基于改进深度森林算法的软件缺陷预测

薛参观 燕雪峰

(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 软件缺陷预测是合理利用软件测试资源、提高软件性能的重要途径。为处理软件缺陷预测模型中浅层机器学习算法无法对软件数据特征进行深度挖掘的问题,提出一种改进深度森林算法——深度堆叠森林(DSF)。该算法首先采用随机抽样的方式对软件的原始特征进行变换以增强其特征表达能力,然后用堆叠结构对变换特征做逐层表征学习。将深度堆叠森林应用于Eclipse数据集的缺陷预测中,实验结果表明,该算法在预测性能和时间效率上均比深度森林有明显的提升。

关键词 软件缺陷预测,深度森林,深度堆叠森林,随机抽样,堆叠结构

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.029

Software Defect Prediction Based on Improved Deep Forest Algorithm

XUE Can-guan YAN Xue-feng

(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Software defect prediction is an important way to rationally use software testing resources and improve software performance. In order to solve the problem that the shallow machine learning algorithm cannot deeply mine the characteristics of software data, an improved deep forest algorithm named deep stacking forest (DSF) was proposed. This algorithm firstly adopts the random sampling method to transform the original features to enhance its feature expression ability, and then uses the stacking structure to perform layer-by-layer representation learning for the transform features. The deep stacking forest was applied for the defect prediction of Eclipse dataset. The experimental results show that the algorithm has significant improvement in the predicting performance and time efficiency than the deep forest.

Keywords Software defect prediction, Deep forest, Deep stacking forest, Random sampling, Stacking structure

1 引言

在软件工程领域,任何检测和验证手段都不可能发现并排除所有的软件缺陷[1]。在实际工程项目中,开发一个没有任何缺陷的软件系统是不可能的,开发人员即使小心翼翼、精益求精,也不能排除软件系统中仍然存在一些未知的错误或意外的缺陷。若软件系统出现缺陷,对于开发团队来说,其维护将变得非常困难,而对于用户来说,则可能遭受经济损失。软件缺陷预测技术则给开发团队提供了一次重新测试缺陷模块或文件的机会,使其在有缺陷倾向的模块上投入更多的时间,在无缺陷倾向的模块上花费更少的时间,从而更好地利用软件项目的资源。

软件缺陷预测[2]是根据软件历史开发数据及已发现的缺陷,借助机器学习等方法来预测软件项目中的缺陷模块或数目等。目前常用的软件缺陷预测方法有神经网络[3]、贝叶斯网络[4]、随机森林[5]、集成学习[6]、支持向量机[7]等,但是这些方法都属于浅层机器学习,不能完美地表达出非结构化数据之间的复杂关系。当数据量达到一定程度时,浅层结构算法的学习能力不如深层结构算法[8]。

深度学习[9]是一种通过分层结构的分阶段信息处理来探索无监督的特征学习和模式分析、分类的技术。然而,目前常见的深度结构学习算法都是基于神经网络的[10]。周志华等[11]提出了一种新型的深度结构学习算法——深度森林(Multi-Grained Cascade Forest, gcForest),它是一种基于决策树的多层分类器,主要由多粒度扫描和级联森林两部分构成。多粒度扫描层用来提取原始数据的特征,级联森林用来不断地精炼分类。每一层的结构都是随机森林[12]和完全随机树森林[13]的组合,随机森林是输入特征到标注概率分布的映射,而完全随机树森林是输入特征到输入特征聚类的映射。深度森林的出现,为深度学习在深度神经网络之外的方法打开了一扇门。

本文将深度森林算法应用于软件缺陷预测,并针对多粒

到稿日期:2017-07-11 返修日期:2017-10-24 本文受十三五重点基础科研项目(JCKY2016206B001),十三五装备预研项目(41401010201),江苏省软件新技术与产业化协同创新中心资助。

薛参观(1986-),男,硕士生,主要研究方向为系统建模与仿真,E-mail:277815319@qq.com;燕雪峰(1975-),男,博士,教授,主要研究方向为软件工程方法论、系统建模与仿真等,E-mail:yxf@nuaa.edu.cn(通信作者)。

度扫描和级联森林在软件缺陷预测中存在的不足，提出一种改进的深度森林算法——深度堆叠森林(Deep Stacking Forest，DSF)。本文将深度堆叠森林算法应用于 Eclipse 数据集[15]的软件缺陷预测，实验结果表明，该算法在预测性能和时间效率上均比深度森林算法更优。

