软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(11):3072−3079 [doi: 10.13328/j.cnki.jos.005345]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于代价敏感间隔分布优化的软件缺陷定位*

解 铮, 黎 铭

(计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

通讯作者: 黎铭, E-mail: lim@lamda.nju.edu.cn

摘 要: 在大型软件项目的开发与维护中,从大量的代码文件中定位软件缺陷费时、费力,有效地进行软件缺陷自动定位,将能极大地降低开发成本.软件缺陷报告通常包含了大量未发觉的软件缺陷的信息,精确地寻找与缺陷报告相关联的代码文件,对于降低维护成本具有重要意义.目前,已有一些基于深度神经网络的缺陷定位技术相对于传统方法,其效果有所提升,但相关工作大多关注网络结构的设计,缺乏对训练过程中损失函数的研究,而损失函数对于预测任务的性能会有极大的影响.在此背景下,提出了代价敏感的间隔分布优化(cost-sensitive margin distribution optimization,简称CSMDO)损失函数,并将代价敏感的间隔分布优化层应用到深度卷积神经网络中,能够良好地处理软件缺陷数据的不平衡性,进一步提高缺陷定位的准确度.

关键词: 软件挖掘;机器学习;缺陷定位;卷积神经网络;间隔分布优化
中图法分类号: TP311

中文引用格式: 解铮,黎铭.基于代价敏感间隔分布优化的软件缺陷定位.软件学报,2017,28(11):3072−3079. http://www.jos.org.cn/1000-9825/5345.htm

英文引用格式: Xie Z, Li M. Cost-Sensitive margin distribution optimization for software bug localization. Ruan Jian Xue Bao/ Journal of Software, 2017,28(11):3072−3079 (in Chinese). http://www.jos.org.cn/1000-9825/5345.htm

Cost-Sensitive Margin Distribution Optimization for Software Bug Localization

XIE Zheng, LI Ming

(State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

Abstract: It is costly to identify bugs from numerous source code files in a large software project. Thus, locating bug automatically and effectively becomes a worthy problem. Bug report is one of the most valuable source of bug description, and precisely locating related source codes linked to the bug reports can help reducing software development cost. Currently, most of the research on bug localization based on deep neural networks focus on design of network structures while lacking attention to the loss function, which impacts the performance significantly in prediction tasks. In this paper, a cost-sensitive margin distribution optimization (CSMDO) loss function is proposed and applied to deep neural networks. This new method is capable of handling the imbalance of software defect data sets, and improves the accuracy significantly.

Key words: software mining; machine learning; bug localization; convolutional neural network; margin distribution optimization

在大型软件项目的维护中,随着软件规模的扩大,软件的缺陷修复往往费时又费力,其开销在维护成本中所占的比重极大.因此,有效地对软件缺陷进行挖掘,对于提高软件质量具有重要意义[1,2].现有的软件缺陷挖掘技术利用不同的信息挖掘软件的缺陷[3−10],例如,对源代码特征建模,预测程序模块的缺陷数量或密度;通过分析程序执行轨迹或与缺陷报告之间的关联,定位程序缺陷位置等.

* 基金项目: 国家自然科学基金(61422304, 61272217)
Foundation item: National Natural Science Foundation of China (61422304, 61272217)
本文由复杂环境下的机器学习研究专刊特约编辑胡清华教授、张道强教授、张长水教授推荐.
收稿时间: 2017-05-09; 修改时间: 2017-06-16; 采用时间: 2017-08-23

