Expert Systems with Applications 42 (2015) 3508–3516

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Investigation and improvement of multi-layer perceptron neural networks for credit scoring

Zongyuan Zhao [a], Shuxiang Xu [a], Byeong Ho Kang [b], Mir Md Jahangir Kabir [a], Yunling Liu [c,*], Rainer Wasinger [a]

[a] School of Engineering and ICT, University of Tasmania, Launceston, Tasmania, Australia
[b] School of Engineering and ICT, University of Tasmania, Hobart, Tasmania, Australia
[c] College of Information and Electrical Engineering, China Agricultural University, Beijing, China

ARTICLE INFO

Article history:
Available online 10 December 2014

Keywords:
Credit scoring
Neural networks
Back propagation

ABSTRACT

Multi-Layer Perceptron (MLP) neural networks are widely used in automatic credit scoring systems with high accuracy and efficiency. This paper presents a higher accuracy credit scoring model based on MLP neural networks that have been trained with the back propagation algorithm. Our work focuses on enhancing credit scoring models in three aspects: (i) to optimise the data distribution in datasets using a new method called Average Random Choosing; (ii) to compare effects of training–validation–test instance numbers; and (iii) to find the most suitable number of hidden units. We trained 34 models 20 times with different initial weights and training instances. Each model has 6 to 39 hidden units with one hidden layer. Using the well-known German credit dataset we provide test results and a comparison between models, and we get a model with a classification accuracy of 87%, which is higher by 5% than the best result reported in the relevant literature of recent years. We have also proved that our optimisation of dataset structure can increase a model's accuracy significantly in comparison with traditional methods. Finally, we summarise the tendency of scoring accuracy of models when the number of hidden units increases. The results of this work can be applied not only to credit scoring, but also to other MLP neural network applications, especially when the distribution of instances in a dataset is imbalanced.

1. Introduction

Credit scoring, or credit rating, is the set of decision models and their underlying techniques that help lenders judge whether an application of credit should be approved or rejected (Thomas, Edelman, & Crook, 2002). Although credit scoring systems cannot themselves forecast profits, they have been widely used in banks and other financial institutions (Jensen, 1992). So far, these systems have shown the ability to decrease credit risk and reduce "bad" loans. From this point of view, a highly accurate scoring system can save on costs and in this way increase profits.

Credit scoring systems can generally be divided into two kinds: new credit application judgement and prediction of bankruptcy after lending. The first kind uses personal information and the financial status of a loan applicant as inputs to calculate a score. If the score is higher than a "safe" level, the applicant will have a high probability of exhibiting good credit behaviour. On the contrary, a low score means high risk for the loan, so the lender needs to take careful consideration of the application. The second kind of credit scoring focuses on the credit record of existing customers. From the payment history of a customer, a financial institution can predict a customer's payment ability and alter his/her credit level. This paper only focuses on new credit application judgement scoring systems.

Compared with traditional credit scoring, which is calculated by professional bank managers, automatic scoring has some obvious advantages: it saves cost and time for evaluating new credit applications; and it is consistent and objective (Marques, Garcia, & Sanchez, 2013). However, some current computational approaches are not as capable as experienced loan experts on judgement accuracy. As the accuracy of scoring can greatly affect the interests of financial institutions, researchers continually strive to improve and enhance scoring accuracy rates. In recent years, artificial intelligence has shown its advantages in credit scoring in comparison with linear probability models, discriminant analysis, and other statistical techniques (Saberi et al., 2013).

* Corresponding author.
E-mail addresses: Zongyuan.Zhao@utas.edu.au (Z. Zhao), Shuxiang.Xu@utas.edu.au (S. Xu), Byeong.Kang@utas.edu.au (B.H. Kang), mmjkabir@utas.edu.au (M.M.J. Kabir), lyunling@163.com (Y. Liu), rainer.wasinger@utas.edu.au (R. Wasinger).

http://dx.doi.org/10.1016/j.eswa.2014.12.006
0957-4174/

Among all of the computational methods available, MLP models are widely utilised (Jensen, 1992; Oreski, Oreski, & Oreski, 2012; Zhong, Miao, Shen, & Feng, 2014) because they provide competitive prediction ability against other methods (West, 2000; Šušteršič, Mramor, & Zupan, 2009). In Werbos (1975), the back-propagation (BP) algorithm was developed and has now been widely used in training MLP feed-forward neural networks. Memetic pareto artificial neural networks (MPANN) have been used to optimise the BP algorithm using a multi-objective evolutionary algorithm and a gradient based local search (Abbass, 2003). This training method can reduce training time and at the same time enhance classification accuracy. That paper also presented a self-adaptive version called SPANN, which was faster than BP and able to largely reduce computational complexity. Many tests showed that RBF, LS-SVM and BP classifiers yielded very good performance with eight credit scoring datasets (Baesens et al., 2003). At the same time however, some linear classifiers such as LDA and LOG also generated good results. This indicated that the performance differences between some models were not obvious (Marqués, García, & Sánchez, 2012). Another test received similar results by testing the accuracy of several automatic scoring models using the German, Australian, and Japanese credit datasets (Bache & Lichman, 2013). It reported that compared with BP, the C4.5 decision tree performed a little better for credit scoring but both of them could achieve high accuracies. Also, Nearest Neighbour and Naïve Bayes classifiers appeared to be the worst in their tests.

Improvements to neural networks include altering the ratios of training and testing datasets, the number of hidden nodes, and the training iterations. A nine learning schemes with different training-to-validation data ratios was investigated and got the implementation results with the German dataset (Khashman, 2010). That paper concluded that the learning scheme with 400 cases for training and 600 for validation performed best with an overall accuracy rate of 83.6%. An emotional neural network (Khashman, 2008) is a modified BP learning algorithm. It has additional emotional weights that are updated using two additional emotional parameters: anxiety and confidence. When comparing emotional neural networks with conventional networks for credit risk evaluation, experimental results have shown that both models are effective, but that the emotional models outperform the conventional ones in decision making speed and accuracy (Khashman, 2011). Another enhancement is the artificial metaplasticity MLP, which is especially efficient when fewer patterns of a class are available or when information inherent to low probability events is crucial for a successful application. This model achieved an accuracy of 84.67% for the German dataset, and 92.75% for the Australian dataset (Marcano-Cedeño, Marin-de-la-Barcena, Jimenez-Trillo, Piñuela, & Andina, 2011). Fuzzy numbers can replace crisp weights and biases to overcome uncertainties and complexities in financial datasets (Khashei, Rezvan, Hamadani, & Bijari, 2013). Results here have shown that this hybrid classification model outperformed the traditional ANNs and are also better than SVM, KNN (k-Nearest Neighbours), and others.

