深度信念网软件缺陷预测模型

甘 露 臧 洌 李 航
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 软件缺陷预测技术在检测软件缺陷、保证软件质量方面发挥了重要的作用。利用神经网络分类算法构建的软件缺陷预测模型得到了广泛的应用。但是利用神经网络分类算法训练历史数据只能进行"浅层学习",无法对数据特征进行深度挖掘。针对该问题,利用多层限制玻尔兹曼机叠加成深度信念网,先进行特征集成与迭代,并对这些特征数据进行深度学习,构建了基于深度信念网的软件缺陷预测模型(DBNSDPM)。仿真实验表明,本模型预测的准确性与传统的神经网络缺陷预测模型预测的准确性相比有显著提高。

关键词 软件缺陷预测,限制玻尔兹曼机,深度学习,深度信念网

中图法分类号 TP392 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.049

Deep Belief Network Software Defect Prediction Model

GAN Lu ZANG Lie LI Hang
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Software defect prediction technology plays an important role in detecting software defect and ensuring software quality. Using the neural network classification algorithm to build software defect prediction model has been used widely. But the neural network classification algorithm to train historical data is only shallow learning, this algorithm can't deeply extract data features. To solve this problem, the deep belief network software defect prediction model (DBNSDPM) by using a deep belief nets which is composed of multilayer restricted Boltzmann machine was constructed. This model conducts feature integration and iteration firstly, then the characteristic data can be studied deeply. The simulation experiment proves that the prediction accuracy of DBNSDPM improves significantly than the traditional neural network prediction model.

Keywords Software defect prediction, Restricted Boltzmann machine, Deep learning, Deep belief network

1 引言

软件缺陷预测技术[1]自 Briand 于 1992 年提出以来一直备受关注。近年来,众多研究人员都对软件缺陷预测技术进行了各种研究,很多软件缺陷预测模型。Nagappan 等人[2]于 2006 年提出了从度量中提取原子组件并使用这些组件进行缺陷预测;Kim 等人[3]于 2007 年通过获取软件改变的历史数据缓存位置来预测缺陷;之后,Zimmermann 和 Nagappan[4]又提出了基于结构测量模型的依赖图;Menzies 等人[5]于 2010 年建立了基于静态编码属性的缺陷预测模型;陈翔等人[6]于 2016 年指出了度量元的设定是影响缺陷预测性能的三个重要影响因素之一。从这些工作中可以发现,简单的复杂性指标(如代码行数)可用于构建有效的预测模型。目前主流的缺陷预测模型有支持向量机模型[7]、神经网络模型[8]、C4.5 决策树模型和贝叶斯网络[9]模型,其中人工神经网络缺陷预测模型由于训练过程简单、时间短等特点在软件缺陷预测工作中发挥了巨大的作用。

人工神经网络是一种机器学习方法,在模式识别和语音识别领域发挥了巨大作用,在软件缺陷预测领域也有很多研究者建立了基于神经网络的缺陷预测模型。神经网络是模仿大脑的神经系统接受信号、产生脉冲、处理并作出反应的生理现象来构造的,含有多个感知器,用来接受输入信号。因此,可以训练一个神经网络,通过调整元素之间的连接值(权重)来得到特定的分类函数。然而,支持向量机、神经网络模型、C4.5 决策树和贝叶斯网络模型等机器学习方法只能称为"浅层学习",无法对数据的特征进行深度挖掘,多层的神经网络虽然对数据特征有进一步的探索,但是由于缺乏有效的参数优化方法,多层神经网络的研究一直停止不前,直到 2006 年 Hinton 提出包含多个层级的深度网络结构的机器学习过程:

$$\mathrm{sgn}(y)=\begin{cases}1, & \text{if } y>0\\ -1, & \text{otherwise}\end{cases}$$

从此,深度学习[10]引起了学术界的广泛关注。随后,由一些限制波尔兹曼机(Restricted Boltzmann Machine, RBM)[11]组成的深度自编码网络也得以研究,这些网络能够更有效地学

到稿日期:2016-03-03 返修日期:2016-08-31

甘 露(1991—),女,硕士生,主要研究方向为软件测试,E-mail:1321710271@qq.com;臧 洌(1964—),女,硕士,副教授,主要研究方向为网络安全及软件可靠性;李 航(1991—),男,硕士生,主要研究方向为机器学习。