近年来,缺陷报告作为大型软件项目中缺陷修复的重要参考,受到越来越多的重视.当软件崩溃或程序行为与预期相违背时,软件用户将使用中遇到的程序错误进行记录并加以描述,与软件自动生成的相关信息一同生成缺陷报告,并提交到软件的开发维护团队.开发人员则会根据缺陷报告所描述的信息,定位缺陷所在位置并进行修复.然而,对于大型项目而言,人工进行缺陷定位往往需要从数以千计的程序文件中进行排查,时间开销极大.为了能够通过缺陷报告中对程序缺陷的描述定位存在缺陷的源程序文件,需要分析以自然语言书写的缺陷报告描述与以程序语言编写的源代码之间的关联.通常,可以认为一个缺陷报告中描述了 1 个或多个缺陷,进而与 1 个或多个源代码文件相关联.目前,多数现有研究将源代码文件作为普通的自然语言处理,将源代码文件和缺陷报告一并使用词袋模型表示,并在相同的特征空间计算相似度.例如,Poshyvanyk 等人[5]提出了一种利用奇异值分解计算缺陷报告与源代码文件相似性的方法;Lukins 等人[6]使用生成式的 LDA(latent Dirichlet allocation)对每个源代码文件和缺陷报告进行建模,并依据其相似度判断与缺陷报告相关的源代码文件;Gay 等人[7]使用向量空间模型(vector space model,简称 VSM)对源代码文件与缺陷报告进行建模并衡量相似度;Zhou 等人[8]提出了 BugLocator 方法,通过计算缺陷报告与已修复的历史缺陷报告相似性的方法提升单纯使用 VSM 计算相似性的缺陷定位性能.近年来,基于深度学习的技术被应用于解决软件工程的相关问题.Lam 等人[9]将 auto encoder 用于对缺陷报告与源代码文件的特征提取;Huo 等人[10]提出了能够提取程序语言结构信息的新型卷积操作,对程序与自然语言学习统一特征映射,进一步提升了缺陷定位的准确度.

然而,现有基于深度学习的缺陷定位技术大多集中于新型网络结构的设计,而对网络损失函数的设计缺乏应有的关注.已有工作[9−11]大多在输出层使用 sigmoid 函数作为激活函数(或在多分类任务中使用 softmax 激活函数),采用交叉熵损失进行训练.这种损失函数对距离分类边界较远的异常点比较敏感,限制了缺陷定位性能的提升.目前,已有理论工作从间隔分布的角度对 AdaBoost 算法的良好泛化性能给出了解释[12,13].Zhang 等人[14]将这一思想应用于分类任务,提出了最优间隔分布机(optimal margin distribution machine,简称 ODM),并取得了良好的效果,证明了优化样本的间隔分布有利于提升分类性能.然而在缺陷预测这类数据高度不平衡的任务中,直接应用 ODM 往往难以处理数据的不平衡性.因此,本文提出了代价敏感的间隔分布优化(cost-sensitive margin distribution optimization,简称 CSMDO)损失函数,能够在特征空间中优化样本的间隔分布,并对正负样例引入不同的损失,使得模型能够从不平衡的数据中有效地进行学习.本文还提出了将 CSMDO 应用于神经网络训练的优化方法.在多个软件数据集上的实验结果表明,CSMDO 在现有的网络结构上能够显著提升缺陷定位的精度.

本文第 1 节介绍软件缺陷定位问题模型.第 2 节叙述代价敏感的间隔分布优化损失函数并提出优化方法.第 3 节介绍实验及结果.第 4 节进行总结.

1 软件缺陷定位模型

1.1 问题形式化

对于大型软件,缺陷报告是其开发维护过程中的重要参考.用户或开发者在发现软件行为与预期发生偏差时,可以在缺陷报告中对软件缺陷进行描述并提交至软件的开发团队.由于缺陷报告与源代码文件的数量庞大,软件缺陷定位旨在将缺陷报告与其所描述的错误相关的源代码文件定位出来,以减少开发者的工作量.令 $\mathcal{C}=\{c_1,c_2,\ldots,c_{n1}\}$表示全体源代码文件,$\mathcal{R}=\{r_1,r_2,\ldots,r_{n2}\}$表示全体缺陷报告,$y_{ij}\in\mathcal{Y}=\{+1,-1\}$表示缺陷报告与源代码文件是否相关.本文将一对缺陷报告与源代码文件的二元组作为样本,试图学习一个预测函数 $f:\mathcal{R}\times\mathcal{C}\to\mathbb{R}$,则 $\mathrm{sgn}(f(r_i,c_j))\in\mathcal{Y}=\{+1,-1\}$表示模型预测 r_i 与 c_j 是否相关.这一学习问题可以被重写为求解优化问题:

$$\min_f \ \mathcal{L}(f)=\lambda\sum_{i,j}L(f(r_i,c_j),y_{ij})+\Omega(f) \tag{1}$$

其中,$L(\cdot,\cdot)$为经验风险,$\Omega(\cdot)$为结构风险.参数λ用于对二者进行折中,使得模型具有较好的泛化性能.