There are some hybrid systems that have implemented neural networks as part of a larger whole construction of a combination system. In (Lee & Chen, 2005), a two-stage hybrid modelling procedure with ANN and multivariate adaptive regression splines (MARS) is presented. After using MARS in building the credit scoring model, the obtained significant variables then served as the input nodes of the ANN. However, the improvements were not obvious. In fact, ensemble systems like this one performed better only in one of the three datasets in the experiments of Tsai and Wu (2008). The authors compared MLP with multiple classifiers or classifier ensembles and concluded that the ability of hybrid systems was not better than usual methods, which meant that all methods needed to be considered when making financial decisions.

As to other methods, support vector machine (SVM) and genetic algorithms (GA) are also used for credit rating with good performance. In Hens and Tiwari (2012), a SVM model was refined by reduction of features using an F score and took a sample instead of a whole dataset to create the credit scoring model. Test results showed that this method was competitive in the view of accuracy as well as computational time. In Chi and Hsu (2012), they selected important variables for use by a GA to combine a bank's internal behavioural rating model and an external credit bureau model. This dual scoring model underwent more accurate risk judgment and segmentation to further discover the parts which were required to be enhanced in management or control from a mortgage portfolio. Other than SVM and GA, Clustering-Launched Classification (CLC) is also available and may perform better than SVM (Luo, Cheng, & Hsieh, 2009). A multi-criteria quadratic programming (MCQP) model was proposed based on the idea of maximising external distance between groups and minimising internal distances within a certain group (Peng, Kou, Shi, & Chen, 2008). This model could solve linear equations to find a global optimal solution and obtained the classifier and at the same time used kernel functions to solve nonlinear problems. Compared to SVM it seemed more accurate and scalable to massive problems. Decision tree (DT) is another good alternative method. A dual strategy ensemble trees was developed based on bagging and random subspace (Wang, Ma, Huang, & Xu, 2012). This DT model reduced influences of noise data and redundant attributes of data to get relatively higher classification accuracy.

Recently, imbalanced datasets (i.e. where instances belonging to one class heavily outnumber instances in other classes) have attracted attention, and some work has shown that appropriate ratios of different kinds of instances can augment classification accuracy. In Brown and Mues (2012), they used five real-world datasets to test the effect of good/bad credit instance ratio. Results showed that linear discriminant analysis (LDA) and logistic regression (LOG) performed acceptable rating accuracy with both slightly imbalanced datasets and highly imbalanced ones. To avoid the effect of imbalance data distribution, a dynamic classifier and dynamic ensemble selection of features were added in the scoring model. This resulted in better performance compared to ensemble static classifiers (Xiao, Xie, He, & Jiang, 2012).

As there are usually some irrelevant attributes in credit datasets, optimisation of input attribution is especially important for SVM systems. Two feature selection methods – a probabilistic classifier with a proper prior structure, and multiple kernel learning using a specific kernel calculation strategy – showed good performance with choosing parts of features in the German dataset (Gonen, Gonen, & Gurgen, 2012). Ping and Yongheng (2011) designed a hybrid SVM-based model using a neighbourhood rough set to select input features and using a grid search to optimise RBF kernel parameters. This model showed its ability when selecting input features and performed better in scoring compared with other hybrid classifiers. A novel feature selection model was proposed based on a rough set and scatter search method called RSFS (Wang, Hedar, Wang, & Ma, 2012). In RSFS, conditional entropy is regarded as heuristic to search optimal solutions. Tests on the Australian dataset and the German dataset demonstrated its competitive performance in saving computational costs and improving classification accuracy. In Yu, Yao, Wang, and Lai (2011), they not only optimised input features by the design of experiment (DOE) method, but also proposed a weighted least squares SVM that emphasised the importance of different classes. Tests in the German dataset and the Australian dataset showed that this combination method had acceptable accuracy with less computation time.

In general, a neural network with back propagation algorithm can score credit applications with high accuracy. Consistent exper-

iments have shown that this method is appropriate for use in credit scoring. However, there are still some problems for this model:

1. As the ratios of approved/rejected instances in the datasets are usually not balanced and sometimes highly imbalanced, the training process of a model may have to deal with more approved instances than rejected ones. This may cause bad performance when applying the learned model to the test datasets.
2. The numbers of instances used in test, validation, and training sets are always limited. More data used in training means less used in validation and test, which may result in low accuracy. As all three parts need as many instances as possible, the distribution of data is vital for the final test results.
3. The number of hidden units affects the accuracy and the time costs of a model. More hidden units may lead to high computing time, but insufficient units cannot achieve high accuracy. The appropriate number can only be learned through experimentation with different models followed by selection of the best according to the test results.

In this paper, we aim to solve these problems by designing a novel data distribution method named "Average Random Choosing", and through optimisation of the MLP structure as outlined below:

1. Optimise the ratio of approved/rejected cases in input instances. Usually a training dataset is randomly chosen from the origin dataset, or a 10-fold validation method is used for the purpose which is basically the same as random selection. In this paper, we propose an Average Random Choosing method. With this method, the rate of different class data stays the same in training as the global rate, but the dataset is chosen randomly, which also assumes fairness. We compare the accuracies of models made by this method with those from models based on the random choosing method.
2. Test the effect of different ratios of training-validation-testing data. Khashman (2010) tested different ratios of training–validation data and obtained the best ratio by choosing the one with the highest accuracy. In our model however, there are three kinds of dataset: training, validation, and test data. Ratios of 6:2:2, 8:1:1 and 9:0.5:0.5 are tested. We also choose the scheme with the highest accuracy as the most suitable one.
3. Improve the structure of MLP networks. We train 34 models with the number of hidden neurons set to a number between 6 and 39, and train each of them 20 times with different input instances. Then we use these models to score instances in test sets. The average and highest accuracy of each group is recorded and we choose the model with the highest accuracy as the best one.

