APPLYING NEURAL NETWORKS ON CREDIT SCORING WITH SMALL DATASET AND HAND-CRAFTED FEATURES

[1]CHIEN-HSUN HUANG, [2]BING-MIN WANG, [3]DARBY TIEN-HAO CHANG

Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan.
E-mail: [1]goho302jo03@gmail.com, [2]e24011069@gmail.com, [3]darby@mail.ncku.edu.tw

Abstract - Credit scoring is importantin financial industry because of its economic importance. A recent study shows that ensemble methods that combine multiple base learnersoutperform individual learners. However, the pipeline to harmony heterogeneous base learners is complex.This work aims to apply neural network techniques to develop anindividual credit scoring learner. The resultant model achieves comparable performance to heterogeneous ensemble methods with simpler construction process and shorter training time. In addition to the model, a contribution of this work is to demonstrate the potential as well as provide a guideline to applying neural networks on small data with hand-crafted features.

Keywords - Credit risk, credit scoring, machine learning, neural network

I. INTRODUCTION

Credit risk assessment,or credit scoring, has attracted significant attention in financial industry over the last decade because of its economic importance(Crook, Edelman, & Thomas, 2007). In June 2018, the volumes of industrial and consumer loans held by commercial banks in the United States was 2209.6 and 1,472.8billions of dollars, respectively(Reserve, 2015).Therefore, a mathematical model for credit risk assessment is urgent to help traditional manual approaches(Lessmann, Baesens, Seow, & Thomas, 2015).Credit risk models can be roughly divide into three types: probability of default (PD), exposure at default (EAD) and loss given default (LGD) (Crook, et al., 2007). This work focuses on the PD models, which is a binary classifier of "approving" or "rejecting" loan applications. Various methods have been adopted for constructing PD models. For examples, statistical methods such as linear discriminate analysis (LDA) (Altman, 1968), quadratic discriminant analysis (QDA) (Friedman, Hastie, & Tibshirani, 2001), and logistic regression (LR) (Wiginton, 1980) have been widely used in PD models because the they are simple for implementation and interpretation. On the other hand, machine learning methods such as neural network (NN) (West, 2000; Zhao, et al., 2015), support vector machine (SVM) (Huang, Chen, & Wang, 2007; Min & Lee, 2005), and decision tree (DT) (Lee, Chiu, Chou, & Lu, 2006; Nie, Rowe, Zhang, Tian, & Shi, 2011) provide an alternative for complex models that are harder to interpret but have better accuracy.(Xia, Liu, Da, & Xie, 2018) A recent study shows that ensemble methods that combine multiple base algorithms can aggregate different hypotheses to form a more robust hypothesis and to achieve a better performance. (Xia, et al., 2018). Moreover, heterogeneous ensemble methods performed better because their base learners are more diverse than homogeneous ones. Despite of the excellent accuracy, however, the pipeline to harmony heterogeneous base methods is complex. Researchers have to figure out an optimal solution to the three phases of ensemble learning: pool generation, base model selection, and integration. Since 2010, NN has many improvements(Bahdanau, Cho, & Bengio, 2014; Klambauer, Unterthiner, Mayr, & Hochreiter, 2017). For example, He et al. proposed a weight initialization method designed for networks with nonlinear rectifiers(He, Zhang, Ren, & Sun, 2015). This weight initialization enables researchers to train very deep networks.Another example is scaled exponential linear unit (SELU), which is an activation function with self-normalized properties which prevent vanishing gradients(Klambauer, et al., 2017). Training deep networks with SELU as the activation function is highly robust. Srivastava et al. proposed dropout, which dynamically hide random neurons during training to prevent co-adapting among neurons (Srivastava, Hinton, Krizhevsky, Sutskever, &Salakhutdinov, 2014). The resultant network with dropout acts like an ensemble of numerous sub-networks. These improvements reveal a potential solution to build an ensemble-like model without the complex ensemble learning pipeline. However, one of the most important reasons of current success of deep learning today is that most applications have abundant training data for these algorithms to succeed.A rough rule of thumb is that 10 million is a minimum size of each class for a supervised deep learning algorithm to match manual performance(Goodfellow, Bengio, & Courville, 2016). Unfortunately, to the extent of our knowledge, the size of credit risk data is much smaller than this scale. Another important advantage of deep learning over other supervised machine learning algorithms is that it can extract high-level abstract features from raw data by building complex concepts out of simpler ones(Goodfellow, et al., 2016). However, the features of credit risk data are hand-crafted, where NN cannot make full of the feature abstraction advantage.Considering the size of feature characteristics, whether these improvements on NN

can be extended to PD models remains a question.