2　深度森林

深度森林是一种新型的深度结构学习算法，首先采用多粒度扫描的方法对原始输入特征进行变换以增强特征表达能力，再使用级联结构做逐层表征学习。

深度森林用多粒度扫描来增强对特征关系的处理能力。多粒度扫描是使用滑动窗口来扫描原始输入特征，从正/负训练样本中提取的所有特征向量被视为正/负实例，利用从相同大小的窗口中提取的实体来分别训练一个随机森林和一个完全随机树森林，然后将这些森林生成的类向量聚合为原始输入特征的变换特征向量。

深度森林用级联结构对变换特征向量进行逐层表征学习。其中，级联的每一层接收由其上一层处理的特征信息，并将该层的处理结果输出到下一层。在深度森林中，级联结构的每层包含两个随机森林和两个完全随机树森林。需要注意的是，在扩展一个新的层后，整个级联的性能将在验证集上进行评估，如果没有显著的性能增益，训练过程将被终止。因此，级联结构中层的数量是自动确定的。

相比于深度神经网络，深度森林具有明显的优势：首先，深度森林的超参数比深度神经网络少得多，而且深度森林对超参数设定的性能具有相当高的鲁棒性；其次，深度森林不需要调参的过程，并且可以进行并行化计算，效率优势异常明显。尽管深度森林的性能优越，但将其用于构建软件缺陷预测模型时仍存在一定的不足。

1)软件缺陷预测是一个二元分类问题，即对软件模块进行质量分析，根据分类规则将其分为有故障倾向类或无故障倾向类。因为软件缺陷预测的主要目的是预测软件中的模块是否有故障倾向，所以本文将有故障倾向的模块设为正例，将无故障倾向的模块设为负例。多粒度扫描和级联结构都是取所生成的类向量来聚合成变换特征向量或增强特征向量，这将造成特征空间的冗余。因为对于软件缺陷预测问题，其模块属于正类的概率和属于负类的概率的和为 1，即两个概率是线性相关的，如果两个概率都用来聚合特征向量，则会造成特征空间冗余，增加算法的空间复杂度。

2)多粒度扫描对于在空间上有关联的特征(如图像匹配、语音识别等)具有显著效果，而对于在空间上无关联的特征(如软件缺陷预测、文本分类等)将可能丢失重要信息。究其原因，对于在空间上无关联的特征，多粒度扫描在一定程度上降低了两端特征的重要性。多粒度扫描时，第一个特征和最后一个特征都仅被扫描到一次，也就是说，这两个特征都仅被使用一次，假如第一个特征或最后一个特征的重要度很大，多粒度扫描就不能有效地利用这个重要的特征。

3)深度森林在级联的每一层都将输出和原始输入聚合在一起作为下一层的输入，这种做法是为了使得级联的输出不断向某一个值靠近，即让级联趋于收敛和稳定。但是，级联在训练时仅仅把上一层的输出与原始输入聚合在一起作为下一层的输入，这将降低级联的收敛速度，从而降低算法的效率。

3　深度堆叠森林

针对深度森林在软件缺陷预测方面的不足，提出改进的深度森林算法——深度堆叠森林(Deep Stacking Forest，DSF)。深度堆叠森林首先采用随机抽样的方法对原始输入特征进行变换，再使用堆叠结构做逐层表征学习。

3.1　随机抽样

针对深度森林中多粒度扫描可能丢失重要信息的问题，采用随机抽样的方法变换原始输入特征。随机抽样是从原始输入特征中按不同规模随机抽取特征，从有缺陷/无缺陷的训练样本中抽取的所有特征组成的向量被视为有缺陷/无缺陷的实例，将相同大小规模抽取的实例组成一个训练实体，抽取的训练实体将分别训练一个随机森林和一个完全随机树森林，然后用这些森林预测原始训练样本生成类向量，将其正例概率值聚合起来形成原始输入特征的变换特征向量。

如图 1 所示，假设原始输入特征为 400 维，使用 100 维的特征规模随机抽样 50 次，则每次从原始输入特征中随机抽取 100 维特征组成一个实体，共生成 50 个实体；每个实体将分别训练一个随机森林和一个完全随机树森林，然后用生成的森林来训练原始训练数据集，则产生两个类向量，将所有实体产生类向量的正例概率值聚合起来便生成了一个 100 维的变换特征向量，即 400 维原始特征向量对应于 100 维变换特征向量。