This work differs from existing relevant research in the following ways: (i) we optimise datasets used in training, validation, and testing. Our new method thus guarantees fairness and also suits imbalanced datasets, (ii) we compare 34 models with different numbers of hidden units, to obtain the model with the highest accuracy and efficiency. Additionally, (iii) we explore the tendency of model accuracy when the number of hidden units increases.

The structure of this paper is as follows: Section 2 gives a brief introduction of the German dataset which is used in our tests. Section 3 describes our data processing methods and the MLP network models. Section 4 presents the experimental results and compares the accuracies of different models and methods. Finally Section 5 summarises this work and gives directions for possible future work.

2. The German dataset

In this work, we use the German dataset which is available publicly on the UCI Machine Learning Repository website (Bache & Lichman, 2013). It is a real world dataset with 21 features including 20 attributes recording personal information and financial history of applicants. The last feature is labelled as approved (marked as 1) or rejected (marked as 2). Some of these attributes are numerical, but others are qualitative and cannot be computed in the training of neural networks. Thus, a numerical version of the dataset is used in this work. It transforms all qualitative variables to numeric and adds four more attributes. The meanings of the original attributes are described in Table 1.

This dataset contains 1000 instances, with 700 approved application cases and 300 rejected ones. These instances are presented randomly.

The German credit dataset is widely used as a benchmark and has had many scoring models. In recent years, different models have been utilised on this dataset to demonstrate the ability of neural networks to solve credit scoring problems. The accuracies of some representative models are listed in Table 2.

Some of the accuracies as listed above are average rates in a group of models and others are the best rates. The "scoring models" in Table 2 are basic models used in experiments, and many of them have been improved. From this table we can see that there

Table 1
Original attributes in the German dataset.

Number	Description	Class
Attribute 1	Status of existing checking account	Qualitative
Attribute 2	Duration in month	Numerical
Attribute 3	Credit history	Qualitative
Attribute 4	Purpose	Qualitative
Attribute 5	Credit amount	Numerical
Attribute 6	Savings account/bonds	Qualitative
Attribute 7	Present employment since	Qualitative
Attribute 8	Instalment rate in percentage of disposable income	Numerical
Attribute 9	Personal status and sex	Qualitative
Attribute 10	Other debtors/guarantors	Qualitative
Attribute 11	Present residence since	Numerical
Attribute 12	Property	Qualitative
Attribute 13	Age in years	Numerical
Attribute 14	Other instalment plans	Qualitative
attribute 15	Housing	Qualitative
Attribute 16	Number of existing credits at this bank	Numerical
Attribute 17	Job	Qualitative
Attribute 18	Number of people being liable to provide maintenance for	Numerical
Attribute 19	Telephone	Qualitative
Attribute 20	Foreign worker	Qualitative

Table 2
Some representative models in recent years and their accuracies.

Article name	Scoring models	Accuracy (%)
Marcano-Cedeño et al. (2011)	MLP	84.67 ± 1.5
Xiao et al. (2012)	Ensemble	82.03
Brown and Mues (2012)	LS-SVM	81.9
Khashei et al. (2013))	MLP	81.3
Khashman (2011)	MLP	81.03
Hens and Tiwari (2012)	SVM	80.42
Wang, Ma, et al. (2012)	DT	78.52
Setiono, Baesens, and Mues (2011)	Re-Rx	78.47
Yu et al. (2011)	SVM	78.46
Vukovic, Delibasic, Uzelac, and Suknovic (2012)	Case-based reasoning model	77.4
Ping and Yongheng (2011)	SVM	76.6
Gonen et al. (2012)	SVM	75.4
Marqués, García, and Sánchez (2013)	SVM	71.8

are lots of models that have used this dataset and there is no significant difference between their performances. The highest one is 84.67 ± 1.5%, achieved by Marcano-Cedeño et al. (2011) using a MLP model. The average accuracy of all the models is 79.08%. This table only includes the results from journal papers published between 2011 and 2013. There were lots of experiments published in previous years, but the accuracies were not better. As the accuracies from models related to this dataset are still not high enough, it is a very challenging goal to improve this classification accuracy rate.

3. Data processing and credit rating model

3.1. Average Random Choosing method

The imbalance of data classes (where instances belonging to one class heavily outnumber instances in other classes) usually exist in credit datasets. The reason is that in real life the number of successful credit applications is usually larger than that of rejected applications. Thus, in the training of neural networks there should be more instances of approved applications in order to get a better scoring model. From the point of test data, as the original dataset is imbalanced, it is reasonable to keep the same ratio (i.e. approved:failed) in the test set. However, fairness cannot be guaranteed if the ratio between good and bad cases changes.

Another problem of data processing is the ratio of training-validation-test sets. All three sets should represent a relevant number of instances. Usually, more instances for training can lead to a higher chance of getting a better model. However, as the amount of data is limited, more data used for training means less for validation and testing. This will cause bad or unequal test performance. It is important to choose the best ratio since this affects the scoring model.

To solve these problems, we propose a method to process credit data. Suppose the total amount of instances is n, and the ratio of good applications in the dataset is p. Then the amount of good and bad applications is

$$\textit{Good applications} : p * n,$$

$$\textit{Bad applications} : (1-p) * n,$$

Then suppose the ratio of data used in training is t and in validation is v. Then we have

$$\textit{Training data} : n * t$$

$$\textit{Validation data} : n * v$$

$$\textit{Test data} : n * (1-t-v)$$

Table 3
Amount of instances in each group.