This work aims to generate a single NN-based PD model to make the pipeline simple while maintain a good performance as heterogeneous ensemble methods. This work identified three factors that have been shown critical to NN performance in recent studies and conducted a systematic analysis on the factors for credit risk data. The first factor is activation function, which introduces nonlinearity; the second is weight initialization, which expedites learning; the third is regularization, which prevents overfitting. The resultant model achieves comparable performance to heterogeneous ensemble methods with simpler construction process and shorter training time. In addition to the model, a contribution of this work is to demonstrate the potential as well as provide a guideline to applying NN on small data with hand-crafted features.

II. DATASET

This work uses two credit risk datasets, German and Australian datasets, as the evaluation benchmark. The two datasets are publicly available in the UCIMachine Learning Repository (Dheeru & Karra Taniskidou, 2017) and have been widely adopted to evaluate PD models. The original dataset of UCI German Credit Datacomprises of 13 categorical and seven numerical features. The UCI Machine Learning Repository also provides a numerical version of the German dataset which contains 24 numerical features. Since many machine learning algorithms require numerical features, the numerical version is adopted in this work. The German dataset contains 1000 samples, where 700 cases are labelled good the other 300 bad cases are labelled bad.The dataset of UCI Australian Credit Approvalcomprises eight categorical and six numerical features.The categorical values are transformed into integers for easy processing. The Australian dataset contains 690 samples, where 307cases are labelled good/approvedand the other 383cases are labelled bad/rejected.

III. METHOD

Subsections 0-0 describes the NN factors identified in this work respectively. Subsection 0 describe how this work analyzed these factors.

3.1. Activation function

The nonlinearity of NN models comes from activation functions. In the last decade, sigmoid and tanh have been used as default activation functions. However, Glorot and Bengio have shown that sigmoid and tanh may lead neurons into saturation,where the gradient is close to zero(Glorot & Bengio, 2010). Small gradients largely decrease the efficiency of back propagation, the so-called gradient vanishing problem. Various activation functions have been proposed to prevent gradient vanishing to expedite the training of NN models.This work analyzessigmoid, its hard variant, tanh,softsign, ReLU and softplus as well as four modern improvements:LeakyReLU, PReLU, ELU and SELU.Table 1 shows a summary of the ten activation functions analyzed in this work.

Table 1 Activation functions analyzed in this work

Activation function	Equation
Sigmoid	$\frac{1}{1+e^{-x}}$
Hardsigmoid	$\max\left(0,\min\left(1,\frac{x+1}{2}\right)\right)$
Tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Softsign	$\frac{x}{1+\lvert x\rvert}$
ReLU	$\begin{cases}0, & x<0\\ x, & x\geq 0\end{cases}$
Softplus	$\ln(1+e^{x})$
LeakyReLU	$\begin{cases}0.01x, & x<0\\ x, & x\geq 0\end{cases}$
PReLU[1]	$\begin{cases}\alpha x, & x<0\\ x, & x\geq 0\end{cases}$
ELU[1]	$\begin{cases}\alpha(e^{x}-1), & x<0\\ x, & x\geq 0\end{cases}$
SELU[2]	$\begin{cases}\lambda\alpha(e^{x}-1), & x<0\\ \lambda x, & x\geq 0\end{cases}$

[1] α is a trainable parameter. [2] λ =1.0507 and α=1.67326.