习低维码。Hinton 等人还于 2006 年提出了深度信念网(Deep Belief Networks,DBN)的一种快速学习算法[12];Hinton 于 2007 年提出了基于多层网络特征的探测器,介绍了多层生成模型的局限性以及 RBM 是如何找到一种有效的深层生成算法模型的关键所在,并且详细地介绍了如何使用 RBM 去学习多层特征网络;Wang 和 Liu 等人[13]提出利用 DBN 自动学习源代码的语义特征,并利用学习到的特征训练和构建缺陷预测模型;Yang 等人[14]利用 DBN 对 14 个基本特征进行特征提取,然后利用回归算法建立分类模型以进行软件缺陷预测。简单来说,深度信念网[15]就是一种多层神经网络,它利用无监督训练方法解决了多层神经网络的训练优化问题,通过多层网络结构进一步挖掘数据特征。本文针对传统分类方法无法深度挖掘数据特征以及多层神经网络缺乏有效的权值训练方法的缺点,利用深度信念网自身的结构特点以及将无监督和有监督训练方法结合的训练方法,构建了 DBNSDPM,并利用 DBNSDPM 对缺陷数据进行了预测,取得了较好的预测结果。

2 传统的神经网络软件缺陷预测模型

以感知器的单元为基础是最简单的神经网络模型。感知器以一个实数值向量作为输入,计算这些输入的线性组合,若得到的结果大于事先规定的阈值,则输出 1,反之输出 -1。假设输入为 x_1 到 x_n,那么感知器的计算输出为:

$$o(x_1,x_2,\cdots,x_n)=\begin{cases}1, & \text{if } w_0+w_1x_1+w_2x_2+\cdots+w_nx_n>0\\ -1, & \text{otherwise}\end{cases} \tag{1}$$

其中,w_i 叫做权值,用来决定输入变量 x_i 对感知器输出的影响力,如果添加一个假想量 $w_0=1$,那么上面的不等式约束就可改为 $\sum_{i=0}^{n}w_ix_i>0$,或以向量的形式表示为 $\vec{w}\cdot\vec{x}>0$。那么感知器的函数也可以简单表示为:

$$o(\vec{x})=\text{sgn}(\vec{w}\cdot\vec{x}) \tag{2}$$

一系列确定的基本单元相互连接形成的网络为多层网络,多层网络的输入为可见结点,中间的结点为隐藏结点,最后一层为输出结点。由限制玻尔兹曼机构成的深度信念网是一个多层神经网络,玻尔兹曼机以自身结构的特点以及独特的训练方法克服了多层神经网络权值难以训练的缺点,从而使得多层神经网络得以广泛应用。

3 深度信念网软件缺陷预测模型

3.1 样本的类不平衡处理

类不平衡问题是软件缺陷预测领域常见的问题之一,由于 80%的软件缺陷存在于 20%的模块中,因此软件缺陷的数据远远小于非缺陷的数据,这样会造成在软件缺陷数据集中正例的数据远远的小于负例的数据。类的不平衡问题往往会对软件缺陷预测模型的准确率造成较大影响,所以为了降低类不平衡对预测准确性的影响,实验前对数据集进行了欠抽样处理。

欠抽样指通过去除多余样本的数目来减少类的不平衡程度,通过随机删除缺陷数据使得其与缺陷数据样本的数量的差异减少来降低不平衡程度,相关的伪代码如算法 1 所示。

算法 1 TomekLinks(P,tl_list)

输入:训练集的输入数据 P

输出:计算 Tomek Links 的结果集 tl_list

```
1. 将训练集 P 中的正例复制到 DateSetP 中;
2. 将训练集 P 中的负例复制到 DateSetN 中;
3. for each ia in DataSetP do
4.     for each ib in DataSetN do
5.         for each ik in P do
6.             Find=false;
7.             if dist(ia,ik)<dist(ia,ib) or dist(ib,ik)<dist(ia,ib)
8.                 Find=true;
9.                 break;
10.            end if
11.        end for
12.        if Find 为 true
13.            tl_list.push(<ia,ib>);
14.        end if
15.    end for
16. end for
```

