个阶段是训练阶段，第二个阶段是预测阶段。

训练阶段是使用历史数据对模型进行训练，输入模型的数据是经过 CFS 算法处理后的数据。将数据分为训练数据和测试数据，然后输入模型训练。训练过程如算法 1 所示。

算法 1　CS-ANN 训练过程

参数：神经网络算法正则化系数 λ，发现概率 p，搜索范围 $[Lb, Ub]$，迭代次数 iter，布谷鸟数量 n

输入：数据集 TrainData $\in R^{m \times r}$，testData $\in R^{n \times r}$。

输出：最优鸟窝 best_nest，最佳阈值 best_t。

```
设置参数
input_layer_size←r，hidden_layer_size←2 × r/* input_layer_size 是神经网络输入层的节点数，hidden_layer_size 是隐藏层的节点数* /
nd←(input_layer_size + 1) * hidden_layer_size + hidden_layer_size + 1 /* nd 表示神经网络总权值的个数，即是布谷鸟算法要寻找的鸟窝数量* /
for i = 1: n do /* 对 n 个布谷鸟的鸟窝进行初始化，每个布谷鸟的鸟窝即是一个解决方案，布谷鸟算法就是寻找出最优的解决方案* /
    使用 [Lb, Ub] 内的随机数初始化鸟窝 nest(i) ∈ R^{1×nd}
end for
将 input_layer_size、hidden_layer_size、λ，TrainData 和 nest 输入神经网络算法，使用式(3)计算每一个布谷鸟的鸟窝产生的代价 cost[n]，并得到当前代价最小的鸟窝 best_nest 和最小代价 cmin /* 首次计算出当前状态下的最佳鸟窝* /
while 迭代次数小于 iter do
    通过 Lévy 飞行方式在 [Lb, Ub] 内搜索新的鸟窝 new_nest[n] /* 布谷鸟数量不变，通过全局搜索方式寻找新的布谷鸟，新的布谷鸟对应新的鸟窝* /
    将 input_layer_size，hidden_layer_size，λ，TrainData 和 new_nest[n] 输入神经网络算法，使用式(3)计算每一个新鸟窝产生的代价 new_cost[n]，并得到当前代价最小鸟窝 new_best_nest 和最小代价 cnew
    for i = 1, …, n do
        若 new_cost[i] < cost[i]，用 new_cost[i] 替换 cost[i]，new_nest[i] 替换 nest[i] //好的解决方案替换差的解决方案
    end for
    若 cnew < cmin，用 cnew 替换 cmin，new_best_nest 替换 best_nest //保留最好的解决方案
    以概率 p 淘汰部分鸟窝，并以局部游走方式在 [Lb, Ub] 内寻找新的鸟窝替换被淘汰的鸟窝，得到新鸟窝 new_nest[n] /* 此过程通过方程式(4)实现，以概率 pa 淘汰通过式中的 s ⊕ H(p_a − ε) 部分实现* /
end while
使用 testData 和 best_nest 进行测试，得到测试结果
在 [0.000, 0.999] 间改变阈值，计算每个阈值下的 F1 值，保存使得 F1 值最大的阈值 best_t
```

预测阶段是使用新数据以及训练阶段得到的最优权值和最佳阈值输入神经网络算法预测模块是否有缺陷倾向，其中新数据的度量属性与训练数据的属性是相同的。

3　实验结果和分析

3.1　参数设置的说明

目前神经网络的隐层数和隐层节点数的确定尚无确定的理论依据，研究人员对隐层一般设置一层或两层，隐层节点个数一般是输入节点数的两倍。本文设置隐层为一层，设置隐层节点数是输入层的两倍。实验设置了不同的权值搜索范围，通过对比分析发现搜索范围在 $[-1, 1]$ 时，布谷鸟算法的收敛速度较快，找出的权值也较好，所以本文设置的搜索范围在 $[Lb, Ub] = [-1, 1]$；布谷鸟算法的提出者 Yang 通过大量仿真实验得出布谷鸟算法中的布谷鸟数量 n 在 $[15, 25]$ 间可满足大多数的优化问题，本文实验设置 $n = 25$；布谷鸟算法的结束条件可以设置一个阈值，即是找到的权值使得神经网络算法的代价函数计算的值小于这个阈值时停止迭代，或者设置迭代次数，本次实验中选择后者，设置迭代次数 iter $= 10^5$。实验中算法的