1.2 网络结构

对于文本分类和特征提取问题已经有一些成熟的网络结构,例如文献[15,16].但相关研究[10]指出:由程序语

言编写的源代码包含重要的结构化信息,直接应用处理自然语言的方式对源代码文件进行特征提取会造成严重的信息损失.本文通过如图 1 所示的网络结构进行前述的学习过程,缺陷报告输入至自然语言卷积层进行特征学习,而源代码文件则使用针对结构化程序语言设计的卷积层学习特征表示.之后,两者的异构特征经过全连接层学习统一特征映射 $\boldsymbol{x}_{ij}=\phi(r_i,c_j)$后,供输出层进行预测.输出层采用代价敏感的间隔分布优化层计算代价来训练网络,使得网络可以通过优化样本在特征空间的间隔分布,使学到的预测函数 $f(r_i,c_j)$具有更好的泛化性能.

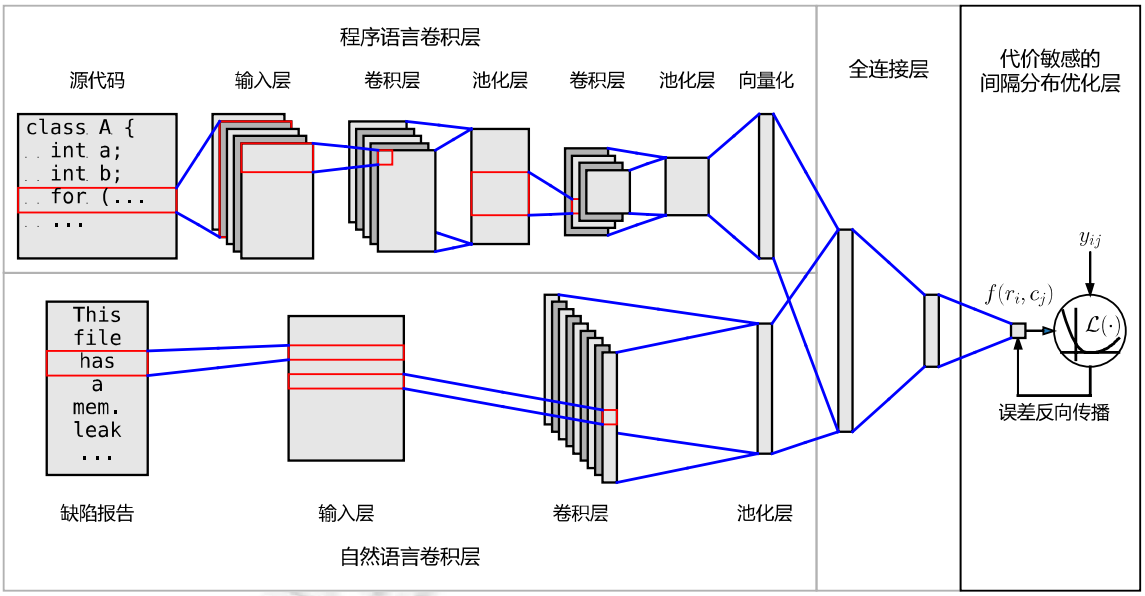

Fig.1 Convolutional neural network structure for bug localization

图 1 用于缺陷定位的卷积神经网络结构

本文使用 one-hot 方式对缺陷报告和源程序中的单词进行编码后输入网络,这一方式已被证实能够有效处理文本数据[17].例如,对于一个文本中出现过的单词集合 V={“bug”,“class”,“contains”,“file”,“is”,“this”},句子 S=“this file contains bug.”可以表示成

$$\boldsymbol{x}=\begin{bmatrix}0&0&0&0&0&1\\0&0&0&1&0&0\\0&0&1&0&0&0\\1&0&0&0&0&0\end{bmatrix}\begin{matrix}\text{this}\\\text{file}\\\text{contains}\\\text{bug}\end{matrix} \tag{2}$$

基于 one-hot 编码,缺陷报告将被表示成一个二维的矩阵.源程序文件的每一行源代码编码为矩阵,每个源程序文件则被表示为一个三维的张量.

