第51卷 第7期 西 安 交 通 大 学 学 报 Vol. 51 No. 7
2017年7月 JOURNAL OF XI'AN JIAOTONG UNIVERSITY Jul. 2017

DOI：10.7652/xjtuxb201707022

应用非线性加权的集成学习软件缺陷序列预测算法

贾晓琳，樊帅帅，罗雪，朱晓燕
（西安交通大学电子与信息工程学院学院，710049，西安）

摘要：针对当前软件缺陷序列预测算法准确度不高的问题，提出了基于非线性加权的集成学习软件缺陷序列预测算法(NLWEPrediction)。该算法在常见线性集成预测算法的基础上增加了非线性回归项，回归项代表了集成预测算法中基预测算法之间的相互关系，修正了线性集成预测的偏差，并通过梯度下降法求解了模型中的参数。实验表明：NLWEPrediction在14个软件缺陷数据集上的均方误差均小于250，并且平均绝对误差均小于13。通过与基预测算法、集成预测Bagging、Stacking算法和只考虑两个分类器关系的非线性加权集成学习算法进行对比，可以看出，NLWEPrediction预测算法的均方误差和平均绝对误差显著减小，预测精度显著提高，说明在线性集成预测算法基础上增加非线性回归项，能够有效提高集成学习算法的分类效果。

关键词：软件缺陷序列；预测算法；软件缺陷；集成学习
中图分类号：O121.8 **文献标志码**：A **文章编号**：0253-987X(2017)07-0156-06

Prediction Algorithm for Software Defect Series Based on Nonlinear Weighted Ensemble Learning

JIA Xiaolin, FAN Shuaishuai, LUO Xue, ZHU Xiaoyan
(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: Aiming at the problem of the basic prediction algorithm with relative low prediction accuracy, a novel and improved prediction algorithm NLWEPrediction is proposed based on nonlinear weighted and ensemble learning. It combines the advantages of linear ensemble learning and the relationship between the base predict algorithms, which corrects the prediction deviation and uses gradient descent method to calculate the model parameters. The experiments proved that the NLWEPrediction's mean squared error in datasets is lower than 250, and the mean absolute difference is lower than 13. The algorithm was compared with its four base prediction algorithms, other two ensemble prediction algorithms Bagging and Stacking and original NLWEPrediction for efficiency analysis. Experimental results showed that NLWEPrediction has obviously low mean square error and average absolute error. The prediction accuracy is improved. So, adding the nonlinear regression terms can improve the capability of ensemble classifier.

Keywords: software defect series; prediction algorithm; software defect; ensemble learning

收稿日期：2017-01-12。 **作者简介**：贾晓琳(1963—)，女，高工，硕士生导师。 **基金项目**：国家自然科学基金资助项目(61402355)；中央高校基本科研业务费专项基金资助项目(jj2014050)。
网络出版时间：2017-04-17 **网络出版地址**：http://www.cnki.net/kcms/detail/61.1069.T.20170417.1202.004.html

软件缺陷预测是改善软件开发质量、提高测试效率的重要途径[1]。近年来，软件缺陷序列预测研究得到国内外相关科研人员的重视，涌现出很多软件缺陷序列预测算法，主要分为基预测和集成学习预测算法，常用的基预测算法有 LR(linearregression)、MP(MultilayerPerceptron)、SMOreg、GP(GaussianProcesses)预测算法[2-5]，集成学习预测算法有 Bagging、Stacking 预测算法[6-7]。Adhikari 等在研究线性回归的基础上，增加了不同基预测算法之间的关系，且通过时间序列数据集进行验证[8]，该算法相对于线性集成预测算法，预测精度显著提高，但其只考虑到两个分类器之间的关系，并没有考虑到多个分类器之间的关联关系。本文在对预测算法进行改进的基础上，提出了基于非线性加权的集成学习软件缺陷序列预测算法(NLWEPrediction)。