3.2. Weight initialization

Once the network topology and activation functions are chosen, one can start training a NNmodel from initial weights. Good weight initialization increases model stability as well as expedite learning convergence (Glorot & Bengio, 2010). A naïve initialization is based on Gaussian distribution, which is adequate for convex objective functions. However, as NN models is getting deeper and more complex in recent years, the objective functions of practical models are far from convex functions.A critical problem of random initialization is the variance of the back-propagated gradients might vanish or explode in very deep networks (Glorot & Bengio, 2010). Therefore, weight initialization has become an important sub-area in deep learning.This work analyzes random, LeCun, Xavier, He and orthogonal initializations. The first four initializations have two variants: normal or uniform distributions. Therefore, this work analyzes nineinitialization functions in total (Table 2).

Table 2 Weight initializations analyzed in this work

Method	Equation
Random (normal)	$X\sim\mathcal{N}(0,0.05^{2})$
Random (uniform)	$X\sim U[-0.05,0.05]$
Random (truncated)[1]	$X\sim\mathcal{N}(0,0.05^{2})$

LeCun (normal)	$X \sim \mathcal{N}\left(0, \frac{1}{n_{\text{in}}}\right)$
LeCun (uniform)	$X \sim U\left[-\sqrt{\frac{3}{n_{\text{in}}}}, \sqrt{\frac{3}{n_{\text{in}}}}\right]$
Xavier(normal)	$X \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}} + n_{\text{out}}}\right)$
Xavier (uniform)	$X \sim U\left[-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right]$
He(normal)	$X \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$
He (uniform)	$X \sim U\left[-\sqrt{\frac{6}{n_{\text{in}}}}, \sqrt{\frac{6}{n_{\text{in}}}}\right]$
Orthogonal[2]	Random orthogonal matrices

$n_{\text{in}}$is size of the previous layer and $n_{\text{out}}$is the size of the current layer.[1]Values farer than two standard deviations from the mean are dropped.[2]Weights of the same layer are initialized together.

3.3. Regularization

Regularization is an important tool in machine learning to prevent overfitting, where a model achieves excellent performance on training data but shows terrible generalization ability on test data. This work analyzes three regularizations widely used in NN studies: early stopping, parameter norm penalty and dropout. Early stopping is a simple but effective regularization in iterative learning algorithms(Yao, Rosasco, & Caponnetto, 2007).The training process stops when the validation loss does not improve for a number, denoted as patient in the context, of epochs. The patient is a hyper-parameter in early stopping.Parameter norm penalty prevents overfitting by simplifying model complexity (Krogh & Hertz, 1992). In practice, parameter norm penalty is achieved by augmenting the loss function with an error of parameternorm.A coefficient, denoted as λ in the context,is used to control the ratio of the augmented error to the actual loss. The λ is a hyper-parameter of parameter norm penalty. This work considers two popular parameter norm penalties, L1 and L2, which corresponds to 1- and 2- vector norms, respectively.Unlike early stopping and parameter norm penalty that are generic to machine learning models, dropout is a NN-specific regularization proposed by (Srivastava, et al., 2014). The concept of dropout is to prevent neuron co-adaption via dropping neurons. Because neurons are dropped randomly, models with dropout have to learn not to rely on some specific critical neurons.From another point of view, networks dropped from some network can be considered as sub-networks to it. Therefore, dropout makes NN act like an ensemble of smaller model instead of a big complex one. These characteristics of dropout improves the robustness of models. In practice, a hyper-parameter, p, is introduced to specify the probability of preserving a neuron.

3.4. Factor analysis process

Different network topologies, activation functions, weight initializations and regularizations have both advantages and disadvantages. A systematic analysis of them on the credit risk data is needed. However, thethree NNfactorshave different characteristics when analyzing.The sets of activation functions and weight initializations are finite therefore can be exhaustively tested. The set of regularizationsis finite, but all of them have real-valued parameters. The network topology is uncountable if arbitrary topology such as CNN and RNN is allowed. This work focuses on a special set of fully connected networks, where the network topology can be formulatedby real-valued parameters.The tree-structured Parzen estimator (TPE) is adopted to efficiently explore hyper-parameter combinations (Bergstra, Bardenet, Bengio, & Kégl, 2011). The TPE is a Bayesian hyper-parameters optimization approach. Unlike grid search which tests numerous hyper-parameter combinations independently, TPE generates promising hyper-parameter combinations based on the test results of previous hyper-parameter combinations. Therefore, TPE has two advantages over grid search. First, the computational cost of TPE does not increase exponentially with the number of hyper-parameters. Second, the hyper-parameter combinations generated by TPE are not limited by a pre-defined set of grids.

