LETTER

Cost-Sensitive and Sparse Ladder Network for Software Defect Prediction

Jing SUN†, Yi-mu JI†,†, Shangdong LIU†, Nonmembers, and Fei WU††(b), Member

SUMMARY Software defect prediction (SDP) plays a vital role in allocating testing resources reasonably and ensuring software quality. When there are not enough labeled historical modules, considerable semi-supervised SDP methods have been proposed, and these methods utilize limited labeled modules and abundant unlabeled modules simultaneously. Nevertheless, most of them make use of traditional features rather than the powerful deep feature representations. Besides, the cost of the misclassification of the defective modules is higher than that of defect-free ones, and the number of the defective modules for training is small. Taking the above issues into account, we propose a cost-sensitive and sparse ladder network (CSLN) for SDP. We firstly introduce the semi-supervised ladder network to extract the deep feature representations. Besides, we introduce the cost-sensitive learning to set different misclassification costs for defective-prone and defect-free-prone instances to alleviate the class imbalance problem. A sparse constraint is added on the hidden nodes in ladder network when the number of hidden nodes is large, which enables the model to find robust structures of the data. Extensive experiments on the AEEEM dataset show that the CSLN outperforms several state-of-the-art semi-supervised SDP methods.

key words: semi-supervised learning, software defect prediction, ladder network, cost-sensitive learning, sparse auto-encoder

1. Introduction

Software testing plays an essential role in ensuring software quality. Nevertheless, with the size and complexity of software increases, the labor cost of testing increases dramatically. Thus, software defect prediction (SDP) methods based on machine learning were proposed, which can automatically predict whether the software module is defective or not.

Traditional SDP methods construct a prediction model based on the labeled historical modules, but labeling data demands experts’ knowledge and is highly time-consuming. To solve this problem, researchers usually adopt the semi-supervised learning manner to build a prediction model. For example, Abaei et al. [1] proposed a semi-supervised hybrid self-organizing map for SDP method (HySOM). Zhang et al. [2] proposed nonnegative sparse graph based label propagation approach for SDP (NSGLP), Wu et al. [3] used limited labeled and abundant unlabeled modules with a semi-supervised cost-sensitive kernel dictionary learning model (CKSDL) to predict the defect.

Most of semi-supervised SDP methods use traditional features (hand-craft feature). As we know, the feature representations learnt from deep learning have powerful discriminant ability, besides semi-supervised deep learning algorithms have been proposed recently. For example, Pseudo-Label [4] uses a deep learning network as the classifier and acquires the pseudo label, and then, it trains the network parameters with the pseudo-labeled dataset. Although it attempts to make use of unlabeled data in addition to a small amount of labeled data, the classification result relies on the prediction accuracy of predicted labels. The Mean Teacher [5] averages model weight instead of label prediction to improve test accuracy and enables training with fewer labels than Pseudo-Label. However, the supervised learning and unsupervised learning parts are independent of each other, and their advantages are not thoroughly combined. Ladder Network [6] consists of stacked denoising auto-encoder with a reconstruction loss as the unsupervised term and classification network with a cross-entropy loss as the supervised term. Besides, there are skip connections between each layer of encoders and decoders in the auto-encoder, which make the unsupervised term focus on relevant details found by supervised term. As a result, the model can achieve more effective feature to the task.

Ladder network has already been applied in many fields, such as image recognition [6] and emotion recognition [7], and has achieved competitive performance. As we know, SDP is a binary classification problem, ladder network may be applied in SDP. However, when we applied it to SDP, we found that superior prediction performance can be achieved in most cases, but a small number of results are still not ideal when we do a large number of experiments in the same project. Thus, it is urgent to improve the robustness of the extracted deep feature representations of Ladder Network. Besides, different misclassifications will occur different costs. When the non-defective module is misclassified as defective, the cost denotes the waste of time and resources to review the module; when defective module is misclassified as non-defective, the cost means the missed opportunity to correct a defect that may appear in the system testing or even after software delivery, which will occur huge economy loss [8]. What more, the defective modules are insufficient in most datasets. As a result, how to obtain effective and robust deep feature representations is a significant chal-

Manuscript received November 6, 2019.
Manuscript revised January 8, 2020.
Manuscript publicized January 29, 2020.
†The authors are with College of Compute, Nanjing University of Posts and Telecommunications (NJUPT), Nanjing 210003, China.
††The author is with the College of Automation, NJUPT, Nanjing 210003, China.
a) E-mail: jiym@njupt.edu.cn (Corresponding author)
b) E-mail: wufei_8888@126.com (Corresponding author)
DOI: 10.1587/transinf.2019EDL8198
lengte.