1　集成学习时间序列预测算法

为了表述方便，用 $\boldsymbol{Y}=[y_1,y_2,\cdots,y_N]^{\mathrm{T}}$ 表示时间序列，$\hat{\boldsymbol{Y}}^{(i)}=[\hat{y}_1^{(i)},\hat{y}_2^{(i)},\cdots,\hat{y}_N^{(i)}]^{\mathrm{T}}$ 为从第 $i=1,2,3$ 预测算法得到的预测结果，$\boldsymbol{Y}$ 的集成预测值为 $\hat{\boldsymbol{Y}}^{(c)}=[\hat{y}_1^{(c)},\hat{y}_2^{(c)},\cdots,\hat{y}_N^{(c)}]^{\mathrm{T}}$，其中

$$\hat{y}_k^{(c)}=\omega_0+\omega_1\hat{y}_k^{(1)}+\omega_2\hat{y}_k^{(2)}+\omega_3\hat{y}_k^{(3)}+\theta_1\upsilon_k^{(1)}\upsilon_k^{(2)}+\theta_2\upsilon_k^{(1)}\upsilon_k^{(3)}+\theta_4\upsilon_k^{(2)}\upsilon_k^{(3)} \tag{1}$$

$$\upsilon_k^{(i)}=(y_k^{(i)}-\mu^{(i)})/\sigma^{(i)}$$

$$\forall\, i=1,2,3;\ k=1,2,3,\cdots,N$$

式中：$\mu^{(i)}$、$\sigma^{(i)}$ 为 $\hat{Y}^{(i)}$ 的平均值、标准差；ω_i 为基预测器线性分量的权重；θ_i 为两个基预测器线性分量的权重；$\upsilon_k^{(i)}$ 为预测值与预测均值的差和预测值标准差的比值。计算 $\hat{y}_k^{(c)}$ 考虑到了非线性的部分，如果有 n 种预测模型，那么就会有 $\binom{n}{2}$ 种非线性分量，因为只考虑预测算法两者之间的关系，该研究将 3 个预测算法进行集成，因此有 3 个非线性项。预测算法的向量为

$$\hat{\boldsymbol{y}}_k^{(c)}=\boldsymbol{F}\boldsymbol{\omega}+\boldsymbol{G}\boldsymbol{\theta} \tag{2}$$

式中：$\boldsymbol{\omega}=[\omega_0,\omega_1,\omega_2]^{\mathrm{T}}$；$\boldsymbol{\theta}=[\theta_1,\theta_2,\theta_3]^{\mathrm{T}}$；$\boldsymbol{1}=[1,1,\cdots,1]^{\mathrm{T}}$；$\boldsymbol{F}=[\boldsymbol{1}\mid\hat{\boldsymbol{Y}}^{(1)}\mid\hat{\boldsymbol{Y}}^{(2)}\mid\hat{\boldsymbol{Y}}^{(3)}]_{N\times4}$；$\boldsymbol{G}=\begin{bmatrix}\upsilon_1^{(1)}\upsilon_1^{(2)} & \upsilon_1^{(2)}\upsilon_1^{(3)} & \upsilon_1^{(1)}\upsilon_1^{(3)}\\ \vdots & \vdots & \vdots\\ \upsilon_N^{(1)}\upsilon_N^{(2)} & \upsilon_N^{(2)}\upsilon_N^{(3)} & \upsilon_N^{(1)}\upsilon_N^{(3)}\end{bmatrix}$。

式(2)中的权值可通过最小化时间序列预测结果的误差平方和求得，计算式为

$$\varepsilon_1=\sum_{k=1}^{N}(y_k-\hat{y}_k^{(c)})^2=$$

$$(\boldsymbol{Y}-\boldsymbol{F}\boldsymbol{\omega}-\boldsymbol{G}\boldsymbol{\theta})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{F}\boldsymbol{\omega}-\boldsymbol{G}\boldsymbol{\theta})=\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}-2\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{b}+\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\omega}+2\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{Z}\boldsymbol{\theta}-2\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{d}+\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{\theta} \tag{3}$$

式中：$\boldsymbol{V}=[\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}]_{4\times4}$；$\boldsymbol{b}=[\boldsymbol{F}^{\mathrm{T}}\boldsymbol{Y}]_{4\times1}$；$\boldsymbol{Z}=[\boldsymbol{F}^{\mathrm{T}}\boldsymbol{G}]_{4\times3}$；$\boldsymbol{d}=[\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Y}]_{3\times1}$；$\boldsymbol{U}=[\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}]_{3\times3}$。