	Training dataset	Validation dataset	Test dataset
Good application	$n*p*t$	$n*p*v$	$n*p*(1-t-v)$
Bad application	$n*(1-p)*t$	$n*(1-p)*v$	$n*(1-p)*(1-t-v)$

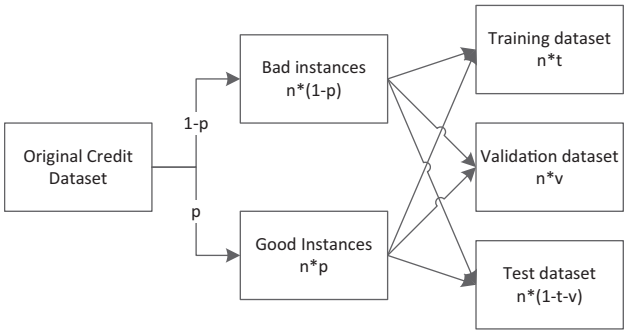

Fig. 1. Flow of data processing and the amount of instances in each group .

As we want the ratio of good to bad applications to stay the same in the training data (as in the original data), the training, validation and test data can be divided into good cases and bad cases. Table 3 shows the amount in each group:

The flow of processing data is listed in Fig. 1:

From the original dataset, two different kinds of data, bad instances and good instances are divided into two groups. Then both of them are divided into training, validation and test datasets randomly. This step should be repeated for each new network training session, which can minimise the effect of unordinary instances. This way, it is more likely to get a good data distribution which promotes a good scoring model. Pseudo of this Average Random Choosing method is described as follows:

Algorithm

Original Dataset D = $\{x_i, t_i \mid i = 0,1 \ldots n\}$
1. Select out all samples of class A A = $\{x_i, t_i \mid t_i = 1\}$,
2. Select out all samples of class B B = $\{x_i, t_i \mid t_i = 2\}$
3. Select instances for test from class A $T_A = \{x_i, t_i \mid i = \text{randperm}(n * p * (1-t-v)), (x_i, t_i) \in A\}$
4. Select instances for test from class B $T_B = \{x_i, t_i \mid i = \text{randperm}(n*(1-p)*(1-t-v)), (x_i, t_i) \in B\}$
5. The test instances group $T = T_A \cup T_B$
6. Samples in class A that are used to train and validate $S_A = A - T_A$
7. Samples in class B that are used to train and validate $S_B = B - T_B$
8. while(not reach the max epoch)
9. Choose out samples used for training in class A $S_{At} = \{x_i, t_i \mid i = \text{randperm}(n * p * t), (x_i, t_i) \in S_A\}$
10. Choose out samples used for training in class B $S_{Bt} = \{x_i, t_i \mid i = \text{randperm}(n*(1-p)*t), (x_i, t_i) \in S_B\}$
11. The training group is $S_t = S_{At} + S_{Bt}$
12. The validation group is $S_v = (S_A - S_{At}) \cup (S_B - S_{Bt})$
13. Calculate output $y_i = \emptyset\left(\sum_{s=0}^{h} w_s \emptyset\left(\sum_{r=0}^{n} w_{sr} x_r\right)\right), x_r \in S_t$
14. Update MLP by BP. Error signal is $e(i) = t_i - y_i$ $t_i \in S_t$
15. Validate MLP $y_i = \emptyset\left(\sum_{s=0}^{h} w_s \emptyset\left(\sum_{r=0}^{n} w_{sr} x_r\right)\right), x_r \in S_v$
16. Performance of MLP $v = \frac{\sum_{i=0}^{n*v} e(i)}{n*v}$
17. if($v > v_0$) stop training
18. else $v_0 = v$
19. end while

By this method, all groups (training, validation, and test) have the same ratio of good to bad instances. Traditional 10-fold validation divides a dataset into 10 blocks of data randomly before training starts. The advantage is that the dataset can be trained 10 times with different datasets for training and validation. However, it is not really random choosing and is not suited for imbalanced datasets. In some extreme circumstances, there could be one group with only one class of data. This is obviously unable to judge the performance of the model. Our average random method chooses instances randomly from both classes of data. It can choose data randomly, which ensures more instance combinations can be used for training. Also, the training, validation, and test datasets have the same ratio of good to bad instances. This is specially designed for imbalanced datasets by guaranteeing enough testing and training data from both classes. Comparison experiments will be presented below to choose the best training–validation–test ratio.

3.2. MLP scoring mode

In this work, we use a feed-forward neural network which contains three layers. The first layer is the input layer with 24 neurons since the dataset contains 24 input features (attributes). The last layer is the output layer with only one neuron, which stands for the score of an application. As "1" stands for approved cases and "2" for rejected cases in the dataset, here we define a threshold value for the score. If the output is less than 1.5, we then regard it as a good application; else it will be treated as a bad one. Only one hidden layer is used to reduce computing complexity. Fig. 2 is the structure of a feed-forward MLP neural network.

In this work we use the back propagation (BP) algorithm to train the network. This means that the weights are altered by feeding back the differences between output signals and desired output values. It uses the gradient decent method to control the speed of training. The neuron activation function is the tangent function.

The number of hidden neurons can have a great impact on the performance of the network. Here we build 34 different MLP models, with the number of hidden units varying from 6 to 39. Too few hidden units cannot solve complex credit rating problems, while too many of them may result in low efficiency and accuracy as well. Our experiment results also prove that this scope is large enough since some of the models can achieve high accuracies.

During the procedure of training, validation is used to control over-fitting. As the iterations continue, the error rate of the network goes down until it reaches the lowest point. After that point, the error rate does not drop any further. This is called over training or over fitting and the training procedure should be stopped when the error rate reaches this lowest point. Here we use some data to validate the network after every iteration. Validation data is not used to train the network; it is more like the test data. After each iteration, we calculate the error rate with validation data and compare it with the result of the last iteration. If the new one is larger, then the iteration stops immediately. Else, the training and validation process goes on. In this way, we can avoid over fitting.

3.3. Discussion

The advantage of the Average Random Choosing method is that the distributions of instances in different classes are balanced in training, validation, and test datasets. Thus, the class imbalance problem can be alleviated to some extent. It should be noted that the new method can also be effective on slightly imbalanced or balanced datasets compared with the traditional random choosing method, as the traditional method may aggravate imbalance in training and other groups.

