无监督缺陷模块序列预测模型：一个工作量感知的评价

赵东晓，周毓明
（南京大学计算机科学与技术系，南京 210046）

摘要：无监督的缺陷模块序列预测模型不需要历史缺陷信息，能够将模块按照包含缺陷的可能性从大到小排序。在得到这种潜在缺陷模块序列之后，人们可根据现有资源情况从该序列中选取排在前面的模块进行审查/测试。为探明这种缺陷模块序列预测模型的实用价值和影响因素，该文在代表性的无监督学习方法 K-Means 和 X-Means 的基础上，提出多种簇间/簇内排序方法，通过联合使用簇间排序和簇内排序得到潜在缺陷模块序列，然后采用工作量感知的评价指标对序列的预测性能进行评价。在多个开源系统上的实验结果表明，在特定的簇间/簇内排序方法下，这种预测模型显著高于随机猜测模型，因此具有实用价值。特别地，簇间排序比簇内排序对缺陷预测性能具有更大的影响。

关键词：软件工程；易错性；预测；聚类；排序；工作量敏感
中图分类号：TP311.5

Using unsupervised models to predict fault-proneness ranking for modules: an effort-aware experimental evaluation

Zhao Dongxiao, Zhou Yuming
(Department of Computer Science and Technology, Nanjing University, Nanjing 210046)

Abstract: Unsupervised fault-proneness ranking prediction models could rank modules according to their predicted fault-proneness in descending order without historical fault information. With such a ranking list in hand, developers can simply select as many modules from the top of the list for inspecting or testing as available resources will allow. This paper employs K-Means and X-Mean, two representative unsupervised methods, to develop unsupervised fault-proneness ranking prediction models. To this end, this paper first uses the combination of inter-cluster ranking and inner-cluster ranking to obtain a ranking list of modules from the most fault-prone to the least fault-prone. Then, this paper employs an effort-aware performance indicator to evaluate the prediction performance of this module ranking. Furthermore, this paper examines the combinations of different inter-cluster and inner-cluster ranking methods. The objective is to examine the actual usefulness of such an unsupervised ranking model and the factors that have an influence on its prediction accuracy. The experimental results from several open-source systems show that such an unsupervised ranking model significantly outperforms random model when appropriate inter-cluster ranking and inner-cluster ranking are used. In particular, the experimental results also show that the inter-cluster ranking has a larger influence on the prediction performance of fault-proneness ranking than inner-cluster ranking.

Keywords: software engineering; fault-proneness; prediction; cluster; ranking; effort-aware

0 引言

在软件开发过程中，软件缺陷的产生不可避免，而软件缺陷的存在有可能会带来灾难[1]。信息社会的到来，使得软件需求量不断增大，新开发的软件也大量增加，这些软件因为开发周期的变短导致的软件缺陷也不断增加，软件开发者在软件测试上花费的代价越来越大。软件缺陷预测作为一种有效的软件质量控制手段，成为近几年的研究热点[2-9]。

作者简介：赵东晓，（1981-），男，硕士研究生，主要研究方向：软件度量。
通信联系人：周毓明，教授，主要研究方向：实证软件工程。E-mail: zhouyuming@nju.edu.cn

目前已有大量的软件缺陷预测研究，但这些模型都或多或少的存在一定的问题。首先，对当前最为流行的有监督缺陷预测模型而言，它们的建模过程需要软件的历史缺陷信息，不适用于缺乏这些信息的新开发软件[10-14]。其次，现有的无监督缺陷预测模型大都用于分类，简单地将待预测样本分为有缺陷类和无缺陷类，没有按照样本的缺陷密度进行排序，不能提供软件模块的缺陷倾向性序列，不利于软件缺陷的审查和验证[15-18]。再次，对当前的半监督缺陷预测模型而言，它们的建模过程需要有专业人员参与，由这些专家负责对少量的缺陷信息进行标记，显然这种模型的预测效果受限于辅助专家的经验。同时，这部分缺陷信息的标记仍然需要付出相当大的代价[19-21]。特别地，大多数缺陷预测模型都采用基于混淆矩阵的评价指标，如预测准确率、精度、召回率、F 值等[15-18,20,22-25]。这种评价标准只考虑了模型的预测能力，没有考虑使用这些模型指导代码检测和审查时的工作量与预测结果的关系，无法保证应用这些模型时的成本效益。这些问题的存在，使得这些预测模型的应用受到很大的限制，因此有必要针对这些不足寻找一种更合理的预测模型。

与已有研究不同的是，本文提出一种基于聚类的、不需专家人员辅助的软件缺陷序列预测模型，并采用了工作量敏感的评价指标对该模型进行评价。我们的做法是，首先选用 K-Means 和 X-Means 算法对待预测样本进行聚类，然后对聚类后的类簇进行第一次排序确定类簇的缺陷倾向递减序列，对类簇内样本进行第二次排序确定缺陷密度递减序列，将经过两步排序后的样本序列作为软件缺陷倾向预测序列。整个预测过程自动完成，不需要有专业人员参与预测。最后结合软件工业实际，考虑了代码审查或检测时的成本效益，采用了 CE（Cost Effectiveness）指标对模型性能进行评价。本文的主要工作如下：