利用梯度下降法[9]求 ε_1 的最小值，可获得预测方程的参数，即求下列方程组的解

$$\begin{cases}\dfrac{\partial}{\partial\boldsymbol{\omega}}\varepsilon_1=0\\ \dfrac{\partial}{\partial\boldsymbol{\theta}}\varepsilon_1=0\end{cases}\Leftrightarrow\begin{cases}\boldsymbol{V}\boldsymbol{\omega}+\boldsymbol{Z}\boldsymbol{\theta}=\boldsymbol{b}\\ \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{\omega}+\boldsymbol{U}\boldsymbol{\theta}=\boldsymbol{d}\end{cases} \tag{4}$$

计算式中未知数的解

$$\left.\begin{aligned}\boldsymbol{\theta}_{\mathrm{opt}}&=(\boldsymbol{U}-\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{V}^{-1}\boldsymbol{Z})^{-1}(\boldsymbol{d}-\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{V}^{-1}\boldsymbol{b})\\ \boldsymbol{\omega}_{\mathrm{opt}}&=\boldsymbol{V}^{-1}(\boldsymbol{b}-\boldsymbol{Z}\boldsymbol{\theta}_{\mathrm{opt}})\end{aligned}\right\} \tag{5}$$

得到预测方程中的参数的解，即可求得时间序列 $\boldsymbol{Y}$ 的预测序列。

2　集成学习软件缺陷序列预测算法

本文所提基于非线性加权的集成学习软件缺陷序列预测算法是在上述时间序列预测算法的基础上改进并实现的。本文算法将原预测算法应用到了软件缺陷序列预测领域，而且考虑了基预测算法之间的所有相互关系。

2.1　算法理论基础

参照式(1)，可得

$$\hat{y}_k^{(c)}=\omega_0+\omega_1\hat{y}_k^{(1)}+\omega_2\hat{y}_k^{(2)}+\omega_3\hat{y}_k^{(3)}+\omega_4\hat{y}_k^{(4)}+\theta_1\upsilon_k^{(1)}\upsilon_k^{(2)}+\theta_2\upsilon_k^{(1)}\upsilon_k^{(3)}+\theta_3\upsilon_k^{(1)}\upsilon_k^{(4)}+\theta_4\upsilon_k^{(2)}\upsilon_k^{(3)}+\theta_5\upsilon_k^{(2)}\upsilon_k^{(4)}+\theta_6\upsilon_k^{(3)}\upsilon_k^{(4)}+\alpha_1\upsilon_k^{(1)}\upsilon_k^{(2)}\upsilon_k^{(3)}+\alpha_2\upsilon_k^{(1)}\upsilon_k^{(2)}\upsilon_k^{(4)}+\alpha_3\upsilon_k^{(1)}\upsilon_k^{(3)}\upsilon_k^{(4)}+\alpha_4\upsilon_k^{(2)}\upsilon_k^{(3)}\upsilon_k^{(4)}+\beta\upsilon_k^{(1)}\upsilon_k^{(3)}\upsilon_k^{(2)}\upsilon_k^{(4)} \tag{6}$$

式中：α_i 为 3 个基预测器线性分量的权重；β_i 为 4 个基预测器线性分量的权重。NLWEPrediction 算法在时间序列预测算法的基础上添加了多个基分类器之间的关联关系。

NLWEPrediction 算法预测 $\hat{y}_k^{(c)}$ 考虑到非线性组成部分。假设有 n 种预测算法，那么就会有 $\binom{n}{2}+\binom{n}{3}+\cdots+\binom{n}{n-1}+\binom{n}{n}$ 个非线性分量。NLWEPrediction 对 4 种基预测算法进行集成，所以该预测算法的回归方程中非线性项有 11 项。为了简洁描述，可将 NLWEPrediction 算法表示为向量形式

$$\hat{\boldsymbol{Y}}=\boldsymbol{F}\boldsymbol{\omega}+\boldsymbol{G}\boldsymbol{\theta}+\boldsymbol{P}\boldsymbol{\alpha}+\boldsymbol{Q}\boldsymbol{\beta} \tag{7}$$

式中：$\boldsymbol{\omega}=[\omega_0,\omega_1,\omega_2,\omega_3,\omega_4]^{\mathrm{T}}$；$\boldsymbol{\theta}=[\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,$