3.2 DBNSDPM 框架

DBNSDPM 的思想:通过叠加 RBM,从特征中提取特征,从而得到更好的功能特性。一个 RBM 拥有一层可见结点和一层隐藏结点,其中可见结点是输入结点,而隐藏结点是处理输入并进行特征集成的,DBNSDPM 可由若干个 RBM 组成,一个 RBM 的隐藏结点是下一个 RBM 的输入结点,因此组成的 DBNSDPM 含有一层输入结点和若干层隐藏结点。通过若干层隐藏结点对特征的提取和迭代能够充分地挖掘输入数据之间的复杂关系,有利于正确地分析处理样本,得到正确有效的结论。

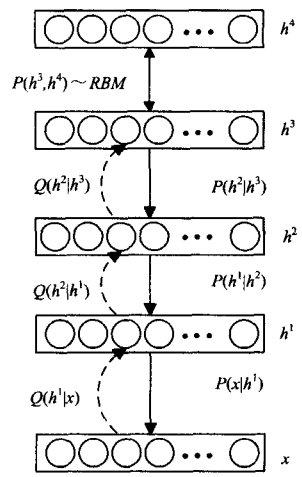

图 1 DBNSDPM 的框架图

DBNSDPM 的框架如图 1 所示,主要由 4 个 RBM 组成,其结构为 22-20-10-10-2,第一个 RBM 有 22 个可见结点,代表缺陷数据集的 22 个属性,20 个隐藏结点作为第二个 RBM 的输入,并且第二个 RBM 有 10 个隐藏结点,这也是第三个 RBM 的输入,第三个 RBM 有 10 个隐藏结点,第四个 RBM 有 10 可见结点和两个隐藏输出,这两个输出即是所需的实验结果。DBNSDPM 的训练总原则:第一步,运用无监督方法训

练 DBNSDPM 的每一层结点;第二步,运用有监督方法对每层结点的权值进行调优。本文提出的软件缺陷预测模型先运用无监督算法训练每个 RBM,再利用有监督算法对每个权值进行微调,DBNSDPM 采用的有监督算法是神经网络算法。

3.3 DBNSDPM 算法

限制玻尔兹曼机包括两种类型的单元:可见单元和隐藏单元。其可以排列成两层,第一层是可见单元,代表输入,隐藏单元依赖于输入值。它可以被看作非线性特征的探测器。

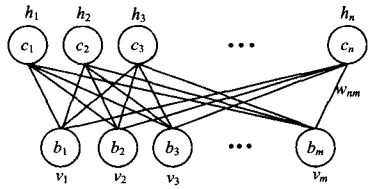

图 2　玻尔兹曼机

限制玻耳兹曼机的结构如图 2 所示,限制玻尔兹曼机采用 m 个可见结点 $V=(V_1,V_2,\cdots,V_m)$来表示输入数据的属性值,用 n 个隐藏结点 $H=(H_1,H_2,\cdots,H_n)$来表示输入数据的属性之间的依赖关系,这种依赖关系称为特征。所有 V 和 H 的值都在 0 和 1 之间,并且根据吉普斯抽样得到了它的联合概率分布以及能量公式:

$$p(v,h)=\frac{1}{Z}e^{-E(v,h)} \tag{3}$$

$$E(v,h)=-\sum_{i=1}^{n}\sum_{j=1}^{m}w_{ij}h_iv_j-\sum_{j=1}^{m}b_jv_j-\sum_{i=1}^{n}c_ih_i \tag{4}$$

对于所有的 $i\in(1,\cdots,n)$,$j\in(1,\cdots,m)$,w_{ij} 是连接 V_j,H_i 之间的权值,b_j 和 c_i 是第 j 个可见结点和第 i 个隐藏结点的偏量。RBM 结点之间的连接仅限于可见结点和隐藏结点之间,同层结点之间不存在连接,所以在给定可见结点后,隐藏结点之间的概率是独立的,反之亦然。

$$p(h|v)=\prod_{i=1}^{n}p(h_i|v),p(v|h)=\prod_{i=1}^{m}p(v_i|h) \tag{5}$$

可以计算得到可见结点的边缘分布概率为:

$$p(v)=\frac{1}{Z}\sum_h p(v,h)=\frac{1}{Z}\sum_h p(v,h)e^{-E(v,h)}$$

$$=\frac{1}{Z}\prod_{j=1}^{m}e^{b_jv_j}\prod_{i=1}^{n}(1+e^{c_i+\sum_{j=1}^{m}w_{ij}v_j}) \tag{6}$$