（1）设计了四种基于聚类的缺陷序列预测排序方法；

（2）采用 CE 评价指标对基于这四种排序方法的缺陷序列预测模型进行评价；

（3）通过比较采用不同排序算法情况下缺陷预测序列的性能，指出了不同的排序算法对缺陷预测序列构造的影响。

本文主要结构如下：首先简要介绍软件缺陷预测的有关概念和研究现状；第一节介绍软件缺陷预测研究的相关工作；第二节介绍基于 K-Means 和 X-Means 的软件缺陷预测模型及建模步骤；第三节介绍工作量感知的缺陷预测性能评价指标；第四节报告实验结果；最后一节给出本文的结论。

1 相关工作

软件缺陷序列预测一般包含三个步骤，首先是建立软件缺陷预测模型，然后利用建立的模型选择合适的度量对待预测对象包含缺陷的几率进行预测，最后根据预测结果按照包含缺陷的几率从大到小的顺序将各对象进行排序，最终得到的序列即为软件缺陷序列预测的结果。在这三个步骤中，大量的工作集中在了缺陷序列预测模型的建立和缺陷序列预测模型性能的评价上，这两步也是软件缺陷序列预测中的重要步骤。

缺陷预测模型普遍采用机器学习方法进行建模。其中，许多研究工作使用有监督学习算法建立缺陷预测模型，例如逻辑回归、决策树、支持向量机、朴素贝叶斯、人工神经网络等算法[10-14]。因为有监督学习算法在缺陷预测领域的应用的不足，研究者开始尝试用其他类型的机器学习算法进行软件缺陷的预测，首先被引入的想法就是半监督的思想。在 2004 年，Zhong 等人应用 K-Means 和 Neural-Gas 算法进行缺陷预测，让辅助专家参与建模过程，他们分别在 NASA 的 KC2 和 JM1 数据集上建立了缺陷预测模型，并进行了实验验证。他们的实验结果表明，在大型数据集和有噪音数据集上 K-Means 性能比 Neural-Gas 聚类算法要好，

但 Neural-Gas 聚类算法具有更好的普适性，其在大部分数据集上都有着较低的平均误差[17,18]。同样地，Seliya 等提出了一种基于约束的半监督聚类方案，该方案以 K-Means 聚类算法为基础[20]。他们发现，跟纯粹的基于无监督学习算法的预测模型相比，这种方法更能够帮助参与预测的专业人员做出更好的估计。Catal 等以 Fuzzy C-Means 算法为基础建立了缺陷预测模型，他们的实验结果表明该算法与 K-Means 算法聚类效果相当[26]。

对上述这些采用半监督学习思想建立的缺陷预测模型而言，需要有辅助专家参与聚类后类簇的缺陷信息标记。然而，获得缺陷信息标记需要付出相当高的代价，同时模型的预测性能受专家经验的影响很大。为此，人们分别在降低缺陷信息标记代价和摆脱辅助专家依赖方面做出了尝试。文献[21]在降低缺陷信息的标记代价方面做出了尝试。他们提出了一种两阶段主动学习算法，首先利用聚类算法得到预测数据集中的代表信息样本，然后利用代表信息样本结合支持向量机算法得到关键信息样本，最后对整个数据集进行有监督的分类学习；由于该算法仅需要对代表信息样本进行缺陷信息的标记，从而减少了缺陷信息标记的代价。文献[16]在摆脱辅助专家依赖方面做出了研究，他们提出一种自动确定聚类后类簇倾向的方法。他们利用度量的阈值作为倾向标记的指标，首先应用 K-Means 算法确定类簇，然后通过比较类簇中各度量的均值与度量阈值向量中对应的阈值大小来确定类簇的缺陷倾向。为了解决 K-Means 算法中聚类中心数目难以确定的问题，Catal 等提出了一种基于 X-Means 算法的缺陷预测模型[15]。他们的实验结果表明，该算法对不包含噪音的数据具有很好的性能。然而，在存在噪音的数据上，该算法会导致不合理的聚类。

不论是基于半监督的软件缺陷预测建模，还是以基于无监督的缺陷预测建模，总是以提高预测性能为目标，众多研究者也在这个命题下对各种提高预测性能的方法进行了研究。文献[22]利用基于四叉树的 K-Means 聚类算法对缺陷预测进行了研究，他们首先利用四叉树算法来确定 K-Means 聚类时的类簇中心点，然后利用 K-Means 算法对数据进行聚类，最后将这种结合后的算法应用到软件缺陷预测领域。他们的实验结果表明，这种算法在大多数情况下能够降低缺陷预测的错误率。文献[23]提出了一种基于模糊聚类非负矩阵分解的缺陷预测方法，该方法解决了最近邻算法处理多属性缺陷数据时性能偏低的问题，克服了非负矩阵分解时易陷入局部最优的不足，提高了不平衡软件缺陷数据的预测精度。文献[24]利用了程序开发人员相关的一些信息进行聚类，对软件缺陷进行预测。他们发现，这种采用非程序代码本身直接相关的信息对软件缺陷预测的方法的准确率偏低。文献[25]提出了一种谱聚类与混沌免疫相结合的软件缺陷预测方法，用混沌免疫聚类算法替换谱聚类中的 K-Means 算法。他们的研究表明，这种结合的算法可以促进软件缺陷数据预测精度的提高。