$\theta_6]^{\mathrm{T}}$；$\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\alpha_3,\alpha_4]^{\mathrm{T}}$；$\boldsymbol{\beta}=[\beta]^{\mathrm{T}}$；$\mathbf{1}=[1,1,\cdots,1]^{\mathrm{T}}$；$\boldsymbol{F}=[\mathbf{1},\hat{Y}^{(1)},\hat{Y}^{(2)},\hat{Y}^{(3)},\hat{Y}^{(4)}]_{N\times 5}$；$\boldsymbol{G}=$

$$\begin{bmatrix} \upsilon_1^{(1)}\upsilon_1^{(2)} & \upsilon_1^{(1)}\upsilon_1^{(3)} & \upsilon_1^{(1)}\upsilon_1^{(4)} & \upsilon_1^{(2)}\upsilon_1^{(3)} & \upsilon_1^{(2)}\upsilon_1^{(4)} & \upsilon_1^{(3)}\upsilon_1^{(4)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \upsilon_N^{(1)}\upsilon_N^{(2)} & \upsilon_N^{(1)}\upsilon_N^{(3)} & \upsilon_N^{(1)}\upsilon_N^{(4)} & \upsilon_N^{(2)}\upsilon_N^{(3)} & \upsilon_N^{(2)}\upsilon_N^{(4)} & \upsilon_N^{(3)}\upsilon_N^{(4)} \end{bmatrix};$$

$\boldsymbol{P}=$

$$\begin{bmatrix} \upsilon_1^{(1)}\upsilon_1^{(2)}\upsilon_1^{(3)} & \upsilon_1^{(1)}\upsilon_1^{(2)}\upsilon_1^{(4)} & \upsilon_1^{(1)}\upsilon_1^{(3)}\upsilon_1^{(4)} & \upsilon_1^{(2)}\upsilon_1^{(3)}\upsilon_1^{(4)} \\ \vdots & \vdots & \vdots & \vdots \\ \upsilon_N^{(1)}\upsilon_N^{(2)}\upsilon_N^{(3)} & \upsilon_N^{(1)}\upsilon_N^{(2)}\upsilon_N^{(4)} & \upsilon_N^{(1)}\upsilon_N^{(3)}\upsilon_N^{(4)} & \upsilon_N^{(2)}\upsilon_N^{(3)}\upsilon_N^{(4)} \end{bmatrix};$$

$$\boldsymbol{Q}=\begin{bmatrix} \upsilon_N^{(1)}\upsilon_N^{(2)}\upsilon_N^{(3)}\upsilon_N^{(4)} \\ \vdots \\ \upsilon_N^{(1)}\upsilon_N^{(2)}\upsilon_N^{(3)}\upsilon_N^{(4)} \end{bmatrix}。$$

式(7)中的权值可通过最小化软件缺陷序列预测值的误差平方和求得，计算公式为

$$\varepsilon_2=\sum_{k=1}^{N}(y_k-\hat{y}_k^{(c)})^2=$$
$$(\boldsymbol{Y}-\boldsymbol{F\omega}-\boldsymbol{G\theta}-\boldsymbol{P\alpha}-\boldsymbol{Q\beta})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{F\omega}-\boldsymbol{G\theta}-\boldsymbol{P\alpha}-\boldsymbol{Q\beta})=\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}-2\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{F\omega}-2\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{G\theta}-$$
$$2\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{P\alpha}+\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F\omega}+2\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{G\theta}+2\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{P\alpha}+$$
$$2\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{Q\beta}+\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G\theta}+2\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{P\alpha}+2\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q\beta}+$$
$$\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P\alpha}+2\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{Q\beta}+\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q\beta}-2\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Q\beta} \quad (8)$$

利用梯度下降法[9]计算 ε_2 的最小值，求解式(7)中的 4 个参数向量，然后利用已知参数的集成回归方程对软件缺陷序列进行预测。

2.2 算法描述

本文选择 LR、MP、SMOreg 和 GP 预测算法作为基预测算法，算法伪代码包括模型构建和软件缺陷序列预测两个部分，伪代码实现如下。

输入：软件缺陷序列历史数据集 $Y=[y_1,y_2,\cdots,y_n]^{\mathrm{T}}$ 和基预测算法 $M_i(i=1,2,\cdots,m)$

输出：$\hat{Y}_{\text{testing}}=[\hat{y}_{j+1},\hat{y}_{j+2},\cdots,\hat{y}_N]^{\mathrm{T}}$