However, this method will perform badly with extremely imbalanced datasets, as in these datasets, one or more classes have too few instances because our method does not create new instances. In these cases, some oversampling methods to generate new instances should be considered.

4. Experiments

In this work we design experiments to determine a competitive MLP model for credit scoring. We use Matlab software running on a 2.5 GHz PC with 4 GB RAM, and the Windows 7 OS. There are three aspects of the experiments. The first aspect is to find the best amount of data used in training, validation, and testing, respectively. Then based on the results, the second aspect is to focus on our new instance choosing method. Thirdly, we discuss the number of hidden neurons by training each model 20 times with different initial weights for each kind of model in all experiments. All instances for training, validation, and testing are randomly chosen. The best and average error rates are listed and discussed.

4.1. Choosing the training-validation-test data ratio

In the experiments, we use 3 different ratios of data, 800:100:100, 900:50:50 and 600:200:200 to determine the most suitable one. To be more accurate, all groups of data are chosen randomly from the German dataset. The number of hidden neurons varies from 6 to 39, which ensures that every group can get their best model. Additionally, each kind of model is trained 20 times. We record the lowest error rate and average rate of each kind of model. Test results are listed in Table 4.

The results show that a ratio of 800:100:100 performs better in nearly all aspects in regards to accuracy rate. The lowest error rate, 0.17, is achieved with 15 hidden units. This is also the best model of all. The average error rate can indicate an overall performance of some model groups. The model with 10 hidden units seems more stable, with an average error rate of 0.2295, which is lower than the others. For all models, the average lowest error rate is around 0.208, indicating that it is more likely to get a very low error rate with this training–validation–test ratio.

The ratio of 600:200:200 gives more data to testing, which leads to insufficient data for training. Thus it gets high error rates in regards to both the lowest value and the average value. Although the ratio 900:50:50 has more instances for training and the lowest error rate is very close to the best one, the shortness of testing data leads to a high average value, which means that there is a low possibility to get a good model.

4.2. Training with the Average Random Choosing method

In this section, we compare the models trained with data from our Average Random Choosing method. Nothing has changed except that we control the percentage of approved/rejected instances in each dataset. The overall percentage is 70% for approved instances and 30% for rejected instances. So this ratio stays the same in the training, validation, and test data groups. The ratio of training–validation–test is 800:200:200, which performs best in our previous tests (see Section 4.1). The results are listed in Table 5.

Comparing with the best and the average error rates in Table 4, the results from this round of testing are clearly better. The lowest error rate is 0.13, which is 0.04 lower than the previous experiments. The average value of the lowest error rate is around 0.155, which is a 25% improvement (the previous best rate was

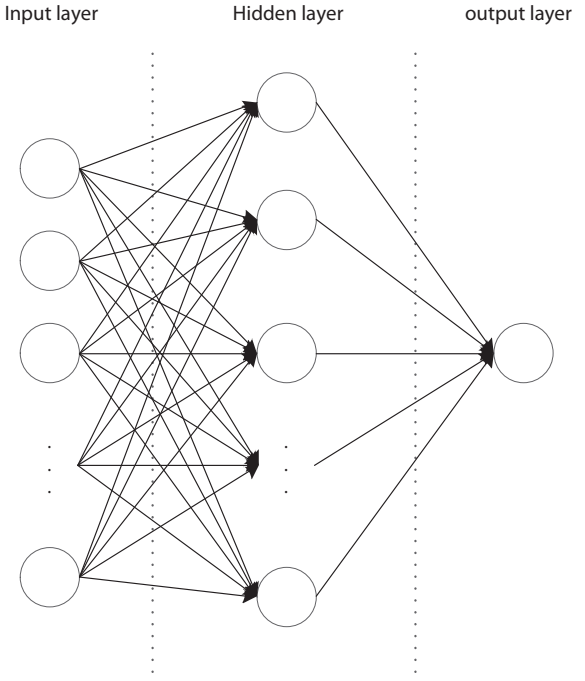

Fig. 2. Typical structure of a MLP neural network.

0.208, as seen in Section 4.1). This indicates that, with this kind of data, it is more likely to get a high accuracy model that has high predictability.

To be more objective, we use another credit rating dataset to validate the performance of our method. The Australian credit dataset contains 690 instances and 15 attributes. 307 of all instances belong to class 1 and the other 383 instances belong to class 2. Thus, this credit rating dataset is not highly imbalanced. A similar training method is applied to this dataset. However, as this dataset only contains 15 attributes, which is less than the German credit dataset, the number of hidden nodes is set from 3 to 32. Test results are listed in Table 6.

Test results indicate that our novel method can also improve the performance of credit rating with the Australian credit dataset. The error rate declines from 12.9% to 11.3% with test data and 10.7% to 10.5% in validation. This test proves that the Average Random Choosing method can still optimise the training of neural networks even when the dataset is not highly imbalanced.

Test results also show that better organised data can enhance model performance, especially when the dataset is imbalanced between its classes. As the German dataset is a real world application, our Average Random Choosing method can be easily generalised to handling other credit datasets. However, when the dataset is not highly imbalanced, our new method can also improve the performance to some extent.

Table 4
Test results and comparison of different ratios of data. Lowest error rate represents the best model we got after 20 times training; Average error rate is the average result after 20 times training.