In this paper, based on the ladder network [6], we propose a novel SDP method named cost-sensitive and sparse ladder network (CSLN). We design sparse constraint into the unsupervised term, and impose the cost-sensitive constraint on the supervised term. Then, we utilize the deep feature representations extracted by the improved ladder network to predict whether the software module is defective or not. Compared with the previous methods, the main contributions are as follows:

1. CSLN designs the improved version of the effective Ladder Network model to extract the deep feature representations of the software modules. And the CSLN makes the lower levels (first two levels) of network to help the high levels (the top level) of network to extract the features that are relevant to the task and also contain detailed information, which is effective for the prediction.

2. To make the deep feature representations learned by the original network be more robust, CSLN imposes sparsity constraint into the unsupervised term of the ladder network, such that model can discover the stable structure and abundant robust features of data. In addition, the CSLN defines different misclassification costs for the fault-prone and non-fault-prone modules in the supervised cross-entropy loss of the ladder network to deal with the class imbalance problem. Extensive comparison experiments conducted on the public AEEEM dataset have demonstrated the superiority of the proposed CSLN.

2. Proposed Approach

2.1 Semi-Supervised Learning with Ladder Network

We give an overview of the ladder network in this section. Assuming that there are N labeled modules \( \{x_1, x_2, \ldots, x_N\} \), the labels of them are defined as \( \{y^*_1, y^*_2, \ldots, y^*_N\} \), and unlabeled modules are \( \{x_{N+1}, x_{N+2}, \ldots, x_{N+M}\} \), where \( M \gg N \). As shown in Fig. 1, there is a deep denoising auto-encoder in the ladder network. \( x_n \) is the \( n \) th input module, and \( N \) is the Gaussian noise, \( y_n \) and \( \hat{y}_n \) are the outputs of the clean encoder and noisy encoder respectively. The \( z^{(l)}_n \) is the hidden representation at layer \( l \) of the module \( n \) in clean encoder, and \( z^{(l)}_n \) and \( \hat{z}^{(l)}_n \) are the hidden representations of the noisy encoder and decoder version respectively. The noise \( N \) is injected into all the hidden layers in the noisy encoder. The output of the noisy encoder is put into the decoder which can map it into a reconstructed representation of the input sample. Besides, there is a skip connection between every layer of noisy encoder and decoder, which can make the information flow freely between encoder and decoder, as a result, there are two inputs at each layer of the decoder, and the one is from the layer above, the other one is from the corresponding layer in the clean encoder. At the same time, the given input is encoded by a clean encoder, which shares parameters with noisy encoder. The detailed information about ladder network can be found in [6].

The objective function of the ladder network is the sum of the supervised cross-entropy loss on the top of the encoder and the unsupervised denoising square error loss of the decoder, it can be defined as follows

\[
J = -\sum_{n=1}^{N} \log P(y_n = y^*_n | x_n) + \sum_{n=N+1}^{M} \sum_{l=1}^{L} \lambda_l \|z^{(l)}_n - \hat{z}^{(l)}_n\|^2
\]

(1)

where the first term of the loss function measures the difference between the noisy output \( y_n \) and the actual label \( y^*_n \). The second term of the loss function measures the difference between \( z^{(l)}_n \) and \( \hat{z}^{(l)}_n \) of layer \( l \). \( \lambda_l \) is the denoising multipliers, and we tune it in the range \( \{0.1, 0.2, 0.3, \cdots, 10\} \).

Although the ladder network has been used in many fields, it is the first time to be applied in SDP. We found that the deep feature representations extracted by the original ladder network model are not efficient and robust enough for achieving stable and desirable prediction results. At the same time, we consider the misclassification cost issue due to that the misclassification of defective modules results in higher-risk cost than that of defect-free ones. Thus, we provide an improved version of the Ladder Network, which will be described in the following content.

2.2 Cost-Sensitive and Sparse Ladder Network (CSLN) for SDP

As we know, it is hard to misclassify a minority (defective) class into a majority (non-defective) class, as the research [10] indicates the learning from imbalanced data sets and learning when costs are unequal can be handled in a similar manner. Besides, cost-sensitive learning is a good solution to the class imbalanced problem by incorporating different penalties into the classification progress. In our paper, we add the penalty factor cost\((i, j)\), which can increase the punishment when a defective module is predicted as a defect-free modules and vice versa.