对这些已有的预测模型而言，虽然实验结果表明具有较高的预测性能，但是它们大都采用基于混淆矩阵的评价指标，主要通过考察预测结果的假阳性率、真阳性率(召回率)、精度、准确率、F 值(精度和召回率的调和平均数)和 AUC 等指标来判断模型的性能。这种评价方法只考虑模块是否包含缺陷，忽略了模块规模的不同，将各个规模的模块同等对待。显然，这与代码审查或测试阶段不同规模模块对工作量的要求不同的事实不符。因此，人们提出应该使用一种“基于代码行的 Alberg 图”的评价方法来评价潜在预测模型的性能[27-30]。在 2010 年，Arisholm 等对在使用不同度量建模后的模型性能评价方法进行了详细的比较[27]。他们的研究显示，代码度量和过程度量等不同度量集的缺陷预测能力在使用 AUC 这一指标进行评价时在统计上没有显著的差别。然而，当使用基于代码行的 Alberg 图来评价它们的潜在缺陷模块序列预测性能时，应用这些不同的度量建立的模型在性能上有显著差异，这一实验结果也为 Mende 等的研究所证实[30]。更进一步，Arisholm 等在“基于代码行的 Alberg 图”

上还提出应该用 CE 指标作为模型性能的评价指标[27]。

2 无监督的缺陷模块序列预测模型

2.1 K-Means 聚类算法

K-Means 算法是一种基于样本间相似性度量的聚类算法，此算法以提前设置的聚类个数 k 为参数，把需要聚类的 n 个样本对象分到 k 个类簇之中。聚类的目的是使得类簇内部具有较高的相似性，而类簇之间具有较低的相似性[31]。在 K-Means 聚类算法中，通常用两点之间的距离来衡量两点的相似性。K-Means 算法的基本步骤为：

(1) 从 n 个样本对象任意选择 k 个对象作为初始聚类中心；

(2) 计算每个样本与这些中心的距离，将样本划入具有最小距离的中心所在类簇；

(3) 重新计算每个类簇的中心；

(4) 计算目标函数，当满足收敛条件时，则算法终止；如果条件不满足则回到步骤（2）。

2.2 X-Means 聚类算法

X-Means 算法无需提前设置准确的聚类个数，只需给出聚类个数的数值范围，可以通过聚类过程自动计算准确的类簇数值[32]。该算法通过不断搜索聚类中心点和聚类个数来求优化一个目标函数，最终输出目标函数最大的聚类模型作为最终结果。算法分为两步，第一步运行传统的 K-means 算法直至收敛，类簇个数 k 的初值是用户给定的类簇个数的下限，然后计算模型的目标函数值。如果求得的目标函数值比目前记录的最大的目标函数值要大，则更新最大目标函数值及最大目标函数值对应的聚类个数和其中心点。第二步是改进结构过程，先判断聚类个数是否达到类簇个数上限，如果达到了聚类个数上限，则输出最大目标函数值对应的聚类个数和其中心点，然后退出算法，否则将新增加一个聚类中心点，这样就构成了 k+1 个中心点，以这 k+1 个中心点为初始中心点，返回第一步进行(k+1)-Means 聚类。

2.3 基于聚类的缺陷模块序列预测

文献[33]将聚类技术应用到软件缺陷模块序列预测领域，经实验验证，该方法具有较好的性能。与其类似，基于聚类的缺陷序列预测算法可以采用如下步骤：

(1) 数据预处理，备份模型评价相关度量；

(2) 设置聚类算法，进行聚类运算，得到聚类结果；

(3) 进行类簇间排序；

(4) 进行类簇内部排序；

(5) 将经过两步排序后的序列作为缺陷预测序列，评价该序列的预测性能。

2.4 基于簇类排序和簇间排序的缺陷模块序列生成

在基于聚类的软件缺陷模块序列预测中，在第（2）步得到聚类结果后，为了得到缺陷模块预测序列，需要设置一定合理的排序方法，将各类簇和类簇内部的各个模块按照缺陷倾向的可能性进行排序。排序的目的是为了提高预测序列的性能。结合该预测序列的性能评价指标 CE 的求解过程，经过排序后的模块序列应该具有如下特点：(1)软件系统中有缺陷的模块应该尽量在序列的前部，无缺陷的模块应该尽量在序列的后部；(2)预测序列中模块的缺陷密度大体上应符合从大到小的顺序。所以，排序策略的设计应该从这两点出发。

在文献[34]中，研究者对软件系统中每一模块包含缺陷的概率与该模块的规模之间的关

系进行了研究。他们的结果表明，随着模块规模的增加，单位代码包含缺陷的概率会降低。因而，在进行模块缺陷序列预测时，一种简单的排序方法就是各模块按照其包含源代码的行数进行升序排列，将这种排序后的序列作为预测序列。记这种排序方法为 LOCMU（Line of Code Manual Up）。

同样利用文献[34]的结论，由于单位代码包含缺陷的概率随着模块规模的增加而降低，那么，在经过聚类后，类簇总代码行越小，该类簇包含缺陷的概率应该越大。利用这一结论，将聚类后各类簇按照总的代码行从小到大的顺序进行排序，类簇内部各模块按照代码行从小到大排序，经过两次排序后得到的模块序列，其缺陷可能性应该是从大到小，记这种排序方法为 MSSB（Minimum-Size-Cluster Started Size Based）。