Number of hidden neurons	800:100:100		900:50:50		600:200:200	
	Lowest error rate	Average error rate	Lowest error rate	Average error rate	Lowest error rate	Average error rate
6	0.21	0.2415	0.21	0.2925	0.245	0.273
7	0.21	0.251	0.2	0.251	0.225	0.273
8	0.19	0.239	0.21	0.2765	0.22	0.2615
9	0.18	0.2385	0.18	0.2315	0.245	0.27675
10	0.18	0.2295	0.2	0.243	0.24	0.27075
11	0.2	0.242	0.18	0.2635	0.225	0.26275
12	0.21	0.246	0.19	0.2675	0.23	0.26575
13	0.2	0.256	0.21	0.2705	0.2	0.26625
14	0.21	0.258	0.19	0.258	0.205	0.26525
15	0.17	0.2595	0.21	0.26	0.24	0.2715
16	0.22	0.246	0.23	0.26	0.23	0.266
17	0.19	0.248	0.21	0.245	0.235	0.2745
18	0.21	0.2595	0.21	0.293	0.21	0.26725
19	0.2	0.255	0.24	0.277	0.235	0.28
20	0.21	0.2585	0.21	0.262	0.22	0.2655
21	0.22	0.256	0.21	0.2835	0.24	0.2665
22	0.2	0.2545	0.22	0.2985	0.23	0.26475
23	0.19	0.259	0.21	0.2675	0.24	0.27075
24	0.19	0.26	0.2	0.26	0.245	0.27375
25	0.22	0.257	0.21	0.2845	0.24	0.27175
26	0.22	0.2725	0.19	0.276	0.235	0.28575
27	0.2	0.261	0.22	0.287	0.24	0.2735
28	0.21	0.2625	0.23	0.2755	0.235	0.2755
29	0.21	0.262	0.19	0.266	0.225	0.273
30	0.23	0.2765	0.23	0.278	0.21	0.2665
31	0.24	0.2715	0.23	0.2775	0.25	0.28575
32	0.22	0.265	0.2	0.2775	0.245	0.272
33	0.23	0.2755	0.22	0.273	0.245	0.27725
34	0.22	0.272	0.22	0.281	0.24	0.28575
35	0.22	0.2675	0.22	0.2885	0.235	0.28225
36	0.24	0.2835	0.19	0.285	0.245	0.2895
37	0.22	0.278	0.23	0.3175	0.225	0.29125
38	0.17	0.28	0.22	0.301	0.21	0.27825
39	0.24	0.294	0.2	0.284	0.235	0.284
Best	0.17	0.2295	0.18	0.2315	0.2	0.2615
Average	0.208235	0.259882	0.209412	0.273897	0.231618	0.27375

Table 5
Test results and comparison of two instance choosing methods. Each model is trained 20 times with different initial weights.

Number of hidden neurons	Average Random Choosing method				Pure random choosing method	
	Lowest test error rate	Average error rate	lowest validation error rate	Average validation error rate	Lowest test error rate	Average error rate
6	0.14	0.2145	0.1265	0.1502	0.21	0.2415
7	0.15	0.211	0.1228	0.1441	0.21	0.251
8	0.17	0.2215	0.1231	0.1438	0.19	0.239
9	0.13	0.2085	0.1284	0.1472	0.18	0.2385
10	0.13	0.221	0.1270	0.1471	0.18	0.2295
11	0.16	0.223	0.1234	0.1439	0.2	0.242
12	0.13	0.225	0.1297	0.1491	0.21	0.246
13	0.16	0.2225	0.1163	0.1431	0.2	0.256
14	0.14	0.2105	0.1113	0.1407	0.21	0.258
15	0.16	0.219	0.1103	0.1422	0.17	0.2595
16	0.14	0.2085	0.1245	0.1399	0.22	0.246
17	0.19	0.2415	0.1196	0.1433	0.19	0.248
18	0.17	0.221	0.1022	0.1342	0.21	0.2595
19	0.14	0.2135	0.1124	0.1407	0.2	0.255
20	0.16	0.2215	0.1006	0.1402	0.21	0.2585
21	0.16	0.215	0.1207	0.1368	0.22	0.256
22	0.15	0.2195	0.1217	0.1351	0.2	0.2545
23	0.18	0.2265	0.1143	0.1353	0.19	0.259
24	0.14	0.215	0.1249	0.1377	0.19	0.26
25	0.15	0.223	0.1007	0.1394	0.22	0.257
26	0.14	0.224	0.0884	0.1310	0.22	0.2725
27	0.17	0.2175	0.1227	0.1380	0.2	0.261
28	0.15	0.2235	0.1123	0.1345	0.21	0.2625
29	0.15	0.2265	0.1100	0.1339	0.21	0.262
30	0.15	0.2275	0.0916	0.1304	0.23	0.2765
31	0.18	0.228	0.0892	0.1283	0.24	0.2715
32	0.18	0.234	0.1013	0.1295	0.22	0.265
33	0.17	0.2345	0.1070	0.1287	0.23	0.2755
34	0.17	0.228	0.1023	0.1285	0.22	0.272
35	0.14	0.229	0.0932	0.1243	0.22	0.2675
36	0.14	0.2275	0.1055	0.1311	0.24	0.2835
37	0.16	0.2265	0.0963	0.1258	0.22	0.278
38	0.19	0.2385	0.0801	0.1324	0.17	0.28
39	0.14	0.2375	0.0854	0.1215	0.24	0.294
Best	0.13	0.2085	0.0801	0.1215	0.17	0.2295
Average	0.1553	0.2231	0.1102	0.1368	0.2082	0.2599

Table 6
Test results and comparison of two instance choosing methods with the Australian credit dataset. Each model is trained 20 times with different initial weights.

Number of hidden neurons	Average Random Choosing method		Pure random choosing method	
	Average test error	Average validation error	Average test error	Average validation error
3	0.1406	0.1315	0.1290	0.1239
4	0.1292	0.1329	0.1486	0.1343
5	0.1389	0.1286	0.1601	0.1258
6	0.1341	0.1314	0.1420	0.1349
7	0.1401	0.1253	0.1406	0.1252
8	0.1570	0.1196	0.1355	0.1221
9	0.1534	0.1373	0.1333	0.1283
10	0.1377	0.1133	0.1355	0.1233
11	0.1304	0.1286	0.1428	0.1216
12	0.1522	0.1209	0.1543	0.1162
13	0.1304	0.1165	0.1399	0.1248
14	0.1401	0.1271	0.1630	0.1188
15	0.1268	0.1267	0.1370	0.1191
16	0.1196	0.1142	0.1449	0.1194
17	0.1292	0.1157	0.1377	0.1193
18	0.1510	0.1265	0.1312	0.1184
19	0.1353	0.1227	0.1536	0.1171
20	0.1280	0.1204	0.1522	0.1148
21	0.1147	0.1170	0.1543	0.1134
22	0.1220	0.1201	0.1304	0.1181
23	0.1437	0.1284	0.1464	0.1118
24	0.1498	0.1265	0.1384	0.1159
25	0.1449	0.1273	0.1536	0.1163
26	0.1425	0.1284	0.1522	0.1146
27	0.1413	0.1182	0.1464	0.1151
28	0.1437	0.1169	0.1377	0.1105