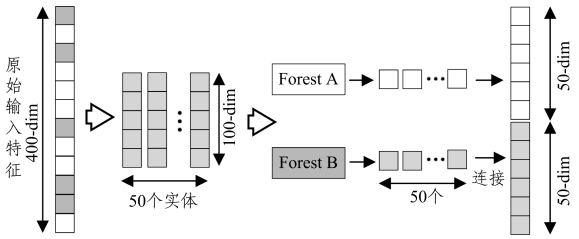

图 1　随机抽样示意图

Fig. 1　Random sampling

由于随机抽样是按规模随机抽取的原始特征，因此每个特征被抽到的概率是相同的，即所有特征具有相同的重要度。根据随机抽样的不同规模，可以设置不同的抽样次数，规模小时抽样次数多，规模大时抽样次数少，这样可以有效地利用原始输入特征。图 1 仅仅表示了一种规模的随机抽样，通过使用多种不同规模、不同次数的随机抽样，最终的变换特征向量将包含更多的特征(见 3.3 节图 3)。

3.2　堆叠结构

针对深度森林中级联结构收敛速度慢的问题，深度堆叠森林采用堆叠结构做逐层表征学习。堆叠结构的核心思想源自 Wolpert[14]提出的“堆叠泛化(stacked generalization)”，即结构中每一个新层的输入都是由该层之前的所有层的输出和原始输入聚合在一起组成的。深度堆叠森林中，堆叠结构是对随机森林和完全随机树森林做“堆叠”，其每一层都由两个随机森林和两个完全随机树森林组成。堆叠结构如图 2 所

示，其中虚线表示副本。

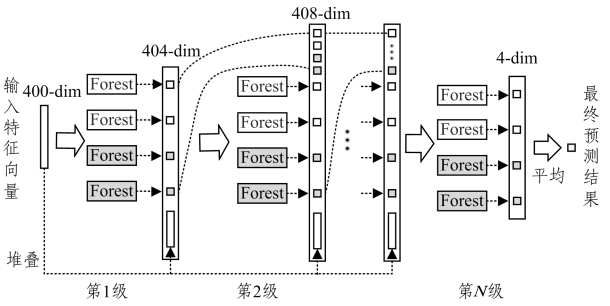

图 2 堆叠结构示意图

Fig. 2 Stacking structure

如图 2 所示，假设有 400 维输入特征，堆叠结构的第一层将分别训练两个随机森林和两个完全随机树森林，每个森林都生成一个类向量，取其正例概率值聚合成堆叠结构第一层的增强特征向量，这样就生成了一个 4 维的增强特征向量，将之与输入特征向量聚合在一起作为第二层的输入，即第二层将有 404 维的输入特征；同样地，经过堆叠结构第二层后生成一个 4 维的增强特征向量，将之与输入特征向量及第一层增强特征向量聚合在一起后作为第三层的输入，即第三层将有 408 维的输入特征；以此类推，直至训练过程终止。

需要注意的是，在扩展一个新的层后，随机抽取训练集的 80％作为验证集，将剩余的 20％作为评估集，整个堆叠结构的性能将在验证集上进行评估，如果没有显著的性能增益，训练过程将终止。因此，堆叠结构中层的数量也是自动确定的。

3.3 总体结构

深度堆叠森林的总体结构如图 3 所示。假设原始输入特征为 400 维，并且有 3 种规模被用于随机抽样，分别为 100 维特征抽样 200 次，200 维特征抽样 100 次，300 维特征抽样 50 次。因此，100 维特征的随机抽样将使每个原始输入特征生成 200 个 100 维的实体，每个实体分别训练一个随机森林和一个完全随机树森林，则生成两个类向量，将其正例概率值聚合起来，生成一个 400 维的变换特征向量。类似地，规模为 200 和 300 维特征的随机抽样将分别为每个原始输入特征生成一个 200 维和 100 维的变换特征向量。将 3 个变换特征向量连接起来，将获得一个 700 维的变换特征向量，这就是通过随机抽样对原始输入特征的重新表示。换句话说，每个 400 维的原始特征向量由 700 维的变换特征向量重新表示。