在文献[33]中，在经过聚类算法后，为了获得最大的评价指标，作者设置了一种以最大类簇为起点的基于欧氏距离的排序方法。其基本思想是认为，通常情况下有缺陷的模块占整个系统模块的比例比较小，而无缺陷的模块占整个系统模块的比例较大。作者认为，在聚类后的的类簇中，类簇规模（类簇中包含的样本数）最大的模块集不包含缺陷的可能性较大，类簇规模最小的模块包含缺陷的可能性较大。基于这种观点，该文中设计了如下排序策略：以最大类簇（包含样本最多的类簇）中心点为起点，各类簇按照其中心点到起点的欧氏距离从小到大的顺序排序，类簇内部各样本同样按照到起点的欧氏距离从小到大的顺序排序。经过两次排序后的序列认为其缺陷可能性是从小到大，记这种排序算法为 LSDB（Largest-Cluster Started Distance Based）。

受文献[33]启发，聚类后规模最小的类簇是有缺陷类簇的可能性最大，结合前面提到的具有较高评价指标 CE 值的缺陷预测序列要满足的条件中的两点，可以将最小类簇中心点作为起点，剩余各类簇按照其中心点距离起点从小到大的顺序排序，类簇内部各样本按照距离起点从小到大的顺序排序，经过这样两次排序后的序列的缺陷可能性应该是从大到小，记这种排序方法为 SSDB（Smallest-Cluster Started Distance Based）。

最后，结合文献[33]和文献[34]，在基于聚类的软件缺陷序列预测过程中，经过聚类后，类簇间排序采用类似 SSDB 方法，以规模最小的类簇为起点，各类簇按照其中心距离最小类簇中心点的欧氏距离从小到大进行排序。每一类簇内部排序采用类似 MSSB 中的方法，按照模块代码行升序进行排列。经过这样两次排序后的序列，各类簇和类簇内部各模块都是按照缺陷可能性从大到小排序，最终所得序列的缺陷可能性也应该是从大到小。记这种排序方法为 SSDSB（Smallest-Cluster Started Distance and Size Based）。

3 工作量感知的缺陷模块排序性能评价

对于缺陷预测序列的评价，考虑到代码审查或测试阶段对工作量的要求，我们采用“基于代码行的 Alberg 图”上的 CE 指标。基于代码行的 Alberg 图是如图 1 所示的一种二维坐标图。

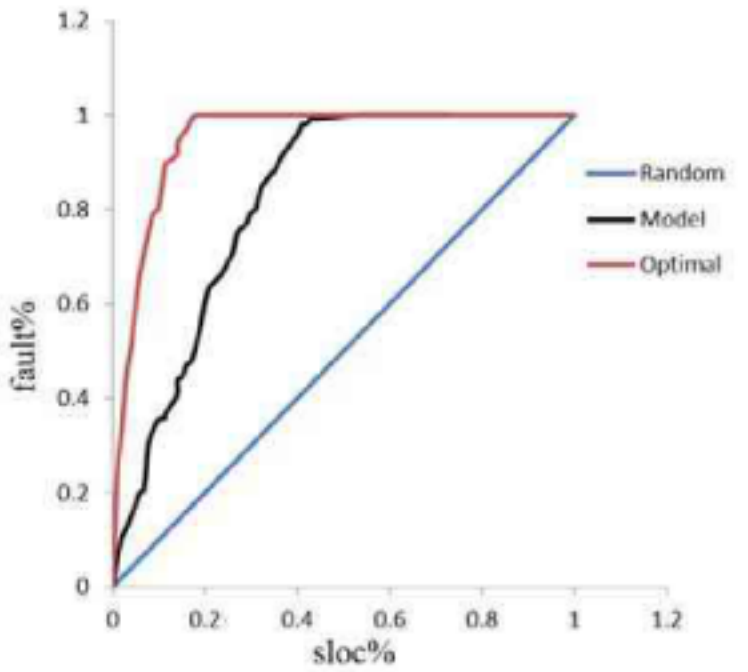

图 1 基于代码行的 Alberg 图
Fig. 1 LOC-based Alberg-diagram

该图横坐标用从给定的模块序列中选取的模块规模(代码行数)占系统总规模的百分比来表示，纵坐标用所选取的模块中包含的实际缺陷数目占系统缺陷总数的百分比表示。当横坐标从 0%变化到 100%时，缺陷预测模型就对应于 Alberg 图上的一条曲线（图中 Model 所代表的曲线）。Alberg 图上还有另外两条曲线：一条对应于“理想最优模型”（图中 Optimal 代表的曲线），另一条对应于“随机模型”（图中 Random 代表的曲线）。所谓“理想最优模型”，就是将模块按实际缺陷的密度（缺陷数目/源代码行）进行降序排序，由该序列构成的 Alberg 图曲线近似为“理想最优模型”曲线。而在“随机模型”中，模块按随机顺序进行排序，其评价性能在 Alberg 图上表现为一条经过（0,0）和（1,1）的直线。然后，分别由这三条曲线、直线 $y = 0$ 和直线 $x = \pi$ 之间的面积（AUC，Area Under Curve），来表示缺陷预测模型预测“潜在缺陷模块预测序列”的性能。

“基于代码行的 Alberg 图”上的 CE 指标由如下公式（公式 1）获得：

$$CE_\pi = \frac{AUC_\pi(Model) - AUC_\pi(Random)}{AUC_\pi(Optimal) - AUC_\pi(Random)} \quad \text{（公式 1）}$$