其他参数设置在前文中已经进行了说明。

3.2 模型的评价标准

如表 1 所示，是评价二分类预测模型常用的方法。有缺陷的模块标记为 1，无缺陷的模块标记为 0。若实际值为 1 且预测值也为 1，即是模型预测出了有缺陷的模块，则记为 TP；同样地，若实际值为 0 且预测值也为 0，即是模型正确预测模块无缺陷，则记为 *TN*。如果实际值为 1，预测值为 0，则模型把有缺陷的模块预测为无缺陷的模块，记为 *FN*；如果实际值为 0，预测值为 1，则模型把无缺陷的模块预测为有缺陷，记为 *FP*。根据这四个值，可以计算出以下定义的评价标准。

表 1 混淆矩阵

predicted	actual	
	1	0
1	*TP*(true positive)	*FP*(false positive)
0	*FN*(false negative)	*TN*(true negative)

3.2.1 *pd*、*pf*、*p* 和 acc

pd 是缺陷的概率(probability of detection，*pd*) ，也称召回率，可使用式(11) 计算，它表示有缺陷的模块被正确分类为有缺陷模块的数量和总样本中有缺陷模块总数的比例。

$$pd = \frac{TP}{TP + FN} \tag{11}$$

pf 是假预测的概率(probability of false alarm，*pf*) ，表示无缺陷模块被错误分类为有缺陷模块的数量和无缺陷模块总数的比例，可用式(12) 表示。

$$pf = \frac{FP}{FP + TN} \tag{12}$$

查准率(precision，*p*) 如式(13) 所示，它表示有缺陷模块被正确分类为有缺陷模块的数量和被分类为有缺陷模块总数的比例。

$$p = \frac{TP}{TP + FP} \tag{13}$$

准确率(accuracy，acc) 表示被正确分类的模块数量和总模块数量的比例，可用式(14) 计算。

$$\text{acc} = \frac{TP + TN}{TP + FP + TP + TN} \tag{14}$$

3.2.2 *F*1 值、G-mean、ROC 和 AUC

*F*1 值定义为查准率 *p* 与召回率 *pd* 之间的调换平均数。它综合考虑了相互矛盾的 *p* 和 *pd*，*F*1 值高则说明模型比较理想，不仅能够找出更多的缺陷，且无缺陷模块分类为有缺陷的错误较低。*F*1 可用式(15) 计算。

$$F1 = \frac{2p \times pd}{p + pd} \tag{15}$$

由表 1 可以看出数据是不平衡的，仅用准确率不能准确地评估模型对不平衡数据的测试性能。在不平衡数据的环境中，G-mean 和 AUC 是常用的评估预测器的方法[23]。在实验中，把有缺陷的作为正类(positive class) ，把无缺陷的作为负类(negative class) 。G-mean 表示的是正类和负类的召回率的几何平均数。一个好预测器应该具有高准确率和 G-mean。G-mean 高说明正确分类有缺陷模块和无缺陷模块的概率都很高。在软件缺陷预测中，G-mean 可用式(16) 计算。

$$\text{G-mean} = \sqrt{pd \times (1 - pf)} \tag{16}$$

受试者工作特征曲线(receiver operating characteristic curve，ROC 曲线) 是由变更所有可能的阈值产生的坐标点(*pf*，*pd*) 而生成的曲线，它反映的是 *pd* 与 *pf* 之间的权衡。ROC 曲线也被广泛应用于评估不同的临床计算系统和机器学习方法等其他领域。AUC 值是 ROC 曲线下的面积，AUC 的值越高就说明分类器的性能越好。

3.3 实验数据

NASA MDP(metrics data program) 是由美国航空航天局发布的专门用于构建软件缺陷预测模型的公开数据集，是实验最常用的数据。文献 [24] 指出 NASA 提供的原始数据集并不适合直接用于数据挖掘，需要经过数据清理(data cleansing) 才适合用于实验。所以本实验数据是从 tera-PROMISE[25] 数据库中选取的经过数据清理后的八个 NASA 数据集。表 2 中分别列出了这些数据集的属性。由表可以看出，这些数据中依然存在过多的属性，数据清理过程只是对数据中的重复数据和矛盾数据进行了处理，但是其中依然存在过多的冗余属性，造成了数据的高维度，这些冗余属性会降低预测器的性能，所以需要进一步对数据进行预处理。