(1) $j\leftarrow Y$ 数据集大小$*(2/3)$；

(2) $Y_{\text{testing}}\leftarrow$将 Y 中的实例 0 到 j 划分到训练集；

(3) $Y_{\text{testing}}\leftarrow$将 Y 中的实例 j 到 Y 的实例数减去 j 划分到测试集；

(4) $\omega\leftarrow\phi,\theta\leftarrow\phi,k\leftarrow$基预测算法数；

(5) **For** $i=1$ to m **do**

在训练集 Y_{testing} 上训练 M_i，然后在测试集 Y_{testing} 上使用交叉验证计算预测结果，获取测试集 Y_{testing} 上的列向量 $\boldsymbol{F}$；

End For

(6) 根据式(7)计算其他向量 $\boldsymbol{G}$、$\boldsymbol{P}$、$\boldsymbol{Q}$ 的值；

(7) 使用优化算法计算权重 $\boldsymbol{\omega}_{\text{opt}}$、$\boldsymbol{\theta}_{\text{opt}}$、$\boldsymbol{\alpha}_{\text{opt}}$ 和 $\boldsymbol{\beta}_{\text{opt}}$；

(8) 使用 $\boldsymbol{\omega}_{\text{opt}}$、$\boldsymbol{\theta}_{\text{opt}}$、$\boldsymbol{\alpha}_{\text{opt}}$ 和 $\boldsymbol{\beta}_{\text{opt}}$，$\boldsymbol{F}$、$\boldsymbol{G}$、$\boldsymbol{P}$、$\boldsymbol{Q}$ 根据式(7)计算预测矩阵 $\hat{\boldsymbol{Y}}_{\text{testing}}$；

Return $\hat{\mathbf{Y}}_{\text{testing}}$

步骤(1)～(7)是模型构建，首先利用基预测算法在训练集上训练算法模型，并获取 4 种预测算法在测试集上的预测序列，并将预测序列加入到向量 $\boldsymbol{F}$ 中；然后根据式(7)计算不同基预测模型之间的相互关系向量，即计算向量 $\boldsymbol{G}$、$\boldsymbol{P}$、$\boldsymbol{Q}$，最后按照式(8)计算该算法的参数，至此建模过程结束。步骤 8 是对软件缺陷序列进行预测，预测过程就是在测试集上根据式(7)得到集成预测器的预测结果。

3 实验及分析

3.1 实验环境与评价指标

本文采用 Apache、C++ Standard Library、eclipse、Firefox、GNOME、Project ESME、Project HT TP Server、Project XML Beans 和 Tomcat 等项目，实验数据包含 Apache_V2.0.35.arff、C++ Standard Library_v4.1.3、eclipse-JDT-1.0、eclipse-JDT-3.0、eclipse-JDT-Core-2.0.1、eclipse-JDT-Core-3.7、eclipse-JDT-UI-1.0、eclipse-PDE-UI-3.1、Firefox-1.0 Branch、GNOME_V2.0、Project ESME_V1.1、Project JUDDI_V3.0.4、Project XML Beans_V2.2、Tomcat 3-3.1.1 Final。

算法在 Intel(R) Core(TM) i5-2400、java version 1.8.0_40、Matlab-R2015 实验环境下实现，预测过程中训练集与测试集比例为 2∶1，分别采用均方误差(MSE)和平均绝对误差(MAE)两种评价指标来评价软件缺陷预测精度。

3.2 实验结果及分析

3.2.1 算法预测精度对比 通过与 4 种基预测算法、集成学习 Bagging、Stacking 预测算法及改进前的时间序列预测算的均方误差和平均绝对误差进行比较，来验证 NLWEPrediction 预测算法的有效性，结果如表 1、2 所示。

由表 1 可知，NLWEPrediction 算法相比于其他 4 种基预测算法，在所有数据集上 MSE 显著减小，预测精度显著提高。同时，NLWEPrediction 算法的预测精度随着 4 种基预测算法精度的提高而提高，这是由于该预测算法考虑各个基预测算法之间相互关系的影响，所以预测结果比单独的预测器要