Table 6 (*continued*)

Number of hidden neurons	Average Random Choosing method		Pure random choosing method	
	Average test error	Average validation error	Average test error	Average validation error
29	0.1341	0.1182	0.1304	0.1081
30	0.1220	0.1099	0.1464	0.1082
31	0.1135	0.1055	0.1601	0.1072
32	0.1389	0.1119	0.1399	0.1103
Best	0.1135	0.1055	0.1290	0.1072
Average STD	0.0384	0.0139	0.0389	0.0142

Table 7
Test results of the best model with highest accuracy and efficiency.

	800:100:100		900:50:50		600:200:200		Average Random Choosing method	
	Lowest error rate	Average error rate	Lowest error rate	Average error rate	Lowest error rate	Average error rate	Lowest error rate	Average error rate
9 Hidden neurons	0.18	0.2385	0.18	0.2315	0.245	0.2768	0.13	0.2085
Best of all	0.17	0.2295	0.18	0.2315	0.2	0.2615	0.13	0.2085
Average	0.2082	0.2599	0.2094	0.2739	0.2316	0.2738	0.1553	0.2230

4.3. Number of hidden neurons

In Table 4, we also list the result of validation for each kind of model, including the lowest and average rates among the 20 different models. Regarding the test data error rates, the lowest ones seem to have been achieved when the number of hidden neurons is 9, 10, or 12, respectively. However, with these numbers, many other models also get competitive results. When the number of hidden units is 6, 14, 16, 19, 24, 26, 35, 36, and 39, respectively, the lowest rate is 0.14, which is only a little higher than 0.13 so they can still be regarded as good models. As to the average rate, the model with 9 hidden neurons gets the lowest one, which is 0.2085. The average of all models is however only 0.223, which means that there is little difference between models when only accuracy is considered. This means that there is no ubiquitous principle for the relationship between the number of hidden neurons and the accuracy of a model.

However, computation time is also very important to neural network research. More hidden neurons can lead to more computation time. Thus, if some models produce the same accuracy, the one with less hidden neurons is preferred. In our experiments, the network with 9 hidden neurons wins in most cases (shown in Table 7). As such, this model is chosen as the most suitable for the German credit dataset in our experiments.

It is more interesting when checking the validation error rates. As validation data is also a kind of test data (but with a different purpose), the error rate can also reflect predictability. It is always higher than the accuracy of test data because a training process will not stop until a validation gets high accuracy. In our experiments, the accuracy of validation can reach almost 0.92 with 38 hidden units in the model. This is 0.05 higher than the best result achieved with the test data. Although this value is not as objective as the accuracy from pure test data, it indicates that the MLP model has the ability of rating credit applications more precisely. Also, there is an interesting tendency found in the validation dataset. As the number of hidden units gets larger, the error rate seems to get lower. This is shown in Fig. 3.

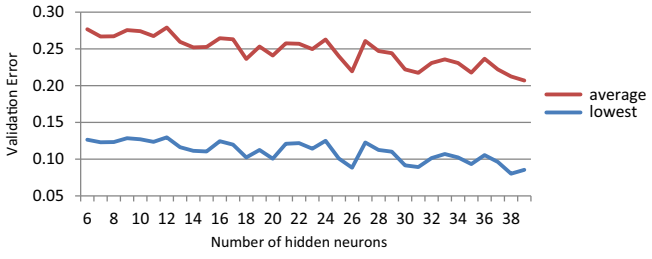

Fig. 3. Tendency of the model's error rates when the number of hidden units increases.

4.4. Summary of experiments

In the first round of experiments, we compare the accuracies of models trained with different ratios of training–validation–test data. The 800:100:100 combination gets the highest accuracy, so this ratio is most suitable for building an acceptable model. After that, we use our Average Random Choosing method to optimise the dataset. The ratio of approved/rejected instances in the German dataset, 7:3, is precisely implemented in the datasets for training, validation, and testing. Results show that our new method can remarkably enhance the accuracy, from 83% to 87% for the best model. Finally, models with 9 hidden units perform best out of all models in terms of high accuracy and low computational time. Also, we find an interesting relationship between the number of hidden neurons and validation accuracy: the more hidden units a model has, the higher the accuracy it may get when validated.

Compared with the results in other relevant articles with the same benchmark dataset, our model achieves a high accuracy of 87%, which is almost higher by 5% than the best result reported in the relevant literature so far. Our best model contains 9 hidden neurons, using our Average Random Choosing method. The ratio of training–validation–test data is 800:100:100. If we take validation results into consideration, the highest accuracy reaches 92%, which is almost 10% higher than the best result from existing models.

5. Conclusions and discussion

In this paper, we presented a high performance credit rating model using MLP neural networks. The MLP model contains 9 hidden units, trained using the BP algorithm. The ratio of training–validation–test data is 800:100:100, and all instances in these

groups are chosen by our Average Random Choosing method. This credit rating model can achieve an accuracy of 87% in our experiments, which is higher by 5% than the best results of existing work related to the German dataset. We also prove that our method can be extended to other credit datasets, like the Australian credit dataset. Our Average Random Choosing method can improve the performance of the MLP training with both highly imbalanced datasets and slightly imbalanced ones. Thus, it can be applied to other training approaches with different real world datasets. Additionally, we discover that the error rate of MLP has a tendency of decreasing when the number of hidden units increases. This tendency should be researched more deeply in the future.