自然语言卷积神经网络采用了一层卷积层和一层池化层,这一网络结构已被证明能够有效处理自然语言文本分类任务[18].程序语言卷积神经网络择由 2 个卷积层和 2 个池化层组成:第 1 层卷积层采用高度为 2 的卷积核,提取相邻语句的顺序结构信息;第 2 层卷积层采用了变化高度的卷积核,能够将程序不同粒度的结构化信息加以提取.自然语言卷积神经网络和程序语言卷及神经网络分别学习到缺陷报告与源代码文件的特征表达后,将被拼接一并通过全连接层学习一个统一的特征映射.代价敏感的间隔分布优化输出层对预测输出与期望输出计算分布优化损失并反向传播,优化了分类边界在统一特征空间中的间隔分布,这将在后面详细叙述.

本文提出的模型相对于现有的常用深度神经网络模型,需要学习的参数数量更少,训练所需的计算开销较小,能够很好地处理具有大量缺陷报告与程序代码的软件项目.

2 代价敏感的间隔分布优化层

2.1 代价敏感的间隔分布优化问题

支持向量机长久以来被证明是一种极为有效的学习算法,其核心思想为极大化最小间隔,即样本到分类边界的距离.与此同时,间隔理论也被应用于解释一些其他算法的良好泛化性能,例如 AdaBoost.Schapire 等人[19]首次尝试使用间隔理论解释 AdaBoost 不易于过度拟合的现象.Breiman[20]猜测最小间隔在其中起到重要作用,并提出了一种优化最小间隔的类 boosting 算法 Arc-gv,但泛化性能却较差.随后,Rayzin 等人[21]发现,Arc-gv 虽然以最小间隔为优化目标,但是具有较差的间隔分布,导致其泛化性能较差.近年来,已有相关理论工作证明 AdaBoost 算法优化了样本的间隔分布,对其泛化性能起到了至关重要的作用[12,13].Zhang 等人基于这一思想,提出了 ODM[14].ODM 通过对||$\boldsymbol{w}$||的缩放,将间隔均值固定为 1;通过引入对样本间隔偏离均值的惩罚,ODM 可以同时优化间隔均值与方差.

在缺陷定位问题中,每个缺陷报告所描述的缺陷只与少数源代码文件相关,这使得缺陷定位问题面临着数据的高度不平衡性.为了应对数据的不平衡性,本文将间隔分布优化问题通过代价敏感的方式扩展至非对称情况,形式化如下:

$$\begin{aligned}\min_{\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\varepsilon}}\ & \frac{1}{2}\boldsymbol{w}^{\top}\boldsymbol{w}+\frac{1}{m}\left(C_{+}\sum_{y_i=+1}\xi_i^2+C_{-}\sum_{y_i=-1}\xi_i^2+C_0\sum_{i=1}^{m}\varepsilon_i^2\right)\\ \text{s.t.}\ & y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i\geq 1-D-\xi_i,\\ & y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i\leq 1+D+\varepsilon_i,\\ & \xi_i\geq 0,\varepsilon_i\geq 0,i=1,\ldots,m.\end{aligned}\tag{3}$$

其中,第 1 项为结构风险项,用于优化样本间隔均值;第 2 项为经验风险项,用于优化样本间隔方差.参数 D 引入了一段类似支持向量回归中的不敏感区域,保持了解的稀疏性.参数 C_+与 C_-分别控制在间隔过小的正类与负类样本上的损失.当正类为少数类时,令 $C_+>C_-$,可使模型将正例误分为负例的代价高于将负例误分为正例的代价.参数 C_0 控制样本间隔过大时的损失.

2.2 损失函数形式化与优化

为了能够使用梯度方法训练神经网络,优化问题(3)可被重写为对于代价敏感的间隔分布优化损失函数的无约束优化问题:

$$\begin{aligned}\min_{\boldsymbol{w}}\ \mathcal{L}(\boldsymbol{w})=&\frac{1}{2}\boldsymbol{w}^{\top}\boldsymbol{w}+\frac{C_{+}}{m}\sum_{y_i=+1}\max\{0,1-D-y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i\}^2+\\ &\frac{C_{-}}{m}\sum_{y_i=-1}\max\{0,1-D-y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i\}^2+\frac{C_0}{m}\sum_{i=1}^{m}\max\{0,y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i-1-D\}^2\end{aligned}\tag{4}$$

对于神经网络的训练,计算目标函数在所有样本上的梯度进行训练开销极大.下面的定理说明了在单个随机抽样的样本计算的梯度是全梯度的无偏估计.