表 1　本文算法与基预测算法的 MSE 指标对比

数据集	LR 算法	MP 算法	SMOreg 算法	GP 算法	本文算法
Apache_V2. 0. 35	902. 93	951. 00	2 406. 60	3 959. 40	16. 57
C++StdLib_v4. 1. 3	1 428. 25	82. 00	1 194. 38	268. 50	5. 12
eclipse-JDT-1. 0	186. 96	1 358. 36	9 034. 32	26 706. 72	15. 56
eclipse-JDT-3. 0	29 499. 71	128 107. 71	13 655. 32	74 232. 13	243. 81
eclipseJDTCor2. 0. 1	22. 80	91. 05	576. 19	1 302. 09	9. 16
eclipseJDTCore3. 7	85 799. 78	19 129. 80	2 037. 17	46 646. 14	0. 48
eclipse-JDT-UI-1. 0	546. 33	1 094. 36	2 403. 70	9 114. 78	3. 20
eclipse-PDE-UI-3. 1	691 475. 70	24 723. 64	55 885. 99	20 635. 60	88. 22
Firefox-1. 0 Branch	162. 16	25. 56	30. 50	107. 58	2. 31
GNOME_V2. 0	239. 88	433. 75	3 432. 50	1873. 50	5. 87
ESME_V1. 1	2 370. 50	322. 63	1 619. 75	574. 88	16. 53
JUDDI_V3. 0. 4	21. 09	47. 82	850. 45	445. 82	5. 87
XMLBeans_V2. 2	10. 29	4. 24	112. 29	112. 47	0. 12
Tomcat 3-3. 1. 1 Final	1. 20	3. 40	0. 95	0. 80	0. 03

表 2　本文算法与基预测算法的 MAE 指标对比

数据集	LR 算法	MP 算法	SMOreg 算法	GP 算法	本文算法
Apache_V2. 0. 35	28. 00	29. 80	45. 67	57. 40	9. 17
C++StdLib_v4. 1. 3	35. 50	8. 75	32. 375	14. 75	1. 91
eclipse-JDT-1. 0	11. 87	34. 48	93. 07	161. 48	5. 09
eclipse-JDT-3. 0	100. 94	203. 90	71. 00	158. 53	9. 09
eclipseJDTCor2. 0. 1	4. 37	8. 00	23. 57	35. 41	7. 13
eclipseJDTCore3. 7	232. 28	130. 44	38. 55	193. 97	0. 47
eclipse-JDT-UI-1. 0	21. 66	31. 57	48. 16	94. 20	1. 25
eclipse-PDE-UI-3. 1	817. 76	144. 37	230. 68	133. 94	12. 56
Firefox-1. 0 Branch	18. 97	10. 28	3. 97	9. 38	1. 16
GNOME_V2. 0	13. 37	20. 25	53. 00	38. 50	5. 87
ESME_V1. 1	45. 00	17. 88	37. 25	21. 63	1. 11
JUDDI_V3. 0. 4	21. 09	47. 82	850. 45	445. 82	7. 04
XMLBeans_V2. 2	3. 12	1. 88	9. 35	9. 05	0. 26
Tomcat3-3. 1. 1 Final	1. 00	1. 70	0. 85	0. 80	0. 07

好得多。由于不同基预测算法之间的相互关系修正了线性集成预测的预测偏差，因此软件缺陷序列的预测精度显著提高。由表 2 可知，NLWEPrediction 的平均绝对误差 MAE 相比于基预测算法均显著减小，故增加了基预测算法之间的相互关系的集成学习，预测算法能显著提高软件缺陷序列的预测性能。

表 3 将 NLWEPrediction 预测算法与集成预测算法 Bagging 和 Stacking 进行比较，与其他 4 种预测算法相比，Bagging、Stacking 集成学习预测算法预测精度都显著提高。这是因为集成学习预测算法是将弱的预测算法以不同的集成方式结合，得到强的预测算法。由表 3 可知，在大多数数据集上，非线性加权集成预测算法比 Bagging、Stacking 集成预测算法的预测结果更好。表 4 将 NLWEPrediction 与改进前的预测算法对比，由表 4 可知改进后的算法相比于改进前的算法，在所有测试集上 MSE 指标和 MAE 指标都显著减小，其主要原因是改进前的非线性加权集成预测算法只考虑了基预测算法两者之间的关系，而本文提出的 NLWEPrediction 算法全面考虑了基预测算法之间所有关系。本文选择