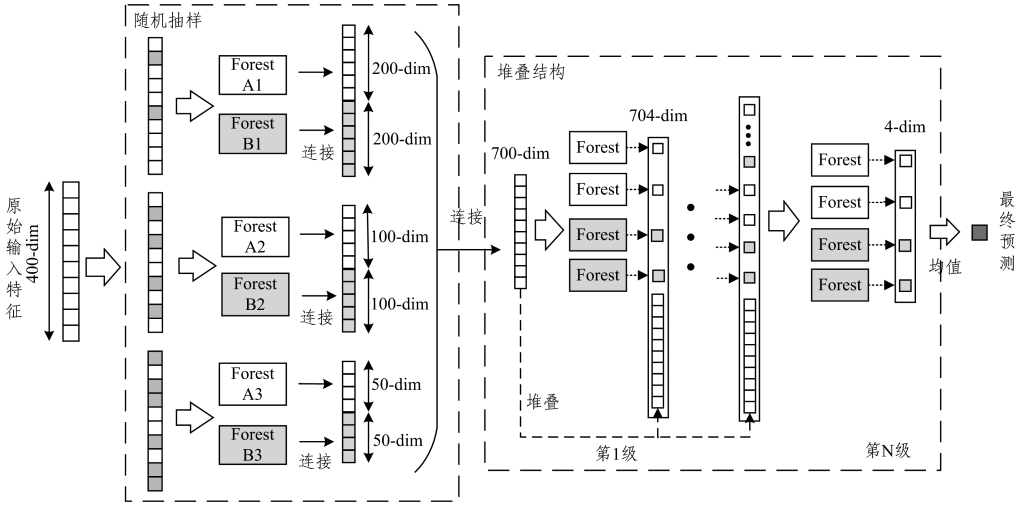

图 3 深度堆叠森林的总体结构示意图

Fig. 3 Overall structure of deep stacking forest

之后，700 维变换特征向量将被传递到堆叠结构。如果堆叠结构的每层由 4 个森林（两个随机森林和两个完全随机树森林）组成，则在堆叠第一层结束时将获得 704 维的特征向量。然后，将这些特征向量输入到下一层的堆叠结构中。重复该过程，直到验证性能表明堆叠结构的扩展应该被终止。

测试时，给出一个测试实例，它将通过随机抽样过程获得相应的 700 维变换特征向量，然后通过堆叠结构预测，直到最后一层。最终的预测结果将通过最后一层的 4 个正例概率值取平均得到，若该平均值大于或等于 0.5，则预测该测试实例为正例，否则为负例。

4 实验与分析

4.1 实验准备

4.1.1 实验环境

实验所用计算机的配置为 Intel(R) Xeon(R) E3-1230 3.30GHz 处理器，8GB 内存。实验所运行的软件环境为安装在 Windows 7 操作系统下的 MATLAB R2014a。

4.1.2 实验数据

本实验采用的数据来源于 Eclipse 标准数据集[15]，此数据集是软件缺陷预测研究领域使用最为广泛的公共数据集之一，可以从 Eclipse Bug Data[1)] 获取。

Eclipse 数据集中有 6 个 ARFF 格式文件，分别对应于 Eclipse 3 个版本 2 种粒度下的故障记录。一种是以 package 为最小粒度进行故障数目统计，即统计该 package 中包含的故障总数；另一种是以 file 为最小粒度进行故障数目统计，即统计一个 file 中包含的故障数量，其故障数据记录又分为发布前故障数（*pre*）和发布后故障数（*post*）。本文中的所有实验数据均采用以 file 为最小粒度的发布后故障数据记录。

Eclipse 数据集中每个 file 级故障记录包含 198 个特征，

1) https://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/

主要包括代码行数度量特征、复杂性度量特征和基于语义语法树的度量特征。由于本文研究的是分类任务的软件缺陷预测，而 Eclipse 数据集给出的是缺陷数量 *post*（指软件发布后的缺陷数量），因此需将故障数量 *post* 转换成模块是否有缺陷的类标 *hasDefects*。转换方式为：

$$hasDefects=\begin{cases}0, & post=0\\ 1, & post\neq 0\end{cases}$$

表 1 列出了 Eclipse 数据集 file 级数据的统计信息。

表 1　Eclipse 数据集 file 级数据的统计信息

Table 1　Statistical information of file data of Eclipse dataset

数据集	特征数	样本数	缺陷数	缺陷率/%
file 2.0	198	6729	975	14.49
file 2.1	198	7888	854	10.83
file 3.0	198	10593	1568	14.80