We construct the cost matrix of the ladder network, as shown in Table 1, where cost\((i, j)\) \((i, j \in 0, 1, \cdots, c)\) is the cost of classifying modules in class \( i \) as class \( j \). Due to
there are two classes in SDP, \( c \) is set as 1. The diagonal elements in the cost matrix are zero because correct classification does not incur any costs. The goal of cost-sensitive learning is to generate a prediction model with minimum misclassification cost by considering the cost factors.

Similar to the work [9], we use the function \( f(i) \) to describe the importance of the \( i \)-class, and it is defined as follows:

\[
f(i) = \sum_{j=0}^{c} \text{cost}(i,j) \quad i = 0, 1, \ldots, c
\]  

(2)

By adding the cost-sensitive constraint into the first term of the objective function (4), we present the function of the supervised term (in Cross-Entropy loss formulation) of CSLN:

\[
\text{loss}_s = -\frac{1}{N} \sum_{n=1}^{N} f(y_n) \log P(\hat{y}_n = y_n | x_n)
\]  

(3)

where \( \hat{y}_n \) is the output label of \( x_n \) in the model, \( N \) is the number of the labeled samples. It is important to note that we use the noisy output \( \hat{y}_n \) in training process, and using the clean output \( y_n \) to test the performance of model.

Inspired by [11], reinforcing the sparsity of weights in the deep neural network is useful for extracting valid structure and stable feature of the data. Besides, considering the unstable performance of the original ladder network applied in the SDP, it is necessary to impose sparse constraint on the ladder network to enhance the robustness of the feature. We design the sparse constraint into the loss function, so that the value of each component in the output vector of the encoder is as close to 0 as possible, and the sparse representations are obtained.

Assuming that the output of the activation function of the \( j \)-th hidden unit is \( h_j(x) \), \( m \) is the neuron number of the previous layer and \( x_i \) is the \( i \)-th input neuron. And then the average activation of the \( j \)-th neuron is:

\[
\hat{\rho}_j = 1/m \sum_{i=1}^{m} h_j(x_i)
\]  

(4)

Given that the sparse constraint coefficient is \( \rho \). The value of this coefficient is determined empirically and is usually defined as a value less than 0.05. Thus, the loss function of the unsupervised term after imposing sparse constraint is:

\[
\text{loss}_u = \sum_{n=N+1}^{M} \sum_{i=1}^{L} \left( \lambda_i \left\| x_n^{(l)} - x^*_n^{(l)} \right\| + \beta \sum_{j=1}^{s} KL(\rho || \hat{\rho}_j) \right)
\]  

\[
\sum_{j=1}^{s} KL(\rho || \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{(1-\rho)}{(1-\hat{\rho}_j)}
\]  

(6)

where \( s \) is the number of the hidden units. Besides, \( KL(\rho || \hat{\rho}_j) \) is called KL-divergence, which measures the degree of difference in probability distribution between \( \rho \) and \( \hat{\rho}_j \). \( \beta \) controls the weight of the sparsity penalty term. Following in [12], we set \( \beta \) as 3.

Finally, the total objective function can be given as:

\[
\text{Loss} = \text{loss}_s + \text{loss}_u.
\]  

(7)

The function can be optimized by gradient descent method.

3. Experiments

3.1 Experimental Design

The proposed CSLN in this paper is compared with the existing classical semi-supervised SDP methods based on machine learning, such as NSGLP [2], and CKSDL [3] and we also compare it with a deep learning-based method HySOM [1]. The comparison experiment settings are as the same as the setting of CKSDL. Besides, we also select the original ladder network [6], which is called LN in this paper, for comparison. And we perform LN with sparse constraint which we call it as SLN, besides, we perform LN with cost-sensitive factor which we call it as CLN. We compare the complete CSLN with these two incomplete versions.

For a given project, we randomly select a certain percentage (e.g., 80%) to train the prediction model and using the remaining modules to test the model. For the training dataset, we change the percentage of labeled modules from 10% to 30%, and we take the case that 20% modules are used as labeled modules as an example. To avoid the randomness caused by the random division of training and testing, we conduct 20 experiments to report the average results. We set the cost-sensitive factors parameters as \( \text{cost}(0, 1) : \text{cost}(1, 0) = 1 : (v/b) \), where \( v \) is the number of the defective modules, and the \( b \) is the number of the defect-free modules.