上述公式中的 $AUC_\pi(c)$ 是曲线 c(*Random*、*Model* 或 *Optimal*)、横坐标轴 $y = 0$ 和直线 $x = \pi(0 \le \pi \le 1)$ 所围部分的面积，该面积可以通过公式（公式 2）计算得出：

$$AUC_\pi(c) = \int_0^\pi c(t)dt \quad \text{（公式 2）}$$

从公式 1 可以明显看出，随机模型的 CE_π 值恒为 0。对于任意的预测曲线而言，CE_π 的值应该在区间(-1,1)之间，当 $CE_\pi < 0$ 时，表明该预测曲线的性能比随机模型还要差；当 $CE_\pi > 0$ 时，表明该预测曲线性能优于随机模型。CE_π 越接近于 1，表明模型 Model 越具有良好的缺陷预测性能。同时可以看出，对于一个软件系统而言，其理想最优模型是将模块按实际缺陷的密度（缺陷数目/源代码行数）进行降序排序，所以在该模型下，软件的模块序列是固定的，也就是 $AUC_\pi(Optimal)$ 是确定的。另外，$AUC_\pi(Random) = 0.5$。因此，CE_π 的值和 $AUC_\pi(Model)$ 具有线性关系。也就是说，要想获得一个比较大的 CE_π 值，只要使得 $AUC_\pi(Model)$ 具有比较大的值即可。此时，该预测序列应该与由“理想最优模型”确定的模块序列具有更多的相似性，在图上的曲线应该是一条尽量靠近 Optimal 的曲线，即具有较好预测性能的模型 Model 对应的曲线应该是一条比较“上凸”的曲线。

4 实验设置和结果

4.1 数据集

我们的实验在 equinox 3.4、JDT Core 3.4、lucene 2.4、mylyn 3.1 和 PDE UI 3.1 五个系统上进行。我们首先利用商业工具 Understand for Java 收集了 WMC、DIT、NOC、LCOM2、CBO、RFC 和 SLOC 这七个度量，并将这些度量当作模块的特征用来聚类。对于每一个系统，网站http://bug.inf.usi.ch/上提供了每个模块中包含的缺陷数目。在我们的实验中，缺陷数据不参与聚类过程，只用来评价各种聚类方法的性能。表 1 给出了这七个代码度量的定义和来源。

表 1 度量的定义及出处

Table 1 The definitions and the references of the metrics

度量名	定　义	来源
WMC	每个类的加权方法数	[35]
DIT	继承树深度。在继承树中，该节点与根节点之间的距离	[35,36]
NOC	直接继承该类的子类数	[35,36]
LCOM2	内聚性缺失 2，指类中未使用任何公共属性的方法数减去使用了公共属性的方法数，若该值小于 0，则 LCOM2 置为 0	[35]
CBO	除了与该类有继承关系的类，与该类耦合的类数。如果某个类使用了另一个类的方法或属性，则该类与另一个类存在耦合关系	[35]
RFC	类影响集包括该类的方法集合 M，以及直接或间接被 M 中的方法调用的函数。RFC 是该类响应集包含的元素的个数	[35]
SLOC	源代码行数	

对于 LCOM2 度量而言，由于其定义要求一个类中必须实现了方法，否则该度量没有意义。在五个系统中，有些类中只有属性而没有方法。对这种类而言，我们将 LCOM2 值赋为“Undefined”。在后续的实验分析中，所有 LCOM2 值为“Undefined”的类都被排除掉。表 2 给出了这五个系统上收集的数据集，包括每个系统的模块数目、排除了 LCOM2 值为“Undefined”的类后的模块数目和排除了 LCOM2 值为“Undefined”的类后的源代码行数等信息。

表2 实验对象描述性信息

Table 2 Experimental object descriptive information

系统名	开发语言	总模块数	预处理后模块数	预处理后代码行数	预处理后包含缺陷的模块数	预处理后缺陷数
Equinox 3.4	Java	322	278	41844	126	241
JDT Core 3.4	Java	997	990	263792	205	373
Lucene 2.4	Java	670	666	83697	63	96
Mylyn 3.1	Java	1860	1654	149869	232	320
PDE UI 3.1	Java	1455	1446	151658	204	305

4.2 实验结果

我们首先对这五个数据集进行预处理，删除其中在这 8 个度量上无定义的样本。然后对预处理后的这五个数据集，分别用第 2.4 节描述的四种排序方法进行实验。试验中，样本间的距离求解时采用的是欧氏距离。在实验中，当以 K-Means 聚类算法建立聚类模型时，分别设置聚类个数 k 为样本总数的 1%、2%、5%和 10%。我们分别计算了 $\pi=0.2$ 和 $\pi=1.0$ 这两种情况下的 CE_π 的值。

首先是采用 K-Means 算法进行聚类。表 3-表 7 分别当 $\pi=0.2$ 时五个数据集上的实验结果，表 8-表 12 表示当 $\pi=1.0$ 时在五个数据集上的实验结果。其中，plk 列对应的数值是实验中聚类参数的设置，其中，p 值表示 K-Means 聚类时设置的类簇个数占样本总数的百分比，k 值表示采用当前百分比时的具体聚类个数；表中各算法列对应的值是采用该排序算法时得