表 3 本文算法与其他两种集成预测算法的预测精度对比

数据集	Bagging 算法		Stacking 算法		本文算法	
	均方误差	平均绝对误差	均方误差	平均绝对误差	均方误差	平均绝对误差
Apache_V2. 0. 35	32. 00	5. 60	876. 80	29. 60	16. 57	9. 17
C++StdLibv4. 1. 3	280. 88	15. 88	507. 38	21. 88	5. 12	1. 91
eclipse-JDT-1. 0	43. 23	6. 44	341. 67	18. 44	15. 56	5. 09
eclipse-JDT-3. 0	129 763. 81	207. 74	131 019. 26	210. 74	243. 81	9. 09
eclipseJDTCor2. 0. 1	1. 00	1. 00	4. 00	2. 00	9. 16	7. 13
eclipse-JDT-Cor3. 7	6 638. 02	80. 77	319 080. 00	564. 77	0. 48	0. 40
eclipse-JDT-UI-1. 0	1. 00	1. 00	100. 00	10. 00	3. 20	1. 25
eclipse-PDE-UI-3. 1	9. 00	3. 00	42 436. 00	206. 00	88. 22	12. 56
Firefox-1. 0 Branch	1. 00	1. 00	1. 00	1. 00	2. 31	1. 16
GNOME_V2. 0	126. 13	10. 88	1 678. 63	40. 88	5. 87	5. 87
ESME_V1. 1	135. 25	11. 25	418. 75	20. 25	16. 53	1. 11
JUDDI_V3. 0. 4	226. 45	13. 18	738. 18	26. 18	5. 87	7. 04
XML Beans_V2. 2	85. 18	8. 71	288. 47	16. 71	0. 12	0. 26
Tomcat3-3. 1. 1 Final	8. 15	2. 75	8. 15	2. 75	0. 03	0. 07

表 4 本文算法与改进前算法预测结果对比

数据集	改进前算法		本文算法	
	均方误差	平均绝对误差	均方误差	平均绝对误差
Apache_V2. 0. 35. arff	23. 89	11. 23	16. 57	9. 17
C++Standard Library_v4. 1. 3	13. 11	2. 63	5. 12	1. 91
eclipse-JDT-1. 0	52. 47	27. 59	15. 56	5. 09
eclipse-JDT-3. 0	502. 16	58. 87	243. 81	9. 09
eclipse-JDT-Core-2. 0. 1	19. 86	16. 35	9. 16	7. 13
eclipse-JDT-Core-3. 7	1. 56	0. 94	0. 48	0. 40
eclipse-JDT-UI-1. 0	14. 31	6. 75	3. 20	1. 25
eclipse-PDE-UI-3. 1	101. 29	19. 28	88. 22	12. 56
Firefox-1. 0 Branch	9. 56	5. 71	2. 31	1. 16
GNOME_V2. 0	17. 29	4. 97	5. 87	5. 87
Project ESME_V1. 1	55. 37	12. 57	16. 53	1. 11
Project JUDDI_V3. 0. 4	49. 52	10. 29	5. 87	7. 04
Project XML Beans_V2. 2	1. 35	0. 79	0. 12	0. 26
Tomcat 3-3. 1. 1 Final	0. 21	0. 14	0. 03	0. 07

4 种基预测算法，增加了 11 个非线性分量，因此能最大程度修正线性集成预测算法的偏差，发挥了线性集成的优势和完全预测修正的功能。

在实验过程中，由于考虑多个分类器之间的关联关系，计算量相应增加，本文算法在 Intel(R)Core(TM)i5-2400 CPU@3. 10GHz 系统上的运行时间比改进之前慢 20%。但是，使用集成学习的目的就是牺牲时间换取更高的分类效果，因此本文算法牺牲一定的时间来换取更高的分类效果仍然有一定意义。

3. 2. 2 算法精度影响因素 NLWEPrediction 算法是对 4 种基预测算法的非线性集成，故 NLWEPrediction 算法的预测精度随着基预测算法预测精度的变化而变化，对 NLWEPrediction 算法的影响如图 1 所示。由图 1 可知，在软件缺陷序列预测中，NLWEPrediction 算法预测精度随着 4 种基预测算法预测精度的提高而提高，这体现了集成学习思想的本质，即将弱分类器集成为强分类器。如