4.1.3　实验方法

由于 Eclipse 数据集的 file 级数据在结构上相同，因此取其中一个版本的数据作为训练数据来学习模型，可以分别预测 3 个版本的数据，这样利用 file 级数据就可以进行 9 次预测和验证。

当训练集和测试集来自同一个版本时，采用十折交叉验证，即将数据集平均分成 10 份，每次取其中 1 份作为预测集，剩余 9 份作为训练集，构建软件缺陷预测模型并进行分类预测。实验共进行 10 轮，将 10 轮实验的平均值作为最终结果。

4.1.4　超参数的设置

超参数是指机器学习模型里的框架参数，如深度堆叠森林中，随机抽样时的森林个数、森林中树的棵数、抽样规模及抽样次数等，都是 DSF 的超参数。

深度森林的超参数与文献[11]中的设置相同。为了更好地进行类比，深度堆叠森林的超参数仿照深度森林的超参数进行设置，如表 2 所列。不同之处在于：在深度堆叠森林中，随机抽样时，采用 3 种规模、3 种次数对原始特征进行抽样，抽样规模分别为$\lceil d/16\rceil$，$\lceil d/9\rceil$，$\lceil d/4\rceil$，对应的抽样次数分别为 200 次、100 次和 50 次。

表 2　gcForest 和 DSF 的超参数设置

Table 2　Hyper-parameter setting of gcForest and DSF

深度森林 gcForest	深度堆叠森林 DSF
森林类型： 随机森林、完全随机树森林	森林类型： 随机森林、完全随机树森林
多粒度扫描时的森林： 森林个数：2 森林中树的棵树：30 滑动窗口的尺寸：$\lceil d/16\rceil$，$\lceil d/9\rceil$，$\lceil d/4\rceil$	随机抽样时的森林： 森林个数：2 森林中树的棵树：30 随机抽样的规模：$\lceil d/16\rceil$，$\lceil d/9\rceil$，$\lceil d/4\rceil$ 随机抽样次数：200，100，50
级联时的森林： 森林个数：4 每个森林中树的棵数：1000	堆叠时的森林： 森林个数：4 每个森林中树的棵数：1000

4.2　实验结果及分析

首先对 Eclipse 数据集的 file 级数据进行预处理，即将模块缺陷数量 *post* 转换为是否有缺陷的类标 *hasDefects*；然后用 gcForest 算法构建软件缺陷预测模型进行实验，共进行 9 次；最后用 DSF 算法构建软件缺陷预测模型进行实验，同样进行 9 次。

采用正确率、准确率、召回率和 F1-度量 4 个评价指标对实验结果进行评估。实验得到的软件缺陷预测结果具体如表 3 及图 4－图 7 所示。

表 3　gcForest 和 DSF 的分类预测结果

Table 3　Classified prediction results of gcForest and DSF

训练集	测试集	gcForest				DSF			
		正确率	准确率	召回率	F1-度量	正确率	准确率	召回率	F1-度量
file 2.0	file 2.0	0.8845	0.6708	0.4031	0.5036	0.8861	0.6522	0.4601	0.5399
	file 2.1	0.8555	0.3208	0.2998	0.3099	0.8569	0.3361	0.3302	0.3331
	file 3.0	0.8505	0.4912	0.2832	0.3592	0.8427	0.4486	0.2730	0.3394
file 2.1	file 2.0	0.8501	0.4543	0.1733	0.2509	0.8520	0.4760	0.2133	0.2946
	file 2.1	0.8978	0.5705	0.2230	0.3206	0.8968	0.5446	0.2858	0.3748
	file 3.0	0.8453	0.4397	0.1652	0.2401	0.8453	0.4518	0.2124	0.2889
file 3.0	file 2.0	0.8575	0.5155	0.2728	0.3568	0.8575	0.5124	0.3384	0.4077
	file 2.1	0.8581	0.3221	0.2810	0.3002	0.8556	0.3277	0.3173	0.3224
	file 3.0	0.8662	0.5911	0.3240	0.4186	0.8637	0.5554	0.3974	0.4633