任何 0～1 之间的分布都可以用 RBM 的 m 个可见结点和 $k+1$ 个隐藏结点来建模,其中 k 表示目标集的分布,m 表示输入数据的属性元素的个数。

在给出输入向量 x_1 以后,由式(3)可以进一步得到:

$$p(h_i=1|x)=\frac{e^{c_i+w_ix}}{1+e^{c_i+w_ix}}=sigm(c_i+w_ix) \tag{7}$$

$$p(x_j=1|h)=sigm(b_j+w_j'h) \tag{8}$$

DBNSDPM 联合分布概率定义如下:

$$p(x,h^1,\cdots,h^4)=(\prod_{k=0}^{2}p(h^k|h^{k+1}))p(h^{3-1},h^4) \tag{9}$$

其中,$x=h^0$,$p(h^{k-1}|h^k)$是 k 层深度信念网中的 RBM 的条件概率,$p(h^3,h^4)$是顶层 RBM 的联合分布概率。在图 2 中,顶层 h^3 和 h^4 给出的是联合分布,其中 Q 是后验概率,用来表示 p 的可能性,在 DBNSDPM 的训练过程中用 $Q(h^{k-1}|^k)$来代表 $p(h^{k-1}|h^k)$的估计值的原因是可以很容易地从 $Q(h^{k-1}|^k)$中抽样出 x 的同义表示。具体的 DBNSDPM 的训练过程为:首先从 $Q(h^1|x)$抽样第一个 RBM 的隐藏结点值 h^1,当期望值大于玻尔兹曼分布 $Q(h^1|x)$时可用平均场方法计算 $\hat{h}^1=E[h^1|x]$来代替h^1,然后将 h^1 作为第二层结点的输入值,计算 h^2 或者$\hat{h}^2$,依次类推,完成整个 DBNSDPM 的训练后即可得到 DBNSDPM 的初始权值 w^i 和偏置 c^i,最后运用神经网络算法对 DBNSDPM 权值进行微调,应用神经网络算法对权值进行微调以后相当于该网络具有了分类能力,即可通过该算法建立软件缺陷预测模型来完成对软件模块的缺陷预测,具体的 DBNSDPM 伪代码如算法 2 所示。

算法 2　DBNSDPM($P,lr,length,W,b,c,mean\_field\_computation$)

输入:训练网络的输入数据 P;RBM 训练的学习率 lr;训练的层数 length;在 k 层中,可见层相对于 RMB 训练的偏置 b^k(k 从 1 到 length);在 k 层中,隐藏层相对于 RMB 训练的偏置 c^k(k 从 1 到 length);布尔值 mean_field_computation,当且仅当在每个附加层的训练数据是通过 mean-field 渐进获取而不是随机抽样时为 true,否则为 false

输出:k 层的权值矩阵 W^k(k 从 1 到 length)

```
1. for k=1 to lengh do
2.   初始化 W^k=0,b^k=0,c^k=0;
3.   while 没有停止标志 do
4.     从 P 中抽样,得到样本 x,并且赋值给 h^0// h^0=x
5.     for i =1 to k-1 do
6.       if mean_field_computation then
7.         h_j^i=Q(h_j^i=1|h^(i-1));// 对于 h^i 中所有的元素 j
8.       else
9.       从 Q(h_j^i|h^(i-1)) 抽取样本 h_j^i;//对于 h^i中所有的元素 j
10.       end if
11.     end for
12.     RBMtrain(h^(k-1),lr,W^k,b^k,c^k) //计算 Q(h^k|h^(k-1)),用于将来使用
13.   end while
14. end for
15. for k =1 to lengh do
16. 对于输出单元计算误差 δ_k:
17. δ_k=o_k(1-o_k)(t_k-o_k)
18. 对于每个隐藏单元 h 计算误差项 δ_h:
19. δ_h=o_h(1-o_h)Σ_{k∈outputs} w_k^h δ_k
20. 更新权值 w_j^i=w_j^i+Δw_j^i
21. end for
```