定理 **1**. 如果样例($\boldsymbol{x}_i$,y_i)是从训练集中随机抽样而得,则在随机采样的单个样本上的梯度$\nabla\mathcal{L}(\boldsymbol{w},\boldsymbol{x}_i,y_i)$是全梯度$\nabla\mathcal{L}(\boldsymbol{w})$的无偏估计.

证明:$\mathcal{L}(\boldsymbol{w})$的梯度为

$$\begin{aligned}\nabla\mathcal{L}(\boldsymbol{w})=&\boldsymbol{w}+\frac{2C_{+}}{m}\sum_{i=1}^{m}(y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i+D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i<1-D)\mathbb{I}(y_i=+1)+\\ &\frac{2C_{-}}{m}\sum_{i=1}^{m}(y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i+D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i<1-D)\mathbb{I}(y_i=-1)+\\ &\frac{2C_0}{m}\sum_{i=1}^{m}(y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i-D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^{\top}\boldsymbol{x}_i>1+D)\end{aligned}\tag{5}$$

其中,$\mathbb{I}(\cdot)$是指示函数,当条件成立时取值为 1,否则取值为 0.根据期望的线性性,可得:

$$\begin{aligned}\mathbb{E}[\nabla\mathcal{L}(\boldsymbol{w},\boldsymbol{x}_i,y_i)] &= \boldsymbol{w}+2C_+\mathbb{E}[(y_i\boldsymbol{w}^\top\boldsymbol{x}_i+D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i<1-D)\mathbb{I}(y_i=+1)]+\\ &\quad 2C_-\mathbb{E}[(y_i\boldsymbol{w}^\top\boldsymbol{x}_i+D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i<1-D)\mathbb{I}(y_i=-1)]+\\ &\quad 2C_0\mathbb{E}[(y_i\boldsymbol{w}^\top\boldsymbol{x}_i-D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i>1+D)]\\ &= \boldsymbol{w}+\frac{2C_+}{m}\sum_{i=1}^{m}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i+D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i<1-D)\mathbb{I}(y_i=+1)+\\ &\quad \frac{2C_-}{m}\sum_{i=1}^{m}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i+D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i<1-D)\mathbb{I}(y_i=-1)+\\ &\quad \frac{2C_0}{m}\sum_{i=1}^{m}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i-D-1)y_i\boldsymbol{x}_i\mathbb{I}(y_i\boldsymbol{w}^\top\boldsymbol{x}_i>1+D)\\ &= \nabla\mathcal{L}(\boldsymbol{w})\end{aligned} \tag{6}$$

因此,$\nabla\mathcal{L}(\boldsymbol{w},\boldsymbol{x}_i,y_i)$是$\nabla\mathcal{L}(\boldsymbol{w})$的无偏估计. □

根据这一结论,使用代价敏感间隔分布优化层的神经网络可以避免在全梯度上进行训练.本文中,网络使用小批量梯度下降(mini-batch gradient descent)算法,利用网络反向传播的梯度对权重进行更新.

3 实 验

为了验证代价敏感的分部间隔优化损失的有效性,本文在多个开源软件数据上进行了实验.对比的方法包含多个现有的缺陷定位方法以及本文所使用的网络结构的变体.

3.1 实验设置

实验中所使用的 4 个数据集分别取自几个开源软件项目,项目的每个版本的源代码可从项目的版本管理系统获得,缺陷报告则可以抓取自项目的在线错误追踪系统.通过对版本管理系统的提交记录与修改信息进行解析,可以得到真实的缺陷报告与源代码文件的关联性信息,作为训练数据的标记[22].

这些数据集在已有的研究中已被广泛使用[8,9],其中,JDT(Java development tools)项目包含为 Eclipse 平台提供完整 Java IDE 的功能的一系列插件,PF(eclipse platform)项目包含组成 Eclipse 基础设施的框架和公共服务,PDE(plug-in development environment)项目是用于开发 Eclipse 插件的工具,AspectJ 项目是 Java 的面向方面程序设计的扩展.表 1 中展示了数据集的统计信息.