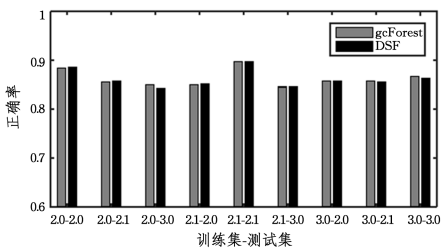

图 4　gcForest 和 DSF 在正确率上的对比

Fig. 4　Accuracy comparison of gcForest and DSF

从图 4 中可以看出，使用 gcForest 算法的预测正确率在 0.84～0.90 之间，平均值为 0.86；使用 DSF 算法的预测正确率在 0.84～0.90 之间，平均值为 0.86。由此可见，使用 gcForest 算法和使用 DSF 算法对 Eclipse 数据集的分类预测结果在正确率上基本都稳定在 0.84 以上，说明这两种算法都有很好的预测能力。

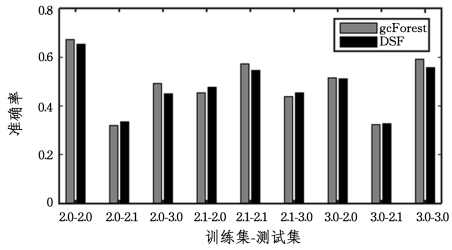

图 5　gcForest 和 DSF 在准确率上的对比

Fig. 5　Precision comparison of gcForest and DSF

从图 5 中可以看出，使用 gcForest 算法的预测准确率在 0.32～0.67 之间，平均值为 0.49；使用 DSF 算法的预测准确率在 0.32～0.65 之间，平均值为 0.48。由此可见，使用 gcForest 算法和使用 DSF 算法对 Eclipse 数据集的分类预测结果在准确率上虽略有下降，但基本持平，说明这两种算法在预测准确率上的性能相当。

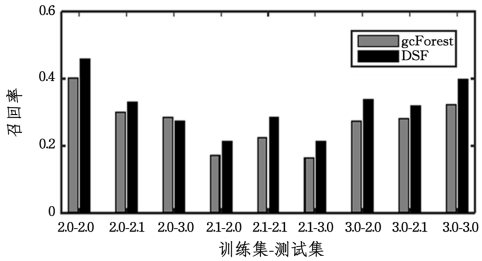

图 6 gcForest 和 DSF 在召回率上的对比

Fig. 6 Recall comparison of gcForest and DSF

从图 6 中可以看出，使用 gcForest 算法的预测召回率在 0.16～0.41 之间，平均值为 0.27；使用 DSF 算法的预测召回率在 0.27～0.47 之间，平均值为 0.32，并且基本都比使用 gcForest 算法的预测召回率高(有一种情况例外)。由此可见，使用 DSF 算法比使用 gcForest 算法对 Eclipse 数据集的分类预测结果在召回率上有明显提升，说明 DSF 算法比 gcForest 算法在召回率上具有明显的优势。

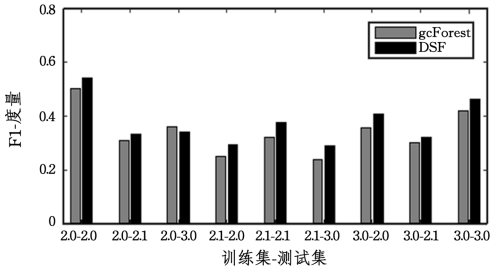

图 7 gcForest 和 DSF 在 F1-度量上的对比

Fig. 7 F1-measure comparison of gcForest and DSF

从图 7 中可以看出，使用 gcForest 算法的预测 F1-度量在 0.24～0.51 之间，平均值为 0.34；使用 DSF 算法的预测 F1-度量在 0.28～0.54 之间，平均值为 0.38，并且基本都比使用 gcForest 算法的预测 F1-度量高(有一种情况例外)。由此可见，使用 DSF 算法比使用 gcForest 算法对 Eclipse 数据集的分类预测结果在 F1-度量上有明显提升，说明 DSF 算法比 gcForest 算法在 F1-度量上具有明显的优势。

综合实验结果，DSF 算法在软件缺陷预测上的性能优于 gcForest 算法，并且在召回率和综合评价指标 F1-度量上具有明显的优势。

4.3 运行时间

图 8 显示了使用 gcForest 算法和使用 DSF 算法对 Eclipse 数据集进行缺陷预测时在运行时间上的对比(十折交叉验证时取其中一折的运行时间)。可以看出，使用 DSF 算法的运行时间均比使用 gcForest 算法的短，并且训练集样本越大，算法运行时间缩短得越多，从而说明 DSF 算法相比 gcForest 算法在效率上具有明显的优势。