其中,RBMtrain 函数是对每一层 RBN 的训练,在上述训练过程中需注意一些参数的选择:如果学习速率过大,重建误差通常显著增加,权值可能过大;如果学习速率降低,而网络是正常学习,重建误差通常会下降,但是这未必是一个好现象,因为在某种程度上伴随着长期较低的学习速率,随机权重更新将会产生噪音,lr 的值选择为 1,momentum 的值通常初始化为 0.5,mini-batch 的选值通常在 10～100 之间,这里选择的是 10;至于隐藏层数,则根据具体使用的稀疏目标来确定,如果很小,则可以使用更多的隐藏结点;反之,可以选择更少的隐藏结点。

4 实验内容及结果

4.1 数据集

NASA 数据集被广泛地运用在缺陷预测技术的研究中，数据来源于 McCabe 和 Halstead 属性的源代码，本文采用的是 NASA 的 MDP 公开数据集，从中选取了 JM1，KC1，MC1，PC2 和 PC5 几个数据量较多的子数据集，如表 1 所列。

表 1 NASA MDP 数据集

	语言	代码行数/k	模块数	缺陷率
JM1	C	315	10878	19.0
KC1	C++	43	2107	15.0
MC1	C&C++	63	9466	0.7
PC2	C	26	5589	0.4
PC5	C++	164	17186	3.0

在典型的缺陷预测技术研究中，通常使用混淆矩阵相关的评估性能指标对预测结果进行评估，如准确率(Accuracy)、精准率(Precision)、查全率(Recall)、F-度量(F-measure)等。混淆矩阵包括两栏(见表 2)，包含正确的正例(TP)、错误的正例(FP)、错误的负例(FN)和正确的负例(TN)。TP 是预测含缺陷并且实际为缺陷的模块数，FP 是预测含缺陷并且实际为不含缺陷的模块数，FN 是预测不含缺陷并且实际为缺陷的模块数，TN 是预测不含缺陷并且实际不含缺陷的模块数。

准确率：$Accuracy=\frac{TN+TP}{TN+TP+FN+FP}$

精准率：$Precision=\frac{TP}{TP+FN}$

查全率：$Recall=\frac{TP}{TP+FP}$

F-度量：$F\text{-}measure=\frac{(\alpha^2+1)Precision*Recall}{\alpha^2(Precision+Recall)}$

在实验中，α 的取值为 1。

表 2 混淆矩阵

	预测值	
含缺陷模块	正确的正例(TP)	错误的正例(FP)
不含缺陷模块	错误的负例(FN)	正确的负例(TN)

4.2 实验结果及分析

首先将 MDP 数据集进行数据处理，对每个属性进行归一化；其次将残缺的属性补零；最后对数据集进行平衡化处理。采用三折交叉实验法，将每个数据集划分成 3 份，依次将每份作为测试集，剩下两份作为训练集，每组实验做 10 次，结果取 30 次实验的平均值。对神经网络软件缺陷预测模型(NN)进行实验，同样对数据进行归一化、属性补零等处理，采用三折交叉实验法，取 30 次实验的平均值作为最终结果。将 DBNSDPM 实验结果与 NN 实验结果进行对比，得出最终结论。

实验结果采取 Accuary，Recall，Precision 和 F-Measure 4 个度量指标进行评价，得到详细的软件缺陷预测结果，如表 3 及图 3—图 6 所示。

从图 3 中可以看出，使用 NN 对 JM1，KC1，MC1，PC2 和 PC5 这 5 个软件缺陷集的预测准确率在 0.53～0.87 之间，在 MC1 数据集上最高达到 0.87，在 PC2 上最高达到 0.53，平均值为 0.73。使用 DBNSDPM 对 JM1，KC1，MC，PC2 和 PC5 这 5 个软件缺陷集的预测准确率在 0.68～0.91 之间，其中在 MC1 数据集上最高达到 0.91，在 PC2 上最高达到 0.68，平均值为 0.82，比 NN 的平均预测准确度高 0.09，其中在 PC2 数据集上 DBNSDPM 比 NN 高 0.15。由此可见，使用 DBNSDPM 所得的准确率比使用 NN 的准确率更高。