However, as to extremely imbalanced data, i.e. when the ratio f two classes is higher than 99:1, or there are too few instances in one class, the performance of our Average Random Choosing method will be hindered. This is because our method does not generate any new instances. In this situation, either oversampling of the minority class or other data mining models should be considered.

Future work will focus on giving explanations of the credit scoring results automatically. The reasons for rejection are important to both applicants and financial institutions. Feature selection methods will be researched for choosing out more important factors when evaluating credit applications. This will significantly reduce computational complexity of the scoring model and enhance its performance. Big data problems affect credit scoring as well, since the data collecting speed of banks is inconceivably high. Credit scoring models that can be updated with new data could be necessary when pursuing higher predicting accuracies.

Acknowledgement

This work is supported by National Key Technology R&D Program of China during the 12th Five-Year Plan Period (Project Number: 2012BAJ18B07).

References

Abbass, H. A. (2003). Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Computation, 15*, 2705–2726.

Bache, K., & Lichman, M. (2013). {UCI} Machine learning repository. In University of California, Irvine, School of Information and Computer Sciences.

Baesens, B., Gestel, T. V., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *The Journal of the Operational Research Society, 54*, 627–635.

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications, 39*, 3446–3453.

Chi, B.-W., & Hsu, C.-C. (2012). A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. *Expert Systems with Applications, 39*, 2650–2661.

Gonen, G. B., Gonen, M., & Gurgen, F. (2012). Probabilistic and discriminative group-wise feature selection methods for credit risk analysis. *Expert Systems with Applications, 39*, 11709–11717.

Hens, A. B., & Tiwari, M. K. (2012). Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method. *Expert Systems with Applications, 39*, 6774–6781.

Jensen, H. L. (1992). Using neural networks for credit scoring. *Managerial Finance, 18*, 15.

Khashei, M., Rezvan, M. T., Hamadani, A. Z., & Bijari, M. (2013). A bi-level neural-based fuzzy classification approach for credit scoring problems. *Complexity, 18*, 46–57.

Khashman, A. (2008). A modified backpropagation learning algorithm with added emotional coefficients. *IEEE Transactions on Neural Networks, 19*, 1896–1909.

Khashman, A. (2010). Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications, 37*, 6233–6239.

Khashman, A. (2011). Credit risk evaluation using neural networks: Emotional versus conventional models. *Applied Soft Computing, 11*, 5477–5484.

Lee, T.-S., & Chen, I. F. (2005). A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with Applications, 28*, 743–752.

Luo, S.-T., Cheng, B.-W., & Hsieh, C.-H. (2009). Prediction model building with clustering-launched classification and support vector machines in credit scoring. *Expert Systems with Applications, 36*, 7562–7566.

Marcano-Cedeño, A., Marin-de-la-Barcena, A., Jimenez-Trillo, J., Piñuela, J. A., & Andina, D. (2011). Artificial metaplasticity neural network applied to credit scoring. *International Journal of Neural Systems, 21*, 311–317.

Marqués, A. I., García, V., & Sánchez, J. S. (2012). Exploring the behaviour of base classifiers in credit scoring ensembles. *Expert Systems with Applications, 39*, 10244–10250.

Marques, A. I., Garcia, V., & Sanchez, J. S. (2013). A literature review on the application of evolutionary computing to credit scoring. *Journal of the Operational Research Society, 64*, 1384–1399.

Marqués, A. I., García, V., & Sánchez, J. S. (2013). On the suitability of resampling techniques for the class imbalance problem in credit scoring. *Journal of the Operational Research Society, 64*, 1060–1070.

Oreski, S., Oreski, D., & Oreski, G. (2012). Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. *Expert Systems with Applications, 39*, 12605–12617.

Peng, Y., Kou, G., Shi, Y., & Chen, Z. (2008). A Multi-criteria convex quadratic programming model for credit data analysis. *Decision Support Systems, 44*, 1016–1030.

Ping, Y., & Yongheng, L. (2011). Neighborhood rough set and SVM based hybrid credit scoring classifier. *Expert Systems with Applications, 38*, 11300–11304.

Saberi, M., Mirtalaie, M. S., Hussain, F. K., Azadeh, A., Hussain, O. K., & Ashjari, B. (2013). A granular computing-based approach to credit scoring modeling. *Neurocomputing, 122*, 100–115.

Setiono, R., Baesens, B., & Mues, C. (2011). Rule extraction from minimal neural networks for credit card screening. *International Journal of Neural Systems, 21*, 265–276.

Šušteršič, M., Mramor, D., & Zupan, J. (2009). Consumer credit scoring models with limited data. *Expert Systems with Applications, 36*, 4736–4744.

Thomas, L. C., Edelman, D. B., & Crook, J. N. (2002). *Credit scoring and its applications*. Philadelphia, PA: SIAM.

Tsai, C.-F., & Wu, J.-W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert Systems with Applications, 34*, 2639–2649.

Vukovic, S., Delibasic, B., Uzelac, A., & Suknovic, M. (2012). A case-based reasoning model that uses preference theory functions for credit scoring. *Expert Systems with Applications, 39*, 8389–8395.

Wang, J., Hedar, A.-R., Wang, S., & Ma, J. (2012b). Rough set and scatter search metaheuristic based feature selection for credit scoring. *Expert Systems with Applications, 39*, 6123–6128.

Wang, G., Ma, J., Huang, L., & Xu, K. (2012a). Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems, 26*, 61–68.

Werbos, P. J. (1975). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Harvard University.

West, D. (2000). Neural network credit scoring models. *Computers & Operations Research, 27*, 1131–1152.

Xiao, J., Xie, L., He, C., & Jiang, X. (2012). Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications, 39*, 3668–3675.

Yu, L., Yao, X., Wang, S., & Lai, K. K. (2011). Credit risk evaluation using a weighted least squares SVM classifier with design of experiment for parameter selection. *Expert Systems with Applications, 38*, 15392–15399.

Zhong, H., Miao, C., Shen, Z., & Feng, Y. (2014). Comparing the learning effectiveness of BP, ELM, I-ELM, and SVM for corporate credit ratings. *Neurocomputing, 128*, 285–295.