表 2 数据集的属性

dataset	language	metrics/#	instances	defect/%
CM1	C	38	327	12.8
KC2	C ++	22	522	20.5
KC3	Java	40	194	18.6
MC1	C	39	327	12.8
PC1	C	38	705	8.7
PC3	C	38	1 077	12.4
PC4	C	38	1 458	12.2
JM1	C	22	7 782	21.5

本次研究使用 WEKA 工具[26] 中的 CFS 算法进行数据预处理。表 3 是处理后的每个数据集的属性。文献 [8，9] 详细介绍了每个属性的含义以及计算方法。

表 3 数据选择的属性

dataset	select metrics
CM1	loc_comments，loc_executable，halstead_content，num_unique_operators，percent_comments
KC2	essential_complexity，intelligence，halstead_effort，num_unique_operands，num_unique_operators
KC3	branch_count，loc_code_and_comment，normalized_cylomatic_complexity
MC1	call_pairs，loc_code_and_comment，loc_comments，cyclomatic_density，normalized_cylomatic_complexity，num_unique_operators，number_of_lines，percent_comments
PC1	loc_blank，loc_code_and_comment，loc_comments，cyclomatic_density，loc_executable，parameter_count，halstead_content，normalized_cylomatic_complexity
PC3	loc_blank，loc_code_and_comment，loc_comments，halstead_content，halstead_length，maintenance_severity，normalized_cylomatic_complexity，num_unique_operands，percent_comments
PC4	loc_code_and_comment，condition_count，parameter_count，percent_comments
JM1	loc_blank，loc_code_and_comment，loc_comments，design_complexity，halstead_content，halstead_effort，halstead_volume，num_unique_operands，num_unique_operators，loc_total

3.4 结果及分析

在实验中，使用十折交叉验证来评估模型。将每一个数据分成 10 份，轮流将其中九份作为训练数据，一份作为测试数据来进行实验，如此循环 10 次。在每次循环的测试中，采用计算 *F*1 值的方法确定最佳阈值，使得 *F*1 值最大的阈值设置为此数

据集的最佳阈值，目的是为了保证预测器能够找出更多的缺陷，并且使得把无缺陷模块分类为有缺陷模块的错误降到最低；然后计算在此阈值下的评价指标，作为此次循环的测试结果；最后取 10 次结果的均值作为对模型的评估。实验流程如图 2 所示。

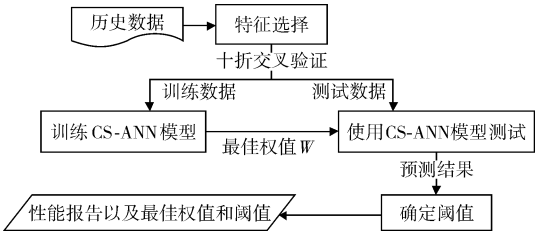

图 2　实验流程

如表 4 所示，所有数据集的最佳阈值，以及在最佳阈值下的 pd、pf、acc、$F1$、G-mean、AUC 值，是 10 次循环结果的均值和方差。

表 4　八个数据集产生的最佳阈值和 pd、pf、acc、$F1$、G-mean、AUC 的均值和方差

数据集	阈值	pd	pf	acc/%	$F1$	G-mean	AUC
CM1	0.544	0.760 ± 0.200	0.243 ± 0.160	76.4 ± 0.125	0.481 ± 0.112	0.740 ± 0.062	0.729 ± 0.104
KC2	0.577	0.755 ± 0.167	0.139 ± 0.089	84.7 ± 0.057	0.670 ± 0.064	0.798 ± 0.069	0.849 ± 0.049
KC3	0.546	0.852 ± 0.178	0.213 ± 0.258	79.8 ± 0.196	0.673 ± 0.215	0.796 ± 0.155	0.753 ± 0.187
MC1	0.532	0.451 ± 0.156	0.036 ± 0.059	95.2 ± 0.057	0.397 ± 0.183	0.648 ± 0.103	0.815 ± 0.091
PC1	0.554	0.685 ± 0.225	0.089 ± 0.081	88.8 ± 0.063	0.505 ± 0.142	0.776 ± 0.115	0.857 ± 0.060
PC3	0.552	0.666 ± 0.181	0.173 ± 0.099	80.7 ± 0.069	0.460 ± 0.071	0.730 ± 0.063	0.815 ± 0.038
PC4	0.572	0.653 ± 0.156	0.073 ± 0.045	89.5 ± 0.029	0.599 ± 0.066	0.771 ± 0.081	0.901 ± 0.028
JM1	0.555	0.609 ± 0.072	0.348 ± 0.075	64.3 ± 0.045	0.423 ± 0.027	0.626 ± 0.018	0.654 ± 0.020
mean		0.679	0.164	82.4	0.526	0.736	0.797