3.2 Datasets

In this experiment, we make use of five projects JDT, PDE, ML, EQ, LC in AEEEM dataset [13]. The brief properties of AEEEM is shown in Table 2. Besides, we removed the repeated items with the same metric label in original dataset in our experiment. We use the harmonic mean of precision and recall as the AEEEM project for evaluation.

| Project | Number of metrics | Number of total modules | Number of defective modules | Percentage of defective modules |
|---------|-------------------|-------------------------|----------------------------|--------------------------------|
| EQ      | 61                | 324                     | 129                        | 39.81%                         |
| JDT     | 61                | 997                     | 206                        | 20.66%                         |
| LC      | 61                | 691                     | 64                         | 9.26%                          |
| ML      | 61                | 1862                    | 245                        | 13.16%                         |
| PDE     | 61                | 1497                    | 209                        | 13.85%                         |

Table 1: Cost matrix of the CSLN

|                | Predict as defective | Predict as defect-free |
|----------------|----------------------|------------------------|
| Defective      | cost (0,0)           | cost (0,1)             |
| Defect-free    | cost (1,0)           | cost (1,1)             |
3.3 Experimental Results

As shown in Fig. 2, the average value of F-measure of compared methods improves with the increasing of the percentage of the labeled modules. The CSLN achieves the best performance among all methods. When the percentage of the labeled modules is set as 20% in Table 3, CSLN improves the prediction accuracies at least by 0.20 (± 0.65–0.45) than the existing semi-supervised SDP methods, and the CSLN also improves the performance of the LN.

The CSLN uses well-designed deep learning semi-supervised method to extract deep feature representations, which has stronger discrimination ability than the traditional handcraft feature. Different from the compared method, CSLN uses the labeled data and the unlabeled data directly rather than using the pseudo-labeled data like [4]. Besides, the result also demonstrates that the sparsity constraint and the cost-sensitive factor improve the accuracy of the prediction.

4. Conclusions

In this paper, we propose the CSLN model to address the challenging semi-supervised SDP task. The CSLN improves ladder network with cost-sensitive learning and sparse constraint to extract effective and robust deep feature representations. Extensive experiments show that the CSLN can achieve a promising result and outperform the related semi-supervised SDP methods.

Acknowledgments

This work was supported by the National Key Re & D Program of China (2017YFB1401302, 2017YFB0202200), Outstanding Youth of Jiangsu Natural Science Foundation (BK20170100), Key R&D Program of Jiangsu (BE2017166), National Natural Science Foundation of China (61702280), Natural Science Foundation of Jiangsu Province (BK20170900), CCF-Tencent Open Fund WeBank Special Funding.

References

[1] G. Abaei, A. Selamat, and H. Fujita, “An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction,” Knowledge-Based Systems, vol.74, no.2, pp.28–39, 2015.

[2] Z.-W. Zhang, X.-Y. Jing, and T.-J. Wang, “Label propagation based semi-supervised learning for software defect prediction,” Automated Software Engineering, vol.24, no.1, pp.47–69, 2017.

[3] F. Wu, X.-Y. Jing, Y. Sun, J. Sun, L. Huang, F. Cui, and Y. Sun, “Cross-project and within-project semi-supervised software defect prediction: A unified approach,” IEEE Trans. Rel., vol.67, no.2, pp.581–597, 2018.

[4] D.-H. Lee, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” Workshop on Challenges in Representation Learning, pp.2–3, 2013.

[5] A. Tarvainen and H. Valpola, “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results,” Advances in Neural Information Processing Systems, pp.1195–1204, 2017.

[6] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, “Semi-supervised learning with ladder network,” Advances in Neural Information Processing Systems, pp.3546–3554, 2015.

[7] J. Deng, X. Xu, and Z. Zhang, “Semi-supervised autoencoders for speech emotion recognition,” IEEE/ACM Trans. Audio, Speech, Language Process., pp.31–43, 2017.

[8] T.M. Khoshgattaar, E. Geleyn, L. Nguyen, and L. Bullard, “Cost-sensitive boosting in software quality modeling,” IEEE International Symposium on High Assurance Systems Engineering, pp.51–60, 2002.

[9] M. Liu, L. Miao, and D. Zhang, “Two-stage cost-sensitive learning for software defect prediction,” IEEE Trans. Rel., vol.63, no.2, pp.676–686, 2014.

[10] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” IEEE Trans. Knowl. Data Eng., vol.18, no.1, pp.63–77, 2006.

[11] A. Ng, “Sparse autoencoder,” CS294A Lecture notes, vol.72, pp.1–19, 2001.

[12] L. Meng, S. Ding, and Y. Xue, “Research on denoising sparse autoencoder,” International Journal of Machine Learning and Cybernetics, vol.8, no.5, pp.1719–1729, 2017.

[13] M. D’Ambros, M. Lanza, and R. Robbes, “An extensive comparison of bug prediction approaches,” IEEE Working Conference on Mining Software Repositories, pp.31–41, 2010.