Table 1 Statistics of the data sets

表 1 数据集统计信息

数据集	缺陷报告数	源代码文件数	平均每报告关联的文件数
JDT	12 826	2 272	4.39
PF	14 893	1 012	6.79
PDE	4 034	2 970	8.34
AspectJ	1 734	1 136	1.73

如表 1 所示,每个项目中潜在可能包含缺陷的源代码文件众多,但与每个缺陷报告相关联的文件数量极少,数据集具有高度的不平衡性.为此,我们选用 Top 10 Rank,AUC(area under ROC curve)与 MAP(mean average precision)作为实验的评价指标[8].

实验用对比的方法选取了一些现有的缺陷定位方法,其中,BugLocator[8]为 Zhou 等人提出的一种基于信息检索的缺陷定位方法,使用修改的向量空间模型计算缺陷报告与源代码文件的相似度以将两者进行关联;Two-Phase[18]为 Kim 等人提出的两阶段缺陷定位方法,该方法首先通过朴素贝叶斯筛除无关的缺陷报告,再通过向量空间模型预测含有错误的源代码文件;HyLoc[9]为 Lam 等人提出的缺陷定位方法,将 auto encoder 与向量空间模型相结合,以根据缺陷报告定位有缺陷的源代码文件.这些方法在实验中都使用了各自文献中的推荐参数.

另外,为了验证本文损失函数的有效性,实验还对比了使用两种常见损失函数的方法变体.这些方法都使用

第 2 节所述的相同的网络结构,仅改变了输出层的损失函数,以说明损失函数对性能的影响.其中,NP-CNN^{logistic}使用在分类任务中最常用的交叉熵损失(cross entropy),也即 Logistic 回归中的 logistic 损失.NP-CNN^{hinge}使用 SVM 中优化的 hinge 损失,加入正则化项后,优化这一损失函数相当于优化特征空间中样例的最小间隔.这些基于神经网络的方法通过在每个数据集上进行交叉验证选择最优参数.实验中,使用本文提出的代价敏感的间隔分布优化损失的方法记为 CSMDO.

3.2 实验结果

每个数据集的实验被重复 10 次并计算平均指标,以消除随机因素的影响.Top 10 Rank,MAP 和 AUC 分别在表 2~表 4 中列出.在实验结果中,显著优于/差于 CSMDO 的结果被分别以•/◦标注(Wilcoxon 符号秩检验,显著性水平 0.05).

Table 2 Top 10 Rank of compared methods on all data sets

表 2 对比方法在不同数据集上的 Top 10 Rank 指标

Data set	BugLocator	Two-Phase	HyLoc	NP-CNN^{logistic}	NP-CNN^{higne}	CSMDO
JDT	0.742◦	0.642◦	0.789◦	**0.938**	0.933	0.935
PF	0.725◦	0.732◦	0.760◦	0.894	0.867◦	**0.897**
PDE	0.673◦	0.617◦	0.718◦	0.842◦	0.851◦	**0.861**
AspectJ	0.622◦	0.570◦	0.741◦	0.848◦	0.842◦	**0.871**
Avg.	0.691	0.640	0.752	0.880	0.873	**0.891**

Table 3 MAP of compared methods on all data sets

表 3 对比方法在不同数据集上的 MAP 指标

Data set	BugLocator	Two-Phase	HyLoc	NP-CNN^{logistic}	NP-CNN^{higne}	CSMDO
JDT	0.441◦	0.372◦	0.455◦	0.522◦	0.525◦	**0.533**
PF	0.350◦	0.398◦	0.410◦	0.537	0.532◦	**0.539**
PDE	0.422◦	0.381◦	0.552◦	0.624◦	0.632◦	**0.674**
AspectJ	0.418◦	0.397◦	0.433◦	0.545◦	0.546◦	**0.577**
Avg.	0.408	0.387	0.463	0.557	0.559	**0.581**