255 到的缺陷模块预测序列的性能评价指标。

表 3 Equinox 3.4 K-Means $CE0_{.2}$ 结果
260 Table 3 Equinox 3.4 K-Means $CE_{0.2}$ results

plk	$CE_{0.2}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|3	0.103	0.051	0.243	0.13	0.354
0.02|6	-0.15	0.035	0.266	0.114	0.354
0.05|14	-0.143	-0.212	0.196	-0.064	0.354
0.10|28	-0.143	-0.01	0.288	0.004	0.354

表 4 JDT Core 3.4 K-Means $CE_{0.2}$ 结果
Table 4 JDT Core 3.4 K-Means $CE_{0.2}$ results

plk	$CE_{0.2}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|10	-0.131	-0.018	0.114	0.05	0.078
0.02|20	-0.123	-0.004	0.076	-0.014	0.078
0.05|50	-0.12	-0.104	0.115	-0.056	0.078
0.10|99	-0.147	-0.035	0.089	-0.043	0.078

表 5 Lucene 2.4 K-Means $CE_{0.2}$ 结果
265 Table 5 Lucene 2.4 K-Means $CE_{0.2}$ results

plk	$CE_{0.2}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|7	-0.16	-0.01	0.141	-0.163	0.048
0.02|13	0.049	0.069	0.161	0.189	0.048
0.05|33	0.064	0.096	0.12	0.063	0.048
0.10|67	0.063	-0.006	0.094	0.008	0.048

表 6 Mylyn 3.1 K-Means $CE_{0.2}$ 结果
Table 6 Mylyn 3.1 K-Means $CE_{0.2}$ results

plk	$CE_{0.2}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|17	-0.072	0.014	-0.015	-0.007	0.102
0.02|33	-0.067	0.077	-0.009	0.04	0.102
0.05|83	-0.071	0.001	-0.008	0.002	0.102
0.10|165	-0.083	0.013	-0.013	-0.084	0.102

表 7 PDE UI 3.1 K-Means $CE_{0.2}$ 结果
270 Table 7 PDE UI 3.1 K-Means $CE_{0.2}$ results

plk	$CE_{0.2}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|14	-0.009	-0.009	0.06	-0.031	0.011
0.02|29	-0.032	-0.037	0.039	-0.032	0.011
0.05|72	-0.018	-0.055	0.01	-0.065	0.011
0.10|145	-0.036	-0.053	0.026	-0.056	0.011

表 8 Equinox 3.4 K-Means $CE_{1.0}$ 结果
Table 8 Equinox 3.4 K-Means $CE_{1.0}$ results

plk	$CE_{1.0}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|3	0.106	-0.146	0.213	0.067	0.269
0.02|6	-0.133	-0.149	0.33	-0.099	0.269
0.05|14	-0.134	-0.092	0.2	-0.052	0.269
0.10|28	-0.162	0.026	0.258	0.061	0.269

表 9 JDT Core 3.4 K-Means $CE_{1.0}$ 结果
275 Table 9 JDT Core 3.4 K-Means $CE_{1.0}$ results

plk	$CE_{1.0}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|10	-0.09	0.069	0.24	0.088	0.196
0.02|20	-0.171	0.131	0.215	0.023	0.196
0.05|50	-0.14	-0.199	0.245	0.046	0.196
0.10|99	-0.182	0.169	0.227	0.134	0.196

表 10 Lucene 2.4 K-Means $CE_{1.0}$ 结果
Table 10 Lucene 2.4 K-Means $CE_{1.0}$ results

plk	$CE_{1.0}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|7	-0.187	-0.155	0.305	-0.233	0.164
0.02|13	-0.067	0.001	0.451	0.001	0.164
0.05|33	-0.109	0.131	0.362	0.041	0.164
0.10|67	-0.12	0.085	0.243	-0.08	0.164

表 11 Mylyn 3.1 K-Means $CE_{1.0}$ 结果
280 Table 11 Mylyn 3.1 K-Means $CE_{1.0}$ results

plk	$CE_{1.0}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|17	-0.092	0.086	-0.011	0.087	0.193
0.02|33	-0.096	0.098	0.016	0.099	0.193
0.05|83	-0.134	0.044	0.039	0.039	0.193
0.10|165	-0.171	-0.045	0.052	-0.07	0.193

表 12 PDE UI 3.1 K-Means $CE_{1.0}$ 结果
Table 12 PDE UI 3.1 K-Means $CE_{1.0}$ results

plk	$CE_{1.0}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
0.01|14	-0.051	0.023	-0.023	0.033	0.049
0.02|29	0.015	0.017	-0.006	0.02	0.049
0.05|72	-0.023	-0.077	-0.055	-0.068	0.049
0.10|145	-0.021	-0.079	0.016	-0.069	0.049

表 13、表 14 表示采用 X-Means 算法进行聚类时分别取 π =0.2 和 π =1.0 时的结果。此时，对于各个数据集，每种排序方法只对应一个缺陷模块预测序列，所以也只得到一个性能评价指标，其值如表所示。我们也求得了在选定一种排序算法情况下，在所有数据集上得到
285 的 CE_{π} 的平均值，其值如表 13、表 14 中 Average 所在行表示。

表 13 X-Means $CE_{0.2}$ 结果
Table 13 X-Means $CE_{0.2}$ result

数据集	$CE_{0.2}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
Equinox 3.4	-0.143	-0.157	0.225	-0.088	0.354
JDT Core 3.4	-0.138	-0.08	0.132	0.198	0.078
Lucene 2.4	0.092	-0.149	0.132	0.063	0.048
Mylyn 3.1	-0.053	0.051	0.024	0.036	0.102
PDE UI 3.1	-0.02	0.014	0.087	0.019	0.011
Average	-0.0524	-0.0642	0.12	0.0456	0.1186