IV. EXPERIMENTS

In this work, a credit risk dataset is split into twostratified subsets (4/5 for training and 1/5 for testing) where the positive to negative ratio of each subset is kept the same as the original dataset. The testing subset is not used in hyper-parameter analysis to prevent overfitting.Furthermore, data split is repeated multiple times, denoted as an evaluation run in the context,to increase the robustness of evaluation. Table 3 shows the performancemean and standard deviations of 20 evaluation runs, where the performance of a run is the accuracy average of all data splits in that run indicates.

Table 3Robustness of evaluation

	Number of data splits per run	
Method	1	20
German dataset		
LR[1]	0.7672±0.0306	0.7635±0.0059
SVM[2]	0.7523±0.0303	0.7546±0.0063
NN9[3]	0.7915±0.0206	0.7915±0.0055
Australian dataset		
LR	0.8653±0.0260	0.8653±0.0063
SVM	0.8617±0.0282	0.8532±0.0052
NN9	0.8865±0.0883	0.8865±0.0152

[1]Logistic regression;[2]support vector machine; [3]the NN proposed by Zhao et al.(Zhao, et al., 2015), which has one hidden layer of nine neurons.

In Table 3, the standard deviations of German dataset are about 2~3%when conducting only one data split in an evaluation run. The standard deviations were effectively reduced to 0.5~0.6%) when increasing the number of data splits to 20 per evaluation run for all three methods. A similar effect is also observed in the Australian dataset. Therefore, we adopted 20 data splits for an evaluation run in the following experiments.

4.1. Comparison with existing methods

The proposed method is comprehensively compared with existing methods, including five individual learners and five ensemble learners (**Error! Reference source not found.**).Logistic regression (LR) has been considered as an industry standard for a long time. Gaussian processes classifier (GPC) implements Gaussian processes for classification problems (Rasmussen, 2004). The GPC has been used in various fields and in credit scoring in Xia et al.'s work (Xia, et al., 2018). Support vector machine (SVM) is another machine learning algorithm that is widely seen in the credit scoring field.The Bag-SVM and Bag-Random forest is proposed by Breiman et al. (Breiman, 1996), which adopts the bagging algorithm and use decision trees as base learners.XGBoost is proposed by Chen and Guestrin (Chen & Guestrin, 2016), which adopts the gradient boosting algorithm and use decision and regression trees (CARTs) as base learners. XGBoost has been seen in many winning solutions of Kaggle machine learning competitions.

Table 4 Comparison with existing methods

	Accuracy	
Method	German dataset	Australian dataset
Individual learner		
LR[1]	0.7148	0.8551
GPC[2]	0.7626	0.8525
SVM[3]	0.7543	0.8539
DT[4]	0.6790	0.8299
NN9[5]	0.7915	0.8865
Ensemble learner		
Bag-SVM[6]	0.7622	0.8581
Bag-GPC[7]	0.7813	0.8553
RF[8]	0.7504	0.8626
XGB[9]	0.7811	0.8705
Bstacking[10]	0.7866	0.8828
This work	0.8164	0.8871

[1]Logistic regression; [2]Gaussian process classifier; [3]support vector machine; [4]decision tree; [5]the NN proposed by Zhao et al. (Zhao, et al., 2015), which has one hidden layer of nine neurons; [6]bagging + SVM; [7]bagging + GPC; [8]random forest; [9]XGBoost; [10]the heterogeneous ensemble classifier proposed by Xia et al. (Xia, et al., 2018), which uses SVM, RF, GPC and XGBoost as base learners.The results of other methods came from the original papers to prevent any difference of re-implementation.