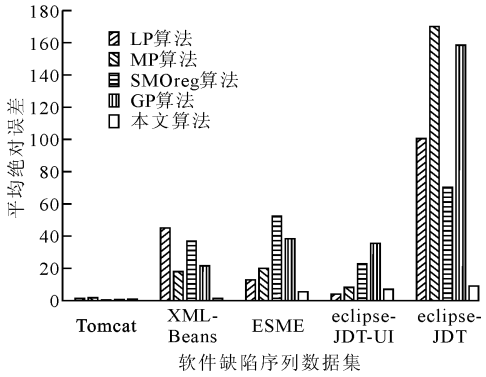

图 1　基预测算法对 NLWEPrediction 算法的影响

果基预测算法在指定数据集上预测精度高，那么 NLWEPrediction 算法预测精度会更高；反之，如果基预测算法在指定数据集上预测精度一般，则 NLWEPrediction 算法预测精度会有所提高，但是相比于强的预测算法，预测性能一般。因此，基预测算法的选择对 NLWEPrediction 预测算法预测精度的提高至关重要。

图 2 给出了训练集实例数对 NLWEPrediction 性能的影响。由图 2 可知，随着软件缺陷序列训练集的增加，算法的性能逐渐提高，当训练集的大小是软件缺陷序列数据集大小的 2/3 时，预测算法精度达到最高，当训练集的大小再增加时，预测算法性能会逐渐降低。经多次实验，确定训练集与测试集的比例为 2∶1时，NLWEPrediction 算法预测结果达到最优。

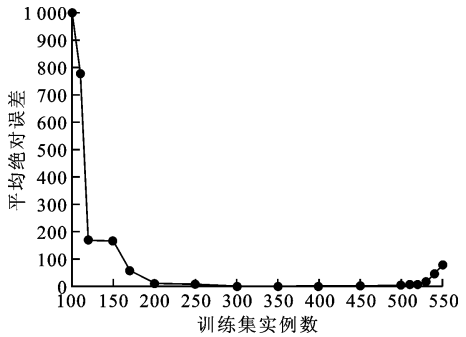

图 2　训练集实例数对 NLWEPrediction 性能的影响

4　结　论

基于非线性加权的集成学习时间序列预测算法基础上，本文提出了基于非线性加权集成学习软件缺陷预测算法。该算法在常见的线性集成预测算法基础上增加了非线性回归项，将基学习器之间的关联关系引入到集成学习中，能够有效提高集成学习器的能力。传统的线性集成没有考虑到基分类器之间存在相关性，或者仅考虑两个基分类器之间的关系，无法全面体现各个分类器在分类结果上的关联关系，NLWEPrediction 算法能够将分类器之间的相关关系体现出来，分类效果较之前的两种算法要好。因此，本文提出的 NLWEPrediction 算法虽然运行时间比其余集成学习算法长，但其分类效果更好。

参考文献：

[1] HE Liang, SONG Qinbao, SHEN Junyi. Boosting-based *k*-NN learning for software defect prediction [J]. Pattern Recognition and Artificial Intelligence, 2012, 25(5): 792-802.

[2] RZTEBLATT Ä D. Linear regression analysis: part 14 of a series on evaluation of scientific publications [J]. Deutsches Ärzteblatt Archiv Medizin Übersichtsarbeit, 2010, 107(44): 776-782.

[3] UMAR S N. Software testing defect prediction model: a practical approach [J]. International Journal of Research in Engineering and Technology, 2013, 2(5): 741-745.

[4] YANG J F, ZHAI Y J, XU D P, et al. SMO algorithm applied in time series model building and forecast [C]// International Conference on Machine Learning and Cybernetics. Piscataway, NJ, USA: IEEE, 2007: 2395-2400.

[5] ROBERTS S, OSBORNE M, EBDEN M, et al. Gaussian processes for time-series modelling [J]. Philosophical Transactions of the Royal Society A, 2013, 371(1984): 20110550.

[6] AUDIBERT J Y. Bagging predictors [J]. Annales De Linstitut Henri Poincare Probability & Statistics, 2004, 40(6): 685-736.

[7] SILL J, TAKACS G, MACKEY L, et al. Feature-weighted linear stacking [EB/OL]. [2016-12-12]. https://arxiv.org/pdf/0911.0460.pdf.

[8] ADHIKARI R, AGRAWAL R K. A novel weighted ensemble technique for time series forecasting [J]. Lecture Notes in Computer Science, 2012, 7301: 38-49.

[9] NESTEROV Y. Introductory lectures on convex optimization [M]. Berlin, Germany: Springer, 2014: 16-38.

（编辑　赵炜）