表 14 X-Means $CE_{1.0}$ 结果
Table 14 X-Means $CE_{1.0}$ result

数据集	$CE_{1.0}$				
	LSDB	SSDB	MSSB	SSDSB	LOCMU
Equinox 3.4	-0.097	-0.031	0.238	-0.061	0.269
JDT Core 3.4	-0.177	0.037	0.259	0.306	0.196
Lucene 2.4	-0.046	-0.36	0.376	0.101	0.164
Mylyn 3.1	-0.085	0.088	0.063	0.092	0.193
PDE UI 3.1	-0.016	0.053	0.114	0.055	0.049
Average	-0.0842	-0.0426	0.21	0.0986	0.1742

4.3 结果分析

从实验结果可以看出，在基于聚类的软件缺陷序列预测模型中，不同的排序算法对结果序列的性能有着明显的影响。在所有情况下，LOCMU 排序算法都具有较好的预测性能。当采用 LSDB 和 SSDB 排序方法时，不论是采用 K-Means 算法建模，还是采用 X-Means 算法建模，在这五个数据集上，所得缺陷序列预测的 CE 值都比较小，在多数情况下甚至为负值，这表明当采用这两种排序方法时，基于 K-Means 算法和 X-Means 算法的缺陷序列预测模型的性能较差。而当采用 MSSB 方法时，从表 3-表 8 可以看出，当 π=0.2 时，该方法并不比 LOCMU 方法有效；而当 π=1.0 时，从表 9-表 13 可以看出，当采用 K-Means 进行聚类并设置聚类百分比为 2%时，多数情况下，该方法要优于 LOCMU；同样地，从表 14 可以看出，当采用 X-Means 算法进行聚类时，多数情况下，该方法同样优于 LOCMU 排序方法。相对于 LSDB、SSDB、SSDSB 而言，MSSB 排序方法同样具有较大的优势。

仔细观察实验结果，可以看出，不论采用哪种聚类算法进行建模，在所有数据集上，采用 SSDB 排序方法和 SSDSB 排序方法所得的预测序列的 CE 值都比较接近，这表明，SSDB 方法和 SSDSB 方法构建的缺陷预测序列的性能之间没有显著的差异。

在基于聚类的软件缺陷序列预测建模过程中，无论采用哪种聚类算法，对于每次聚类的结果而言，MSSB 方法和 SSDSB 方法在进行类簇内部排序时，都是采用模块代码行升序的方法，所以对于同一类簇而言，其排序结果是相同的，不同的是两种方法对应的类簇间的排序结果。而对于 SSDB 方法和 SSDSB 方法而言，其在类簇间排序时采用的方法是相同的，不同的是类簇内部的排序方法。从实验结果可以看出，就缺陷预测序列的性能而言，MSSB 方法和 SSDSB 方法之间有着明显的差异，而 SSDB 和 SSDSB 方法之间差异不明显。这就说明，在基于聚类的软件缺陷预测序列建模过程中，对聚类结果进行排序时，类簇间排序算法对最终缺陷预测序列性能的影响要比类簇内部排序算法的对最终缺陷预测序列性能的影响更加显著。因此，在基于聚类的缺陷序列预测建模过程中，对聚类结果进行排序时，应优先考虑类簇间的排序问题。

5 结束语

本文首先简要介绍了软件缺陷预测领域的相关研究工作，对基于机器学习的软件缺陷序列预测中的关键问题进行了阐述，分析了已有研究中建模方法和评价标准的不足。针对这些不足，本文提出了基于聚类的软件缺陷序列预测模型，着眼于建模过程中的排序问题，文中设计了四种不同的排序算法，并采用了一种工作量敏感的评价指标对该模型的性能进行了实验比较。从五个数据集上的实验结果可以看出，不论是采用 K-Means 聚类还是 X-Means 聚类，当采用 MSSB 排序方法时，所得的缺陷预测序列都能在一定程度上帮助减少代码审查的工作量。同时，从几种排序方法之间的比较可以看出，在聚类后的构建缺陷预测序列的过程中，相比类簇内部的排序方法而言，类簇间的排序方法会对预测序列的性能产生更加明显的影响。

[参考文献] (References)

[1] Littlewood B, Lorenzo S. Software reliability and dependability: a roadmap[C]. Proceedings of The Conference on The Future of Software Engineering, 2000: 175-188.

[2] Catal C, Diri S. A systematic review of software fault prediction studies[J]. Expert Systems with Applications, 2009, 36(4): 7346-7354.

[3] Koru A G, Liu H. An investigation of the effect of module size on defect prediction using static measures[C].Proceedings of The 1st International Workshop on Predictor Models in Software Engineering, 2005:1-5.

[4] Zimmermann T, Nagappan N, Gall H, et al. Cross-project defect prediction: A large scale experiment on data vs. domain vs. Process[C]. Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on the Foundations of Software Engineering, 2009: 91-100.

[5] Fenton N, Neil M. A critique of software defect prediction models[J]. IEEE Transactions on Software Engineering, 1999, 25(5):675-689.