In general, ensemble learners achieved better performance than individual learners.The Bag-GPC yielded good performance on the German dataset; the RF yielded good performance on the Australian dataset. The XGB and Bstacking yielded good performance on both datasets. Finally, this work achieved the best performance overall. In **Error! Reference source not found.**, both NN9 and this work used neural network. The difference of this work to NN9 is including many advanced deep learning techniques from other fields. These results show the potential of applying contemporary deep learning methods on small dataset and hand-crafted features.Another advantage of the classifier proposed by this work is that itbelongs to individual learners. In practice, the process to generate an individual learner or a homogeneous ensemble learner like XGB is simpler than a heterogeneous ensemble learner like Bstacking.

4.2. Suggested hyper-parameters

In addition to a high-performance classifier, animportant contribution of this work is a systematic analysis of NN hyper-parameters.This analysis controls NNfactors one by one for comparison, where other hyper parameters were determined by TPE. Table 5 analyzes activation functions; Table 6 analyzes weight initializers; while Table 7 analyzes regularization methods.Table 5 shows several activation functions (Softsign, ELU and SELU) are good. In addition, Table 6 and Table 7 also helps to know activation functions.In Table 6, the best activation function for 8 / 10 weight initializers was SELU. In Table 7, the best activation fell into three ones (Hard sigmoid, ELU and SELU). As a result, analyzing the three tables jointly provides a more comprehensive view than individually. If a systematic parameter tuning process is performed, many activation functions are good; otherwise SELU is a good choice. For weight initializers, the differences are small in Table 6. Moreover, no definitely winners in Table 5 and Table 7. Therefore, picking any of them or conducting a parameter tuning process is suggested in practice. For regularization, early stopping and enabling dropout are suggested. Parameter norm penalty is suggested to be disabled, otherwise L2 is preferred over L1.

Table 5 Comparison of activation functions

		Suggested hyper-parameter combinations			
Activation function	Accuracy	Weight initializer	Early stopping	Parameter norm penalty	Dropout
Sigmoid	0.8220	-	Enabled	Disabled	Enabled
Hard sigmoid	0.8070	Orthogonal	-	Disabled	-

Tanh	0.8260	Random (normal)	-	Disabled	Enabled
Softsign	0.8435	-	-	Disabled	Enabled
ReLU	0.8280	-	Enabled	Disabled	Disabled
Softplus	0.8315	-	Enabled	Disabled	Disabled
LeakyReLU	0.8155	He (uniform)	-	Disabled	Enabled
PReLU	0.8010	-	-	-	Disabled
ELU	0.8375	Random (normal)	Enabled	Disabled	Enabled
SELU	0.8410	-	-	Disabled	Enabled

Accuracies higher than two standard deviations below the mean of the best are highlighted in bold. The standard deviation came from Table 3.

Table 6 Comparison of weight initializers

		Suggested hyper-parameter combinations			
Weight initializer	Accuracy	Activation function	Early stopping	Parameter norm penalty	Dropout
Random (normal)	0.8275	Hard sigmoid	-	Disabled	Enabled
Random (uniform)	0.8215	Hard sigmoid	Disabled	Disabled	Enabled
Random (truncated)	0.8315	SELU	Enabled	Disabled	Enabled
LeCun (normal)	0.8240	SELU	-	Disabled	Enabled
LeCun (uniform)	0.8370	SELU	-	Disabled	Enabled
Xavier (normal)	0.8270	SELU	Disabled	Disabled	Enabled
Xavier (uniform)	0.8275	SELU	-	Disabled	Enabled
He (normal)	0.8270	-	Enabled	Disabled	Enabled
He (uniform)	0.8275	SELU	-	Disabled	Enabled
Orthogonal	0.8335	SELU	-	Disabled	Enabled

Accuracies higher than two standard deviations below the mean of the best are highlighted in bold. The standard deviation came from Table 3.