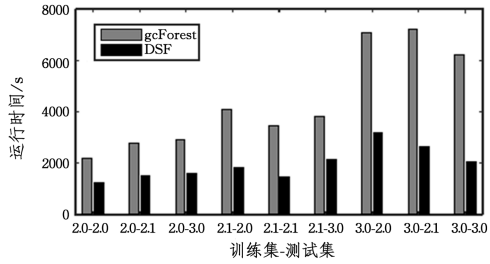

图 8 gcForest 和 DSF 在运行时间上的对比

Fig. 8 Running time comparison of gcForest and DSF

究其原因，以本实验为例，原始输入特征为 198 维，在特征变换阶段，gcForest 算法采用多粒度扫描，则需要分别训练 513 个随机森林和完全随机树森林，其变换后的特征向量为 2052 维；而 DSF 算法采用随机抽样，仅需要分别训练 350 个随机森林和完全随机树森林，其变换后的特征向量为 700 维，无疑比 gcForest 算法的训练速度快。在逐层表征学习阶段，gcForest 算法和 DSF 算法每层都分别需要训练两个随机森林和两个完全随机树森林，但 DSF 算法的为 700 维输入特征向量，相比 gcForest 算法的 2052 维输入特征向量，其训练速度要快得多，因此 DSF 算法比 gcForest 算法在时间效率上具有明显的优势，且在训练样本越多时，其时间效率优势越明显。

结束语 本文针对当前软件缺陷预测中浅层机器学习算法无法对软件数据特征进行深度挖掘的问题，在深度森林算法的基础上提出了深度堆叠森林算法，其首先采用随机抽样的方法对原始输入特征进行变换以增强特征表达能力，然后用堆叠结构对变换特征做逐层表征学习。将深度堆叠森林应用于软件缺陷预测，实验结果表明，深度堆叠森林算法在性能和效率上都比深度森林算法有一定提高。然而，该算法的预测性能仍没有达到预期效果，且没有考虑数据不平衡问题，下一步将研究如何进一步提升该算法的预测性能，并针对训练集中数据不平衡的问题提出解决方法。

参 考 文 献

[1] WANG Q, WU S J, LI M S. Software defect prediction[J]. Journal of Software, 2008, 19(7): 1565-1580. (in Chinese)
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.

[2] CHEN X, GU Q, LIU W S, et al. Survey of static software defect prediction[J]. Journal ofSoftware, 2016, 27(1): 1-25. (in Chinese)
陈翔，顾庆，刘望舒，等. 静态软件缺陷预测方法研究[J]. 软件学报，2016，27(1)：1-25.

[3] JINDAL R, MALHOTRA R, JAIN A. Software defect prediction using neural networks[C]// International Conference on Reliability, INFOCOM Technologies and Optimization. IEEE, 2015: 1-6.

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

[5] KALAI M R，GRACIA JACOB S. Improved Random Forest Algorithm for Software Defect Prediction through Data Mining Techniques[J]. International Journal of Computer Applications，2015，117(23)：18-22.

[6] WANG T，ZHANG Z，JING X，et al. Multiple kernel ensemble learning for software defect prediction[J]. Automated Software Engineering，2016，23(4)：1-22.

[7] THANGAVEL M，NASIRA G M. Support Vector Machine for Software Defect Prediction[J]. International Journal of Applied Engineering Research，2014，9(24)：25633-25644.

[8] SUN Z J，XUE L，XU Y M，et al. Overview of deep learning[J]. Application Research of Computers，2012，29(8)：2806-2810. (in Chinese)
孙志军，薛磊，许阳明，等. 深度学习研究综述[J]. 计算机应用研究，2012，29(8)：2806-2810.

[9] HINTON G E，SALAKHUTDINOV R R. Reducing the dimensionality of data with neural networks[J]. Science，2006，313(5786)：504.

[10] DENG L，DONG Y. Deep Learning：Methods and Applications[M]. Now Publishers Inc.，2014.

[11] ZHOU Z H，FENG J. Deep Forest：Towards An Alternative to Deep Neural Networks[C]// IJCAI-17. 2017：3553-3559.

[12] BREIMAN L. Random Forests[J]. Machine Learning，2001，45(1)：5-32.

[13] LIU F T，TING K M，YU Y，et al. Spectrum of variable-random trees[J]. Journal of Artificial Intelligence Research，2008，32(1)：355-384.

[14] WOLPERT D H. Stacked Generalization[J]. Neural Networks，1992，5(2)：241-259.