Table 4 AUC of comparedmethods on all data sets

表 4 对比方法在不同数据集上的 AUC 指标

Data set	BugLocator	Two-Phase	HyLoc	NP-CNN^{logistic}	NP-CNN^{higne}	CSMDO
JDT	0.781◦	0.724◦	0.813◦	0.881◦	0.917◦	**0.927•**
PF	0.772◦	0.803◦	0.830◦	0.913◦	0.895◦	**0.941•**
PDE	0.753◦	0.763◦	0.824◦	0.914◦	0.925•	**0.927•**
AspectJ	0.683◦	0.664◦	0.761◦	0.857◦	0.852◦	**0.926•**
Avg.	0.747	0.739	0.807	0.891	0.897	**0.930**

从实验结果中可以看出,CSMDO 在各个数据集上均取得了很好的缺陷定位性能.根据表 4,相对于没有使用深度神经网络的 BugLocator,Two-Phase 和 HyLoc 这 3 种方法,CSMDO 在 AUC 上平均提升了 18.3%,19.1%和 12.3%,在其他指标上也有大幅度提升.这一结果说明:对于使用向量空间模型计算源代码文件与缺陷报告的文本相似度的方法而言,CSMDO 的程序语言卷积层能够更好地处理源代码文件的结构化信息,以抽取有助于缺陷定位的高层次的语义特征.

对比在同样网络结构下使用交叉熵损失的 NP-CNN^{logistic},CSMDO 在 Top 10 Rank,MAP 和 AUC 这 3 种指标上平均提升了 1.1%,2.4%和 3.9%,而相对于使用 hinge 损失的 NP-CNN^{hinge},则分别提升了 1.8%,2.2%和 3.3%.根据实验结果,使用 CSMDO 损失函数在全部数据集的 MAP 和 AUC 指标上和在 3 个数据集上的 Top 10 Rank 性能均有所提高.这说明,通过使用代价敏感的间隔分布优化损失优化样本在特征空间中的间隔分布,可以有效

地适应软件缺陷数据的不平衡性,提升网络模型在缺陷定位数据上的泛化性能.

4 结 论

本文从间隔分布优化和应对非平衡数据的角度提出了一种用于分类任务的新型损失函数,即代价敏感的间隔分布优化(CSMDO)损失函数,并将其应用到神经网络的优化求解中.对于该损失函数的优化,相当于优化特征空间中样本的间隔分布,而间隔分布这一概念在相关理论工作中被证实对模型泛化误差的控制具有重要意义.在多个大型软件工程数据集上的实验结果表明,该损失函数能够良好地处理高度不平衡的软件缺陷定位数据,对于同样的网络结构,应用该损失函数可以显著地提升缺陷定位的准确度.事实上,该损失函数并不局限于软件缺陷定位任务中,可以直接推广到更多任务上.

本文的工作揭示了神经网络效果的提升并不只在于网络结构的改变,对于损失函数的良好设计一样十分重要.如何更好地将软件缺陷数据的特性嵌入损失函数,使其能够更好地刻画和解决缺陷定位问题,仍有待进一步研究.

References:

[1] Li M, Huo X. Software defect mining based on semi-supervised learning. Journal of Data Acquisition and Processing, 2016,31(1): 56−64 (in Chinese with English abstract). [doi: 10.16337/j.1004-9037.2016.01.005]

[2] Li M. Software defect mining. Chinese Association for Artificial Intelligence Conmmunication, 2016,2(8):44−49 (in Chinese with English abstract).

[3] Tu WW, Li M, Zhou ZH. Mining software defect factor. Journal of Jilin University (Engineering and Technology Edition), 2012, 42(s1):383−386 (in Chinese with English abstract).

[4] Ding H, Chen L, Qian J, Xu L, Xu BW. Fault localization method using information quantity. Ruan Jian Xue Bao/Journal of Software, 2013,24(7):1484−1494 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4294.htm [doi: 10.3724/SP.J. 1001.2013.04294]

[5] Poshyvanyk D, Gueheneuc YG, Marcus A, Antoniol G, Rajlich VC. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. IEEE Trans. on Software Engineering, 2007,33(6):420−432. [doi: 10.1109/tse. 2007.1016]

[6] Lukins SK, Kraft N, Etzkorn LH. Source code retrieval for bug localization using latent dirichlet allocation. In: Proc. of 15th Working Conf. on Reverse Engineering. IEEE, 2008. 155−164. [doi: 10.1109/wcre.2008.33]