Table 7 Comparison of regularization methods

Regularization				Suggested hyper-parameter combinations	
Early stopping	Parameter norm penalty	Dropout	Accuracy	Activation function	Weight initializer
Disabled	Disabled	Enabled	0.8325	SELU	-
Disabled	L1	Disabled	0.7915	Hard sigmoid	Orthogonal
Disabled	L1	Enabled	0.8130	-	Random (normal)
Disabled	L2	Disabled	0.7925	Hard sigmoid	-
Disabled	L2	Enabled	0.8120	SELU	-
Enabled	Disabled	Disabled	0.8165	Hard sigmoid	Random (normal)
Enabled	Disabled	Enabled	0.8320	SELU	Random (uniform)
Enabled	L1	Disabled	0.8060	Hard sigmoid	-
Enabled	L1	Enabled	0.8155	ELU	Random (truncated)
Enabled	L2	Disabled	0.8165	Hard sigmoid	-
Enabled	L2	Enabled	0.8310	ELU	Orthogonal

Accuracies higher than two standard deviations below the mean of the best are highlighted in bold. The stan dard deviation came from Table 3.

CONCLUSIONS

This work proposes anindividual learner for small dataset with hand-crafted features based on neural networks. The performance outperforms other individual learners and is comparable to complex ensemble learners. This work shows that the advance of neural networks in other fields can be transferred to financial data and concludes a guideline for future works.

REFERENCES

[1] Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. The journal of finance, 23, 589-609.

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[3] Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In Advances in neural information processing systems (pp. 2546-2554).

[4] Breiman, L. (1996). Bagging predictors. Machine learning, 24, 123-140.

[5] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd

international conference on knowledge discovery and data mining (pp. 785-794): ACM.
[6] Crook, J. N., Edelman, D. B., & Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. European Journal of Operational Research, 183, 1447-1465.
[7] Dheeru, D., & Karra Taniskidou, E. (2017). UCI Machine Learning Repository. In.
[8] Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1): Springer series in statistics New York, NY, USA:.
[9] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).
[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning: The MIT Press.
[11] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).
[12] Huang, C.-L., Chen, M.-C., & Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. Expert systems with applications, 33, 847-856.
[13] Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. In Advances in Neural Information Processing Systems (pp. 971-980).
[14] Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In Advances in neural information processing systems (pp. 950-957).
[15] Lee, T.-S., Chiu, C.-C., Chou, Y.-C., & Lu, C.-J. (2006). Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. Computational Statistics & Data Analysis, 50, 1113-1130.
[16] Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research, 247, 124-136.
[17] Min, J. H., & Lee, Y.-C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. Expert systems with applications, 28, 603-614.
[18] Nie, G., Rowe, W., Zhang, L., Tian, Y., & Shi, Y. (2011). Credit card churn forecasting by logistic regression and decision tree. Expert Systems with Applications, 38, 15273-15285.
[19] Rasmussen, C. E. (2004). Gaussian processes in machine learning. In Advanced lectures on machine learning (pp. 63-71): Springer.
[20] Reserve, F. (2015). Assets and Liabilities of Commercial Banks in the United States (H. 8). In: Washington.
[21] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15, 1929-1958.
[22] West, D. (2000). Neural network credit scoring models. Computers & Operations Research, 27, 1131-1152.
[23] Wiginton, J. C. (1980). A note on the comparison of logit and discriminant models of consumer credit behavior. Journal of Financial and Quantitative Analysis, 15, 757-770.
[24] Xia, Y., Liu, C., Da, B., & Xie, F. (2018). A novel heterogeneous ensemble credit scoring model based on bstacking approach. Expert Systems with Applications, 93, 182-199.
[25] Yao, Y., Rosasco, L., & Caponnetto, A. (2007). On early stopping in gradient descent learning. Constructive Approximation, 26, 289-315.
[26] Zhao, Z., Xu, S., Kang, B. H., Kabir, M. M. J., Liu, Y., & Wasinger, R. (2015). Investigation and improvement of multi-layer perceptron neural networks for credit scoring. Expert Systems with Applications, 42, 3508-3516.

★★★