由表 4 可以看出，对所有数据集的 AUC 值都大于 0.5，这就说明预测器的性能是可被接受的[27]。文献[28]研究指出通过人工代码审查只能查出 60% 左右的缺陷，预测模型的准确率必须在 60% 以上才具有实用性。本文提出模型的召回率和准确率都在 60% 以上，说明预测模型具有实用性。

本实验设置了反向传播神经网络(BP)、随机森林(RF)、朴素贝叶斯(NB)和 C4.5 算法四种常用的构建软件缺陷预测模型的算法作为对比，对于每一种算法同样采用十折交叉验证的方法，最后计算评价指标的均值。为了对比不同模型的稳定性，根据模型对八个数据集测试得到的评价指标绘制盒图[29]，如图 3 所示。

由图 3 可以看出，CS-ANN 的 AUC 值要高于其他四种预测模型，而图中 accuracy 显示其他算法(如 BP)的准确率要略高于 CS-ANN，这是由于数据的不平衡造成的，同时也说明了在不平衡数据的环境下，用准确率不能准确反映预测模型的性能。由图 3 的整体性能可以看出，CS-ANN 模型的稳定性较好。

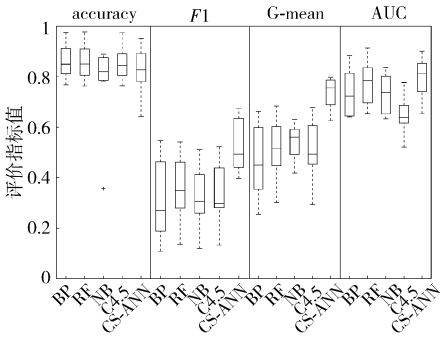

图 3　不同算法构建模型的性能指标盒图

为了显示更多整体性能的细节，表 5 ~ 7 列出了由八个数据集分别产生的不同算法的 $F1$、G-mean 和 AUC 的均值。表中每行使用黑体标注的是该数据集对应的该指标的最优值。由表 5 和 6 可以看出，CS-ANN 对每一个数据产生的 $F1$ 值和 G-mean 值都要高于其他四种算法，说明模型正确预测负类和正类的性能要优于其他四种算法，分类时产生的误分类要低于其他四种算法。BP 算法的准确率虽然高，但是 $F1$ 值和 G-mean 值要比 CS-ANN 模型分别低 20% 和 14% 左右，说明单独使用神经网络算法不能很好地处理数据不平衡问题，分类时找出较多有缺陷模块的同时也产生了较多的误分类。表 7 表明部分数据使用 RF 算法产生的 AUC 略高于 CS-ANN 模型，但平均 AUC 值比 CS-ANN 低两个百分点，说明 CS-ANN 模型的稳定性要优于 RF 算法。

表 5　数据集分别产生的五种算法的 $F1$ 值

数据集	BP	RF	NB	C4.5	CS-ANN
CM1	0.206	0.136	0.256	0.293	**0.481**
KC2	0.546	0.541	0.511	0.522	**0.670**
KC3	0.286	0.355	0.406	0.375	**0.673**
MC1	0.109	0.308	0.120	0.133	**0.397**
PC1	0.417	0.396	0.344	0.271	**0.505**
PC3	0.254	0.251	0.262	0.302	**0.460**
PC4	0.509	0.527	0.418	0.501	**0.599**
JM1	0.170	0.341	0.268	0.291	**0.423**
mean	0.312	0.357	0.323	0.336	**0.526**

表 6　数据集分别产生的每一种算法的 G-mean 值

数据集	BP	RF	NB	C4.5	CS-ANN
CM1	0.395	0.301	0.481	0.492	**0.740**
KC2	0.662	0.684	0.630	0.666	**0.798**
KC3	0.471	0.526	0.585	0.547	**0.796**
MC1	0.255	0.465	0.542	0.294	**0.648**
PC1	0.566	0.538	0.579	0.454	**0.776**
PC3	0.430	0.429	0.504	0.494	**0.730**
PC4	0.631	0.668	0.599	0.677	**0.771**
JM1	0.313	0.505	0.418	0.453	**0.626**
mean	0.465	0.515	0.542	0.510	**0.736**