[7] Gay G, Haiduc S, Marcus A, Menzies T. On the use of relevance feedback in IR-based concept location. In: Proc. of Int'l Conf. on Software Maintenance. 2009. 351−360. [doi: 10.1109/icsm.2009.5306315]

[8] Zhou J, Zhang H, Lo D. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports. In: Proc. of the 34th Int'l Conf. on Software Engineering. 2012. 14−24. [doi: 10.1109/icse.2012.6227210]

[9] Lam AN, Nguyen AT, Nguyen HA, Nguyen TN. Combining deep learning with information retrieval to localize buggy files for bug reports. In: Proc. of the 28th Int'l Conf. on Automated Software Engineering. 2015. [doi: 10.1109/ase.2015.73]

[10] Huo X, Li M, Zhou ZH. Learning unified features from natural and programming languages for locating buggy source code. In: Proc. of the 25th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2016). New York, 2016. 1606−1612.

[11] Mou L, Li G, Zhang L, Wang T, Jin Z. Convolutional neural networks over tree structures for programming language processing. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence. 2016. 1287−1293.

[12] Wang L, Sugiyama M, Jing Z, Yang C, Zhou ZH, Feng J. A refined margin analysis for boosting algorithms via equilibrium margin. Journal of Machine Learning Research, 2011,12:1835−1863.

[13] Gao W, Zhou ZH. On the doubt about margin explanation of boosting. Artificial Intelligence, 2013,203:1−18. [doi: 10.1016/j.artint. 2013.07.002]

[14] Zhang T, Zhou ZH. Optimal margin distribution machine. arXiv:1604.03348, 2016.

[15] Johnson R, Zhang T. Effective use of word order for text categorization with convolutional neural networks. In: Proc. of the Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015. 103−112. [doi: 10.3115/v1/n15-1011]

[16] Zhang X, LeCun Y. Text understanding from scratch. arXiv:1502.01710, 2015.

[17] Kim Y. Convolutional neural networks for sentence classification. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing. ACL, 2014. 1746−1751. [doi: 10.3115/v1/d14-1181]

[18] Kim D, Zeller A, Tao Y, Kim S. Where should we fix this bug? A two-phase recommendation model. IEEE Trans. on Software Engineering, IEEE, 2013,39(11):1597−1610. [doi: 10.1109/tse.2013.24]

[19] Schapire RE, Freund Y, Barlett P, Lee WS. Boosting the margin: A new explanation for the effectiveness of voting methods. In: Fisher DH, ed. Proc. of the 14th Int'l Conf. on Machine Learning (ICML'97). Morgan Kaufmann Publishers, 1997. 322−330.

[20] Breiman L. Prediction games and arcing algorithms. Neural Computation, 1999,11(7):1493−1517. [doi: 10.1162/089976699300016106]

[21] Reyzin L, Schapire RE. How boosting the margin can also boost classifier complexity. In: Cohen WW, Moore A, eds. Proc. of the 23rd Int'l Conf. on Machine Learning (ICML 2006), Vol.148. 2006. 753−760. [doi: 10.1145/1143844.1143939]

[22] Fischer M, Pinzger M, Gall H. Populating a release history database from version control and bug tracking systems. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE, 2003. 23−32. [doi: 10.1109/icsm.2003.1235403]

附中文参考文献:

[1] 黎铭,霍轩.半监督软件缺陷挖掘研究综述.数据采集与处理,2016,31(1):56−64. [doi: 10.16337/j.1004-9037.2016.01.005]

[2] 黎铭.软件缺陷挖掘.中国人工智能学会通讯,2016,2(8):44−49.

[3] 涂威威,黎铭,周志华.软件缺陷因素挖掘.吉林大学学报(工学版),2012,42(s1):383−386.

[4] 丁晖,陈林,钱巨,许蕾,徐宝文.一种基于信息量的缺陷定位方法.软件学报,2013,24(7):1484−1494. http://www.jos.org.cn/1000-9825/4294.htm [doi: 10.3724/SP.J.1001.2013.04294]

解铮(1994－),男,山东济南人,硕士生,主要研究领域为机器学习,软件挖掘.

黎铭(1980－),男,博士,副教授,博士生导师,CCF 专业会员,主要研究领域为软件挖掘,机器学习.