表 3 实验结果统计

	Model	Accuracy	Recall	Precision	F-Measure
JM1	NN	0.74	0.40	0.32	0.36
	DBNSDPM	0.82	0.55	0.45	0.50
KC1	NN	0.81	0.38	0.42	0.48
	DBNSDPM	0.85	0.52	0.56	0.54
MC1	NN	0.87	0.33	1	0.50
	DBNSDPM	0.91	0.42	0.92	0.58
PC2	NN	0.53	0.58	0.52	0.55
	DBNSDPM	0.68	0.85	0.72	0.78
PC5	NN	0.71	0.68	0.23	0.34
	DBNSDPM	0.82	0.79	0.44	0.57

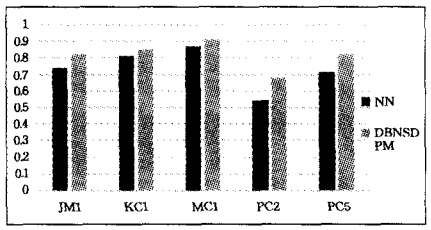

图 3 Accuracy 指标的统计结果

从图 4 中可以看出，使用 NN 对 JM1，KC1，MC，PC2 和 PC5 这 5 个软件缺陷集的预测召回率在 0.33～0.68 之间，在 PC5 数据集上最高达到 0.68，在 MC1 上最高达到 0.33，平均值为 0.47。使用 DBNSDPM 对 JM1，KC1，MC，PC2 和 PC5 这 5 个软件缺陷集的预测召回率在 0.42～0.85 之间，其中在 PC2 数据集上最高达到 0.85，在 MC1 上最高达到 0.42，平均值为 0.62，比 NN 的平均预测准确度高 0.15，其中在 PC2 数据集上 DBNSDPM 比 NN 高 0.27。由此可见，使用 DBNSDPM 所得的召回率比使用 NN 的召回率更高。

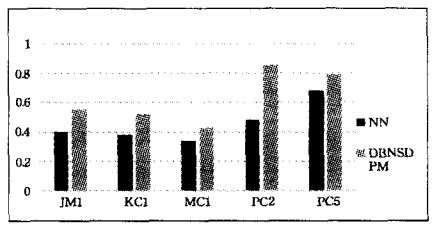

图 4 Recall 指标的统计结果

从图 5 中可以看出，使用精准率在 JM1，KC1，MC，PC2 和 PC5 这 5 个软件缺陷集上分别对 NN 和 DBNSDPM 进行测量时，NN 的精准率在 0.23～1 之间，在 MC1 数据集上最高达到 1，在 PC5 上最高达到 0.23，平均值为 0.50。使用 DBNSDPM 预测的精准率在 0.44～0.92 之间，其中在 MC1 数据集上最高达到 0.92，在 PC5 上最高达到 0.44，平均值为 0.62，比 NN 的平均预测精准率高 0.12，其中在 PC2 数据集上 DBNSDPM 比 NN 高0.20，在 PC5 数据集上 DBNSDPM 比神经网络高 0.21。由此可见，使用 DBNSDPM 所得的精准率比使用 NN 的精准率更高。

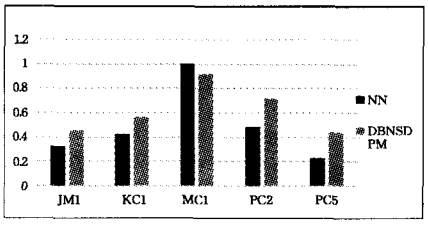

图 5　Precision 指标的统计结果

从图 6 中可以看出,使用 F 度量在 JM1,KC1,MC,PC2 和 PC5 这 5 个软件缺陷集上分别对 NN 和 DBNSDPM 进行测量时,NN 的 F 度量值在 0.34～0.55 之间,在 PC2 数据集上最高达到 0.55,在 PC5 上最高达到 0.34,平均值为 0.45。使用 DBNSDPM 预测的 F 度量值在 0.50～0.78 之间,其中在 PC2 数据集上达到最高 0.78,在 JM1 上最高达到 0.50,平均值为 0.59,比 NN 的平均预测准确度高 0.14,其中在 PC2 数据集上 DBNSDPM 比 NN 高 0.23,在 PC5 数据集上 DBNSDPM 比 NN 高 0.23。由此可见,使用 DBNSDPM 所得的下度量比使用 NN 的 F 度量值更高。