[15] ZIMMERMANN T，PREMRAJ R，ZELLER A. Predicting Defects for Eclipse[C]// International Workshop on Predictor MODELS in Software Engineering. IEEE，2007：9.

(上接第 133 页)

[6] LIU P，YUAN P Y. Routing selection and channel assignment method for mobile Ad Hoc cognitive network[J]. Computer Science，2017，44(3)：141-145. (in Chinese)
刘萍，袁培燕. 移动自组织认知网络中的路由选择与信道分配方法[J]. 计算机科学，2017，44(3)：141-145.

[7] FAN X J，LIU L F，LI S Y. An opportunistic routing algorithm based on emergency scenario in Ad Hoc networks[J]. Computer Technology and Development，2017，27(3)：6-11. (in Chinese)
范晓军，刘林峰，李思颖. 基于应急场景的自组织网络机会路由算法[J]. 计算机技术与发展，2017，27(3)：6-11.

[8] XIAO J，LIU W，TANG L. Routing algorithm for vehicular Ad Hoc network based on task allocation model [J]. Computer Engineering，2017，43(2)：6-15. (in Chinese)
肖晶，刘伟，唐伦. 基于任务分配模型的车载自组织网络路由算法[J]. 计算机工程，2017，43(2)：6-15.

[9] LIU B T，ZHOU Y，CHEN Y R. Research on the routing algorithm optimizing lifetime of wireless Ad Hoc networks [J]. Chinese Journal of Sensors and Actuators，2017，30(3)：463-466. (in Chinese)
刘半藤，周莹，陈友荣. 基于加权路由思想的无线自组织网络生存时间优化算法研究[J]. 传感技术学报，2017，30(3)：463-466.

[10] ZHANG D Y，CHEN Z G，ZHOU H B，et al. Energy-balanced cooperative transmission based on relay selection and power control in energy harvesting wireless sensor network[J]. Computer Networks，2016，104(20)：189-197.

[11] WANG N，LI D，LIU X L. Research of ant-colony-based connected dominating sets routing protocol in wireless Ad hoc networks [J]. Application Research of Computers，2016，33(12)：3822-3827. (in Chinese)
王娜，李丹，刘晓楳. 无线自组织网络中基于蚁群算法结合连通支配集的路由协议[J]. 计算机应用研究，2016，33(12)：3822-3827.

[12] CHEN X W，YUAN X B，LI B Q. Load balancing routing algorithm based on AODV for wireless sensor network [J]. Computer Engineering，2015，41(11)：142-146. (in Chinese)
陈昕韡，袁晓兵，李宝清. 基于 AODV 的无线自组织网络负载均衡路由算法[J]. 计算机工程，2015，41(11)：142-146.

[13] ZHAO C，LI J，DAI K C，et al. Soft sensor modeling for penicillin fermentation process based on adaptive weighted least squares support vector machine [J]. Journal of Nanjin University of Science and Technology，2017，41(1)：100-107. (in Chinese)
赵超，李俊，戴坤成，等. 基于自适应加权最小二乘支持向量机的青霉素发酵过程软测量建模[J]. 南京理工大学学报，2017，41(1)：100-107.

[14] TANG K Z，XIAO X，JIA J H，et al. Adaptive particle swarm optimization algorithm based on discrete estimate strategy of diversity [J]. Journal of Nanjing University of Science and Technology，2013，37(3)：344-349. (in Chinese)
汤可宗，肖绚，贾建华，等. 基于离散式多样性评价策略的自适应粒子群优化算法[J]. 南京理工大学学报，2013，37(3)：344-349.

[15] LIU B T，ZHOU Y，CHEN Y R，et al. Research on the routing algorithm in MANETs based on the energy cost function [J]. Chinese Journal of Sensors and Actuators，2017，30(2)：302-305. (in Chinese)
刘半藤，周莹，陈友荣，等. 基于移动-能量代价函数的无线自组织网络路由策略研究[J]. 传感技术学报，2017，30(2)：302-305.

[16] CAO J L，YU J，WANG L L，et al. An energy-efficient clustering routing protocol for wireless sensor networks[J]. Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition)，2014，26(2)：150-154. (in Chinese)
曹建玲，余俊，王路路，等. 一种能量高效的无线传感器网络分簇路由协议[J]. 重庆邮电大学学报(自然科学版)，2014，26(2)：150-154.