表 7　数据集分别产生的每一种算法的 AUC 值

数据集	BP	RF	NB	C4.5	CS-ANN
CM1	0.641	0.699	0.644	0.622	**0.729**
KC2	0.829	0.786	0.832	0.704	**0.849**
KC3	0.643	0.694	0.661	0.653	**0.753**
MC1	0.653	**0.841**	0.708	0.521	0.815
PC1	0.793	0.830	0.771	0.669	**0.857**
PC3	0.800	0.782	0.766	0.616	**0.815**
PC4	0.885	**0.914**	0.836	0.777	0.901
JM1	0.645	0.653	0.633	0.616	**0.654**
mean	0.736	0.775	0.731	0.647	**0.797**

文献[30]构建了基于多核技术和集成学习(MEKL)的软件缺陷预测模型,旨在解决由于数据不平衡导致的误分类问题。图4所示为使用MEKL和CS-ANN模型对七个数据集产生的$F1$值及$F1$值的均值的折线图。Mean表示七个$F1$值的平均值,MEKL模型的数据取自文献[30]。通过对比说明CS-ANN能够很好地处理数据不平衡问题。文献[31]提出了结合遗传算法和神经网络算法(NN GAPO + B)的软件缺陷预测模型。图5为使用NN GAPO + B和CS-ANN模型对五个数据集产生的AUC值及AUC值的均值的折线图。Mean表示五个AUC值的平均值,NN GAPO + B模型的数据取自文献[31]。由图5可以明显地看出CS-ANN模型的性能优于NN GAPO + B预测模型。

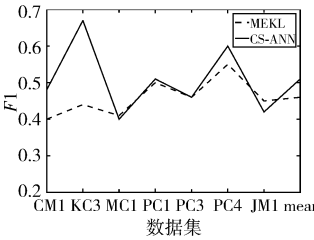

图4 CS-ANN模型的$F1$值对比

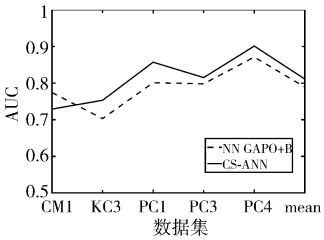

图5 CS-ANN模型的AUC值对比

4 结束语

本文结合布谷鸟搜索算法和人工神经网络算法构建了软件缺陷预测模型。布谷鸟搜索算法是基于布谷鸟寻找鸟窝放置鸟蛋的行为而提出的一种智能搜索算法。使用布谷鸟搜索算法能够很好地寻找人工神经网络的最优权值。模型采用最大化$F1$值的方法确定最佳阈值,使用$F1$值作为判断最佳阈值的标准,能够很好地保证模型在找出更多缺陷的同时把误分类的概率降到最低。在公开的基准数据集上进行了实验,分别设置了传统算法和其他研究者提出算法的对比,证明了本文提出的CS-ANN软件缺陷预测模型具有更好的精准率和稳定性。然而从实验中也可以看出CS-ANN模型在某些数据集上的性能偏低,这是由于数据特征的差异造成的,需要寻找更好的数据预处理方法来提高软件缺陷预测模型的稳定性,这将是下一步工作的重点。

参考文献:

[1] Pressman R S. Software engineering: a practitioner's approach [M]. [S. l.]: Palgrave Macmillan, 2005: 449-479.

[2] Malhotra R. A systematic review of machine learning techniques for software fault prediction [J]. Applied Soft Computing, 2015, 27: 504-518.

[3] Catal C. Software fault prediction: a literature review and current trends [J]. Expert Systems with Applications, 2011, 38(4): 4626-4636.

[4] Siers M J, Islam M Z. Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem [J]. Information Systems, 2015, 51: 62-71.

[5] Laradji I H, Alshayeb M, Ghouti L. Software defect prediction using ensemble learning on selected features [J]. Information & Software Technology, 2015, 58: 388-402.

[6] Pushphavathi T P, Suma V, Ramaswamy V. A novel method for software defect prediction: hybrid of FCM and random forest [C]//Proc of International Conference on Electronics and Communication Systems. [S. l.]: IEEE Press, 2014: 1-5.

[7] Okutan A, Yildiz O T. Software defect prediction using Bayesian networks [J]. Empirical Software Engineering, 2014, 19(1): 154-181.

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

[9] Abran A. Software metrics and software metrology [M]. [S. l.]: Wiley, 2010: 145-191.