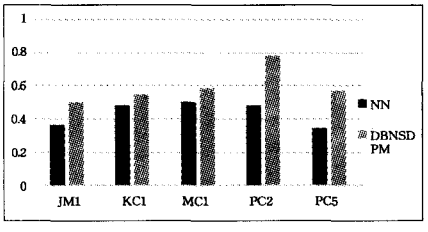

图 6　F-Measure 指标的统计结果

综上,在 JM1,KC1,MC,PC2 和 PC5 这 5 个软件缺陷集上分别对 NN 和 DBNSDPM 使用 Accuary,Recall,Precision 和 F-Measure 4 个度量指标进行测量时,DBNSDPM 的度量值比 NN 普遍高 10%以上,由此可见,与传统的神经网络相比,在软件缺陷预测技术研究领域引入深度学习中深度信念网构成的 DBNSDPM 能提高预测的准确性。

结束语　本文针对现有的软件缺陷预测模型不能深入挖掘数据特征的局限性,提出了利用深度学习中的深度信念网算法,对软件缺陷数据集的数据进行学习,挖掘数据特征,并构建软件缺陷预测模型。通过实际数据和对比实验结果表明,本文提出的 DBNSDPM 对软件缺陷具有较高的预测性和实用性。在后续的工作中,将进一步研究其他非神经网络分类算法与 DBNSDPM 在软件缺陷预测应用上的差异性,并且对 DBNSDPM 的各种参数调节方法以及结构进行改进,使得 DBNSDPM 在软件缺陷预测领域中能够取得更高的准确度。

参 考 文 献

[1] LIANG C C,ZHANG D Y,LIN H J. Integrated Research of Software Defect[J]. Computer Engineering,2006,32(19):88-90. (in Chinese)
梁成才,章代雨,林海静. 软件缺陷的综合研究[J]. 计算机工程,2006,32(19):88-90.

[2] NAGAPPAN N,BALL T,ZELLER A. Mining metrics to predict component failures[C]// Proceedings of the 28th International Conference on Software Engineering. ACM,2006:452-461.

[3] KIM S,ZIMMERMANN T,WHITEHEAD JR E J,et al. Predicting faults from cached history[C]// Proceedings of the 29th International Conference on Software Engineering. IEEE Computer Society,2007:489-498.

[4] ZIMMERMANN T,NAGAPPAN N. Predicting defects using network analysis on dependency graphs[C]// Proceedings of the 30th International Conference on Software Engineering. ACM,2008:531-540.

[5] 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.

[6] CHEN X,GU Q,LIU W S,et al. Research on Static software defect prediction method [J]. Journal of Software,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.

[7] ELISH K O,ELISH M O. Predicting defect-prone software modules using support vector machines[J]. Journal of Systems and Software,2008,81(5):649-660.

[8] JINDAL R,MALHOTRA R,JAIN A. Software defect prediction using neural networks[C]// International Conference on Reliability,INFOCOM Technologies and Optimization. 2014:1-6.

[9] GE H H,JIN C,YE J M. Software Defect Prediction Model Based on Particle Swarm Optimization and Na ve Bayes[J]. Computer Engineering,2011,37(12):36-37. (in Chinese)
葛贺贺,金聪,叶俊民. 基于 PSO 和朴素贝叶斯的软件缺陷预测模型[J]. 计算机工程,2011,37(12):36-37.

[10] 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.

[11] FISCHER A,IGEL C. An introduction to restricted Boltzmann machines[M]// Progress in Pattern Recognition,Image Analysis,Computer Vision,and Applications. Springer Berlin Heidelberg,2012:14-36

[12] HINTON G E,OSINDERO S,TEH Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation,2006,18(7):1527-1554

[13] WANG S,LIU T,TAN L. Automatically learning semantic features for defect prediction[C]// International Conference on Software Engineering. ACM,2016:297-308.

[14] YANG X,LO D,XIA X,et al. Deep Learning for Just-In-Time Defect Prediction[C]// 2015 IEEE International Conference on Software Quality,Reliability and Security (QRS). IEEE,2015:17-26.

[15] SCHÖLKOPF B,PLATT J,HOFMANN T. Greedy Layer-Wise Training of Deep Networks[C]// Conference on Advances in Neural Information Processing Systems. MIT Press,2007:153-160.