[6] D'Ambros M, Lanza M, Robbes R. Evaluating defect prediction approaches: a benchmark and an extensive comparison[J]. Empirical Software Engineering, 2012,17(4-5): 531-577.

[7] Menzies T, Milton Z, Turhan B, et al. Defect prediction from static code features: current results, limitations, new approaches[J]. Automated Software Engineering, 2010, 17(4): 375-407.

[8] Lessmann S, Baesens B, Mues C, et al. Benchmarking classification models for software defect prediction: A proposed framework and novel findings[J]. IEEE Transactions on Software Engineering, 2008, 34(4): 485-496.

[9] Challagulla V, Bastani F, Yen I, et al. Empirical assessment of machine learning based software defect prediction techniques[C]. Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems. 2005: 263-270.

[10] Singh Y, Kaur A, Malhotra R. Predicting software fault proneness model using neural network[C]. Proceedings of the 9th International Conference on Product-Focused Software Process Improvement, 2008:204-214.

[11] Pearce J, Ferrier S. Evaluating the predictive performance of habitat models developed using logistic regression[J]. Ecological Modelling, 2000, 133(3): 225-245.

[12] Turhan B, Bener A. Analysis of Naive Bayes' assumptions on software fault data: An empirical study[J]. Data & Knowledge Engineering, 2009, 68(2):278-290.

[13] Guo L, Ma Y, Cukic B, et al. Robust prediction of fault-proneness by random forests[C]. Proceedings of the 15th International Symposium on Software Reliability Engineering, 2004:417-428.

[14] Xing F, Guo P, Lyu M. A novel method for early software quality prediction based on support vector machine[C]. Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering, 2005:213-222.

[15] Catal C, Sevim U, Diri B. Metrics-driven software quality prediction without prior fault data[J]. Electronic Engineering and Computing Technology, 2010, 60: 189-199.

[16] Catal C, Sevim U, Diri B. Clustering and metrics thresholds based software fault prediction of unlabeled program modules[C]. Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations, 2009: 199-204.

[17] Zhong S, Khoshgoftaar T, Seliya N. Analyzing software measurement data with clustering techniques[J]. IEEE Intelligent Systems, 2004, 19(2): 20-27.

[18] Zhong S, Khoshgoftaar T, Seliya N. Unsupervised learning for expert-based software quality estimation[C]. Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering, 2004: 149-155.

[19] Seliya N, Khoshgoftaar T, Zhong S. Analyzing software quality with limited fault-proneness defect data[C]. Proceedings of the 9th IEEE International Symposium on High Assurance Systems Engineering, 2005: 89-98.

[20] Seliya N, Khoshgoftaar T. Software quality analysis of unlabeled program modules with semi-supervised clustering[J]. IEEE Transactions on Systems, Man, Cybernetics, Part A: Systems and Humans, 2007, 37(2):

201-211.
[21] 马樱. 基于机器学习的软件缺陷预测技术研究[D]. 成都：电子科技大学计算机科学与工程学院,2012.
[22] Bishnu P, Bhattacherjee V. Software fault prediction using quad tree based k-means clustering algorithm[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(6): 1146-1150.
[23] 常瑞花,慕晓冬,李琳琳,等. 基于模糊聚类非负矩阵分解的软件缺陷预测[J]. 宇航学报，2011，32(9):2059-2064.
[24] 陈媛. 基于数据挖掘的软件缺陷预测技术研究[D]. 长春：中国科学院研究生院（长春光学精密机械与物理研究所）,2012.
[25] 慕晓冬,常瑞花,宋国军,等. 基于混沌免疫谱聚类的软件缺陷预测[J]. 高技术通讯，2012，22(12):1219-1224.
[26] Catal C, Sevim U, Diri B. Software fault prediction of unlabeled program modules[C]. Proceedings of the World Congress on Engineering, 2009, 1.
[27] Arisholm E, Briand L, Johannessen E. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models[J]. Journal of Systems and Software, 2010, 83(1): 2-17.
[28] Arisholm E, Briand L. Predicting fault-prone components in a Java legacy system[C]. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, 2006: 8-17.
[29] Arisholm E, Briand L, Fuglerud M. Data mining techniques for building fault-proneness models in telecom Java software[C]. Proceedings of the 18th IEEE International Symposium on Software Reliability, 2007: 215-224.
[30] Mende T, Koschke R. Revisiting the evaluation of defect prediction models[C]. Proceedings of the 5th International Conference on Predictor Models in Software Engineering, 2009.
[31] Tan P, Steinbach M, Kumar V. 数据挖掘导论[M]. 范明，范宏建等译. 北京：人民邮电出版社，2006.
[32] Pelleg D, Moore A. X-means: extending k-means with efficient estimation of the number of clusters[C]. Proceedings of the Seventeenth International Conference on Machine Learning, 2000: 727-734.
[33] 杨明. 基于聚类的潜在缺陷模块序列预测[D]. 南京：东南大学计算机科学与工程学院, 2011.
[34] Koru A, Emam K, Zhang D, et al. Theory of relative defect proneness[J]. Empirical Software Engineering, 2008, 13(5): 473-498.
[35] Chidamber S, Kemerer C. A metrics suite for object-oriented design[J]. IEEE Transactions on Software Engineering, 1994, 20(6): 476-493.
[36] Chidamber S, Kemerer C. Towards a metrics suite for object-oriented design[C]. Conference proceedings on Object-oriented Programming Systems, Languages, and Applications, 1991: 197-211.