[10] Kaur R, Bajaj P. A review on software defect prediction models based on different data mining techniques [J]. International Journal of Computer Science and Mobile Computing, 2014, 3(5): 879-886.

[11] Gondra I. Applying machine learning to software fault-proneness prediction [J]. Journal of Systems & Software, 2008, 81(2): 186-195.

[12] Jin Cong, Jin S W, Ye Junmin. Artificial neural network-based metric selection for software fault-prone prediction model [J]. IET Software, 2012, 6(6): 479-487.

[13] Zheng Jun. Cost-sensitive boosting neural networks for software defect prediction [J]. Expert Systems with Applications, 2010, 37(6): 4537-4543.

[14] Gayathri M, Sudha A. Software defect prediction system using multilayer perceptron neural network with data mining [J]. International Journal of Recent Technology and Engineering, 2014, 3(2): 54-59.

[15] Li Kewen, Chen Chenxi, Liu Wenying, *et al*. Software defect prediction using fuzzy integral fusion based on GA-FM [J]. Wuhan University Journal of Natural Sciences, 2014, 19(5): 405-408.

[16] De Carvalho A B, Pozo A, Vergilio S R. A symbolic fault-prediction model based on multiobjective particle swarm optimization [J]. Journal of Systems and Software, 2010, 83(5): 868-882.

[17] Azar D, Vybihal J. An ant colony optimization algorithm to improve software quality prediction models: case of class stability [J]. Information and Software Technology, 2011, 53(4): 388-393.

[18] Valian E, Mohanna S, Tavakoli S. Improved cuckoo search algorithm for feedforward neural network training [J]. International Journal of Artificial Intelligence & Applications, 2011, 2(3): 36-43.

[19] Wang Huanjing, Khoshgoftaar T M, Napolitano A. A comparative study of ensemble feature selection techniques for software defect prediction [C]//Proc of the 9th International Conference on Machine Learning and Applications. [S. l.]: IEEE Press, 2010: 135-140.

[20] Hall M A. Correlation-based feature selection for discrete and numeric class machine learning [C]// Proc of the 17th International Conference on Machine Learning. San Francisco: Morgan Kaufmann Publishing, 2000: 359-366.

[21] Hagan M T, Demuth H B, Beale M H. Neural network design [M]. Boston: Martin Hagan, 1996: 12-20.

[22] Yang Xinshe, Deb S. Cuckoo search via Lévy flights [C]//Proc of World Congress on Nature & Biologically Inspired Computing. [S. l.]: IEEE Press, 2009: 210-214.

[23] Wang Shuo, Yao Xin. Using class imbalance learning for software defect prediction [J]. IEEE Trans on Reliability, 2013, 62(2): 434-443.

(下转第476页)

间，提高了查询处理效率。

表 2 表示了在面对查询语句中不同查询词数量时，两种查询算法在不同索引结构下的查询时间。从表中可以看到，随着查询词数量的增大，多层自索引结构对查询效率的提升越大。特别是对 MaxScore 算法来讲，在使用多层自索引结构以后，查询词数量基本不会对查询时间造成影响。

表 2 不同查询词数量的查询时间

algorithm	avg	2	3	4	5	>5
WAND	66.7	34.5	56.6	65.5	77.2	99.7
WAND_s	49.4	30.6	46.9	47.7	55.4	66.3
MaxScore	59.4	35.2	50.9	59	66.5	85.6
MaxScore_s	25.4	25.5	25.8	25.1	25.3	25.1

搜索引擎通常不需要返回全部的相关文档，WAND 和 MaxScore 算法的查询结果均为 top-k。图 6 说明了当 k 增大时，多层自索引结构的性能表现。可以观察到，不管是否采用多层自索引结构，查询时间均随 k 值的增大而变长。这是因为返回结果数量的增加意味着结果集文档得分阈值的下降，使得评估文档数增多。但是无论 k 取值如何，WAND_s、MaxScore_s 都分别优于 WAND、MaxScore 算法。

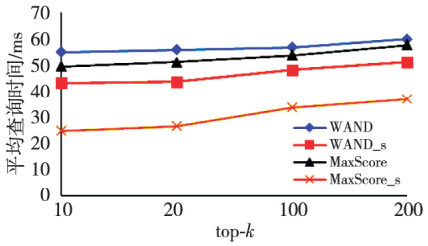

图 6 当 k 值增大时各查询算法的查询时间

5 结束语

本文首先介绍了倒排索引及自索引结构的相关概念；然后设计了具有多层自索引结构的索引压缩与查询系统，实现了 DAAT 中两种经典的动态剪枝算法——WAND 和 MaxScore 算法；最后通过与不带跳转结构索引的多组对比实验，证明了倒排链表多层自索引结构的有效性和高效性。本系统能够在返回相同结果集的前提下，使查询时间减少 21.1%（WAND）和 49.8%（MaxScore）。

参考文献：

[1] Moffat A, Zobel J. Self-indexing inverted files for fast text retrieval [J]. ACM Trans on Information Systems, 1996, 14(4): 349-379.

[2] Campinas S, Delbru R, Tummarello G. SkipBlock: self-indexing for block-based inverted list [C] //Advances in Information Retrieval. Berlin: Springer, 2011: 555-561.

[3] Chierichetti F, Lattanzi S, Mari F, *et al*. On placing skips optimally in expectation [C] //Proc of International Conference on Web Search and Data Mining. New York: ACM Press, 2008: 15-24.

[4] Zhang Jiangong, Long Xiaohui, Suel T. Performance of compressed inverted list caching in search engines [C] //Proc of the 17th International Conference on World Wide Web. New York: ACM Press, 2008: 387-396.

[5] Jonassen S, Bratsberg S E. Efficient compressed inverted index skipping for disjunctive text-queries [C] //Advances in Information Retrieval. Berlin: Springer, 2011: 530-542.

[6] Jonassen S, Bratsberg S E. Improving the performance of pipelined query processing with skipping—and its comparison to document-wise partitioning [J]. World Wide Web, 2014, 17(5): 949-967.

[7] 单栋栋. 搜索引擎中索引剪枝的研究 [D]. 北京: 北京大学, 2013.

[8] Jiang Kun, Yang Yuexiang. Efficient dynamic pruning on largest scores first (LSF) retrieval [J]. Frontiers of Information Technology & Electronic Engineering, 2016, 17(1): 1-14.

[9] 刘小珠，彭智勇，陈旭. 高效的随机访问分块倒排文件自索引技术 [J]. 计算机学报，2010，33(6)：977-987.

[10] Broder A Z, Carmel D, Herscovici M, *et al*. Efficient query evaluation using a two-level retrieval process [C] //Proc of the 12th International Conference on Information and Knowledge Management. New York: ACM Press, 2003: 426-434.

[11] Turtle H, Flood J. Query evaluation: strategies and optimizations [J]. Information Processing & Management, 1995, 31(6): 831-850.

[12] Yan Hao, Ding Shuai, Suel T. Inverted index compression and query processing with optimized document ordering [C] //Proc of the 18th International Conference on World Wide Web. New York: ACM Press, 2009: 401-410.

（上接第 472 页）

[24] Gray D, Bowes D, Davey N, *et al*. The misuse of the NASA metrics data program data sets for automated software defect prediction [C] //Proc of the 15th Annual Conference on Evaluation & Assessment in Software Engineering. [S. l.]: IET, 2011: 96-103.

[25] http://openscience.us/repo/defect/mccabehalsted/ [EB/OL].

[26] Holmes G, Donkin A, Witten I H. WEKA: a machine learning workbench [C] //Proc of the 2nd Australian and New Zealand Conference on Intelligent Information Systems. [S. l.]: IEEE Press, 1994: 357-361.

[27] Menzies T, Stefano J S D. How good is your blind spot sampling policy [C] // Proc of the 8th IEEE International Symposium on High Assurance Systems Engineering. 2004: 25-26.

[28] Shull F, Basili V, Boehm B, *et al*. What we have learned about fighting defects [C] //Proc of IEEE International Symposium on Software Metrics. [S. l.]: IEEE Computer Society, 2002: 249.

[29] Wohlin C, Runeson P, Höst M, *et al*. Experimentation in software engineering [M]. [S. l.]: Springer Science & Business Media, 2012.

[30] Wang Tiejian, Zhang Zhiwu, Jing Xiaoyuan, *et al*. Multiple kernel ensemble learning for software defect prediction [J]. Automated Software Engineering, 2016, 23(4): 569-590.

[31] Wahono R S, Herman N S, Ahmad S. Neural network parameter optimization based on genetic algorithm for software defect prediction [J]. Advanced Science Letters, 2014, 20(10-12): 1951-1955.